Я

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Франчін Вікторії Паолівни*
(ПІБ)

академічної групи *122-19-4*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка вебзастосунку для публікації новин та оголошень університету з використанням Spring Boot і ReactJS*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Реута О.В.*			
розділів:				
спеціальний				
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-4* (група) *Франчін В.П.* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка вебзастосунку для публікації новин та оголошень університету з використанням Spring Boot і ReactJS*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проектно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***14.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***12.06.2023 р.***

Завдання видав _____ *доц. Реута О.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Франчін В. П.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 92 с., 31 рис., 4 дод., 34 джерела.

Об'єкт розробки: клієнтська та серверна сторона сайту для розміщення новин університету.

Мета кваліфікаційної роботи: розробка сайту для публікації новин та створення ефективного інформаційного середовища в університеті.

У вступі розглядається аналіз обраної предметної галузі та наявні технічні рішення, конкретизується мета та призначення кваліфікаційної роботи, наведено обґрунтування актуальності обраної теми роботи та уточняється постановка завдання.

У першому розділі проаналізовано предметну галузь та наявні аналоги розроблюваного застосунку, розглянуто існуючі технічні рішення, обґрунтовано актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації інформаційної системи та її функціоналу, технологій та програмних засобів для розробки застосунку.

У другому розділі описано перелік обраних інструментів для розробки та принципи їх роботи, описано архітектуру створеної ІС та алгоритми її функціонування, описано структуру бази даних, визначено перелік та формат вхідних і вихідних даних, а також проілюстровано процес роботи з застосунком, та приведено використані технічні та програмні засоби.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, виконано розрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні сайту для розміщення важливих новин та оголошень університету, реалізації функцій пошуку та зручного перегляду публікацій користувачами, а також забезпеченні можливості менеджменту контенту на сайті його адміністраторами із використанням додаткового вебзастосунку для адміністрації.

Актуальність розробки сайту для публікації новин НТУ «Дніпровська політехніка» полягає у відсутності єдиної, незалежної від інших, платформи для інформування студентів в той час як більшість сучасних навчальних закладів мають власні новинні портали та блоги, що вбудовані у структуру їх основного сайту або реалізовані як окремий сайт.

Список ключових слів: САЙТ, ВЕБЗАСТОСУНОК, ІНФОРМАЦІЙНИЙ ПОРТАЛ, JAVA, SPRING BOOT SPA, HTML, CSS, JAVASCRIPT, REACTJS.

ABSTRACT

Explanatory note: --- p., --- fig., 3 add., --- sources.

Object of development: client and server side of the site for posting university news.

The purpose of the qualification work: development of a website for publishing news and creating an effective information environment at the university.

The introduction examines the analysis of the selected subject area and the available technical solutions, specifies the purpose and purpose of the qualification work, provides the rationale for the relevance of the chosen topic of work, and specifies the statement of the task.

In the first section, the subject field and existing analogues of the developed application are analyzed, existing technical solutions are considered, the relevance of the task and the purpose of development are substantiated, the task statement is formulated, the requirements for the software implementation of the information system and its functionality, technologies and software tools for the development of the application are specified.

The second section describes the list of selected tools for development and their principles of operation, describes the architecture of the created IS and algorithms for its operation, describes the structure of the database, defines the list and format of input and output data, and also illustrates the process of working with the application, and gives the used technical and software tools.

In the economic section, the labor intensity of the developed information system is determined, the cost of work on creating the application is calculated, and the time for its creation is calculated.

The practical significance lies in the creation of a site for posting important news and announcements of the university, the implementation of search functions and the convenient viewing of publications by users, as well as the provision of the possibility of content management on the site by its administrators using an additional web application for administration.

The relevance of developing a site for publishing news of the Dnipro Polytechnic National Technical University lies in the lack of a single, independent platform for informing students, while most modern educational institutions have their own news portals and blogs, which are embedded in the structure of their main site or implemented as a separate site

List of keywords: SITE, WEB APPLICATION, INFORMATION PORTAL, JAVA, SPRING BOOT SPA, HTML, CSS, JAVASCRIPT, REACTJS.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ІС – інформаційна система;

API – Application Programming Interface;

CMS – Content Management System;

CRUD – Create, Read, Update, Delete

CSS – Cascading Style Sheets;

HTML – Hypertext Markup Language;

HTTP – Hypertext Transfer Protocol;

JS – JavaScript;

JSON – JavaScript Object Notation;

MPA – Multi page Application (багатосторінковий вебзастосунок)

REST – Representational State Transfer;

SPA – Single Page Application (односторінковий вебзастосунок);

SQL – Structured Query Language;

UI – User Interface (інтерфейс користувача);

UX – User Experience (досвід користувача).

ЗМІСТ

ВСТУП

Розроблена інформаційна система призначена для використання в галузі освіти та представляє собою інформаційний портал.

Корпоративний інформаційний портал – це сайт, орієнтований на відносно вузьку аудиторію, в даному випадку – студентів, що навчаються в НТУ “Дніпровська політехніка”, їх викладачів, а також тих, хто планує вступати до цього закладу.

В умовах навчання на дистанційній формі характерне зростання потреби в ефективному забезпеченні інформаційно-комунікаційного середовища у вищих навчальних закладах. В наш час наявність корпоративного інформаційного порталу дозволяє значно полегшити та покращити процес взаємодії між студентами, викладачами та адміністрацією ВУЗу. Такий ресурс надає можливість швидко та зручно отримувати необхідну інформацію про важливі оголошення, актуальні новини та події університету.

Отже, головною метою корпоративного інформаційного порталу є забезпечити швидкий та зручний доступ до актуальної інформації та налагодити ефективну комунікацію між усіма учасниками навчального процесу.

На даний момент основними інформаційними каналами університету є різні соціальні мережі, але вони не охоплюють ту частину аудиторії, яка не користується даними застосунками. Перевагою сайту-порталу є те, що він не залежить від сторонніх платформ та не потребує від користувачів завантаження на їх пристрої додаткових застосунків. Завдяки цьому, такий канал інформації як звичайний вебсайт дозволить публікаціям набирати більше переглядів, ніж їх розміщення лише на сторінках в соціальних мережах.

Для того, щоб інформаційний портал був ефективним, він має відповідати переліку певних вимог:

- стабільно функціонувати при достатньо великій кількості запитів та забезпечувати якомога швидшу відповідь на запити користувачів;
- забезпечувати функції зберігання та обробки даних;

- бути безпечним та захищеним від спроб злому;

- для зручності користувачів необхідно дотримуватися принципу простоти та зрозумілості вебдизайну та інтерфейсу, розробити його із врахуванням до уваги користувальницького досвіду. Також бажано, щоб інтерфейс був адаптивним та однаково добре виглядав на різних розмірах екрану;

- інформаційний портал повинен постійно оновлюватися та доповнюватися новими матеріалами, для цього потрібно забезпечити простоту менеджменту контенту на сайті та швидке розміщення публікацій, надавати функції їх редагування та видалення;

- мати можливість для розвитку та впровадження нового функціоналу до вже розробленої інформаційної системи.

Отже, дана система може значно полегшити та покращити процес співпраці між студентами, викладачами та адміністрацією вузу, що однозначно призведе до підвищення якості освіти.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА

1.1. Загальні відомості з предметної галузі

Вища освіта завжди була й залишається конкурентним ринком, який у наші дні все більше переходить до цифрового формату. Це означає, що для підвищення конкурентоспроможності навчальним закладам необхідний розвинений вебсайт, активна присутність в соціальних мережах, достатні інвестиції в SEO-діяльність та блог, який представляє цінність для своїх читачів. Хороший вебсайт університету може допомогти привернути увагу вступників та стати цінним ресурсом для студентів та викладачів.

Університетський блог новин може охоплювати багато різних тем, наприклад, надавати інформацію про вступ, життя у гуртожитку, навчальні програми, надавати поради з пошуку роботи, просувати діяльність, що спонсорується університетом. Даний інструмент дозволяє знайти більше потенційних студентів, збільшити кількість відвідувань головного сайту університету, забезпечує університет контентом, яким можна поділитись у соціальних мережах, що також збільшить вебтрафік та сприятиме росту популярності університету. Тому серед сучасних закладів вищої освіти стає дедалі популярнішим створення власних сайтів та ведення блогів.

Розроблювана в цій кваліфікаційній роботі інформаційна система виступатиме новинним порталом, в якому публікуватимуться актуальні новини та оголошення Національного технічного університету “Дніпровська політехніка”.

У обраній предметній галузі можна виділити два основні аспекти: специфіка створення ефективних новинних порталів, спрямованих на аудиторію вищого навчального закладу та особливості і технології розробки сучасних вебсайтів.

Так як розроблюваний вебзастосунок є насамперед новинним порталом, то варто окремо розглянути загальні особливості ведення сучасних новинних порталів, їх структуру та функції.

У дослідженні університету Джорджії в США, "Вивчення вебсайтів новин з професійної точки зору"[1], було досліджено моделі інтеграції інтерактивних функцій на новинних сайтах. Результати цього дослідження показали, що вебсайти ЗМІ переважно використовують функції, які дозволяють користувачам взаємодіяти з контентом без можливості впливати на нього. Така тенденція пояснюється тим, що функції, які дозволяють користувачам впливати на наповнення сайту (додавати власні публікації, тощо), порушують історично складений односторонній потік комунікації між ЗМІ та аудиторією. Більш тонкі способи впливу на контент новинного порталу, такі як різні варіанти оцінювання публікацій, перекладають роль визначення рівня важливості публікацій на аудиторію. Інші функції, такі як пересилання публікацій, поширення в соціальних мережах, мультимедійні опції та персоналізація, не викликають подібних конфліктів. Також при проектуванні інформаційного порталу важливу роль відіграє UX, оскільки існує досить багато ефективних рішень, до яких звикла більшість користувачів.

Як приклад хорошої реалізації сайту новин навчального закладу, можна навести блоги університетів ВУЗу King's College London (рис. 1.1) та University of Glasgow (рис. 1.2), що зайняли перше та друге місце в рейтингу на онлайн ресурсі FeedSpot[2]. Наведені сайти об’єднує лаконічний та адаптивний дизайн (рис. 1.1 - рис. 1.2), а також зручна навігація та функції пошуку за великою кількістю тем та розділів.

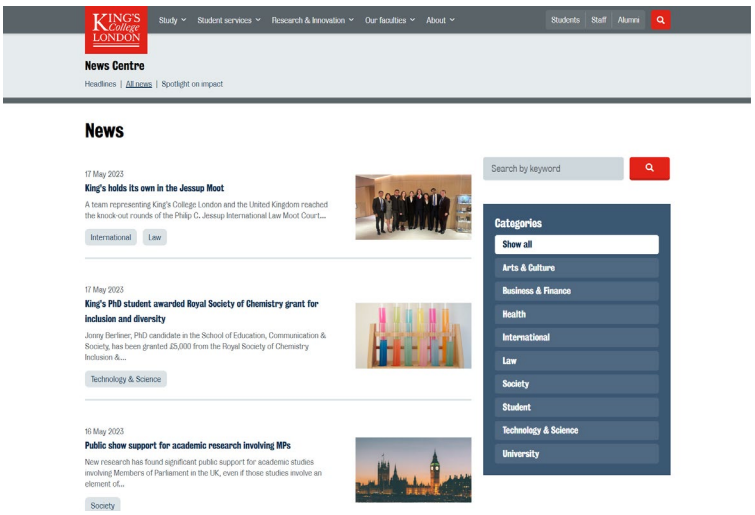

Рис. 1.1. Головна сторінка сайту новин британського ВУЗу King's College London.

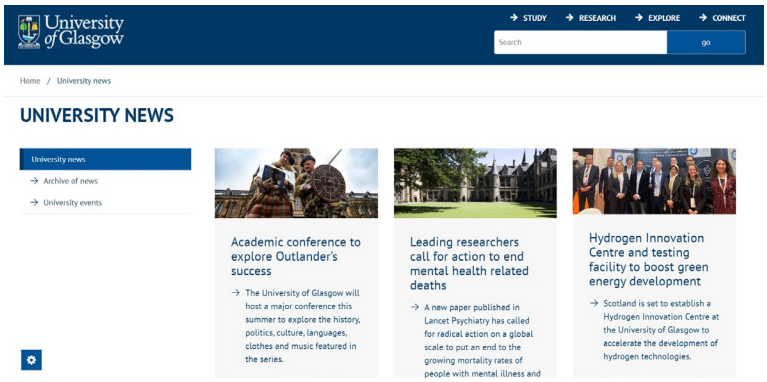

Рис. 1.2. Головна сторінка сайту новин американського університету University of Glasgow.

У наш час існує кілька найпоширеніших способів розробки вебсайтів. Найпростіші з них - використання конструкторів сайтів та платформ CMS. CMS - це платформи для управління контентом на сайті, які надають різні функціональні можливості та інструменти для SEO. Одними з найпопулярніших є WordPress та Wix.

Незважаючи на те, що CMS чудово підходять для розробки більшості типів сайтів, таких як портфоліо, візитки, блоги тощо, розробка сайту з нуля без конструкторів часто необхідна при розробці чогось принципово нового, з унікальною або складною функціональністю, яка не покривається можливостями CMS. Сайти, розроблені з нуля, є більш гнучкими та мають перевагу швидкості завантаження, оскільки вони не використовують базу даних CMS для завантаження контенту. Крім того, написання сайту з нуля дозволяє розробникам

поліпшити свої навички у програмуванні. Переважно саме з цієї причини цей підхід був обраний для даної кваліфікаційної роботи.

Основними інструментами веброзробки залишаються мова розмітки HTML (Hypertext Markup Language), що задає структуру вебсторінок, мова таблиці стилів CSS (Cascading Style Sheets), що відповідає за зовнішній вигляд елементів сторінок, та JavaScript, яка є однією з основних мов програмування для веброзробки і використовується для додавання інтерактивності до вебсторінок, розробки вебдодатків та взаємодії з вебсерверами.

Існує також багато інших інструментів, бібліотек та фреймворків для frontend та backend розробки, а також роботи із базами даних, які розширюють можливості вже існуючих технологій. Інструменти, що було використано в даній роботі детальніше описано у другому розділі.

Вебсайти поділяють на статичні та динамічні. Статичні вебсторінки - це прості сторінки, написані мовами такими, як HTML, CSS і т. д. При отриманні запиту сервер надсилає відповідь клієнту без будь-яких додаткових операцій. На статичних вебсайтах сторінки залишаються незмінними, поки їх не відредагують вручну, а користувач не може змінити їх вміст та взаємодіяти з ними. Динамічні вебсторінки є більш функціональними та забезпечують інтерактивний інтерфейс користувача, але є складнішими в розробці і завантажуються довше, ніж статичні вебсторінки. Вони змінюють вміст або оформлення з кожним запитом до вебсервера і здатні генерувати різний вміст для різних відвідувачів залежно від їх запитів. Типова структура простого динамічного вебсайту наведена на рисунку 1.3.

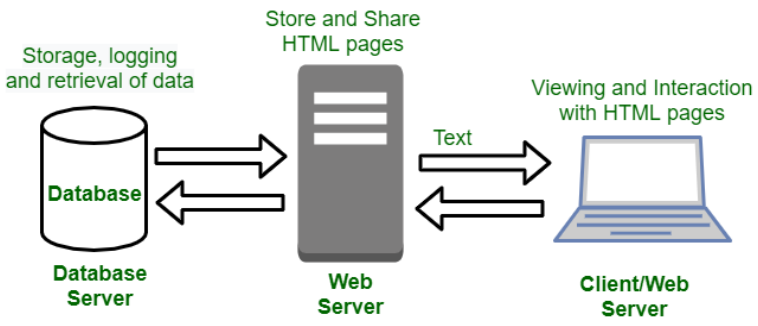

Рис. 1.3. Типова структура динамічного сайту.

SPA та MPA - це дві основні моделі проектування вебдодатків: односторінковий додаток (SPA) та багатосторінковий додаток (MPA). Порівнянню даних моделей архітектури вебзастосунків присвячені роботи[3-4].

Односторінкові додатки (скорочено SPA) - це вебдодатки, які завантажують усі дані під час першого кліку на сторінці та потім лише приймають відповіді на запити у JSON форматі і оновлюють тільки необхідні компоненти, а не перезавантажують сторінку при кожній взаємодії з її елементами. Навпаки, багатосторінкові додатки завантажують всю сторінку повністю. Продуктивність і час відгуку мають велике значення для покращення загального користувацького досвіду. Приблизно 52% користувачів очікують, що вебдодатки завантажуватимуться протягом двох секунд. SPA дозволяє забезпечити швидкість завантаження, простоту розробки та можливість кешування даних, однак має обмеження щодо SEO-оптимізації Для розробки односторінкових додатків використовують такі фреймворки як Angular, Vue, React, Ember.

MPA - це окремо створені кілька сторінок, які об'єднуються разом і утворюють вебсайт. На відміну від SPA, багатосторінкові додатки вимагають декількох рівнів дизайну інтерфейсу, оскільки це набір різних сторінок. Вони перезавантажуються, оновлюючи ресурси, такі як CSS, HTML і скрипти, щоразу, коли користувачі натискають на вкладку. Ці вебсайти передають багато даних від сервера до клієнта і від клієнта до сервера, тому вони мають низьку швидкість роботи. MPA дозволяє легше управляти SEO, однак потребує більшого обсягу коду. Вибір між SPA і MPA залежить від потреб і характеру проекту. Якщо потрібна проста інтерактивність і швидкість, SPA може бути кращим варіантом. Якщо проектом зі складним вмістом та має потребу у SEO, MPA може підійти краще. Враховуючи зростаючу популярність SPA та його переваги, багато додатків переходять до цієї моделі. Приймаючи до уваги ці фактори, для даної кваліфікаційної роботи була обрана модель SPA.

1.2. Призначення розробки та галузь застосування

Об’єктом розробки є інформаційна система “Dniprotech News”, яка може використовуватись для розміщення публікацій, новин, статей від університету НТУ «Дніпровська Політехніка». Її мета полягає у створенні єдиного інформаційного простору для студентів, де розміщуватимуться всі публікації з важливими новинами.

Так як наразі інформування від університету відбувається через декілька різних соцмереж, а точніше Facebook, Telegram та Instagram, є не дуже зручним кожного разу перевіряти всі канали інформування для того, щоб нічого не пропустити. До того ж частина студентів не користується всіма переліченими соцмережами. Тому сайт, який не залежить від інших платформ та який міститиме всі новини в одному місці та швидкому доступі покращить якість поінформованості студентів та забезпечить зручний доступ до інформації.

Розроблена ІС складається із трьох частин: серверна, клієнтська та адміністративна.

Frontend частина клієнта призначена для перегляду та пошуку користувачами публікацій.

Frontend частина адміністратора призначена для менеджменту контенту на основному сайті та реалізує операції CRUD: створення, читання, оновлення, видалення публікацій.

Розроблений сайт має універсальну структуру, тому, залежно від опублікованого контенту його можна використовувати як звичайний блог або налаштувати для використання у будь-яких інших закладах.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки «виконання кваліфікаційної роботи» є:

- освітня програма 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» від 16.05.2023 № 350-с;
- завдання на кваліфікаційну роботу на тему «Розробка вебзастосунку для публікації новин та оголошень університету з використанням Spring Boot і ReactJS».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є розробка новинного сайту університету. Система призначена для надання можливості публікації оголошень адміністрацією та їх зручного перегляду та пошуку відвідувачами сайту. Дана кваліфікаційна робота включає розробку серверу для роботи із базою даних та двох окремих вебзастосунків: один для звичайних користувачів та другий для менеджменту контенту адміністраторами.

Для досягнення мети у розробці проекту, необхідно виконати наступні кроки:

- дослідити предметну галузь поставленої задачі;
- створити попередній дизайн готового сайту;
- розробити базу даних;
- розробити серверну частину застосунку;
- розробити клієнтську частину застосунку для відвідувачів;
- розробити клієнтську частину застосунку для адміністраторів.

Попередній дизайн сайту було виконано у популярному редакторі для розробки інтерфейсів Figma[5]. При безпосередній розробці вебзастосунку деякі деталі дизайну було замінено, однак ці зміни ніяк не впливають на функціонал. Так як для застосунку було обрано архітектуру SPA, дизайн створено із врахуванням зручності керуванням змістом сторінки, шапка та футер сайту

залишатимуться статичними, а контент основної частини змінюватиметься залежно від запиту користувача: результати пошуку або вибір публікації для перегляду. Дизайн головної сторінки сайту та дизайн сторінки для перегляду публікації наведено відповідно на рисунках 1.4-1.5.

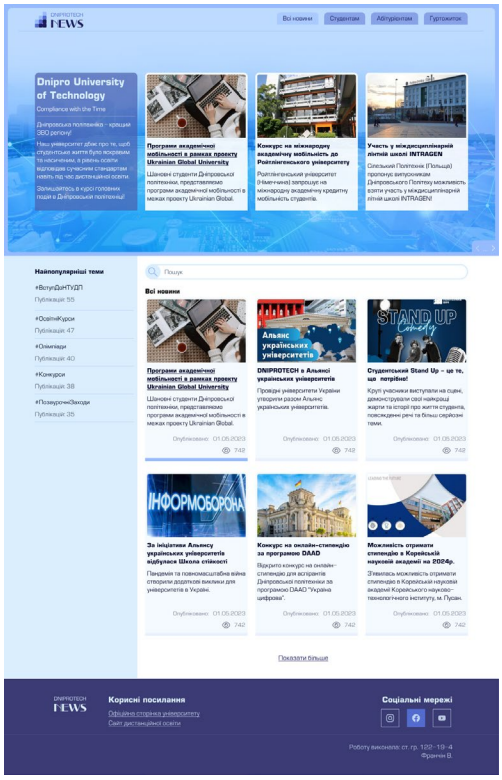

Рис. 1.4. Дизайн головної сторінки сайту.

Рис. 1.5. Дизайн сторінки перегляду публікації.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для розробки клієнтської частини проекту будуть використані фреймворки ReactJS і Tailwind CSS для забезпечення інтерактивного і привабливого інтерфейсу користувача.

Для розробки серверної частини було обрано мову програмування Java, а також будуть використані технології Spring, включаючи Spring Boot, Spring JPA і Spring Web.

Для роботи з базою даних проекту буде використана система керування реляційними базами даних MariaDB.

У вебзастосунку для відвідувачів повинен бути реалізований наступний функціонал:

- слайдер із закріпленими важливими публікаціями;
- список популярних хештегів, що базується на сумарній кількості переглядів публікацій за даним тегом, а також та швидкий пошук по ним;
- зручний перегляд публікацій з можливістю швидкої навігації між ними;
- можливість пошуку публікацій за хештегами та текстом.

У вебзастосунку для адміністраторів повинен бути реалізований наступний функціонал:

- перегляд списку наявних публікацій та списку хештегів;
- створення нових публікацій;
- редагування та видалення доданих раніше публікацій для внесення змін або виправлення помилок;
- можливість закріплення публікації на слайдері, що дозволить виділити важливу інформацію для користувачів;
- додавання та видалення хештегів для керування тегами і підвищення зручності пошуку.

Обидва вебзастосунки повинні мати адаптивний дизайн для зручного перегляду на різних пристроях, що забезпечить позитивний користувацький

досвід (UX). Також необхідно розробити REST API, що реалізовуватиме функціонал CRUD для забезпечення обміну даними між клієнтом і сервером. Дані, які передаються від сервера, будуть мати JSON формат для зручності обробки та взаємодії з клієнтом.

1.5.2. Вимоги до інформаційної безпеки

У проекті відсутня потреба зберігати приватну інформацію про користувачів, яка потребувала б шифрування. Враховуючи це, інформаційна система не вимагає особливих заходів щодо забезпечення конфіденційності даних. Однак, важливо зазначити, що враховуються загальні принципи безпеки, які мають бути виконані для забезпечення надійності системи.

У проекті була обрана структура, що розділяє функціонал адміністраторів та відвідувачів на два окремі вебзастосунки. Це забезпечує відокремлення доступу та захист даних у базі даних. Відвідувачі мають лише можливість завантажувати та переглядати дані з бази даних, тоді як функції для керування вмістом бази даних доступні лише в адміністративному вебзастосунку. Це допомагає запобігти несанкціонованому доступу користувачів та забезпечує контроль над даними.

Перевірка вхідної інформації: Вебзастосунок повинен забезпечувати контроль та перевірку введеної користувачем інформації. Система повинна попереджати користувача про некоректний ввід та не дозволяти введення недопустимих значень. Це допомагає забезпечити правильність та цілісність введених даних.

1.5.3. Вимоги до складу та параметрів технічних засобів

Наведені технічні характеристики є рекомендованими, тобто при наявності технічних засобів із характеристиками не нижче зазначених, розроблена система функціонуватиме відповідно до вимог.

- Операційна система: Windows 10 або Linux;

- Оперативна пам'ять (RAM): не менше 8 ГБ;
- Відеопам'ять (VRAM): 64 МБ;
- Швидкісний Інтернет для забезпечення швидкого завантаження та передачі даних;
- Жорсткий диск (HDD) або накопичувач (SSD) об'ємом не менше 16 ГБ;
- Процесор: рекомендується Intel Core i5, 2.30GHz.

Зі сторони користувача для доступу до системи з будь-якого технічного засобу необхідні лише вихід у інтернет та сучасний браузер, бажано, оновлений до останньої версії.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для коректної експлуатації програми необхідне наступне програмне забезпечення, встановлене на ПК:

- операційна система Windows 10 або Linux;
- Java Version 19;
- система керування базами даних MariaDB, яка є необхідною для зберігання та обробки даних, що використовуються програмою;
- для роботи із кодом серверної частини програми: середовище розробки на Java – Eclipse або IntellijIDEA;
- для роботи із кодом клієнтської частини програми: середовище для розробки вебзастосунків як, наприклад, Visual Studio Code.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Основним функціональним призначенням системи "Dniprotech News" є створення зручного та доступного середовища для публікації новин та інформації для студентів університету НТУ "Дніпровська Політехніка" та всіх зацікавлених користувачів.

Розроблена система включає два вебзастосунки: один для звичайних відвідувачів та інший для адміністраторів. Обидва вебзастосунки мають зручний та інтуїтивно зрозумілий інтерфейс.

Адміністратори мають повний контроль над системою і можуть створювати та публікувати новини, статті або інші інформаційні матеріали від університету.

Кожна публікація має наступні атрибути: заголовок, короткий опис, зображення обкладинки, зміст та перелік тегів. Адміністратори можуть створювати нові публікації, редагувати існуючі та публікувати їх на вебсайті. Важливі публікації можуть бути закріплені у слайдері на головній сторінці, щоб звернути на них особливу увагу відвідувачів.

Система підтримує використання хештегів для класифікації новин. Адміністратори можуть прикріплювати хештеги до публікації для позначення теми або розділу на сайті. Це допомагає організувати публікації та спростити їх пошук для користувачів.

Головна сторінка вебсайту містить слайдер з закріпленими публікаціями, меню навігації, де можна переключатися між основними розділами, та панель популярних хештегів, за допомогою якої відвідувачі можуть швидко переглянути новини, пов'язані з конкретними тегами.

Крім того, користувачам надається можливість виконувати пошук новин за ключовими словами або хештегами. Це дозволяє знайти потрібну інформацію швидко та зручно.

2.2. Опис застосованих математичних методів

У зв'язку зі специфікою обраної області розробки, програма не вимагає використання складних математичних методів, за винятком стандартних математичних операцій, які використовуються для внутрішніх обчислень і логічних операцій.

2.3. Опис використаних технологій та мов програмування

Розроблена ІС має клієнт-серверну архітектуру, що відповідає принципам REST, реалізує RESTful API та використовує HTTP протокол для обміну даними, дані передаються у форматі JSON. Сервер був написаний на мові програмування Java із використанням наступних технологій:

- Spring boot;
- Spring JPA;
- Spring Web;
- MariaDB;
- Hibernate.

Клієнтська частина системи була написана на мові програмування JavaScript та з використанням:

- ReactJS;
- React router dom;
- React redux;
- Redux;
- Tailwind;
- PostCSS;
- Webpack.

Далі наведено короткий опис та принципи функціонування кожної із використаних технологій та мов програмування:

REST та RESTfull API

REST (Representational State Transfer) - це стиль архітектури програмного забезпечення, що використовується для побудови розподілених систем. REST

визначає набір принципів і обмежень, які сприяють створенню масштабованих, міжплатформерних та ефективних веб-додатків[6].

RESTful API[7] - це найпоширеніші API у світі веб-сервісів. Вони використовують протокол передачі гіпертексту (HTTP) для створення, читання, оновлення та видалення даних (CRUD). Вони популярні завдяки своїй простоті, масштабованості, швидкості та здатності працювати з усіма типами даних.

Інтерфейс прикладного програмування (API) - це набір правил, за допомогою яких дві програми спілкуються між собою та обмінюються даними.

RESTful API - це API, який відповідає архітектурним обмеженням Representational State Transfer (REST). Ці обмеження розробки - ряд правил, яких повинні дотримуватися API - дозволяють створювати швидші, масштабовані API, які підтримують всі типи даних. Завдяки цьому RESTful API стали найпоширенішими API у світі, особливо для веб-сервісів.

Архітектурні обмеження REST надають основу для побудови ефективних та масштабованих веб-сервісів. Розглянемо кожне обмеження детальніше:

1) Клієнт-сервер: це обмеження визначає чітку розмежованість ролей клієнта і сервера. Сервер відповідає за зберігання та обробку даних, тоді як клієнт звертається до сервера для отримання цих даних і може вносити зміни за необхідності. Це дозволяє незалежне масштабування та розвиток обох компонентів системи.

2) Єдиний інтерфейс: клієнт повинен отримувати інформацію від сервера в узгодженому та зручному для читання форматі. Уніфікований інтерфейс повинен відповідати кільком принципам, зокрема: використовувати гіпермедіа як двигун стану додатку (HATEOAS), кожне повідомлення, що передається між клієнтом і сервером, повинно містити достатньо інформації для його обробки, тобто запит від клієнта до сервера повинен містити ідентифікатор ресурсу та точно вказувати дію, яку клієнт хоче здійснити (створити, прочитати, оновити або видалити), ресурси, що згадуються в повідомленнях, повинні бути ідентифіковані за допомогою унікального URI (Uniform Resource Identifier), що дозволяє клієнтам однозначно звертатись до конкретних ресурсів на сервері.

3) Відсутність стану: сервер не зберігає жодної інформації про стан клієнта між запитами. Кожен запит клієнта повинен містити всю необхідну інформацію для обробки запиту, і сервер повинен надіслати повну відповідь. Це спрощує масштабування сервера і робить його більш незалежним від клієнтських сесій.

4) Кешування: дозволяє зберігати копії часто запитуваних даних на клієнті або на проміжному сервері (кеші). Коли клієнт або інші клієнти звертаються до тих самих даних, вони отримують відповідь з кешу, що зменшує навантаження на сервер і покращує продуктивність.

5) Багаторівнева система: передбачає розподілення функціональності системи на різні рівні абстракції або шари. Це дозволяє підвищити модульність і масштабованість системи, а також полегшує розвиток та зміну окремих компонентів без впливу на інші частини.

6) Код за вимогою: це обмеження передбачає можливість передачі виконуваного коду від сервера до клієнта. Клієнт може вимагати виконання певних операцій, що реалізовані на сервері, замість того, щоб виконувати їх локально. Це дозволяє зменшити навантаження на мережу та забезпечити більшу гнучкість у виконанні дій.

RESTful API складаються з кінцевих точок (URL-адреси ресурсів), методів (GET, PUT, POST, DELETE), заголовків (метадані та інформація про статус) та даних (інформація про ресурси). Архітектурні обмеження REST, такі як розмежування ролей клієнта і сервера, відсутність стану на сервері та можливість кешування, забезпечують ефективну та незалежну роботу компонентів системи.

Переваги REST архітектури:

Масштабованість: REST дозволяє побудувати масштабовані системи, де можна додавати нові сервери або розподіляти ресурси безперервно.

Міжплатформовість: REST не залежить від конкретних платформ або мов програмування, що дозволяє використовувати різні технології для клієнтів і серверів.

Простота: REST простий у використанні та розумінні. Використання HTTP методів та URI для взаємодії з ресурсами спрощує розробку та розуміння системи.

HTTP

HTTP (The Hypertext Transfer Protocol) – це прикладний протокол для розподілених, спільних, гіпермедійних інформаційних систем, який дозволяє користувачам обмінюватися даними у всесвітній павутині. HTTP надає користувачам можливість взаємодіяти з веб-ресурсами, такими як HTML-файли, шляхом передачі гіпертекстових повідомлень між клієнтами і серверами. Клієнти HTTP зазвичай використовують з'єднання за протоколом управління передачею (TCP) для зв'язку з серверами.

HTTP використовує спеціальні методи запитів для виконання різних завдань, такі як GET і HEAD, POST, PUT, DELETE та інші.

JSON

JSON (JavaScript Object Notation) - це легкий формат обміну даними. Це текстовий формат, який є повністю незалежним від мови[8].

JSON побудований на двох структурах:

- Колекція пар ім'я/значення. У різних мовах це реалізується як об'єкт, запис, структура, словник, хеш-таблиця, список з ключами або асоціативний масив;
- Впорядкований список значень. У більшості мов реалізується як масив, вектор, список або послідовність.

Це універсальні структури даних, які підтримують практично всі сучасні мови програмування. Формат даних, взаємозамінний з мовами програмування, також має базуватися на цих структурах.

В JSON вони набувають такої форми:

Об'єкт - це невпорядкований набір пар ім'я/значення. Об'єкт починається з лівої дужки «{» і закінчується правою дужкою. За кожним іменем слідує двокрапка, а пари ім'я/значення розділені комою.

Java

Мова програмування Java орієнтована на об'єкти, характеризується суворою типізацією. Це універсальна програмна платформа, за допомогою якої розробляють настільні та мережеві, а також мобільні додатки.

Вона має наступні головні принципи:

- Об'єктно-орієнтованість: Java базується на об'єктно-орієнтованій парадигмі, що дозволяє створювати модульний та розширюваний код. Вона підтримує такі концепції, як спадкування, поліморфізм, інкапсуляція та абстракція.
- Переносимість: програми, написані на Java, можуть виконуватися на будь-якій платформі, яка має встановлену відповідну віртуальну машину Java (JVM).
- Безпека: Java має вбудовану систему безпеки, яка захищає від потенційно небезпечного коду. Вона використовує механізми, такі як управління пам'яттю, перевірку меж масивів та автоматичне усунення помилок.
- Велика бібліотека: Java має велику стандартну бібліотеку класів, яка надає готові рішення для багатьох завдань, таких як робота з мережею, робота з базами даних, графічний інтерфейс тощо. Це спрощує розробку програм і прискорює процес.

Spring boot

Spring[9] є найпопулярнішим інструментом при розробці промислових додатків на Java - від великих монолітних додатків до мікросервісів. Одразу варто зазначити, що Spring Boot і Spring Framework - це різні технології. Spring - це ціла екосистема для розробки Java, що включає величезну кількість готових модулів, таких як Spring MVC, Spring JDBC, Spring Security та інші. Більшість цих фреймворків може працювати незалежно один від одного, однак вони забезпечують більшу функціональність у разі спільного їхнього використання. Spring Boot, зі свого боку, є розширенням Spring, Spring, яке надає функцію RAD (Rapid Application Development - швидка розробка додатків) для Spring Framework. Він використовується для створення автономного Spring-додатку,

який можна просто запустити, оскільки він потребує мінімальної конфігурації Spring.

Spring Boot це не окремий фреймворк, а доповнення до Spring, яке полегшує роботу з ним. Spring потрібно конфігурувати для кожного нового проекту. Конфігурація може зайняти багато часу і не дати відчутних переваг у подальшій роботі. Щоб виправити проблему, було створено Spring Boot. Він включає комплекс утиліт для автоматизації налаштування.

Spring Boot:

- автоматично конфігурує проекти на основі одного зі стартових пакетів для них;
- полегшує створення та розгортання додатків на Spring;
- швидко і легко управляє залежностями і довантажує необхідні модулі;
- підтримує вбудований сервер для запуску додатків;
- може автоматично створити та налаштувати базу даних для додатка.

За необхідності налаштування Spring Boot можна змінити, щоб він конфігурував і налаштовував компоненти інакше.

Автоконфігурація - це метод роботи в Spring Boot, що дає змогу скоротити кількість дій, які повинні робити розробники. Він автоматично налаштовує додаток Spring на основі раніше доданих залежностей. Автоконфігурація Spring Boot пропонує кілька надійних функцій за замовчуванням, зберігаючи при цьому велику гнучкість.

Кожен додаток Spring Boot містить вбудований сервер. Вбудований сервер вбудовується як частина програми, що розгортається. Перевагою вбудованого сервера є те, що нам не потрібен попередньо встановлений сервер в середовищі. У Spring Boot вбудованим сервером за замовчуванням є Tomcat.

Переваги Spring Boot:

- створює автономні Spring-додатки, які можна запускати за допомогою Java -jar;

- легко тестує веб-додатки за допомогою різних вбудованих HTTP-серверів, таких як Tomcat, Jetty тощо;
- надає зручні "стартові" POM-файли, щоб спростити конфігурацію Maven;
- надає готові до виробництва функції, такі як метрики, перевірки працездатності та зовнішні конфігурації;
- не має вимог до конфігурації XML;
- пропонує інструмент CLI для розробки та тестування Spring Boot-додатків;
- мінімізує написання шаблонних кодів (код, який повинен бути включений в багато місць з невеликими змінами або без них), конфігурацію XML і анотації;
- підвищує продуктивність і скорочує час розробки.

Spring JPA

Spring Data JPA[10] – це модуль у складі Spring Framework, який надає спрощений спосіб роботи з базами даних в додатках на мові Java. Він пропонує високорівневий API для взаємодії з реляційними базами даних, використовуючи підхід Object-Relational Mapping (ORM).

Зазвичай для реалізації рівня доступу до даних доводиться писати занадто багато шаблонного коду для виконання простих запитів, а також для виконання пагінації та аудиту. Spring Data JPA має на меті значно полегшити реалізацію рівнів доступу до даних, зменшивши зусилля до того обсягу, який дійсно необхідний.

Принцип роботи Spring Data JPA базується на спрощенні розробки доступу до даних. Основні принципи, які використовуються в Spring Data JPA:

- автоматична генерація запитів: Spring Data JPA дозволяє створювати запити до бази даних на основі методів репозиторію. За допомогою конвенцій та назв методів, Spring Data JPA автоматично генерує SQL-запити, що зменшує необхідність ручного написання складних запитів;

- універсальні методи: Spring Data JPA надає набір універсальних методів, які можна використовувати для виконання стандартних операцій доступу до даних, таких як створення, читання, оновлення та видалення записів. Це дозволяє економити час на написанні рутинного коду для доступу до бази даних;

- підтримка пагінації та сортування: Spring Data JPA дозволяє легко виконувати пагінацію та сортування результатів запитів. При розробці можна вказувати кількість записів на сторінку та поле сортування, що спрощує роботу з великими обсягами даних;

- підтримка реляційних відношень: Spring Data JPA надає зручні способи встановлення та управління реляціями між сутностями бази даних. Він підтримує такі типи відношень, як один-до-одного, один-до-багатьох та багато-до-багатьох. В результаті не потрібно вручну виконувати складні операції злиття даних, оскільки Spring Data JPA автоматично керує цими відношеннями.

Переваги використання Spring Data JPA:

- зменшення кількості написаного коду: Spring Data JPA надає абстракцію доступу до даних, що дозволяє розробникам писати менше коду для реалізації операцій з базою даних. Це полегшує розробку, зменшує кількість помилок і прискорює процес розробки;

- більша продуктивність: Spring Data JPA пропонує ряд оптимізацій, які дозволяють покращити продуктивність додатків. Він використовує кешування результатів запитів, оптимізовані запити до бази даних та інші підходи для забезпечення швидкості та ефективності доступу до даних;

- підтримка різних баз даних: Spring Data JPA підтримує різні реляційні бази даних, такі як MySQL, PostgreSQL, Oracle і багато інших. Це дозволяє розробникам працювати з багатьма базами даних, не змінюючи код додатка;

- легка інтеграція з іншими модулями Spring: Spring Data JPA інтегрується з іншими модулями Spring, такими як Spring MVC і Spring Security. Це дозволяє розробникам створювати повноцінні додатки, які використовуються

веб-фреймворком Spring та забезпечують безпеку та інші функціональні можливості.

Spring Data Repository

Spring Data JPA надає три сховища даних, які наведені нижче:

CrudRepository: Пропонує стандартні операції створення, читання, оновлення та видалення. Містить такі методи, як findOne(), findAll(), save(), delete() і т. д.

PagingAndSortingRepository: Розширює CrudRepository і додає методи findAll. Це дозволяє сортувати та отримувати дані у посторінковому режимі.

JpaRepository: Це специфічний для JPA репозиторій, визначений у Spring Data Jpa. Він розширює обидва сховища CrudRepository та PagingAndSortingRepository. Він додає специфічні для JPA методи, такі як flush(), щоб викликати очищення в контексті персистентності.

Spring Web

Spring Web[11] – це модуль фреймворка Spring, який надає підтримку для розробки веб-додатків на мові Java. Він базується на принципах моделі представлення-контролера-репозиторія (MVC)[12] і дозволяє розробникам ефективно створювати веб-додатки, використовуючи прості та повторно використовувані компоненти.

Стартер Spring web використовує Spring MVC, REST і Tomcat як вбудований сервер за замовчуванням. Єдина залежність spring-boot-starter-web транзитивно підтягує всі залежності, пов'язані з веб-розробкою.

Основні принципи роботи Spring Web:

- модель представлення-контролера-репозиторія (MVC): Spring Web використовує архітектурний підхід MVC для розділення логіки додатку на три основні компоненти: модель (Model), яка представляє дані; контролер (Controller), який обробляє запити та взаємодіє з користувачем; та представлення (View), яке відображає дані для користувача. Цей підхід полегшує організацію та розробку веб-додатків;

- інверсія контролю (IoC): Spring Web використовує принцип інверсії керування, що дозволяє розробникам декларативно визначати залежності між компонентами додатку. Це забезпечує більшу гнучкість, підтримує повторне використання коду та полегшує тестування додатків;
- обробка запитів: Spring Web надає потужний механізм обробки запитів, включаючи можливість мапування URL-адрес на методи контролера, обробку параметрів запиту, валідацію даних, обробку форм та інші функції. Це дозволяє розробникам легко створювати ефективні та безпечні веб-додатки.

Spring Web легко інтегрується з іншими модулями фреймворка Spring, такими як Spring Security для забезпечення безпеки, Spring Data для доступу до бази даних та інші, також Spring Web надає зручні інструменти для тестування веб-додатків, включаючи підтримку автоматичного тестування контролерів та взаємодії з HTTP-запитами.

Розширюваність: Spring Web надає гнучкість та розширюваність для розробки веб-додатків завдяки багатому набору інтерфейсів та абстракцій.

MariaDB

MariaDB Server[13] – це система керування базами даних, яка базується на відкритому вихідному коді MySQL та забезпечує високу продуктивність, надійність та розширюваність. Вона розроблена з метою заміни MySQL, зберігаючи при цьому сумісність з його синтаксисом та протоколами.

Ця СКБД пропонує можливості обробки даних як для невеликих, так і для корпоративних завдань. Вона є покращеною версією MySQL[14] та поставляється з численними вбудованими потужними функціями і багатьма додатковими можливостями.

Переваги MariaDB:

- сумісність з MySQL: MariaDB зберігає сумісність з MySQL, спрощуючи перехід між ними;
- висока продуктивність: MariaDB має оптимізації, які забезпечують швидку обробку запитів та високу швидкодію;

- розширені можливості: MariaDB надає додатковий функціонал, такий як географічні типи даних та розширені оптимізації та налаштування бази даних.

Hibernate

Hibernate Framework

Hibernate[15] - це фреймворк Java, який спрощує розробку Java-додатків для взаємодії з базою даних. Це легкий інструмент ORM (Object Relational Mapping - об'єктно-реляційне відображення) з відкритим вихідним кодом. Hibernate реалізує специфікації JPA (Java Persistence API) для збереження даних. Інструмент ORM використовує JDBC API для взаємодії з базою даних.

Об'єктно-реляційне відображення базується на контейнеризації об'єктів та абстракції, яка забезпечує цю можливість. Абстракція дозволяє адресувати об'єкти, отримувати до них доступ і маніпулювати ними без необхідності враховувати, як вони пов'язані з їхніми джерелами даних.

Hibernate – це шар між БД та застосунком. Він може застосовуватись у будь-якому типі застосунків (desktop, web, spring і т. д.). При роботі із ним немає необхідності створювати прямі запити до БД через JDBC. Запити через JDBC виконуються на нижчому рівні. Основна мета Hibernate – працювати з таблицями бази даних як з об’єктами, що дозволяє легше створювати правильний абстрактний рівень для застосунку, застосовуючи принципи ООП. На рисунку 2.1 Persistent Object – це Java об’єкт, який відображає якусь таблицю із бази даних. Окрім цього існує спеціальний Mapping File, який описує зв’язок Java об’єкту з таблицями (також для цього можуть використовуватися анотації), конфігураційний файл, в якому описуються налаштування доступу до бази даних.

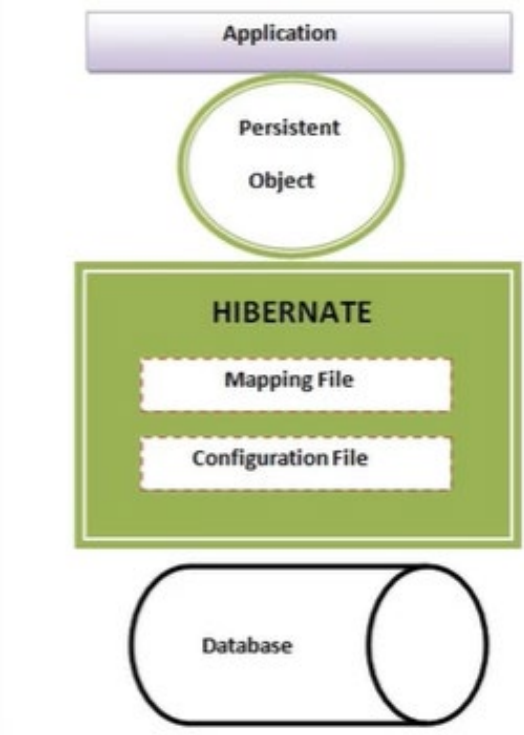

Рис. 2.1. Схема роботи застосунка з Hibernate

JavaScript

JavaScript[16] - безпечна мова програмування, спеціально створена для браузерів, через що не має прямого доступу до низькорівневих операцій з пам'яттю або процесором.

Можливості JavaScript залежать від контексту його використання. Наприклад, Node.js дозволяє взаємодіяти з файлами та здійснювати мережеві запити.

У браузерному середовищі JavaScript[17] може керувати веб-сторінками, реагувати на дії користувача, спілкуватися з веб-сервером та зберігати дані на стороні клієнта.

Проте, в браузерному середовищі JavaScript має деякі обмеження з метою захисту користувача. Наприклад, він не може безпосередньо читати або записувати файли на жорсткому диску, отримувати доступ до операційної системи або надмірно використовувати камеру або мікрофон без згоди користувача. Також він не може безпосередньо звертатися до вкладок або вікон з різних сайтів через політику того ж походження.

Ці обмеження існують для забезпечення безпеки користувача та запобігання можливим зловживанням.

Сьогодні JavaScript може використовуватися не лише у браузерах, але й на серверах та на різних пристроях за допомогою спеціальних програм, відомих як рушії JavaScript.

Кожен браузер має свій власний вбудований рушій, який часто називають "віртуальною машиною JavaScript".

Різні рушії мають свої унікальні кодові назви. Наприклад, V8, що використовується в Chrome, Opera та Edge.

ReactJS

React - це бібліотека JavaScript, яка використовується для створення користувацького інтерфейсу.

При роботі з React[18-19] та Node.js, є можливість використовувати вбудований сервер для запуску веб-додатків під час розробки. У React проектах, зазвичай використовується інструмент Create React App (CRA) для створення нових проектів. При створенні проекту за допомогою CRA, він включає в себе налаштований розробчий сервер, який можна запустити за допомогою команди npm start. Цей сервер автоматично надає доступ до додатку через локальний веб-адрес (зазвичай http://localhost:3000) і оновлює його при зміні вихідного коду.

Він виконується локально на комп'ютері та надає можливість запуску React додатку без необхідності встановлення окремого веб-сервера.

Розробчий сервер, який надається разом з react-scripts, базується на пакеті webpack-dev-server. Webpack є потужним інструментом збірки проектів, а webpack-dev-server є розширенням, що дозволяє запускати локальний сервер для розробки.

Усю структуру веб-сторінки можна представити за допомогою DOM (Document Object Model)[20] - організація елементів html, якими ми можемо маніпулювати, змінювати, видаляти або додавати нові. Для взаємодії з DOM застосовується мова JavaScript.

Однак, коли ми намагаємося маніпулювати елементами HTML за допомогою JavaScript, можемо зіткнутися з погіршенням продуктивності, особливо при зміні великої кількості елементів. Операції над елементами можуть зайняти певний час, що негативно впливає на користувацький досвід. Однак, якщо б ми працювали з об'єктами JavaScript безпосередньо з коду JS, операції відбувалися б швидше.

Для вирішення проблеми продуктивності з'явилася концепція віртуального DOM. Віртуальний DOM представляє легку копію звичайного DOM. І відмінністю React є те, що ця бібліотека працює саме з віртуальним DOM, а не звичайним. Якщо додатку потрібна інформація про стан елементів, відбувається звернення до віртуального DOM. Якщо потрібно змінити елементи веб-сторінки, зміни спочатку вносяться в віртуальний DOM. Потім новий стан віртуального DOM порівнюється з поточним станом. Якщо ці стани відрізняються, React знаходить мінімальну кількість маніпуляцій, необхідних для оновлення реального DOM до нового стану, і виконує їх.

У результаті така схема взаємодії з елементами веб-сторінки працює значно швидше та ефективніше, ніж якби ми працювали з DOM напряму з JavaScript.

Ще однією особливістю React є використання JSX[21]. JSX[22-23] поєднує код JavaScript і XML і надає простий і інтуїтивно зрозумілий спосіб визначення коду візуального інтерфейсу.

React router dom

React Router DOM[24] - це бібліотека для маршрутизації (routing) в React-додатках. Вона надає набір компонентів та методів, які дозволяють організувати навігацію між різними сторінками або відображеннями в React-додатку. На традиційних веб-сайтах браузер запитує документ з веб-сервера, завантажує та оцінює ресурси CSS та JavaScript і рендерить HTML, надісланий з сервера. Коли користувач натискає на посилання, він починає процес заново для нової сторінки.

Маршрутизація на стороні клієнта дозволяє додатку оновлювати URL-адресу після натискання на посилання без повторного запиту іншого документа з сервера. Замість цього ваш додаток може негайно відобразити новий інтерфейс

і зробити запити до даних за допомогою функції fetch, щоб оновити сторінку новою інформацією.

Принцип роботи React Router DOM базується на використанні компонента <BrowserRouter>, який відслідковує поточний URL та співставляє його з заданими шляхами (routes) та відповідними компонентами, які мають бути відображені при збігу URL. Кожен шлях (route) представляється компонентом <Route>, в якому вказується шлях та компонент, що повинен бути показаний при відповідному URL.

React Router DOM також надає можливість передавати параметри (параметри шляху) до компонентів, що дозволяє динамічно змінювати вміст відповідних сторінок в залежності від URL та переданих параметрів. Наприклад, можна мати шлях з параметром /:id, де id може бути будь-яким ідентифікатором, і відповідний компонент може використовувати цей id для відображення конкретного об'єкта з бази даних або іншої джерела даних.

Основна мета використання React Router DOM - це забезпечити реактивну навігацію та маршрутизацію в React-додатках. Вона дозволяє розбити додаток на окремі сторінки або відображення, керувати поточним URL та змінювати вміст сторінок без перезавантаження сторінки.

Redux

Redux є контейнером для управління станом додатка. Redux не прив'язаний безпосередньо до React.js і може також використовуватися з іншими js-бібліотеками та фреймворками.

Ключові моменти Redux:

- сховище (store): зберігає стан програми;
- дії (actions): деякий набір інформації, який виходить від застосунку до сховища і який вказує, що саме потрібно зробити. Для передачі цієї інформації у сховища викликається метод dispatch();
- творці дій (action creators): функції, які створюють дії;
- reducer: функція (або кілька функцій), яка отримує дію і відповідно до цієї дії змінює стан сховища.

Загальну схему взаємодії елементів архітектури Redux можна виразити як на рисунку 2.2.

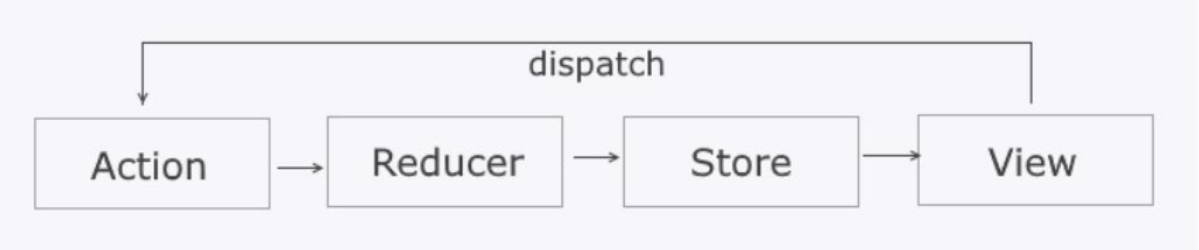

Рис. 2.2. Взаємодія елементів архітектури Redux

З View (тобто з компонентів React) надсилається дія, цю дію отримує функція reducer, яка відповідно до дії оновлює стан сховища. Потім компоненти React застосовують оновлений стан зі сховища.

React redux

React Redux - це інтеграція між бібліотекою Redux та бібліотекою React. Вона надає зручний спосіб зв'язку між сховищем (store) Redux та компонентами React. React Redux додає декларативний підхід до використання Redux у React додатках та надає спеціальні компоненти, які допомагають підключати компоненти до сховища та автоматично оновлювати їх, коли стан змінюється. Він спрощує роботу з Redux у React-орієнтованих додатках та забезпечує більш зрозумілий та зручний спосіб роботи зі станом.

Redux є незалежною бібліотекою для управління станом, тоді як React Redux є пакетом, спеціально створеним для інтеграції Redux з React. Redux надає базові концепції, які дозволяють створювати та оновлювати сховище Redux, тоді як React Redux надає зручний спосіб підключення React компонентів до сховища та автоматичного оновлення їх при зміні стану.

Tailwind CSS

Tailwind CSS[25] - це CSS-фреймворк, який надає широкий набір готових класів для швидкого та простого стилізування веб-елементів. Він пропонує інтерактивний підхід до написання CSS, використовуючи класи, щоб задавати стилі прямо в HTML-коді.

Принцип роботи Tailwind CSS полягає в наданні великого набору класів, які представляють окремі CSS-властивості та значення. Наприклад, замість написання власних CSS-правил для вирівнювання елементів по центру, можна просто додати клас text-center до елемента HTML.

Одна з головних переваг Tailwind CSS - це його гнучкість і простота використання. Завдяки великому набору класів, можна швидко стилізувати елементи без необхідності в написанні власних CSS-правил. Можна також комбінувати класи для створення складних компонентів та макетів. Крім того, Tailwind CSS пропонує можливості адаптивного дизайну, що дозволяє змінювати стилі в залежності від розміру екрану.

Основна перевага Tailwind CSS над простим CSS полягає в ефективності та швидкості розробки. Замість написання великих CSS-файлів і постійного звернення до них, ви можете швидко застосовувати стилі, використовуючи класи безпосередньо в HTML-коді. Це дозволяє зберігати час, спрощує підтримку коду та сприяє швидкому прототипуванню.

Крім того, Tailwind CSS надає можливість налаштовувати тему (theme) та використовувати власні значення. Це дозволяє легко змінювати вигляд вашого додатка шляхом налаштування змінних, таких як кольори, шрифти та відступи.

PostCSS

PostCSS[26] - це інструмент для перетворення вихідного CSS на модифікований CSS за допомогою плагінів JavaScript. Ці плагіни можуть покращити стилі багатьма різними способами, наприклад, виявляти помилки у вашому CSS коді, або використовувати синтаксис CSS наступного покоління і багато іншого. Tailwind CSS є одним з таких плагінів, тому для використання Tailwind потрібно встановити PostCSS.

Webpack

Для створення проекту використовувався Create React App (CRA), тому Webpack вже включений та налаштований внутрішньо. CRA є стартовим шаблоном, який надає попередню конфігурацію для розробки на ReactJS, включаючи Webpack. CRA приховує конфігураційні файли Webpack, такі як

webpack.config.js, для спрощення процесу розробки та забезпечення зручного інтерфейсу для розробників. Весь конфігураційний код Webpack вже налаштований внутрішньо та включений у внутрішні пакети CRA.

Webpack[27] – це збирач модулів. Він аналізує модулі додатка, створює граф залежностей, потім збирає модулі в правильному порядку в один або більше бандл (bundle), на який може посилатися файл "index.html".

Зазвичай при створенні додатка на JavaScript код розділяється на кілька частин (модулів). Потім у файлі "index.html" необхідно вказати посилання на кожен скрипт. Важливо не тільки не тільки підключити всі скрипти, а й розташувати їх у правильному порядку. Якщо завантажити скрипт, що залежить від React, до завантаження самого React, виникне помилка. Webpack вирішує ці проблеми. Збірка модулів є лише одним з аспектів роботи Webpack. Він також виконує завантаження ресурсів: дозволяє імпортувати різні типи ресурсів, такі як CSS файли, зображення, шрифти тощо, автоматично опрацьовує ці ресурси та додає їх до вихідного бандлу або копіює їх у вказану папку.

Webpack може використовувати різні набори правил та надати плагіни для транспіляції коду з сучасних стандартів JavaScript (наприклад, ES6, JSX) у більш сумісний формат, який підтримують різні браузери. Він також може застосовувати різні оптимізації, такі як мінімізація коду, зведення дублікатів, усунення непотрібних частин коду та інші для поліпшення продуктивності додатка.

Розгортання: Webpack дозволяє створити готові до розгортання версії додатка, які можна легко запустити на сервері або відправити на хостинг.

Webpack має також має велику екосистему плагінів, які дозволяють розширити його функціонал.

2.4. Опис структури системи та алгоритмів її функціонування

Розроблена інформаційна система складається із трьох частин: REST-сервер для роботи із базою даних та два вебзастосунки, що реалізують клієнта: для звичайних відвідувачів (головна сторінка сайту) та для адміністраторів.

Схематично структуру та принцип роботи програми можна показати як на рисунку 2.3.

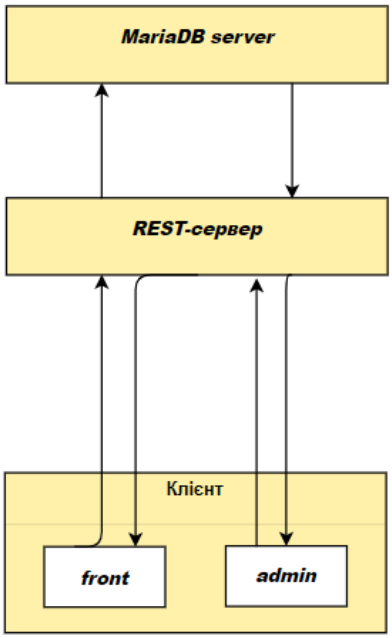

Рис. 2.3. Структура програми

Для роботи із даними в проекті використовується всього дві сутності – articles та tags, що зберігають дані про завантажені публікації та теги, що прикріплюються до них. Було розроблено базу даних, що має наступну структуру:

Таблиця "articles" з колонками:

- id (тип: long, первинний ключ, що автоматично генерується, поле наслідуване від EntityID) – унікальний ідентифікатор статті;
- title (тип: String) – заголовок статті;
- description (тип: String) – короткий опис статті;
- pathToImage (тип: String) – шлях до зображення, пов'язаного зі статтею;
- content (тип: String) – зміст статті;
- created (тип: LocalDate) – дата створення статті;
- locked (тип: boolean) – позначає, чи є стаття закріпленою на слайдері;
- viewed (тип: long) – кількість переглядів статті.

Таблиця "tags" з наступними колонками:

- id (тип: Long, первинний ключ, автоматично генерується, поле наслідуване від EntityID) – унікальний ідентифікатор статті;

- content (тип: String, унікальне поле) – текстове значення тегу.

Проміжна таблиця "article_tag" для зв'язку багато-до-багатьох між таблицями "articles" і "tags" з наступними колонками:

- article_id (зовнішній ключ, посилається на стовпець "id" таблиці "articles");
- tag_id (зовнішній ключ, посилається на стовпець "id" таблиці "tags").

Дану структуру можна представити у вигляді ER діаграми, що наведена на рисунку 2.4.

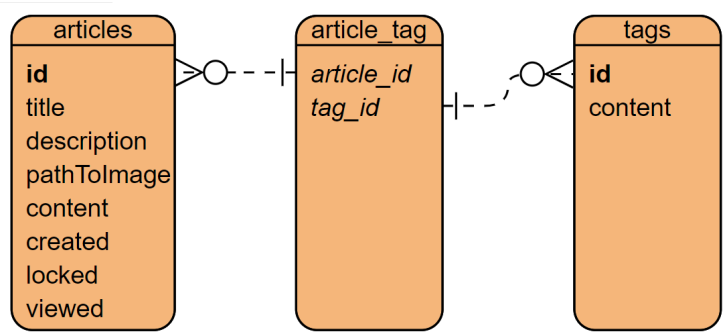

Рис.2.4. ER діаграма бази даних.

Для керування базами даних у системі використовується сервер MariaDB, переглянути структуру створених таблиць у графічному вигляді можна у інструменті HeidiSQL. На рисунках 2.5-2.7 наведено використовувані таблиці та значення їх полів.

Столбцы: Добавить Удалить Вверх Вниз

#	Имя	Тип данных	Длина/Значе...	Беззна...	Разреш...	Zerofill	По умолчанию	Коммент...	Сопоставление
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT		
2	content	VARCHAR	5000	☐	☐	☐	Нет значения по умолчанию		utf8mb4_general_ci
3	created	DATE		☐	☑	☐	NULL		
4	description	VARCHAR	255	☐	☐	☐	Нет значения по умолчанию		utf8mb4_general_ci
5	locked	BIT	1	☐	☐	☐	Нет значения по умолчанию		
6	path_to_image	VARCHAR	255	☐	☐	☐	Нет значения по умолчанию		utf8mb4_general_ci
7	title	VARCHAR	255	☐	☐	☐	Нет значения по умолчанию		utf8mb4_general_ci
8	viewed	BIGINT	20	☐	☐	☐	Нет значения по умолчанию		

Рис. 2.5. Структура таблиці articles, що містить дані про публікацію

Столбцы: Добавить Удалить Вверх Вниз

#	Имя	Тип данных	Длина/Значе...	Беззнаковое	Разрешить N...	Zerofill	По умолчанию	Коммент...	Сопоставление
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT		
2	content	VARCHAR	255	☐	☐	☐	Нет значения по умолчанию		utf8mb4_general_ci

Рис. 2.6. Структура таблиці tags, що містить інформацію про хештег

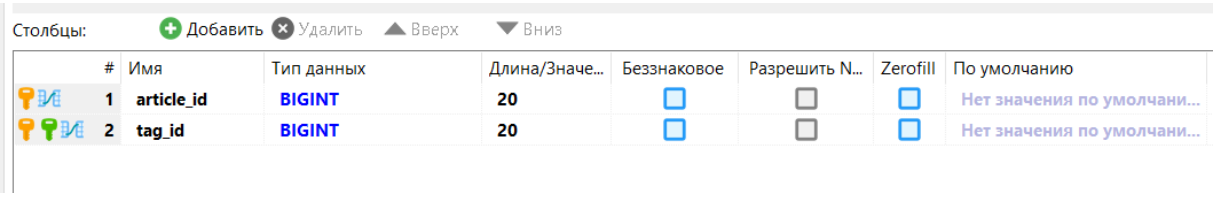

Рис. 2.7. Структура проміжної зв'язуючої таблиці article_tag

Опис програми:

Сервер виконує роль центральної системи, яка надає функціональність і обробляє запити, що надходять від клієнтських додатків.

Розроблений сервер написаний на Java з використанням Spring-фреймворку, має архітектуру REST[28], обробляє HTTP-запити і надсилає відповіді у вигляді даних у форматі JSON. Він використовується для роботи з базою даних і надає API для взаємодії з клієнтськими додатками, написаними з використанням ReactJS.

Використання Spring дозволяє легко створювати API, встановлювати шляхи (endpoints) для обробки різних типів запитів (GET, POST, PUT, DELETE) та взаємодіяти з даними.

Для створення проекту із попереднім автоматичним налаштуванням всіх необхідних для програми залежностей, було використано сервіс Spring Initializr із параметрами, наведеними на рис. 2.8. Цей сервіс генерує дескриптор розгортання системи збірки Maven - pom.xml і головний клас програми.

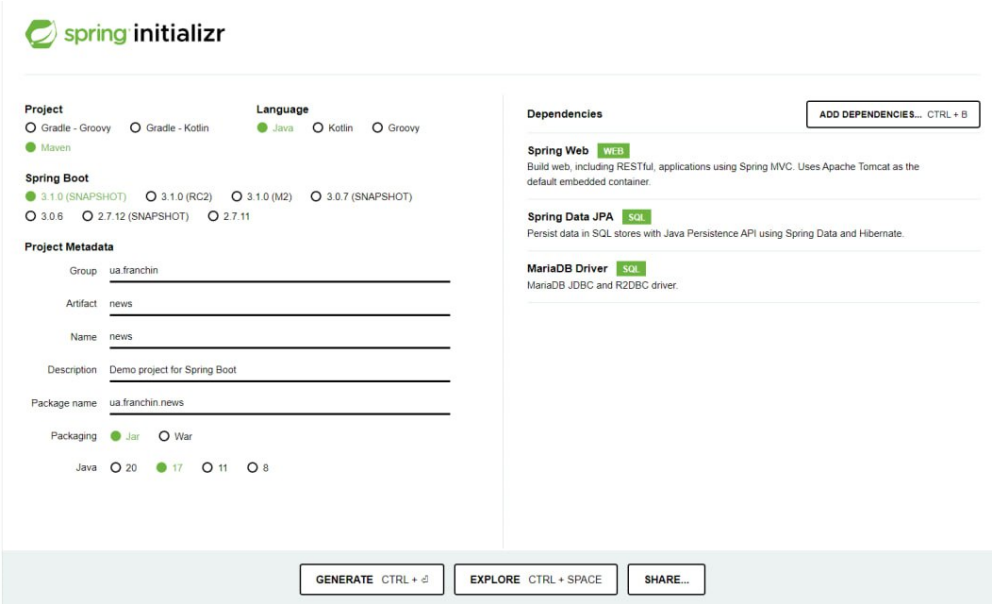

Рис. 2.8. Параметри та залежності, вказані при створенні проекту

Файлова структура готового проекту наведена на рисунку 2.9.

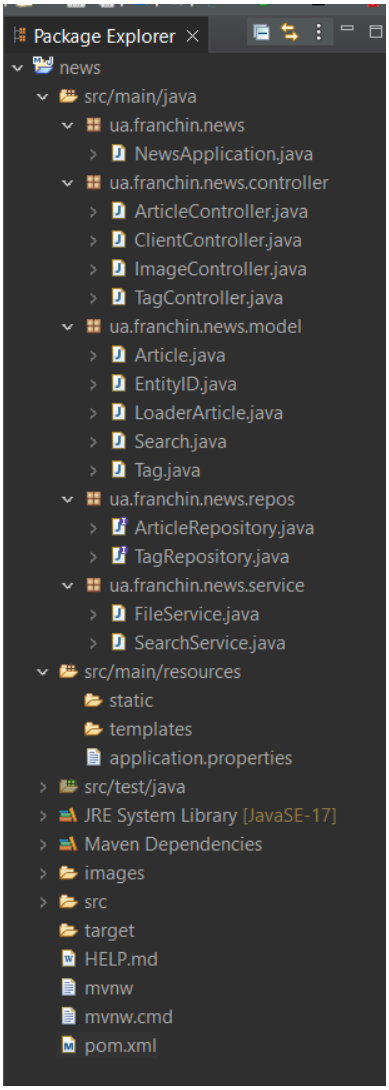

Рис. 2.9. Файлова структура розробленого серверу

Розглянемо детальніше призначення кожного класу.

NewsApplication.java:

Цей клас є точкою входу для серверного додатка. Він містить метод main, який запускає додаток, використовуючи метод SpringApplication.run. Анотація “@SpringBootApplication” вказує, що цей клас є основним класом додатка Spring Boot.

Пакет ua.franchin.news.model представляє модель даних, він містить класи, які визначають сутності і взаємозв'язки між ними в системі.

Класи Article.java і Tag.java представляють відповідно модель статті та тегу в системі. Вони є сутностями JPA (Java Persistence API) і містять анотації JPA, такі як @Entity і @Column, які вказують на те, що клас є сутністю, яка може бути

збережена в базі даних, а анотація @ManyToMany вказує, що існує зв'язок багато-до-багатьох між статтями і тегами. За допомогою анотації @JoinTable, вказуються назви таблиці з'єднання та зовнішніх ключів, які використовуються для зберігання зв'язку.

Приклад використання анотацій JPA у класі наведено на рисунку 2.10.

Рис. 2.10. Приклад використання анотацій JPA для визначення моделі Article

EntityID.java:

Цей клас представляє модель унікального ідентифікатора сутності. Він є абстрактним класом, який використовується для спільного використання коду, пов'язаного з унікальним ідентифікатором, між різними сутностями. Використовуючи анотацію @MappedSuperclass, цей клас вказує, що він є базовим класом для інших сутностей. Це дозволяє спільно використовувати його поля і методи в інших класах-сутностях.

LoaderArticle.java:

Цей клас представляє модель завантаження статті в систему. Він використовується в контролері при обробці запиту на створення нової статті. Дані, що отримуються з фронтенду, передаються у вигляді об'єкту LoaderArticle, і потім використовуються для створення об'єкту Article.

Search.java:

Цей клас представляє модель пошуку статей в системі. Він містить поля, які використовуються для передачі критеріїв пошуку з фронтенду до сервера.

Клас Search має такі основні поля:

keyword: Ключове слово для пошуку вмісту статей за текстом.

tagNames: Список назв тегів для пошуку статей, пов'язаних з цими тегами для пошуку за тегами.

Цей клас використовується в сервісі пошуку для виконання пошуку статей, враховуючи задані критерії.

У пакеті ua.franchin.news.controller знаходяться контролери – компоненти, які відповідають за обробку HTTP-запитів та взаємодію з користувачем через API. Контролери виконують роль прослойки між фронтендом (клієнтською стороною) та бекендом (серверною стороною) додатку. У них використовуються анотації Spring, такі як @RestController[29], @RequestMapping, @GetMapping, @PostMapping, для визначення маршрутів та методів обробки запитів. Наприклад, метод getPost обробляє GET-запит до /article/{pathId} і повертає відповідь, яка містить статтю з бази даних з відповідним ідентифікатором. Анотації в стилі @RequestMapping використовуються для відображення URL-адреси (наприклад /client для цілого класу) або конкретного методу обробника. @RestController - це спеціалізована версія контролера. Вона включає анотації @Controller та @RequestBody (вказує на значення, яке очікується в запиті), і, як наслідок, спрощує реалізацію контролера.

У пакет ua.franchin.news.repository представлені репозиторії – компоненти, які відповідають за доступ до даних у базі даних. Репозиторії забезпечують абстракцію над зберіганням та отриманням об'єктів з бази даних. Для взаємодії із базою даних вони використовують технології Hibernate в поєднанні з JPA. Вони використовуються для виконання різних типів запитів до бази даних, забезпечуючи абстракцію від деталей роботи з базою даних і дозволяючи програмістам працювати з об'єктами та колекціями, а не з реляційними таблицями та SQL-запитами безпосередньо.

Репозиторій ArticleRepository використовується для доступу до даних про статті (Article). Він розширює інтерфейс JpaRepository, що надає базові методи

для роботи з даними, такі як збереження, видалення, оновлення та отримання даних. У цьому репозиторії оголошено кілька власних методів:

- searchArticles – виконує пошук публікацій за тегами;
- searchArticles – виконує пошук публікацій за текстом, який може знаходитись у заголовку, описі або змісті статті;
- findByTagsIn – повертає сторінку публікацій, що містять будь-які з вказаних тегів;
- findByLocked – повертає список публікацій залежно від їх статусу закріплення.

Аналогічно, TagRepository використовується для доступу до даних про теги.

У пакеті ua.franchin.news.service містяться сервіси SearchService та FileService. Сервіси зазвичай виконують різні операції, пов'язані з обробкою даних та взаємодією з репозиторіями та іншими компонентами.

Сервіс SearchService відповідає за виконання пошуку публікацій. Він має наступні методи:

- toPageable – перетворює об'єкт Search на об'єкт Pageable, який використовується для налаштування посторінкового запиту до бази даних;
- extractHashtags – виокремлює хештеги з вмісту пошукового запиту;
- generateAnswer – генерує відповідь у форматі JSON на основі сторінки статей, отриманої з бази даних. Він створює HashMap з вмістом сторінки статей, загальною кількістю сторінок та поточною сторінкою;
- findArticles – цей метод виконує пошук статей на основі пошукового запиту.

Сервіс FileService відповідає за операції з файлами, такі як завантаження, збереження та видалення.

У таблиці 2.1 наведено повний список API, що використовуватиметься клієнтом для взаємодії із сервером.

Табл. 2.1.

Перелік ендпоінтів API у розробленому сервері

GET /article/{pathId}	отримати статтю за її ідентифікатором із збільшенням лічильника переглядів
GET /article/s/{pathId}	отримати статтю за її ідентифікатором без збільшення лічильника переглядів.
GET /article/locked	отримати список заблокованих статей
POST /article/search	пошук статей за певними параметрами у тілі запиту
PUT /client/created	створити нову статтю за даними, переданими у тілі запиту
GET /client/locked/{pathId}	закріпити або відкріпити публікацію за її ідентифікатором
POST /client/update/content/{pathId}	оновити вміст статті за її ідентифікатором
POST /client/update/image/{pathId}	оновити зображення статті за її ідентифікатором. Зображення передається як файл.
POST /client/update/tags/{pathId}	оновити теги статті за її ідентифікатором
DELETE /client/delete/articles/{pathId}	видалити публікацію за її ідентифікатором
GET /image	отримати зображення обкладинки за замовчуванням
GET /image/default	отримати зображення за замовчуванням
GET /image/{imageName}	отримати зображення за його іменем
GET /tag/popular	отримати список популярних тегів

Закінч. табл. 2.1.

GET /tag/all	отримати всі теги
POST /tag/create	створити новий тег за вказаним текстом
DELETE /tag/delete/{pathId}	видалити тег за його ідентифікатором

Клієнтська частина проекту, що включає користувальницький та адміністраторський вебзастосунки, побудована на основі бібліотеки React, яка дозволяє створювати компоненти, керувати станом даних та оновлювати користувацький інтерфейс у відповідь на зміни даних. А також використовує наступні бібліотеки та фреймворки: React router dom, React redux, Redux, Tailwind CSS, Webpack.

Бібліотека React Router використовується для реалізації маршрутизації в додатку. Вона дозволяє визначати шляхи (URL) та пов'язані з ними компоненти, що мають бути відображені при зміні URL.

Redux є бібліотекою для керування станом додатку. Вона забезпечує централізоване зберігання даних та спрощує управління станом між різними компонентами. Redux використовується для збереження та оновлення даних публікацій та збереження даних пошукового запиту між переходами на сторінці.

API-функції використовують метод fetch для виконання HTTP-запитів до сервера і отримання відповідей у форматі JSON. Наприклад, функція loadSearchPage виконує запит POST з використанням методу fetch (рис. 2.11)

```
28
29  export const loadSearchPage = ({ page = 0, size = 6, content = '' }) =>
30      fetch(getArticleURL + '/search', {
31          method: 'POST',
32          headers: { "Content-Type": "application/json" },
33          body: JSON.stringify({ size: size, page: page, content: content })
34      }).then(res => res.status === 200 ? res.json() : defaultArticles);
```

Рис. 2.11. Використання методу fetch для виконання запиту POST

Файлова структура розроблених додатків наведена на рисунку 2.12 (а-б).

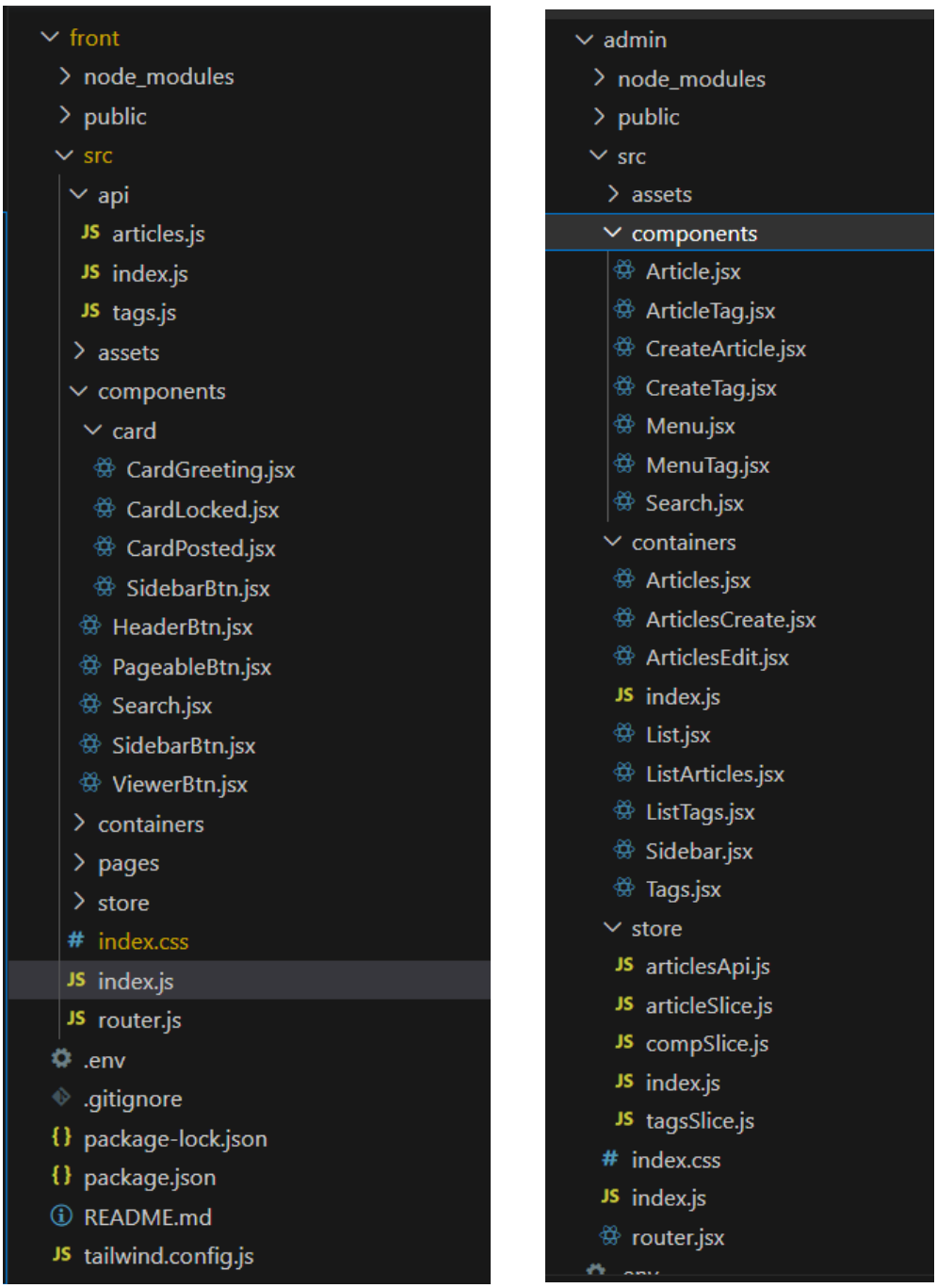

Рис. 2.12. Файлова структура клієнтської частини а) користувальницького вебзастосунку, б) адміністраторського вебзастосунку

React розроблений на концепції багаторазових компонентів. Розробник визначає невеликі компоненти та об'єднуєте їх, щоб сформувати більші компоненти. Багато в чому компоненти поводяться як звичайні функції JavaScript. Вони приймають довільні вхідні дані (так звані "пропси") і повертають React-елементи, які описують те, що ми хочемо побачити на екрані. JSX - це розширення мови JavaScript, яке дозволяє структурувати рендеринг компонентів, використовуючи синтаксис, подібний до HTML. JSX надає можливість писати

HTML-елементи на Javascript і розміщувати їх в DOM, перетворюючи HTML-теги в React-елементи без необхідності використання інших методів, таких як createElement() або appendChild().

2.5. Обґрунтування та організація вхідних та вихідних даних програми

У розробленій програмі дані передаються у форматі JSON. JSON (JavaScript Object Notation) є легким форматом обміну даними, який використовується для структуризації і передачі даних між сервером і клієнтом.

Так як клієнтська частина розділена на два окремі вебзастосунки розглянемо окремо вхідні та вихідні дані для кожного з них.

У клієнтському застосунку для звичайних відвідувачів сайту користувач може виконувати пошук статей за різними критеріями. Для цього використовується об'єкт класу Search, який містить наступні поля:

- text: текстовий критерій пошуку. Статті будуть знаходитись за співпадінням з заголовком, описом або повним текстом статті;
- tags: список тегів, за якими будуть знаходитись статті.

Внутрішній механізм виконання запиту для пошуку для відправки даних використовує метод POST.

Також під час першого завантаження сторінки по замовчанню виконується запит за ідентифікатором сторінки пагінації для отримання відповідного списку публікацій, запит на отримання списку закріплених публікацій, а також пошук всіх тегів для виведення списку із найбільш популярними. Для перегляду публікації надсилається запит за ідентифікатором статі для отримання даних для її подальшого відображення.

Відповідно, вихідними даними будуть результати пошуку. Після виконання пошуку сервер надсилає відповідь зі списком статей, що відповідають критеріям пошуку. Кожна стаття включає інформацію, таку як ідентифікатор, заголовок, опис, зміст і список тегів. Результат пошуку також містить загальну кількість сторінок і поточну сторінку для реалізації пагінації.

У клієнтському застосунку для адміністраторів сайту також окрім пошуку також виконується створення та редагування статей та тегів. При створенні статті користувач надсилає на сервер запит методом PUT з даними статті, такими як заголовок, опис, зміст і список тегів. Ці дані передаються у вигляді об'єкта класу LoaderArticle. При редагуванні статі користувач надсилає відповідну форму із тими ж полями, при цьому у запиті також передається ідентифікатор статі, дані якої потрібно змінити. При видаленні статей виконується запит за методом DELETE та передається лише ідентифікатор статті, яку потрібно видалити. Для створення тегу відправляється запит методом POST та передається текстове значення, яке представляє вміст хештегу, який потрібно створити, а для його видалення відповідно, – ідентифікатор тегу.

Після успішного створення статті сервер надсилає відповідь з інформацією про створену статтю.

Вхідні дані визначаються структурою класів LoaderArticle та Search, а вихідні дані представлені у вигляді структури статей та результату пошуку.

У цій програмі дані передаються у форматі JSON. JSON (JavaScript Object Notation) є легким форматом обміну даними, який використовується для структуризації і передачі даних між сервером і клієнтом.

При створенні статті або виконанні пошуку, дані передаються з клієнта на сервер у вигляді JSON-об'єктів. Наприклад, об'єкт LoaderArticle або Search будуть перетворені в JSON перед відправкою на сервер. Отримані дані з сервера також повертаються у форматі JSON.

Всі ці дані передаються через HTTP-запити до серверної частини програми, і вони обробляються на сервері за допомогою відповідних методів та сервісів для редагування, видалення, створення та видалення даних.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Розробка та тестування проекту виконувались на ноутбуці, що має наступні технічні характеристики:

- операційна система Windows 10;
- процесор Intel Core i5-8300 2.30GHz
- оперативна пам’ять об’ємом 8 ГБ
- відеокарта NVIDIA G-Force GTX 1050
- відеопам’ять 4 ГБ
- жорсткий диск об’ємом 465 ГБ

2.6.2. Використані програмні засоби

Для розробки проекту цієї кваліфікаційної роботи було використано:

- середовище розробки Eclipse IDE для роботи над серверною частиною проекту;

- редактор коду Visual Studio Code (VS Code) для роботи над клієнтською частиною проекту;

- система управління базами даних MariaDB;

- інструмент для управління базами даних HeidiSQL;

- середовище виконання Node.js;

Також було встановлено JDK 19.0.2.

Eclipse IDE є одним з найпопулярніших Java IDE, має модульну архітектуру, що дозволяє легко встановлювати різні плагіни та розширення. Це дає можливість налаштувати робоче середовище під свої потреби та використовувати різні інструменти, бібліотеки та фреймворки. Eclipse IDE також має потужний візуальний редактор коду з підсвічуванням синтаксису, автодоповненням, рефакторингом коду та іншими корисними функціями, надає єдину робочу область, для організації проектів та ресурсі та має вбудовану підтримку популярних систем контролю версій, таких як Git, SVN та CVS.

VS Code – це швидкий і розширюваний редактор коду, призначений для різних мов програмування і технологій. Одною із головних переваг VS Code є велика кількість розширень, які призначені для різних цілей від налаштування теми інтерфейсу до підтримки додаткових мов програмування. VS Code містить розширену вбудовану підтримку для розробки Node.js за допомогою JavaScript і TypeScript, засновану на тих самих базових технологіях, що й Visual Studio. VS Code також включає інструменти для веброзробки, такі як JSX/React, HTML, CSS, SCSS, Less та JSON.

HeidiSQL[30] – це безкоштовний інструмент для керування базами даних з відкритим вихідним кодом. Він надає повну функціональність для керування базами даних, створення та редагування таблиць, виконання запитів і т.д.. HeidiSQL має зручний графічний інтерфейс для редагування та відображення даних в таблицях та надає можливість легко змінювати значення, додавати нові рядки, видаляти та оновлювати дані безпосередньо з інтерфейсу. Він також містить інтегровану довідку для мови SQL, дозволяє підключатися до декількох локальних і/або віддалених серверів баз даних і може використовуватися з параметрами командного рядка. HeidiSQL підтримує наступні бази даних: MariaDB, MySQL, SQL Server, PostgreSQL і SQLite.

Node.js - це однопотокове кросплатформерне середовище виконання з відкритим вихідним кодом і бібліотека, що використовується для запуску вебдодатків, написаних на JavaScript, поза браузером клієнта. Node.js дозволяє розробляти серверні додатки з використанням JavaScript. Він забезпечує зручну модель подій та асинхронність, що дозволяє обробляти багато одночасних з'єднань без блокування потоку виконання. Node.js дозволяє використовувати велику кількість зовнішніх модулів, які полегшують розробку. За допомогою пакетного менеджера npm, можна легко встановлювати та управляти залежностями проекту.

MariaDB — реляційна система керування базами даних із відкритим вихідним кодом, створена як відгалуження (форк) MySQL. MariaDB зберігає велику частину синтаксису та функціональності MySQL та відповідно має

велику сумісність із додатками і бібліотеками, розроблених для MySQL. MariaDB є популярним вибором для розробки вебдодатків, вона надає розробникам широкі можливості для роботи з даними та забезпечує високу продуктивність та розширюваність.

2.6.3. Виклик та завантаження програми

Щоб запустити даний проект на локальному пристрої необхідно встановити та налаштувати сервер субд MariaDB, вказавши необхідні параметри, такі як порт, ім'я користувача та пароль (для даного проекту було встановлено пароль «1945») і запустити сервер. Після налаштування сервера баз даних MariaDB необхідно запустити REST-сервер, який буде взаємодіяти з базою даних і забезпечувати функціональність проекту.

Після запуску REST-сервера потрібно скомпілювати та запустити клієнтську користувальницьку та адміністративну частини проекту. Для цього необхідно мати встановлене середовище Node.js і пакетний менеджер npm. Користувальницький та адміністративний застосунок необхідно запустити на різних портах змінивши конфігурацію сервера. Зазвичай клієнтська частина проекту працює на порту 8080, а адміністративна частина може працювати на іншому, наприклад, 8000. Після цього можна відкрити браузер і перейти за посиланням localhost:8080, щоб отримати доступ до вебсайту для користувачів, та аналогічно за посиланням localhost:8000, щоб перейти до вебсайту для адміністрації.

2.6.4. Опис інтерфейсу користувача

Інтерфейс вебзастосунків для користувачів та адміністраторів суттєво відрізняється, однак, обидва вебзастосунки надають зручний та адаптивний дизайн для зручного перегляду на екранах із різним розширенням. Інтерфейс головної сторінки сайту наведено на рисунку 2.13, а її вигляд на екранах із меншими розширеннями – на рисунку 2.14 (а-б)

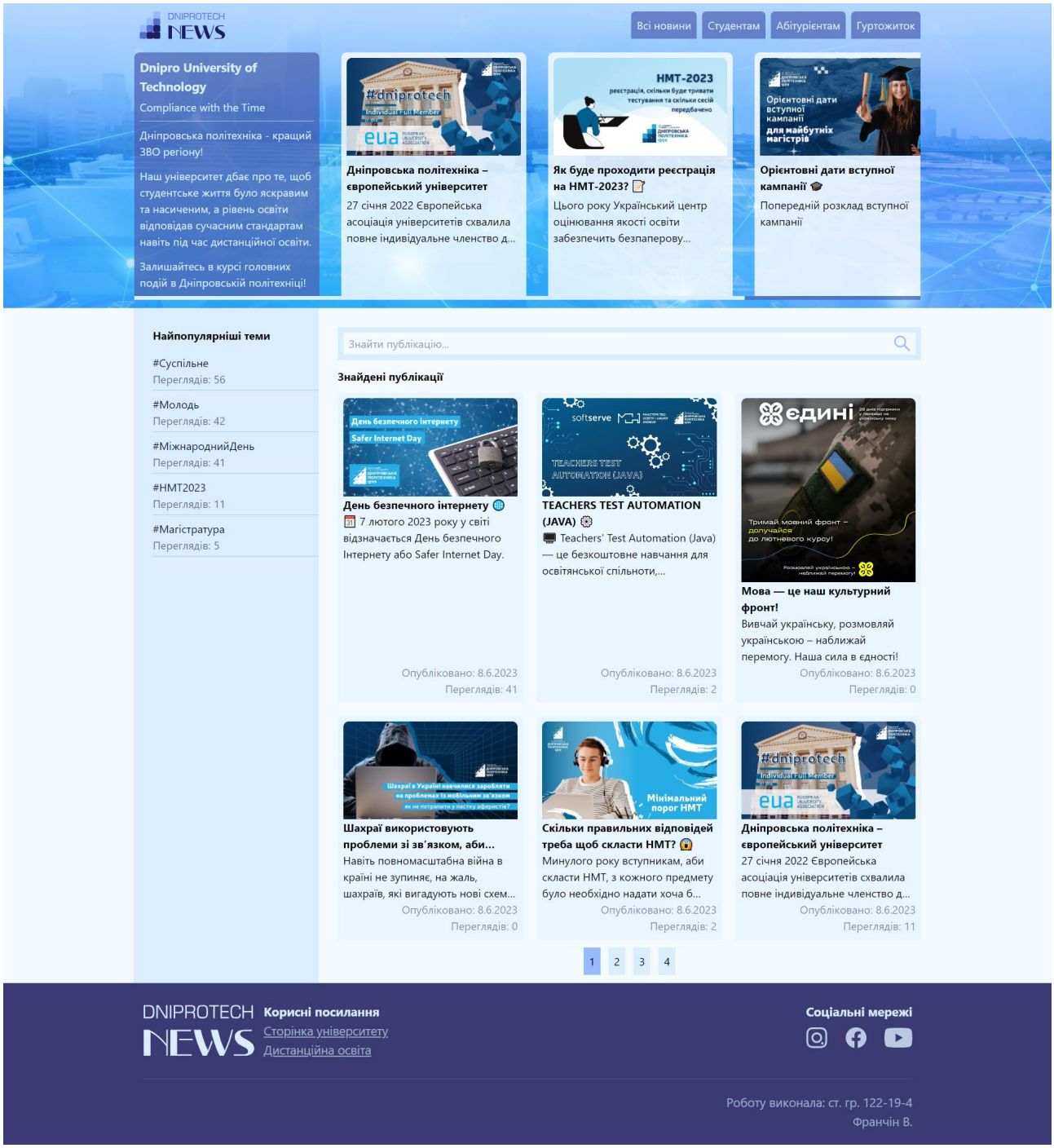

Рис. 2.13. Головна сторінка сайту новин

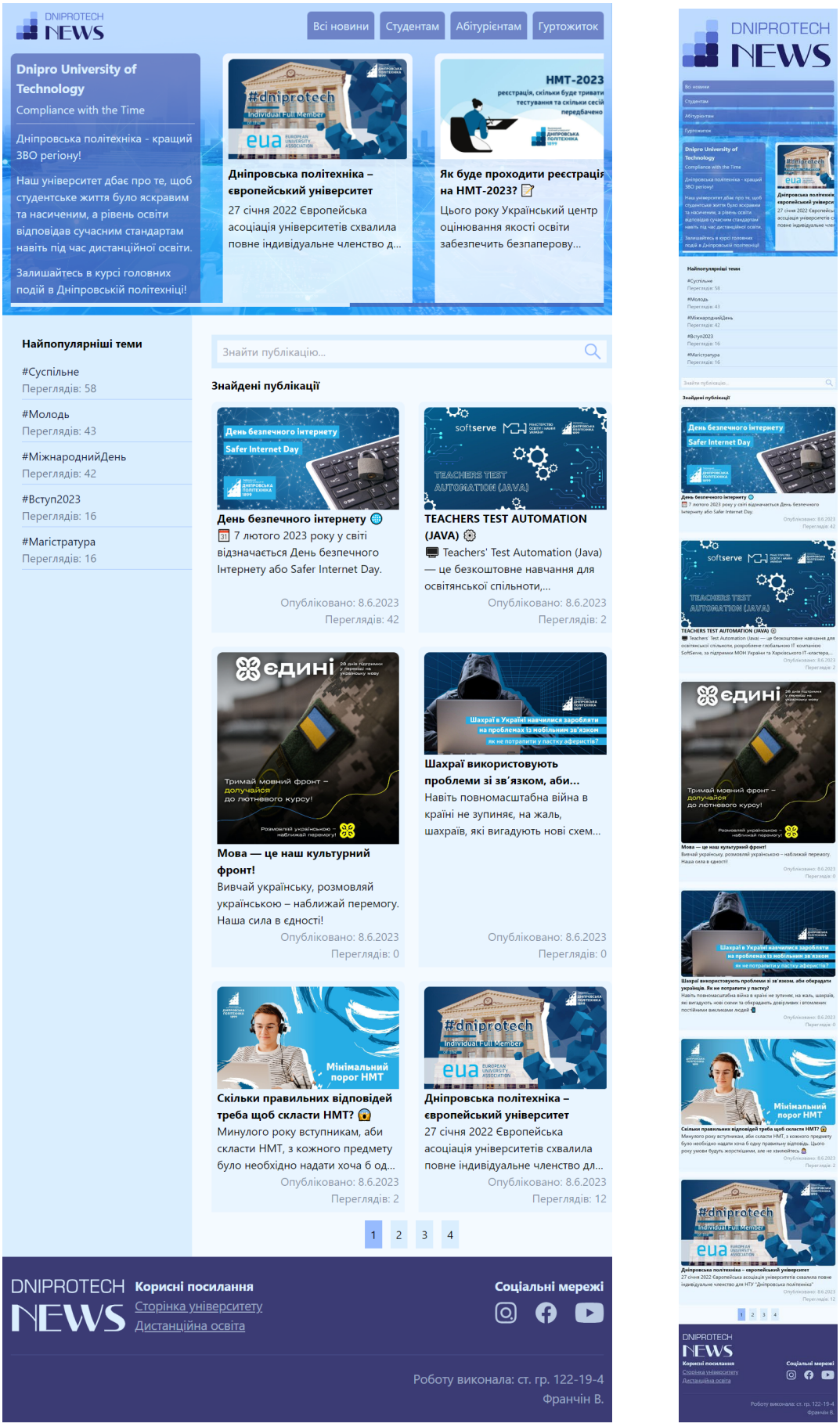

Рис. 2.14. Вигляд головної сторінки а) для планшетної версії, б) для мобільної версії

Розглянемо детальніше основні компоненти даної сторінки: шапка сайту, панель хештегів, список публікацій та футер.

В шапці сайту (рис.2.16) розміщено логотип, який також являється кнопкою повернення на головну сторінку, меню переключення між основними розділами (рис.2.15) та слайдер із закріпленими публікаціями, перший елемент слайдеру є декоративним елементом із привітальним текстом. Переключення по темам виконується за системою пошуку по хештегам: при натисканні на одну із кнопок виконується пошук за відповідним хештегом. Відповідно, розміщення публікації за певною темою керується шляхом прикріплення до неї відповідного з тегів: «Студентам», «Абітурієнтам», «Гуртожиток». Кнопка «Всі новини» скидає попередній фільтр пошуку та завантажує всі наявні публікації.

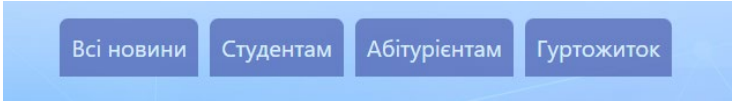

Рис. 2.15. Меню для знаходження новин за основними закріпленими темами

Рис. 2.16. Шапка сайту із, логотипом, меню та слайдером закріплених публікацій

В основній частині сайту з лівої сторони знаходиться панель хештегів (рис. 2.17), де виводиться список із п'яти найпопулярніших тегів та сумарна кількість переглядів публікацій за ним. При натисканні на будь-який з них виконується пошук за даним тегом.

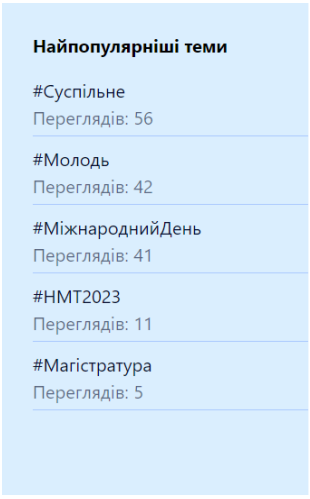

Рис. 2.17. Панель популярних тегів

Публікації виводяться у вигляді карточок (по шість елементів на сторінку пагінації), на яких виводиться обкладинка, назва, короткий опис, дата публікації та кількість переглядів публікації. Над списком знаходиться панель пошуку виводиться текст із інформацією про запит, за яким було виконано пошук. Вигляд області для пошуку публікацій можна побачити на рисунку 2.18.

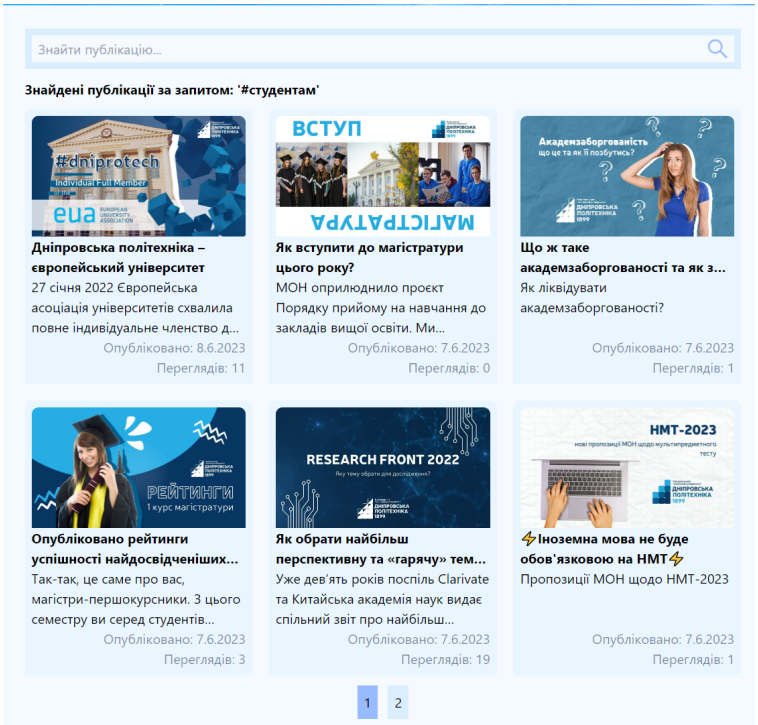

Рис. 2.18. Панель пошуку та список знайдених публікацій

Пошук можна виконувати як за звичайним текстом (рис. 2.19), так і за тегами або переліком тегів, при чому, якщо в запиті знаходяться і теги і звичайний текст, пошук буде виконано саме за тегами (рис 2.20).

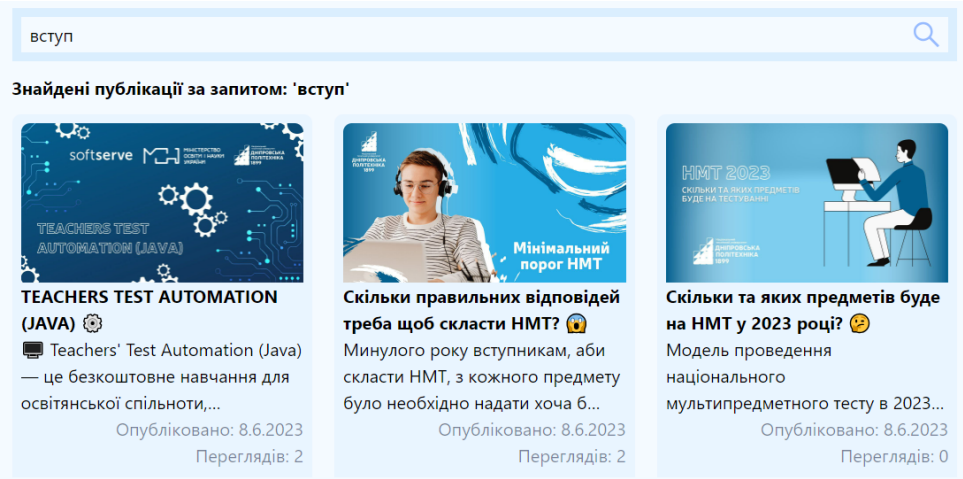

Рис. 2.19. Пошук публікацій за словами в тексті

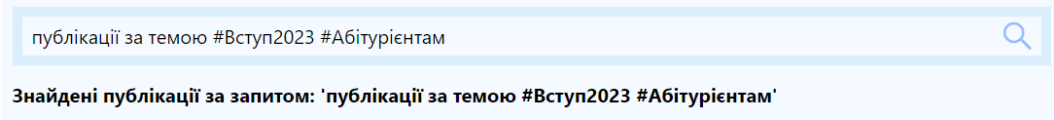

Рис. 2.20. Пошук публікацій за переліком тегів

Футер сайту виконано у мінімалістичному стилі. В ньому знаходяться посилання на інші сайти університету та сторінки у соціальних мережах. Вигляд футеру наведено на рисунку 2.21.

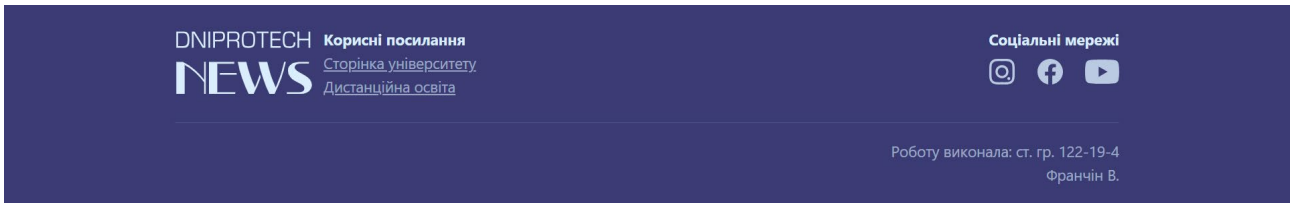

Рис. 2.21. Вигляд футеру сайту

При натисканні на публікацію для її перегляду, відкривається сторінка для перегляду публікації. Загальний вигляд сторінки для десктопної та мобільної версії наведено на рисунку 2.22 (а-б). На даній сторінці залишаються незмінними конмпоненти шапки та футеру. Виводиться основний контент публікації: заголовок, опис, обкладинка та основний текст. Якщо у тексті є посилання, вони виділяються відповідним чином та дозволяють швидко перейти по ним. Теги, закріплені за публікацією прикріпляються у вигляді додаткового рядку в основному тексті, при натисканні на будь-який прикріплений тег, виконується пошук за даним тегом та перехід на головну сторінку для відображення результатів.

Рис. 2.22. Сторінка перегляду публікації а) на десктопній версії б) на мобільній

Інтерфейс головної сторінки вебсайту для адміністрування має наступну структуру: форма введення даних для створення публікації, у кожному полі форми внизу відображається кількість доступних знаків, панель із можливістю переключення між списком доданих раніше публікацій або тегів та пошук за ними. У списку публікацій вгорі знаходяться закріплені на слайдері, а нижче – всі інші. Загальний вигляд сторінки наведено на рисунку 2.23 (а-б).

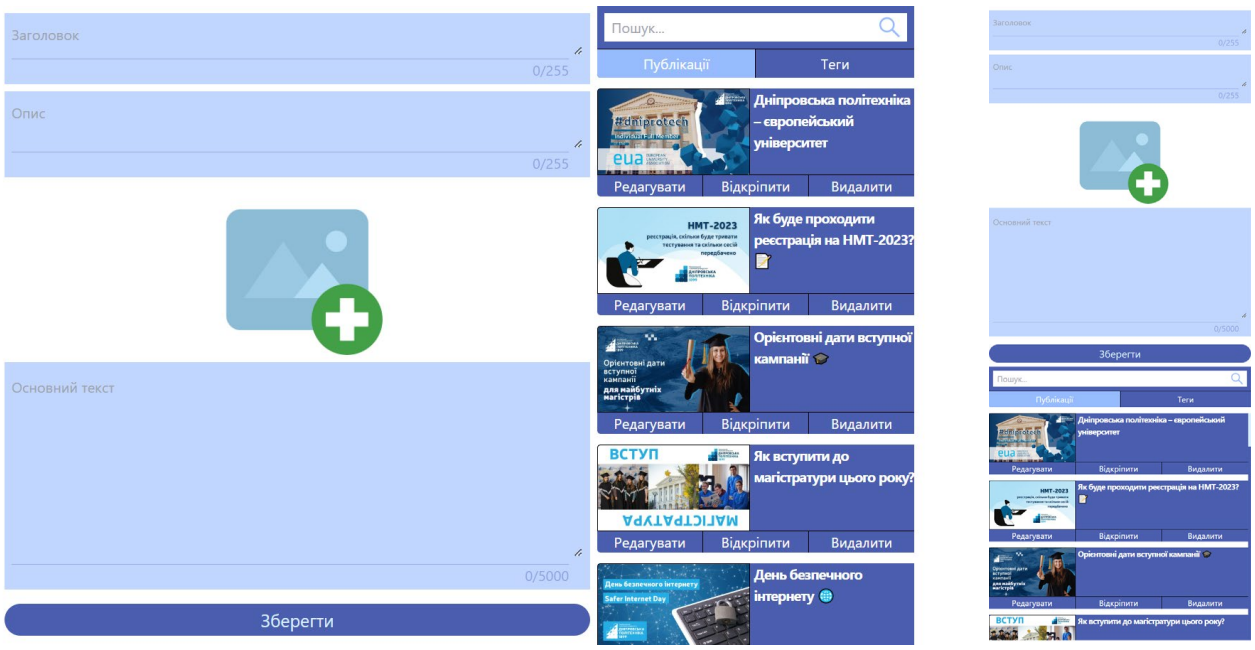

Рис. 2.23. Вигляд сторінки адміністратора а) на десктопній версії, б) на мобільній версії

Для створення публікації обов'язковим є поле назви, а інші опціональними, при спробі зберегти публікацію без назви, буде виведено попередження (рис. 2.24), а при створені публікації без обраного зображення обкладинки їй буде присвоєно зображення за замовченням:

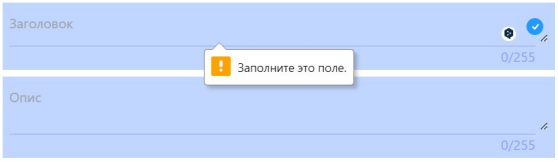

Рис. 2.24. Попередження при спробі зберегти публікацію без назви

У кожної піблікації наявні три кнопки: редагувати, закріпити/відкріпити та видалити. При натисканні кнопки закріплення, публікація закріплюється на слайдері головної сторінки сайту та переноситься на початок списку публікацій на сторінці адміністратора. При натисканні кнопки редагування у форму завантажується інформація про обрану публікацію, а під кнопкою пошуку у боковій панелі з'являється кнопка для переходу назад у форму створення нової публікації. Вигляд сторінки при редагуванні публікації наведено на рисунку 2.25.

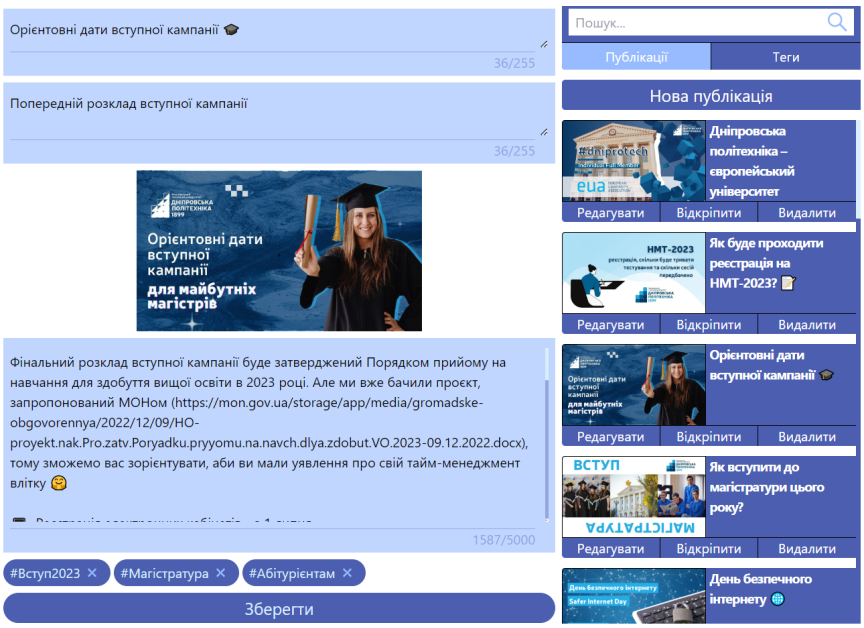

Рис. 2.25. Редагування публікації

Для додавання тегів до публікації необхідно обрати розділ «теги» на боковій панелі. Для зручності всі теги відсортовано у алфавітному порядку. При натисканні на тег він зникає зі списку всіх тегів і відображається у списку під публікацією. Для того, щоб його прибрати, потрібно просто натиснути на хрестик. На боковій панелі тегів також для кожного тега наявна кнопка видалення, а також над списком тегів є кнопка для створення нового хештегу. Процес керування тегами можна побачити на рисунку 2.26.

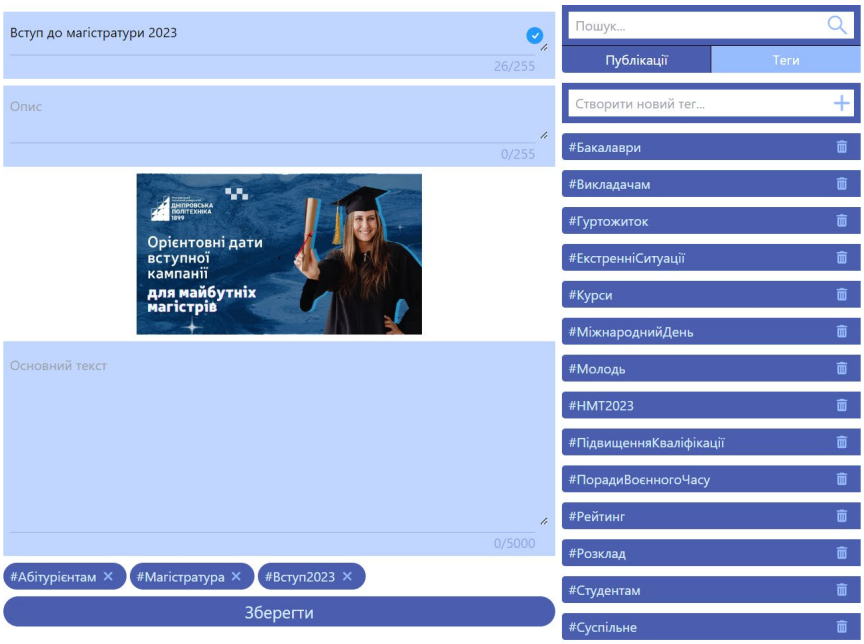

Рис. 2.26. Панель тегів, додавання, створення та видалення тегів

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки інформаційної системи

Початкові дані:

1. передбачуване число операторів програми – 1950;
2. коефіцієнт складності програми – 1,3;
3. коефіцієнт корекції програми в ході її розробки – 0,5;
4. годинна заробітна плата програміста – 123,6 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 0,59 грн/год.

Згідно із даними у дослідженні «Зарплати українських розробників – зима 2023», розміщеному на онлайн ресурсі для української спільноти розробників DOU[31] середня заробітна плата full-stack розробників, які працюють із мовою програмування JavaScript та використовують фреймворк ReactJS, за півроку становить 3500$. Курс валют станом на 10.06.2023, відповідно до даних на сайті Національного банку України[32], становить 36,5686 грн до 1 USD. Відповідно, зарплата розробника у гривнях становить:

36,5686 *3500 = 127 990.1 грн за півроку, або

127 990.1/6 = 2 133,18 грн за місяць

Спираючись на дані у дослідженні на електронному ресурсі, присвяченому веденню бухгалтерського обліку[33], при 40-годинному робочому тижні середня кількість робочих годин у 2023-му році становить 171 година на місяць. Виходячи з цього визначимо заробітну плату розробника за годину:

2 133,18/171 = 123,6 (грн/год)

Згідно із Постановою КМУ №1206 від 28.10.2022р.[34] тарифи на електроенергію для побутових споживачів до 31 березня 2023 року становлять 1,44 грн/кВт·год за весь спожитий обсяг до 250 кВт·год на місяць та по 1,68 грн/кВт·год при споживанні понад 250 кВт·год. Ноутбук, на якому велась розробка, споживає 65 Вт/год[5], або 0,065 кВт/год. За робочий місяць (171 год) ноутбук споживає приблизно 1,115кВт. Вартість для 1 години роботи ноутбука відповідно становить 1,44 * 0,065 = 0,0936 грн/год. Вартість інтернету, використаного при розробці, становить 350 грн/місяць за умовами тарифу Vodafone Red Unlim Plus. Для місяця, в якому 30 днів, вартість інтернету за годину становить 350/720 = 0,49грн/год.

Загальна вартість машино-годин становить:

0,0936 + 0,49 = 0,59 (грн/год).

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_\alpha + t_n + t_{отл} + t_\partial, \text{ людино-годин} \qquad (3.1.1.)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_α – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

t_∂ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів:

$$Q = \boldsymbol{q} \cdot \boldsymbol{C} \cdot (\mathbf{1} + \boldsymbol{p}) \qquad (3.1.2.)$$

де q - передбачуване число операторів (1950);

С - коефіцієнт складності програми (1,3);

p - коефіцієнт корекції програми в ході її розробки (0,5).

Умовне число операторів, розраховане за формулою 3.1.1.

$$Q = 1950 \cdot 1{,}3 \cdot (1 + 0{,}5) = 3\,802.5$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$\boldsymbol{t_u} = \frac{Q \cdot \boldsymbol{B}}{(\mathbf{75}...\mathbf{85}) \cdot \boldsymbol{k}}, \text{ людино-годин} \qquad (3.1.3.)$$

де *Q* – умовне число операторів (3 802.5),

B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2),

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (1,2).

Розрахунок витрат праці на вивчення опису задачі за формулою 3.1.3:

$$\frac{3\,802{,}5 \cdot 1{,}2}{75 \cdot 1.5} = 50.7, \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$\boldsymbol{t_\alpha} = \frac{Q}{(\mathbf{20}...\mathbf{25}) \cdot \boldsymbol{k}}, \text{ людино-годин} \qquad (3.1.4.)$$

де Q – умовне число операторів інформаційної системи (3 802.5);
k – коефіцієнт кваліфікації програміста (1,2).

Розрахуємо витрати праці на розробку алгоритму рішення задачі за формулою (3.1.4):

$$\boldsymbol{t_\alpha} = \frac{3\,802{,}5}{(23) \cdot 1{,}2} = 137.77, \text{ людино-годин}$$

Витрати на складання програми по готовій блок-схемі, розраховуються за формулою:

$$\boldsymbol{t_n} = \frac{Q}{(\mathbf{20}...\mathbf{25}) \cdot \boldsymbol{k}}, \text{ людино-годин} \qquad (3.1.5.)$$

де Q – умовне число операторів інформаційної системи (3 802.5);
k – коефіцієнт кваліфікації програміста (1,2).

Розрахунок за формулою (3.1.5):

$$t_n = \frac{3\,802{,}5}{(23) \cdot 1{,}2} = 137.77 \text{ людино-годин}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$\boldsymbol{t}_{\text{отл}} = \frac{\text{Q}}{(\mathbf{4}...\mathbf{5})\cdot \boldsymbol{k}}, \text{ людино-годин} \quad (3.1.6.)$$

Розрахунок за формулою 3.1.5:

$$\boldsymbol{t}_{\text{отл}} = \frac{\mathbf{2\ 802{,}5}}{(\mathbf{4})\cdot \mathbf{1{,}2}} = \mathbf{792.19}, \text{ людино-годин}$$

- за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = \text{Q} \cdot 1{,}5 \cdot t_{\text{отл}}, \text{ людино-годин} \quad (3.1.7.)$$

Розрахунок витрат праці на налагодження програми за формулою 3.1.7:

$$t_{\text{отл}}^{k} = 1{,}5 \cdot 792.19 = 1188{,}28, \text{ людино-годин}$$

Витрати праці на підготовку документації, розраховуються за формулою:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \text{ людино-годин} \quad (3.1.8.)$$

де $t_{\text{др}}$ – трудомісткість підготовки матеріалів і рукопису.

$$t_{\text{др}} = \frac{\text{Q}}{(15...20)\cdot k}, \text{ людино-годин} \quad (3.1.9.)$$

$t_{\text{до}}$ - трудомісткість редагування, печатки й оформлення документації.

$$t_{\text{др}} = 0{,}75 \cdot t_{\text{др}}, \text{ людино-годин} \quad (3.1.10.)$$

Розрахунок трудомісткості підготовки матеріалів і рукопису за формулою (3.1.9):

$$t_{\text{др}} = \frac{3\ 802{,}5}{20 \cdot 1.2} = 158.44, \text{ людино-годин}$$

Розрахунок трудомісткості редагування, печатки й оформлення документації, за формулою (3.10):

$$t_{\text{др}} = 0{,}75 \cdot 158.48 = 118.83, \text{ людино-годин} \quad (3.1.10.)$$

Розрахунок праці на підготовку документації, за формулою (3.1.8):

$$158.44 + 118.83 = 277.27, \text{ людино-годин} \quad (3.1.8.)$$

Розрахунок трудомісткості розробки програмного забезпечення за формулою (3.1):

t = 50 + 50.7 + 137.77 + 137.77 + 792.19 + 277.27 = 1445.7, людино-годин

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу $З_{Мч}$, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{Мч}, \text{ грн} \quad (3.2.1.)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн} \quad (3.2.2.)$$

де t – загальна трудомісткість, людино-годин (1445,7),

Спр - середня годинна заробітна плата програміста, грн/годину (123,6).

Розрахунок за формулою 3.2.2:

$$З_{ЗП} = 1445,7 \cdot 123,6 = 178688.1, \text{ грн}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{Мч} = \boldsymbol{t}_{отл} \cdot C_{Мч}, \text{ грн} \quad (3.2.3.)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год. (792.19),

$С_{мч}$ - вартість машино-години ЕОМ, грн/год. (0,59).

Розрахунок за формулою 3.2.3:

$$З_{Мч} = 792.1875 \cdot 0,59 = 467.39, \text{ грн}$$

Розрахунок витрат на створення програмного забезпечення за формулою (3.2.1):

$$К_{ПО} = 178688.10016 + 467.39 = 179\,155,49, \text{ грн}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$Т = \frac{t}{B_k \cdot F_p}, \ міс \quad (3.2.4.)$$

де t – загальна трудомісткість, людино-годин (1445.7),

В*k* – число виконавців (1),

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні Fр = 171 година).

Розрахунок очікуваного періоду створення ПЗ за формулою 3.2.4:

$$T = \frac{1445.6966}{1 \cdot 171} = 8{,}45,\ міс$$

Висновки: згідно із розрахунками на розробку сайту для розміщення новин університету буде витрачено приблизно 1445.7 людино-годин, враховуючи всі складові процесу розробки, такі як постановка задачі, дослідження алгоритму її рішення, розробка блок-схеми, програмування, налагодження та підготовка документації. Відповідно, ймовірна очікувана тривалість розробки складатиме 8.45 місяців при стандартному 40-годинному робочому тижні і 171-годинному робочому місяці. Очікувані витрати на розробку даного сайту складатимуть 179 155 грн. Ця вартість включає витрати на заробітну плату виконавців програми та витрати машинного часу, необхідного на налагодження програми на ЕОМ.

ВИСНОВКИ

Метою даної кваліфікаційної роботи була розробка веб орієнтованої ІС, що виступатиме інформаційним порталом для університету. Додаток дозволяє користувачам знаходити та переглядати публікації, розміщені на сайті та виконувати пошук по різним темам. Адміністратори, зі своєї сторони можуть створювати, редагувати та видаляти публікації, керувати їх класифікацією за тематикою за допомогою системи хештегів та закріплювати певні публікації, щоб звернути на них особливу увагу відвідувачів.

Розроблена система складається із трьох частин, завдяки винесенню функціоналу адміністратора у окремий додаток, функції користувача та адміністратора чітко розмежовуються і користувач ніяким чином не може вплинути на дані, що зберігаються у базі даних.

Серверна частина додатку була написана на мові програмування Java. Для її розробки було використано стек технологій Spring, що включає Spring boot, Spring Data JPA, Spring Web, а також Hibernate. Для реалізації бази даних, в якій зберігається інформація про публікації та прикріплені до них теги, було використано MariaDB Server. Розроблений застосунок відповідає принципам REST архітектури та реалізовує REST API для взаємодії клієнта із сервером. Обмін даними між клієнтом та сервером виконується за допомогою протоколу HTTP у форматі JSON.

Клієнтська частина складається із двох окремих застосунків, що реалізовані як SPA: один – головна сторінка сайту, на яку заходять користувачі для перегляду публікацій, а друга – сторінка з якої адміністратори можуть займатися редагуванням контенту та слідкувати за наповненням сайту. Клієнтська сторона була реалізована на мові JavaScript із використанням технологій ReactJS, React Router DOM, React Redux та фреймворку Tailwind CSS, що спростив роботу зі стилями та створенням адаптивного дизайну для застосунків.

У економічному розділі кваліфікаційної роботи було виконано розрахунок часу розробки та оціночної вартості розробленого програмного забезпечення. Згідно із результатами розрахунків час розробки становить 1445.7 годин, або 8 місяців та два тижні. При цьому вартість розробки складатиме 179 155 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Itai Himelboim, Steve McCreery, University of Georgia, USA. New technology, old practices: Examining news websites from a professional perspective. The International Journal of Research into New Media Technologies. 2012. P. 431 – 438. – Режим доступу: https://www.researchgate.net/publication/248391118_New_technology_old_practices_Examining_news_websites_from_a_professional_perspective
2. 80 Best University Blogs and Websites 2023 [Електронний ресурс]. URL: https://blog.feedspot.com/university_blogs/
3. Olga Olshevska, V. Solovei, Y. Bortsova. The difference between developing single page application and traditional web application based on mechatronics robot laboratory onaft application [Електронний ресурс]. – 2018. – Режим доступу: https://www.adcisolutions.com/knowledge/whats-difference-between-single-page-application-and-multi-page-application.
4. Paweł Skólski. Single-page application vs multiple-page application [Електронний ресурс]. – 2016. – Режим доступу: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58
5. Проект дизайну вебзастосунку для користувачів [Електронний ресурс]. – Режим доступу: https://www.figma.com/file/lER4NJ0mK3E7cB5e89OAco/Design-(NTU-DP-News-blog)?type=design&node-id=61%3A304&t=Y7S8dItq2IRxnqb5-1
6. Representational State Transfer (REST) [Електронний ресурс]. – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
7. Natalie Hays. What is a RESTful API, how it works, advantages, and examples [Електронний ресурс]. – 2021. – Режим доступу: https://www.mailgun.com/blog/it-and-engineering/restful-api/

8. Introducing JSON [Електронний ресурс]. – Режим доступу: https://www.json.org/json-en.html
9. Spring Boot Tutorial [Електронний ресурс]. – Режим доступу: https://www.javatpoint.com/spring-boot-tutorial
10. Spring Data JPA documentation [Електронний ресурс]. – Режим доступу: https://spring.io/projects/spring-data-jpa
11. Spring Boot Starter Web [Електронний ресурс]. – Режим доступу: https://www.javatpoint.com/spring-boot-starter-web
12. Web MVC framework [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html
13. MariaDB tutorials [Електронний ресурс]. – Режим доступу: https://www.mariadbtutorial.com/mariadb-basics/mariadb-select/
14. Richard Petterson. MariaDB vs MySQL – Difference Between Them. – 2023. – [Електронний ресурс]. – Режим доступу: https://www.guru99.com/mariadb-vs-mysql.html#:~:text=Key%20Difference%20between%20MariaDB%20and%20MySQL&text=MariaDB%20is%20Open%20Source%2C%20whereas,MariaDB%20is%20faster%20than%20MySQL.
15. Hibernate [Електронний ресурс]. – 2021. – Режим доступу: https://www.theserverside.com/definition/Hibernate
16. Сучасний підручник з JavaScript [Електронний ресурс]. – Режим доступу: https://uk.javascript.info/
17. Інтерактивні WEB-документи Розробка динамічних елементів Web-сайтів з використанням JavaScript-сценаріїв. [Електронний ресурс]. – Режим доступу: https://moodle.znu.edu.ua/pluginfile.php/716094/mod_resource/content/1/JavaSc.pdf
18. Документація React [Електронний ресурс]. – Режим доступу: https://react.dev/learn

19. Samer Buna. All the fundamental React.js concepts, jammed into this one article. – 2017. – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
20. Document Object Model (DOM) [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
21. React JSX [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/react/react_jsx.asp
22. Stephen Arancio. What is JSX. – 2021. – [Електронний ресурс]. – Режим доступу: https://medium.com/@sjarancio/what-is-jsx-e3dda0af3490
23. Writing Markup with JSX [Електронний ресурс]. – Режим доступу: https://react.dev/learn/writing-markup-with-jsx
24. React Router documentation [Електронний ресурс]. – Режим доступу: https://reactrouter.com/en/main/start/concepts
25. Документація Taiwind CSS. [Електронний ресурс]. – Режим доступу: https://tailwindcss.com/docs/installation/using-postcss
26. Ivad Yves. Setup Tailwind CSS in a React project configured from scratch with Webpack | a step-by-step guide [Електронний ресурс]. – Режим доступу: https://dev.to/ivadyhabimana/setup-tailwind-css-in-a-react-project-configured-from-scratch-a-step-by-step-guide-2jc8
27. Webpack documentation [Електронний ресурс]. – Режим доступу: https://webpack.js.org/concepts/
28. Building REST services with Spring [Електронний ресурс]. – Режим доступу: https://spring.io/guides/tutorials/rest/
29. The Spring @Controller and @RestController annotations. – 2023. – [Електронний ресурс]. – Режим доступу: https://www.baeldung.com/spring-controller-vs-restcontroller
30. HeidiSQL features. [Електронний ресурс]. – Режим доступу: https://www.heidisql.com/

31. Ірина Іпполітова, Ігор Яновський. Зарплати українських розробників — зима 2023 [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/salary-report-devs-winter-2023/
32. Курс валют НБУ [Електронний ресурс]. – Режим доступу: https://minfin.com.ua/ua/currency/nbu/
33. Норма тривалості робочого часу – 2023 [Електронний ресурс]. – Режим доступу: https://www.buhoblik.org.ua/kadry-zarplata/vremya/4282-norma-trivalosti-robochogo-chasu-2023.html
34. Тарифи на електроенергію [Електронний ресурс]. – Режим доступу: https://yasno.com.ua/news/yasno_news/Electricity_tariffs_for_the_population_remain_unchanged

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

//Код клієнтського застосунку (головний сайт)

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { RouterProvider } from 'react-router-dom';
import './index.css';
import router from './router';
import { Provider } from 'react-redux';
import store from './store';

ReactDOM.createRoot(document.getElementById('root')).render(
<Provider store={store} children={<RouterProvider router={router} />} />);
```

router.js

```
import { Route, createBrowserRouter, createRoutesFromElements } from "react-router-dom";
import { loadArticleLocked } from "./api/articles";
import { loadTagPopular } from "./api/tags";
import Common from "./pages/Common";
import Main from "./pages/Main";
import Viewer, { loaderViewer } from "./pages/Viewer";
import Articles, { loaderArticles } from "./containers/Articles";

export default createBrowserRouter(createRoutesFromElements(
<Route path="/" element={<Common />} loader={loadArticleLocked}>
        <Route path="/" element={<Main />} loader={loadTagPopular}>
                <Route index element={<Articles />} loader={loaderArticles} />
        </Route>
        <Route path="/view/:page" element={<Viewer />} loader={loaderViewer} />
</Route>
));
```

index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

* {
margin: 0;
padding: 0;
@apply box-border;
}

html, body, #root {
height: 100%;
@apply bg-white;
@apply scroll-hidden;
}

#root {
@apply flex flex-col
}

.scroll-hidden::-webkit-scrollbar {
width: 0;
  }

.custom-scrollbar {
```

```
@apply overflow-x-auto;
}

.custom-scrollbar::-webkit-scrollbar {
height: 6px;
}
.custom-scrollbar::-webkit-scrollbar-track {
@apply bg-info/70
}
.custom-scrollbar::-webkit-scrollbar-thumb {
@apply bg-lights
}
.active {
@apply bg-balance
  }
```

Main.jsx

```
import { Outlet } from "react-router-dom";
import Search from "../components/Search";
import Sidebar from "../containers/Sidebar";

const Main = () =>
<div className="md:flex h-full">
        <Sidebar />
        <div className="flex flex-col grow md:pl-7 md:pt-7 lg:pt-7">
                <Search />
                <Outlet />
        </div>
</div>

export default Main;
```

Common.jsx

```
import { Outlet } from "react-router-dom";
import Footer from "../containers/Footer";
import Header from "../containers/Header";

const Common = () =>
<>
        <Header />
        <div className="grow bg-lights/30">
                <div className="max-w-[1154px] mx-auto h-full">
                        <Outlet />
                </div>
        </div>
        <Footer />
</>

export default Common;
```

Viewer.jsx

```
import { Link, useLoaderData } from "react-router-dom";
import { defaultArticle, loadArticleById } from "../api/articles";
import ViewerBtn from "../components/ViewerBtn";
import { getURL } from "../api";

const Viewer = () => {

const article = useLoaderData() || defaultArticle;
const date = new Date(article.created);
console.log(article)

const renderContent = () => {
```

```
                const regex = /(\(|\s|^)((?:https?:\/\/)[^\s)]+)/g;
                const paragraphs = article.content.split("\n");
                return paragraphs.map((paragraph, index) => {
                  const parts = paragraph.split(regex);
                  return (
                        <div key={index} className="mt-2">
                          {parts.map((part, partIndex) => {
                                if (regex.test(part)) {
                                  //якщо частина є посиланням
                                  const linkParts = part.split(' ');
                                  const url = linkParts[linkParts.length - 1];
                                  const displayedText = part.slice(0, part.lastIndexOf(url));
                                  return (
                                        <span key={partIndex}>
                                          {displayedText}
                                          <Link className="mt-7 mb-4 mx-auto underline text-accent" to={url}
target="_blank" rel="noopener noreferrer">
                                                {url}
                                          </Link>
                                        </span>
                                  );
                                } else {
                                  //якщо частина не є посиланням
                                  return <span key={partIndex}>{part}</span>;
                                }
                          })}
                        </div>
                  );
                });
          };

        return (
                <div className="flex flex-col md:p-8 p-4 md:w-[780px] w-full mx-auto">
                        <div className="md:px-8 text-center">
                                <p className="font-bold text-xl">{article.title}</p>
                                <p className="my-3">{article.description}</p>
                        </div>
                        <img
                                className="mb-3"
                                src={getURL + `/image/${article.pathToImage}`} alt="" />
                        <div className="ndent-4 mt-4 text-justify">
                                {renderContent()}
                        </div>

                        <div className="flex gap-2 text-accent select-none">
                                {article.tags.map((value) => <ViewerBtn key={value.id} value={value} />)}
                        </div>
                        <div className="flex flex-col w-full items-end mt-5 text-text/50 ">
                                <p>Опубліковано: <time dateTime={article.created}>{date.getDate() + "." +
(date.getMonth() + 1) + "." + date.getFullYear()}</time></p>
                                <p>Переглядів: {article.viewed}</p>
                        </div>
                        <Link className="mt-7 mb-4 mx-auto underline text-accent" to={"/"}>На головну</Link>
                </div>
        );
        }

        export const loaderViewer = ({ params }) =>
        loadArticleById(params.page || 0);

        export default Viewer;
```

Store/index.js

```
import { configureStore, createSlice } from "@reduxjs/toolkit";

const slice = createSlice({
name: 'search',
initialState: {
	content: '',
	current: 0,
},
reducers: {
	setContent: (state, action) => {
		state.content = action.payload;
		state.current = 0;
	},
	setCurrent: (state, action) => void (state.current = action.payload),
}
})

export const { setContent, setCurrent } = slice.actions;
export default configureStore({
reducer: slice.reducer
});
```

Articles.jsx

```
import { useEffect, useState } from "react";
import { useSelector } from "react-redux";
import { useLoaderData } from "react-router-dom";
import { defaultArticles, loadSearchPage } from "../api/articles";
import CardPosted from "../components/card/CardPosted";
import store from "../store";
import Pageable from "./Pageable";

const Articles = () => {
const [update, setUpdate] = useState(false);
const [articles, setArticles] = useState(defaultArticles);
const { content, current } = useSelector(store => store);
const loader = useLoaderData();

useEffect(() => {
	setArticles(loader);
	setUpdate(true);
}, [loader]);

useEffect(() => {
	if (!update)
		return;
	loadSearchPage({ page: current, content }).then(setArticles);
	// eslint-disable-next-line
}, [content, current]);

return (
	<>
		<h1 className="text-black font-bold md:ml-0 ml-3 my-3">Знайдені публікації
{content.length > 0 ? `за запитом: '${content}'` : ''}</h1>
		<div className="flex flex-wrap items-stretch justify-left gap-5">
			{articles.content.map((value) => <CardPosted key={value.id} article={value} />)}
		</div>
		<div className="flex justify-center gap-3 my-3 select-none" >
			<Pageable current={articles.current} total={articles.total} />
		</div>
	</>
);
}
```

```
export const loaderArticles = ({ params }) => {
const { content, current } = store.getState();
return loadSearchPage({ page: params.page || current, content });
}

export default Articles;
```

Header.jsx

```
import { Link, useLoaderData } from "react-router-dom";
import { ReactComponent as Logo } from "../assets/images/logo-header.svg";
import HeaderBtn from "../components/HeaderBtn";
import CardGreeting from "../components/card/CardGreeting";
import CardLocked from "../components/card/CardLocked";

const Header = () => {
const locked = useLoaderData();

return (<header className="bg-no-repeat bg-cover bg-center bg-sliderBackground p-3">
        <div className="max-w-[1152px] mx-auto">
                <div className="md:flex justify-between w-full">
                        <Link className="h-auto px-2 cursor-pointer" to="/"><Logo /></Link>
                        <div className="flex flex-col select-none md:flex-row gap-2 md:mt-0 mt-5">
                                <HeaderBtn children={"Всі новини"} />
                                <HeaderBtn tags={"Студентам"} />
                                <HeaderBtn tags={"Абітурієнтам"} />
                                <HeaderBtn tags={"Гуртожиток"} />
                        </div>
                </div>
                <div className="flex gap-8 items-stretch custom-scrollbar mt-5">
                        <CardGreeting />
                        {locked.slice().reverse().map((value) => <CardLocked key={value.id}
article={value} />)}
                </div>
        </div>
</header>);
}

export default Header;
```

Footer.jsx

```
import { ReactComponent as Logo } from "../assets/images/logo-footer.svg";
import { ReactComponent as FaceBook } from "../assets/images/soc-facebook.svg";
import { ReactComponent as Instagram } from "../assets/images/soc-instagram.svg";
import { ReactComponent as YouTube } from "../assets/images/soc-youtube.svg";

const Footer = () => {
return (
        <footer className="bg-footerbg text-lights font-europe text-lg select-none">
                <div className="max-w-[1152px] mx-auto px-3">
                        <div className="md:flex border-b border-balance/20 py-7">
                                <div className="h-[78px] mr-2"> <Logo /></div>
                                <div className="md:mt-0 mt-2 w-full flex items-start justify-between
font-europe">
                                        <div>
                                                <div className="font-bold">Корисні посилання</div>
                                                <ul className="text-md underline text-lights/60">
                                                        <li><a target="_blank" rel="noopener
noreferrer" href="https://www.nmu.org.ua/">Сторінка університету</a></li>
                                                        <li><a target="_blank" rel="noopener
noreferrer" href="https://do.nmu.org.ua/">Дистанційна освіта</a></li>
                                                </ul>
                                        </div>
```

```
                                                <div>
                                                        <div className="font-bold mb-2">Соціальні
мережі</div>
                                                        <ul className="flex justify-between h-8">
                                                                <li><a target="_blank" rel="noopener
noreferrer" href="https://instagram.com/dniprotech?igshid=NTc4MTIwNjQ2YQ=="><Instagram /></a></li>
                                                                <li><a target="_blank" rel="noopener
noreferrer" href="https://m.facebook.com/ntudp"><FaceBook /></a></li>
                                                                <li><a target="_blank" rel="noopener
noreferrer" href="https://youtube.com/@dniprotech"><YouTube /></a></li>
                                                        </ul>
                                                </div>
                                        </div>
                                </div>

                                <div className="flex flex-col justify-end text-end py-5 text-lights/60">
                                        <div>Роботу виконала: ст. гр. 122-19-4</div>
                                        <div>Франчін В.</div>
                                </div>
                        </div>
                </footer>
        );
        }

        export default Footer;
```

Pagable.jsx

```
        import PageableBtn from "../components/PageableBtn";

        const Pageable = ({ current, total }) =>
        <>
                {
                        Array(9).fill(0)
                                .map((_, index) => index + current - 4)
                                .filter(value => value >= 0 && value < total)
                                .map((value) => <PageableBtn key={value} value={value} />)
                }
        </>

        export default Pageable;
```

Sidebar.jsx

```
        import { useLoaderData } from "react-router-dom";
        import SidebarBtn from "../components/SidebarBtn";

        const Sidebar = () => {
        const popular = useLoaderData();
        return (
                <div className="bg-lights min-w-[271px] pl-7 pb-7">
                        <h1 className="text-black pt-7 pb-4 pr-5 font-bold">Найпопулярніші теми</h1>
                        <div className="flex flex-col gap-2">
                                {popular.filter(tag => tag.content !== "Студентам" && tag.content !==
"Абітурієнтам" && tag.content !== "Гуртожиток").slice(0, 5).map((value) => <SidebarBtn key={value.id}
tag={value} />)}
                        </div>
                </div>
        );
        }

        export default Sidebar;
```

HeaderBtn.jsx

```
        import { useDispatch } from "react-redux";
```

```
import { setContent } from "../store";
import { useNavigate } from "react-router-dom";

const HeaderBtn = ({ children, tags = '' }) => {
const dispatch = useDispatch();
const navigate = useNavigate();
const handlerClick = () => {
        dispatch(setContent((tags.length > 0 ? '#' : '') + tags.toLowerCase()));
        navigate('/');
}

return (
        <div
                onClick={handlerClick}
                className="text-lights bg-info/70 hover:text-dark hover:bg-balance/70 transition-[500ms]
md:rounded-b-none rounded-md p-2"
                children={tags.length > 0 ? tags : children}
        />
);
}

export default HeaderBtn;
```

PagableBtn.jsx

```
import { useDispatch, useSelector  } from "react-redux";
import { setCurrent } from "../store";

const PageableBtn = ({ value }) => {
const dispatch = useDispatch();
const currentPage = useSelector((state) => state.current);
const handleClick = () => {
        dispatch(setCurrent(value));
};

const isActive = value === currentPage;

return (
        <div
                onClick={handleClick}
                className={`p-2 bg-lights cursor-pointer ${isActive ? "active" : "className=bg-
balance"}`}
                children={value + 1}
        />
);
}

export default PageableBtn;
```

Search.jsx

```
import { useDispatch } from "react-redux";
import { redirect } from "react-router-dom";
import { ReactComponent as Logo } from "../assets/images/Search.svg";
import { setContent } from "../store";

const Search = () => {
const dispatch = useDispatch();

const handlerSubmit = event => {
        dispatch(setContent(event.target.search.value));
        event.preventDefault();
}

return (
```

```
        <form
                className="flex flex-row relative border-[8px] border-lights"
                onSubmit={handlerSubmit}>
                <input
                        className="grow bg-transparent p-1 pl-2 outline-none"
                        name="search"
                        maxLength="255"
                        type="text"
                        placeholder="Знайти публікацію..."
                        defaultValue={window.sessionStorage.search || ''}
                />
                <button
                        className="right-1 top-1 h-6 self-center pr-2"
                        type="submit"
                        children={<Logo />}
                />
        </form>
);
}

export const actionSearch = async ({ request }) => {
const data = await request.formData();
window.sessionStorage.setItem('search', data.get('search'));
return redirect(`/`);
}

export default Search;
```

SidebarBtn.jsx

```
import { useDispatch } from "react-redux";
import { defaultTag } from "../api/tags";
import { setContent } from "../store";

const SidebarBtn = ({ tag = defaultTag }) => {
const dispatch = useDispatch();

const handlerClick = () =>
        dispatch(setContent('#' + tag.content));

return (
        <div
                className="border-b border-balance/70 pb-1 select-none cursor-pointer"
                onClick={(handlerClick)}
        >
                <div className="text-text">#{tag.content}</div>
                <div className="text-text/60">Переглядів: {tag.viewed}</div>
        </div>
);
}

export default SidebarBtn;
```

ViewerBtn.jsx

```
import { useDispatch } from "react-redux";
import { setContent } from "../store";
import { useNavigate } from "react-router-dom";

const ViewerBtn = ({ value }) => {
const dispatch = useDispatch();
const navigate = useNavigate();
const handlerClick = () => {
        dispatch(setContent('#' + value.content));
        navigate('/');
```

```
}
return <div
        className="cursor-pointer"
        children={'#' + value.content}
        onClick={handlerClick}
/>;
}

export default ViewerBtn;
```

CardGreeting.jsx

```
const CardGreeting = () => {
return (
        <div className="bg-info/70 select-none rounded-t-lg p-2">
                <div className="flex flex-col gap-2 w-[255px] text-lights">
                        <div className="flex flex-col gap-1 pb-2 border-b border-white/60">
                                <p className="text-lg font-bold">Dnipro University of Technology</p>
                                <p>Compliance with the Time</p>
                        </div>
                        <div className="flex flex-col gap-3">
                                <p>Дніпровська політехніка - кращий ЗВО регіону!</p>
                                <p>Наш університет дбає про те, щоб студентське життя було
яскравим та насиченим, а рівень освіти відповідав сучасним стандартам навіть під час дистанційної освіти.
</p>
                                <p>Залишайтесь в курсі головних подій в Дніпровській
політехніці!</p>
                        </div>
                </div>
        </div>
);
}

export default CardGreeting;
```

CardLocked.jsx

```
import { useNavigate } from "react-router-dom";
import { defaultArticle } from "../../api/articles";
import { getURL } from "../../api";

const CardLocked = ({
article = defaultArticle
}) => {
const navigate = useNavigate();
return (
        <div
                className="bg-white/80 select-none rounded-t-lg p-2 cursor-pointer"
                onClick={() => navigate(`/view/${article.id}`)}
        >
                <div className="w-[255px]">
                        <img className="rounded-t-lg pointer-events-none" src={getURL +
`/image/${article.pathToImage}`} alt="" />
                        <h1 className="font-bold mt-2 break-words line-clamp-2 hover:underline"
children={article.title} />
                        <p className="mt-1 break-words line-clamp-3" children={article.description} />
                </div>
        </div>
);
}

export default CardLocked;
```

CardPosted.jsx

```
import { useNavigate } from "react-router-dom";
import { defaultArticle } from "../../api/articles";
import { getURL } from "../../api";


const CardPosted = ({ article = defaultArticle }) => {
const navigate = useNavigate();
const date = new Date(article.created);
return (
	<div
		className="bg-lights/40 select-none md:rounded-b-none rounded-lg p-2 cursor-pointer
lg:basis-[calc((100%-2.5rem)/3)] md:basis-[calc((100%-1.25rem)/2)] basis-[100%] overflow-hidden"
		onClick={() => navigate(`/view/${article.id}`)}
	>
		<div className="w-full h-full flex flex-col justify-between">
			<div>
				<img className="w-full md:rounded-b-none rounded-lg pointer-events-
none" src={getURL + `/image/${article.pathToImage}`} alt="" />
				<h1 className="font-bold break-words line-clamp-2 hover:underline"
children={article.title} />
				<p className="break-words line-clamp-3" children={article.description}
/>
			</div>
			<div className="flex flex-col gap-3">
				<div className="flex flex-col w-full items-end text-text/50">
					<div>Опубліковано: <time
dateTime={article.created}>{date.getDate() + "." + (date.getMonth() + 1) + "." + date.getFullYear()}</time></div>
					<div>Переглядів: {article.viewed}</div>
				</div>
			</div>
		</div>
	</div>
);
}

export default CardPosted;
```

card/SidebarBtn.jsx

```
import { useDispatch } from "react-redux";
import { defaultTag } from "../api/tags";
import { setContent } from "../store";

const SidebarBtn = ({ tag = defaultTag }) => {
const dispatch = useDispatch();

const handlerClick = () =>
	dispatch(setContent('#' + tag.content));

return (
	<div
		className="border-b border-balance/70 pb-1 select-none cursor-pointer"
		onClick={(handlerClick)}
	>
		<div className="text-text">#{tag.content}</div>
		<div className="text-text/60">Переглядів: {tag.viewed}</div>
	</div>
);
}

export default SidebarBtn;
```

api/articles.js

```
import { getURL } from ".";
```

```
const getArticleURL = getURL + `/article`;

export const defaultArticle = {
id: -1,
title: "Обраний пост не вдалося завантажити",
description: "Виникла проблема під час завантаження цього поста. Перезавантажте сторінку або зверніться до цього ресурсу пізніше.",
pathToImage: null,
content: "Виникла проблема під час завантаження цього поста. Перезавантажте сторінку або зверніться до цього ресурсу пізніше. Можливо проблема в інтернет-з'єднанні або у відсутності цієї статті на сайті. Передбачається, що цей ресурс міг бути видалений.",
created: "2001-01-01",
locked: false,
viewed: 0,
tags: [],
}

export const defaultArticles = {
content: [defaultArticle],
current: 0,
total: 1,
}

export const loadArticleById = (id) =>
fetch(getArticleURL + "/" + id).then(res => res.status === 200 ? res.json() : defaultArticle);

export const loadArticleLocked = () =>
fetch(getArticleURL + "/locked").then(res => res.status === 200 ? res.json() : []);

export const loadSearchPage = ({ page = 0, size = 6, content = " }) =>
fetch(getArticleURL + '/search', {
        method: 'POST',
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ size: size, page: page, content: content })
}).then(res => res.status === 200 ? res.json() : defaultArticles);
```

api/index.js

```
export const getURL =
`${process.env.REACT_APP_PROT}://${process.env.REACT_APP_HOST}:${process.env.REACT_APP_PORT}`;
```

api/tags.js

```
import { getURL } from ".";

const getTagURL = getURL + `/tag`;

export const defaultTag = {
id: -1,
content: "Не вдалося завантажити",
viewed: 0,
}

export const loadTagPopular = () =>
fetch(getTagURL + "/popular").then(res => res.status === 200 ? res.json() : []).then(value =>
value.map((value) => ({
        id: value[0],
        content: value[1],
        viewed: value[2],
})));
```

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
```

```
content: [
        "./src/**/*.{js,jsx,ts,tsx}",
],
theme: {
        extend: {
                colors: {
                        text: '#060D2F',
                        dark: '#1E1E62',
                        accent: '#546CCF',
                        balance: '#97BBFE',
                        lights: '#DAEEFE',
                        info: '#4B5EAF',
                        footerbg: '#3B3B74'
                },
                backgroundImage: {
                        sliderBackground: 'url(assets/images/sliderBackground.jpg)',
                        logoBackground: 'url(assets/images/logo.svg)'
                }
        }
},
plugins: [

],
}
```

//Код клієнтського застосунку (сторінка адміністратора)

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import { RouterProvider } from 'react-router-dom';
import router from './router';

ReactDOM.createRoot(document.getElementById('root')).render(
<RouterProvider router={router} />
);
```

router.js

```
import { setupListeners } from "@reduxjs/toolkit/dist/query";
import { Provider } from "react-redux";
import { Outlet, Route, ScrollRestoration, createBrowserRouter, createRoutesFromElements } from "react-
router-dom";
import { ArticlesEdit, ArticlesCreate, Sidebar } from "./containers";
import store from "./store";

setupListeners(store.dispatch)

const Common = () => {
return (
        <Provider store={store} >
                <Outlet />
                <Sidebar />
                <ScrollRestoration />
        </Provider>
)
}

export default createBrowserRouter(createRoutesFromElements(
<Route element={<Common />}>
        <Route index element={<ArticlesCreate />} />
        <Route path="/:id" element={<ArticlesEdit />} />
</Route>
))
```

index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
.scrollbar {
        @apply overflow-y-auto;
}

.scrollbar::-webkit-scrollbar {
        width: 6px;
}

.scrollbar::-webkit-scrollbar-track {
        @apply bg-info
}

.scrollbar::-webkit-scrollbar-thumb {
        @apply bg-lights
}

.scroll-hidden::-webkit-scrollbar {
        width: 0;
  }

.custom-scrollbar {
        @apply overflow-y-auto;
}

.custom-scrollbar::-webkit-scrollbar {
        width: 4px;
}

.custom-scrollbar::-webkit-scrollbar-track {
        @apply bg-info/40
}

.custom-scrollbar::-webkit-scrollbar-thumb {
        @apply bg-lights
}
}

* {
margin: 0;
padding: 0;
@apply box-border;
}

html, body, #root {
height: 100%;
@apply bg-white;
@apply scroll-hidden;
}

#root {
@apply flex md:flex-row flex-col max-w-screen-lg mx-auto;
}
```

Article.jsx

```
import { useNavigate } from "react-router-dom";
import { useDeleteArticleMutation, useLockedMutation } from "../store/articlesApi";
```

```
const Article = ({ value }) => {
const { id, pathToImage, title, locked } = value;
const [fetchDelete] = useDeleteArticleMutation();
const [fetchLocked] = useLockedMutation();
const navigate = useNavigate();


const handleClickLocked = () =>
        fetchLocked(id).unwrap();
const handleClickDelete = () =>
        fetchDelete(id).unwrap();

return (
        <div className="relative bg-info text-white md:rounded-l select-none">
                <div className="flex justify-start border-b border-black">
                        <div className="min-w-fit rounded-tl border-t border-x border-black overflow-
hidden">
                                <img className="h-24 object-contain pointer-events-none" src={

`${process.env.REACT_APP_PROT}://${process.env.REACT_APP_HOST}:${process.env.REACT_APP_P
ORT}/image/${pathToImage}`
                                } alt="" />
                        </div>
                        <p className="font-bold line-clamp-4 tracking-tighter pl-1 pt-0" children={title}
/>
                </div>
                <div className="flex text-center">
                        <h1 className="w-full cursor-pointer hover:bg-balance/60 transition-[50ms]"
onClick={() => navigate('/' + id)} >Редагувати</h1>
                        <div className="border-l border-black" />
                        <h1 className="w-full cursor-pointer hover:bg-balance/60 transition-[50ms]"
onClick={handleClickLocked}>{locked ? 'Відкріпити' : 'Закріпити'}</h1>
                        <div className="border-l border-black" />
                        <h1 className="w-full cursor-pointer hover:bg-balance/60 transition-[50ms]"
onClick={handleClickDelete}>Видалити</h1>
                </div>
        </div>
);
}
export default Article;
```

ArticleTag.jsx

```
import { useDispatch } from "react-redux";
import { ReactComponent as XVAR } from "../assets/x-var.svg";
import { removeTags } from "../store/tagsSlice";

const ArticleTag = ({ value, children }) => {
const dispatch = useDispatch()
return (
        <div className="flex justify-between items-center bg-info text-lights px-2 py-1 rounded-full ">
                <h1 children={'#' + value?.content || children} />
                <XVAR className="fill-balance h-4 mx-2 cursor-pointer" onClick={() =>
dispatch(removeTags(value))} />
        </div>
);
}

export default ArticleTag;
```

CreateArticle.jsx

```
import { useDispatch, useSelector } from "react-redux";
import { useNavigate, useParams } from "react-router-dom";
import { setTags } from "../store/tagsSlice";
```

```
const CreateArticle = () => {
const dispatch = useDispatch();
const navigate = useNavigate()
const { menu } = useSelector(store => store.comp);
const { id } = useParams();

const handlerClick = () => {
        dispatch(setTags([]))
        navigate('/')
}

return (id && menu === 0) && (
        <button className="bg-info text-xl text-white p-1 hover:bg-balance transition-[50ms] md:rounded-l"
onClick={handlerClick}>
                Нова публікація
        </button>
)
}

export default CreateArticle;
```

CreateTag.jsx

```
import { useSelector } from "react-redux";
import { ReactComponent as Plus } from "../assets/plus.svg";
import { useCreateTagMutation } from "../store/articlesApi";

const CreateTag = () => {
const { menu } = useSelector(store => store.comp);
const createTag = useCreateTagMutation()[0];

const handlerSubmit = (event) => {
        event.preventDefault();
        createTag(event.target.content.value).unwrap();
}

return menu === 1 && (
        <form
                className="relative border-[8px] border-info"
                onSubmit={handlerSubmit}
        >
                <input
                        name="content"
                        className="w-full bg-transparent p-1 pl-2 outline-none"
                        placeholder="Створити новий тег..."
                        pattern="[a-za-яіїєґA-ZA-ЯІЇЄҐ0-9]+"
                        type="text"
                        required
                />
                <button
                        className="absolute right-1 top-1 h-6"
                        type="submit"
                        children={<Plus className="fill-balance" />}
                />
        </form>
)
}

export default CreateTag;
```

Решта коду додається окремо.

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему

"Розробка вебзастосунку для публікації новин та оголошень університету з використанням Spring Boot і ReactJS"

студентки групи 122-19-4 Франчін Вікторії Паолівни

Керівник економічного розділу **О.Г. Вагонова**
проф. каф. ПЕП та ПУ, д.е.н

ДОДАТОК В

РЕЦЕНЗІЯ

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему

"Розробка вебзастосунку для публікації новин та оголошень університету з використанням Spring Boot і ReactJS"

студентки групи 122-19-4 Франчін Вікторії Паолівни

Рецензент кваліфікаційної роботи

ДОДАТОК Г

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна робота Франчін.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Франчін.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
news.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Франчін.ppt	Презентація кваліфікаційної роботи