Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Пояснювальна записка

до комплексної бакалаврської дипломної роботи на тему:

«Клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 3. «Розробка панелі адміністратора за допомогою реактивного фреймворка React»

Виконав: студент 2 курсу, групи КІ-20мсз
спеціальності 123 — Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Никитюк В.О.
(прізвище та ініціали)
Керівник к.т.н., доцент каф. ОТ
Семеренко В.П.
(прізвище та ініціали)
«13» 06 2022 р.
Рецензент к.т.н., доцент каф. ЗІ
Куперштейн Л. М.
(прізвище та ініціали)
«14» 06 2022 р.

Допущено до захисту
Зав. кафедри д.т.н., проф. О.Д. Азаров
«15» 06 2022 р.

Вінниця ВНТУ — 2022 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень перший (бакалаврський)
Галузь знань — 12 — Інформаційні технології
(шифр і назва)

Спеціальність — 123 — «Комп'ютерна інженерія»
(шифр і назва)

Освітньо — професійна програма — Комп'ютерна інженерія
(Назва освітньо — професійної програми)

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ, д.т.н, проф.
Азаров О.Д.
"09" 02 2022року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Никитюку В'ячеславу Олександровичу

1 Тема роботи — Клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 3. «Розробка панелі адміністратора за допомогою реактивного фреймворка React», керівник роботи Семеренко В.П., затверджені наказом вищого навчального закладу від 24.03.2022 № 66

2 Строк подання студентом роботи 14.06.2022.

3 Вихідні дані до роботи: контент для наповнення веб-додатку інформаційної системи через адміністративну панель, дані для створення макету веб-інтерфейсу.

4 Зміст текстової частини — огляд основних популярних технологій для організації клієнт-серверної взаємодії. Розробка веб-інтерфейсу адміністративної панелі. Опис основних функціональних можливостей веб-інтерфейсу адміністративної панелі.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): структурна схема ідентифікації користувача, структурна схема додавання новини, дизайн сторінок адміністративної панелі.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Семеренко В. П., к.т.н., доцент каф. ОТ		
1-3	Войцеховська О. В., к.т.н., доцент каф. ОТ		

7 Дата видачі завдання 09.01.2022.

8 Календарний план виконання курсової роботи приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз завдання	08.02.22		вик.
2	Огляд аналогів інформаційних систем інших структурних підрозділів та веб-інтерфейсів	09.02.22	16.02.22	вик.
3	Аналіз сучасних глобальних сховищ	17.02.22	24.02.22	вик.
4	Аналіз технологій для розробки адміністративної панелі для веб-додатку	25.02.22	05.03.22	вик.
5	Вибір колірної палітри та створення прототипу веб-сайту	06.03.22	20.03.22	вик.
6	Створення дизайну веб-інтерфейсу	21.03.22	05.04.22	вик.
7	Розробка фізичної структури веб-додатку. Верстка сторінок веб-сайту	06.04.22	10.04.22	вик.
8	Реалізація та налаштування клієнт-серверної взаємодії. Реалізація авторизації	11.04.22	02.05.22	вик.
9	Реалізація функціональних можливостей веб-інтерфейсу адміністративної панелі	03.05.22	25.05.22	вик.
10	Оформлення пояснювальної записки та ілюстративного матеріалу	26.05.22	10.06.22	вик.
11	Перевірка якості виконання бакалаврської роботи та усунення недоліків	13.06.22		вик.

Студент _____ Никитюк В.О.

Керівник роботи _____ Семеренко В.П.

АНОТАЦІЯ

Комплексна бакалаврська дипломна робота складається з 68 сторінок формату А4, на яких є 25 рисунків, перелік джерел посилань містить 15 найменувань.

В комплексній бакалаврській дипломні роботі проведено аналіз популярних технологій для розробки макету клієнтської частини веб-додатку та адміністративної панелі. Було створено порівняльну характеристику глобальних сховищ, додатків і сервісів для розробки дизайну.

Спроектовано фізичну структуру проекту та розроблено макет веб-інтерфейсу адміністративної панелі, за яким виконано її верстку, використовуючи фреймворк React.

Реалізовано різні функціональні можливості веб-інтерфейсу адміністративної панелі та клієнт-серверну взаємодію з використанням сучасних технологій.

Ключові слова: адміністративна панель, інформаційна система, React, GraphQL, клієнт-сервер, Apollo Client, MobX.

ABSTRACT

The complex bachelor's thesis consists of 68 A4 pages, which contain 25 figures, the list of reference sources contains 15 titles.

In the complex bachelor's thesis the analysis of popular technologies for development of the layout of the client part of the web application and the administrative panel is carried out. A comparative description of global repositories, applications, and design services has been developed.

The physical structure of the project was designed and the layout of the web interface of the administrative panel was developed, according to which its layout was performed using the React framework.

Various functionalities of the web interface of the administrative panel and client-server interaction with the use of modern technologies have been implemented.

Keywords: administrative panel, information system, React, GraphQL, client-server, Apollo Client, MobX.

ЗМІСТ

ВСТУП

В сучасному світі веб-сайти вирішують велику кількість питань користувачів та завдань бізнесу. Навчальні платформи, пошук музики, продаж різноманітних товарів в зручному інтерфейсі та з швидким керуванням великими кількостями інформації.

Зазвичай кожен університет чи коледж мають свої веб-сайти зі всією необхідною інформацією про навчальний заклад: вимоги до вступу, рейтингові списки, загальна інформація про навчальний заклад, список факультетів та ін. Це дає можливість визначитись з вибором університету, знаходячись вдома, а не відвідуючи кожен конкретний заклад. Інформаційні системи структурного підрозділу навчального закладу виконують ту ж функцію, але з детальнішим описом свого напрямку, історією та новинами кафедри, навчальною інформацією та методичними розробками. Такі інформаційні системи є доволі актуальними, особливо в сучасних умовах, коли навчання відбувається переважно онлайн.

Крім красивого дизайну, який буде приваблювати користувачів та дозволить зручно користуватись інтерфейсом або User Interface (UI) / User Experience (UX) потрібно досягти комфортної взаємодії з сайтом зі сторони його керування.

Адміністрування клієнтського додатку полягає в наповненні контентом, його редагуванням: зміна фотографій, додавання нової новини, редагування загальної інформації. Все це також невід’ємна частина підтримки сайту, яку виконує його власник чи інші робітники, що пов’язані з додатком.

Важливою частиною розробки веб-сайту є створення клієнтського інтерфейсу, але зручність адміністративної частини також вирішальна. Чим зручніше буде реалізований дизайн та інструментарій для наповнення або редагування, тим швидше, а отже продуктивніше, можна буде робити певні зміни в основному додатку.

Метою роботи є розробка адміністративної панелі для клієнт-серверного веб-додатку структурного підрозділу навчального закладу, яка дозволить швидко та продуктивно редагувати, видаляти та добавляти необхідну інформацію в клієнтський додаток і керувати правами доступу користувачів, розділяючи їх на гостей, викладачів та адміністраторів, які зможуть користуватись адміністративною панеллю.

Задачі дослідження бакалаврської роботи:

— проаналізувати популярні технології розробки клієнтських інтерфейсів;

— обрати оптимальний стек технологій для проектування адміністративної панелі інформаційної системи підрозділу навчального закладу;

— розробити макет веб-додатку підрозділу навчального закладу;

— розробити структуру та дизайн адміністративної панелі для інформаційної системи підрозділу навчального закладу;

— виконати верстку адміністративної панелі інформаційної системи підрозділу навчального закладу;

— розробити програмний код для забезпечення необхідних функціональних можливостей адміністративної панелі веб-додатку підрозділу навчального закладу.

Об’єкт дослідження — процес проектування та розробки інтерфейсу і функціональної частини адміністративної панелі для інформаційної системи структурного підрозділу навчального закладу.

Предметом дослідження є технології клієнт-серверної взаємодії при розробці адміністративної панелі для зручного керування та адміністрування контенту веб-інтерфейсу інформаційної системи.

Апробація результатів бакалаврської роботи: зроблено доповідь на LI науково-технічній конференції підрозділів Вінницького національного технічного університету [1].

Публікація за темою роботи: В. О. Никитюк. Аналіз методів клієнт-серверної взаємодії при розробці панелі адміністратора за допомогою реактивного фреймворка React / В. О. Никитюк, О. В. Войцеховська // Тези доповіді. Матеріали LI наукової-технічної конференції підрозділів Вінницького національного технічного університету (Вінниця, 2022 р.) [Електронний ресурс]. Режим доступу: до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15104/12730**.**

1 АНАЛІЗ ТЕХНОЛОГІЙ КЛІЄНТ-СЕРВЕРНОЇ ВЗАЄМОДІЇ ПРИ РОЗРОБЦІ ПАНЕЛІ АДМІНІСТРАТОРА ДЛЯ ІНФОМАЦІЙНОЇ СИСТЕМИ ПІДРОЗДІЛУ НАВЧАЛЬНОГО ЗАКЛАДУ

При розробці панелі адміністратора для інформаційної системи структурного підрозділу навчального закладу потрібно проаналізувати доступні популярні та ефективні технології клієнт-серверної взаємодії.

Також основним завданням адміністративної панелі є подання зручного інтерфейсу, який дозволить виконувати адміністрування веб-додатку без спеціальних технічних знань та знань сучасних IT-технологій.

Інтерфейс являє собою графічну структуру програми. Він складається з кнопок, які натискають користувачі, текстів, які вони читають, зображень, полів введення тексту і всіх інших елементів, з якими взаємодіє користувач. Крім того, він включає в себе макет екрану, переходи, анімацію інтерфейсу і кожну мікро-взаємодію. Будь-який візуальний елемент, взаємодія або анімація повинні бути розроблені в процесі дизайну інтерфейсу користувача (UI-дизайну).

UI дизайн — це, в першу чергу, графічний дизайн, який повинен бути естетичним, привабливим, візуально стимулювати користувача та відповідати тематиці веб-додатку [2].

UX дизайн визначається тим, як користувачі взаємодіють з додатком або сервісом. UX дизайн повинен бути інтуїтивно-зрозумілим, навігація в додатку повинна бути логічною та простою.

UX дизайн визначається тим, наскільки легко чи складно взаємодіяти з елементами призначеного для користувача інтерфейсу, створеними дизайнерами.

1.1 Огляд основних популярних технологій для організації клієнт-серверної взаємодії

1.1.1 Огляд бібліотеки Apollo Client

Технологія Apollo Client — це повна бібліотека керування станом для JavaScript, яка дозволяє керувати локальними та віддаленими даними за допомогою GraphQL та використовується для отримання, кешування та зміни даних програми, автоматично оновлюючи інтерфейс користувача [3].

Потрібно прагнути до простого та зрозумілого керування даними. Саме декларативний підхід та кешування Apollo до отримання даних забезпечує написання меншої кількості коду з кращим результатом.

Схематично зв’язок між сервером і клієнтом, використовуючи Apollo, подано на рисунку 1.1.

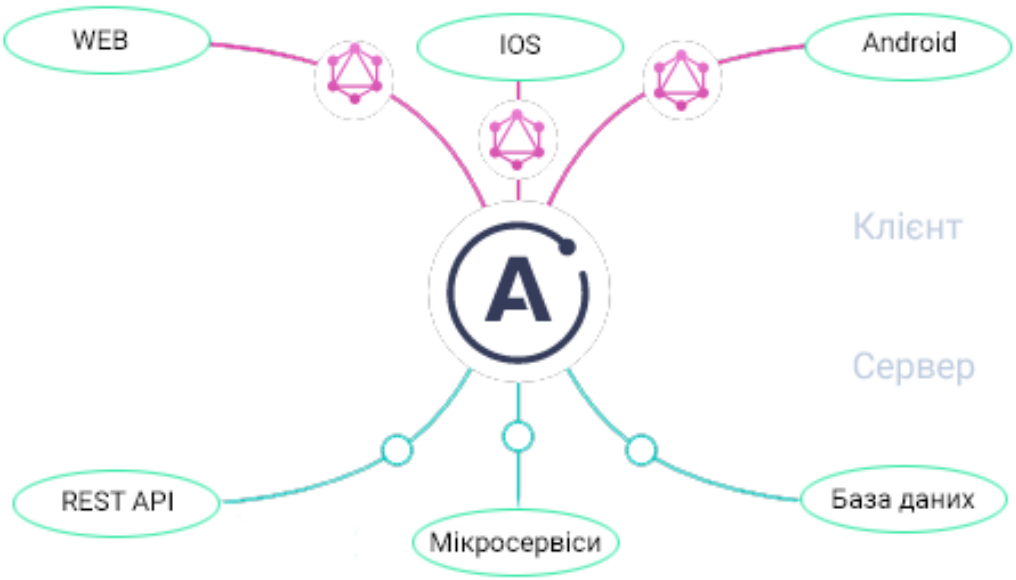

Рисунок 1.1 — Візуальне представлення клієнт-серверної взаємодії з допомогою Apollo

З рисунку 1.1 видно, що взаємодія з базою даних, робота мікросервісів та REST API за допомогою цієї бібліотеки дозволяє використання GraphQL на додатках для IOS чи Android або веб-застосунках.

1.1.2 Огляд фреймворку JavaScript React

Технологія React — це найпопулярніший фреймворк на даний час, який є зручним, гнучким та відносно простим в використанні рішенням для розробки інтерфейсу користувача.

Суттєвою перевагою React перед іншими популярними фреймворками є можливість вибору потрібної кількості бібліотек, які будуть задіяні в проекті, що дозволяє не навантажувати проект зайвими модулями, покращивши цим швидкодію веб-застосунку та його оптимізацію [4].

Компонентний підхід розробки, який передбачений в React, розширяє структурні здібності розробки, так як одні і ті ж компоненти можуть використовуватись в різних частинах проекту, а з можливостями взаємодії з ними через props (об’єкт із даними) можна міняти його відображення або принцип дії залежно від необхідного результату.

Веб-додатки, що виконані за допомогою фреймворку React називаються Single Page Applications (SPA), тому що при переході між сторінками, вкладка браузера не перезавантажується, а змінюється лише динамічна частина веб-інтерфейсу.

1.1.3 Аналіз можливостей сумісного використання технологій Apollo Client та React для отримання даних з серверу

Технологія Apollo Client допомагає структурувати код без ручного відстеження станів додатку (декларативний підхід), що відповідає сучасним методам розробки програмного забезпечення. Основна бібліотека забезпечує вбудовану інтеграцію з React.

Зараз Apollo Client включає три hooks (хуки), які можна використовувати в додатках — у всіх тих місцях, де використовуються компоненти вищого порядку або механізми render props. Це хуки: useQuery, useMutation і useSubscription, які прості в освоєнні, мають безліч переваг перед раніше існуючим REST API. Зокрема, це стосується зменшення розмірів сукупності

вихідних програмних файлів програми та скорочення обсягу шаблонного коду. Це дає можливість зменшити залежність від ширини інтернет-каналу від провайдера, що, в свою чергу, дозволить мінімізувати час відновлення та кешування програмного продукту.

1.1.3.1 Використання відкладеного виконання запитів

Хук useQuery виконує запит на момент виклику функції, однак саме така поведінка системи потрібна не завжди. Наприклад, потрібно створити поле для введення пошукового запиту, що видає користувачеві підказки. Для цього може знадобитись вивести компонент з підказками, які з'являться на основі того, що введе користувач. Але при цьому виконання запиту на пошуковий сервер буде відкладено до того моменту, поки користувач почне щось вводити в поле. Для реалізації подібного сценарію можна скористатись хуком useLazyQuery, який повертає кортеж з функцією execute, як перший аргумент та змінні loading і data, як показано в лістингу 1.1.

Лістинг 1.1 — Синтаксис використання хука useLazyQuery

```
const [execute, { loading, data }] =
useLazyQuery(GET_INVENTORY)
```

Запит не буде виконаний доти, доки не буде викликана функція execute. У цей момент компонент повторно відрендериться і буде застосована звичайна схема виконання запитів, що реалізується за допомогою useQuery.

1.1.3.2 Використання множинних мутацій

Коли виконуються множинні мутації в одному компоненті, використання компонентів вищого порядку або механізму render props може призвести до появи коду, який буде складно зрозуміти. Застосування API render prop передбачає побудову вкладених один в одного компонентів, що дає хибне відчуття структурованості коду та її точної ієрархії. Новий хук

useMutation дозволяє повністю обійти цю проблему, оскільки його використання зводиться до простого виклику функції. У прикладі (лістинг 1.2) показано, як декілька мутацій та запитів можуть взаємодіяти один з одним. Все це відбувається в межах того самого компонента.

Лістинг 1.2 — Використання множинних мутацій в межах одного компонента

```
function Message() {
  const [saveMessage, { loading }] =
useMutation(SAVE_MESSAGE)
  const [deleteMessage] = useMutation(DELETE_MESSAGE)
  const { data } = useQuery(GET_MESSAGE)
  return (<div>
      <p>
        {loading
          ? 'Loading ...'
          : `Message: ${data && data.message ?
data.message.content : ''}`}
      </p>
      <p>
<button onClick={() => saveMessage()}>Зберегти</button>
<button onClick={() => deleteMessage()}>Видалити</button>
      </p>
    </div>)}
```

В лістингу 1.2 описано використання декількох мутацій. Перша мутація використовується після натискання на кнопку «Зберегти» і потрібна для зберігання повідомлення, тоді як кнопка «Видалити» викликає мутацію для видалення повідомлення.

1.1.4 Використання REST API для реалізації POST-запитів

Для реалізації POST-запитів використовується технологія REST API, яку в JavaScript можна застосовувати без додаткових бібліотек. Для цього існує

асинхронна функція fetch, яка першим аргументом приймає URL, на який буде відісланий запит та другим аргументом — приймає об’єкт з додатковими основними параметрами:

— method, тобто тип запиту (GET, POST, PUT, DELETE);

— headers, тобто додаткові заголовки запиту, якщо потрібно вказати;

— body, тобто тіло запиту, в якому передаються додаткова інформація та дані.

Функція fetch повертає Promise, який при отримані відповіді від сервера виконує вказану функцію в якій передається об'єкт response як параметр [5].

1.2 Огляд сучасних глобальних сховищ у взаємодії з технологією React

Технологія глобального сховища — це досить зручний підхід для зберігання стану всього додатку в одному місці та можливістю управління ним.

Корисність такого сховища можна відчути, наприклад, виконуючи задачу з переключенням теми сайту (світла або темна). У звичайному випадку можна було б виконати цю задачу, використовуючи тільки props. Але реалізація такого способу забере великий час розробки і буде зовсім не динамічним рішенням. Прийдеться при створенні кожного нового компоненту вручну прописувати для нього ті ж самі props.

Для зберігання часу розробників, меншої кількості коду та кращої структури проекту існують такі бібліотеки, як MobX та Redux, які виконують схожі задачі, але їх принцип дії суттєво відрізняється.

1.2.1 Огляд глобального сховища Redux та реалізація в ньому архітектурного підходу Flux

Глобальне сховище Redux є контейнером для управління станом програми. Redux не прив'язаний безпосередньо до React.js і може використовуватися з іншими JS-бібліотеками та фреймворками [6].

Основні характеристики Redux:

— сховище (store), яке зберігає стан додатку;

— дії (actions), тобто деякий набір інформації, що походить від додатку до сховища і який вказує, що саме потрібно зробити. Для передачі інформації у сховища викликається метод dispatch();

— творці дій (actions creators), тобто функції, що створюють дії;

— reducer, тобто функція (або кілька функцій), яка отримує дію і відповідно до неї змінює стан сховища.

Загальну схему взаємодії елементів архітектури Redux представлено на рисунку 1.2.

Рисунок 1.2 — Загальна схема взаємодії елементів архітектури Redux

З View (тобто компонентів React) посилається конкретна дія через функцію dispatch, цю дію отримує функція reducer, яка відповідно до описаної дії оновлює стан сховища. Після зміни стану сховища компоненти React застосовують уже оновлений стан. Саме так відбувається зміна глобального стану.

Flux являє собою архітектуру програм, які використовують бібліотеку React.

Програми, що використовують Flux, мають три основні частини: диспетчер (dispatcher), сховище даних (store) та представлення (view), які є стандартними компонентами Redux [7].

Диспетчер представляє у всій цій схемі центральну ланку, яка контролює потік даних у Flux. Диспетчер реєструє сховища та їх зворотні виклики (callback). Коли диспетчер отримує ззовні деяку дію, то через зворотні виклики сховищ диспетчер повідомляє цим сховищам про дію, яка відбулась.

Сховища містять стан програми та її логіку. За своєю дією вони можуть нагадувати модель з патерну MVC, у той же час вони не є одним об'єктом, а керують групою об'єктів. Кожне окреме сховище керує певною областю або доменом програми.

Кожне сховище реєструється в диспетчері разом із зворотними викликами. Коли диспетчер отримує дію, він виконує зворотний виклик, передаючи йому дію як параметр. Залежно від типу дії викликається той чи інший метод усередині сховища, де відбувається оновлення стану сховища. Після оновлення сховища генерується подія, яка вказує на те, що сховище було оновлено. І через цю подію уявлення (тобто компоненти React) дізнаються, що сховище було оновлено і самі оновлюють свій стан.

Представлення оформляють візуальну частину програми. Особливий вид представлень — controller-view є компонентом найвищого рівня, який містить всі інші компоненти. Controller-view прослуховує події, що походять від сховища. Отримавши подію, controller-view передає дані, отримані від сховища іншим компонентам.

Коли controller-view отримує подію від сховища, спочатку controller-view запитує у сховища всі необхідні дані. Потім він викликає свій метод setState() або forceUpdate(), який призводить до виконання компонента методу render(). А це, в свою чергу, приводить до виклику методу render() та оновлення дочірніх компонентів.

Дія є функцією, яка може містити деякі дані, які передаються диспетчеру. Дія може бути викликана обробниками подій у компонентах, наприклад, натисканням на кнопку. Також ініціатором дій може бути якесь інше зовнішнє джерело, наприклад, сервер. Через диспетчер сховище отримує дію та відповідним чином реагує на неї.

1.2.2 Огляд глобального сховища MobX та використання в ньому патерну Observer

Глобальне сховище MobX — це бібліотека, яка робить керування станом простим та масштабованим за рахунок прозорого застосування функціонального реактивного програмування та патерну Observer [8].

Патерн Observer — це поведінковий патерн проектування, який створює механізм підписки, що дає змогу одним об’єктам стежити й реагувати на події, які відбуваються в інших об’єктах [9].

Призначення цього патерна в тому, щоб визначати залежність типу «один до багатьох» між об'єктами таким чином, що при зміні стану одного об'єкта всі, що залежать від нього, сповіщаються про це і автоматично оновлюються.

Цей патерн має такі переваги:

— підтримує принцип вільного зв'язку між об'єктами, які взаємодіють один з одним;

— дозволяє ефективно передавати дані іншим об'єктам, без будь-яких змін у класах Subject або Observer;

— спостерігачі можуть бути додані або видалені в будь-який момент часу.

Схематично реалізацію патерна можна представити на рисунку 1.3. При реалізації шаблону «Observer» зазвичай використовуються такі класи:

— Subject — інтерфейс, що визначає методи для додавання, видалення та оповіщення спостерігачів;

— Observer — інтерфейс, за допомогою якого спостережуваний об'єкт оповіщає спостерігачів;

— ConcreteSubject — конкретний клас, який реалізує інтерфейс Subject;

— ConcreteObserver — конкретний клас, який реалізує інтерфейс Observer.

Рисунок 1.3 — Структура патерна Observer

Глобальне сховище MobX автоматично підписує компоненти React на будь-які зміни, які використовуються під час візуалізації. В результаті компоненти будуть автоматично перемальовуватись, коли відповідні спостережувані дані змінюються. Це також гарантує, що компоненти не будуть перемальовуватись, якщо немає відповідних змін.

На практиці це робить програми, що використовують сховище MobX, добре оптимізованими одразу після інсталяції і простого налаштування, і їм, як правило, не потрібен додатковий код, щоб запобігти надмірному рендерингу. Також механізм підписки є набагато точнішим та ефективнішим порівняно з Redux, в якому залежності даних мають бути оголошені явно або попередньо обчислені.

1.2.3 Порівняльний аналіз MobX та Redux

Порівняльну характеристику глобальних сховищ наведено в таблиці 1.1.

Таблиця 1.1 — Порівняльна таблиця додатків для створення дизайну

Бібліотека	Кількість сховищ	Застосування	Масштабованість	Продуктивність
MobX	Можливість створення декілька глобальних сховищ	Для невеликих і простих додатків	Порівняно менш масштабований	Швидка зміна великої кількості даних
Redux	Можливість створення одного великого сховища	Для великих та комплексних сайтів	Створений для масштабованості в великих проектах	Можливі затримки при обробці великої кількості даних

Використання відразу декількох глобальних сховищ в MobX дозволяє логічно їх розділити, що в результаті забезпечує краще розуміння коду [10]. Так як веб-додаток структурного підрозділу навчального закладу є невеликим (порівняно, наприклад, з інтернет-магазином) і продуктивність глобального сховища MobX краща відносно сховища Redux, було вирішено використовувати MobX при проектуванні адміністративної панелі веб-додатку підрозділу навчального закладу.

Отже, глобальне сховище MobX зручне в використанні і не потребує додаткових налаштувань, дозволяє зберігати глобальний стан веб-додатку з логічним його поділом. Тому при проектуванні адміністративної панелі інформаційної системи структурного підрозділу навчального закладу буде використовуватись сховище MobX.

1.3 Огляд та обгрунтування вибору інструментів для розробки макету веб-інтерфейсу

Крім швидкодії сайту, наповнення його контентом та функціональними можливостями, які він виконує, не менш важливим фактором в сучасному світі

є його дизайн. Щоб абітурієнту, студенту чи викладачу було зручно користуватись веб-додатком, повинні дотримуватись принципи UI та UX. Крім того, є різні застосунки для створення дизайну, чи його прототипу: Figma, Adobe Photoshop, Adobe XD та ін.

Додаток Adobe Photoshop — це графічний редактор, який є найпопулярнішим додатком для редагування растрових зображень, цифрового малювання. Велика кількість самих різноманітних інструментів дозволяє використовувати цей інструмент не тільки для редагування фотографій чи малювання, а і для створення макетів веб-сайтів.

Інтерфейс самого редактору є достатньо складним і різноманітним. Враховуючи велику наповненість функціями різного спрямування, цей додаток використовує багато ресурсів комп’ютера, що є недоліком при виборі інструменту для створення дизайну.

Популярність Adobe Photoshop на ринку дизайну сайтів та мобільних додатків впала після випуску сучасніших інструментів для розробки дизайну.

Одним з таких інструментів є Figma — векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи та завантаження додатку на комп’ютер користувача. Принциповою різницею між Adobe Photoshop та Figma є направленість самого застосунку під конкретні задачі. В випадку Figma, це розробка додатків та веб-інтерфейсів.

Зручність полягає не тільки в створенні дизайну, але і в використанні цього дизайну уже в процесі розробки, що відсутнє в Adobe Photoshop. Також відносно мала кількість необхідних ресурсів комп’ютера для використання Figma впливає на швидкість розробки.

Також можливість створення анімацій прямо в дизайні, або ж повноформатний перегляд макету є ще одними перевагами онлайн-сервісу Figma.

При виборі інструменту для розробки дизайну були проаналізовані та випробувані ще такі додатки як Sketch та Adobe XD.

Перевагами Sketch можна зазначити його швидкодію, так як додаток працює на вбудованих засобах операційної системи, і зручність використання, що забезпечує високу продуктивність праці. Недоліками є можливість використання даного додатку лише на Mac OS та відсутність безкоштовної версії програми (тільки пробний період).

Додаток Adobe XD теж було порівняно з вищезазначеними варіантами. Суттєвою його перевагою є можливість створення та зручного керування станами компонентів дизайну, що допомагає показати розробнику необхідні анімації. Недоліками виступає відносно невеликий інструментарій по роботі з текстом та необхідність платної підписки для роботи [11].

Порівняння розглянутих інструментів для розробки макетів веб-інтерфейсів наведено таблиці 1.2.

Таблиця 1.2 — Порівняльна таблиця додатків для створення дизайну

Назва інструменту	Використання ресурсів	Швидкість відповіді програми	Інтерфейс	Вартість
Figma	~ 2GB+ оперативної пам’яті	Декілька мілісекунд	Інтуїтивний	Безкоштовно для одного користувача
Photoshop	70%+ доступної оперативної пам’яті	Довга застримка	Складний	Підписка, близько 240$ на рік
Sketch	4 GB пам’яті на диску; 2+ GB оперативної пам’яті	З затримкою	Інтуїтивний	Від 99$ перший рік, потім по 79$ за рік (пробний період 30 днів)
Adobe XD	2+ GB пам’яті на диску; 4 GB оперативної пам’яті	Помірна продуктивність	Інтуїтивний	Безкоштовний стартовий пакет, 120$ підписка

Отже, для розробки макету веб-клієнту інформаційної системи підрозділу навчального закладу прийнято рішення про використання інструменту Figma завдяки простоті вивчення, можливості колективної праці над макетом та наявності потрібних інструментів для дизайну.

1.4 Огляд аналогів веб-сайтів структурних підрозділів навчальних закладів

Перед безпосередньою розробкою дизайну веб-клієнту інформаційної системи були розглянуті і проаналізовані сайти інших структурних підрозділів навчального закладу. А саме це веб-сайти кафедри автоматизації та інтелектуальних інформаційних технологій, кафедри системного аналізу та інформаційних технологій та кафедри захисту інформації.

На рисунку 1.4 подана головна сторінка веб-інтерфейсу кафедри захисту інформації Вінницького національного технічного університету [12].

Даний сайт є достатньо інформативним та інтуїтивно зрозумілим. Однією з його проблем є не в повному обсязі виконана колірна палітра. Наприклад, колір фону навігації, контактів та категорій приємний на вигляд і підходить під тематику інформаційної системи, але заголовки новин використовують звичайний синій колір без акценту, який в даному випадку не є правильним вибором. Також, щоб перейти на внутрішні сторінки для додаткової інформації (наприклад, розклад занять або інформація першокурснику), потрібно спочатку перейти на сторінку «Студентам», але це являється зайвою дією.

Проаналізувавши веб-інтерфейс кафедри системного аналізу та інформаційних технологій ВНТУ, визначено його перевагу в колірній палітрі (рисунок 1.5). Кольори сайту приємні для перегляду і не викликають дискомфорту при читанні. Недоліком даного веб-інтерфейсу є його заплутаний дизайн і відразу не зрозуміло де знайти потрібну інформацію.

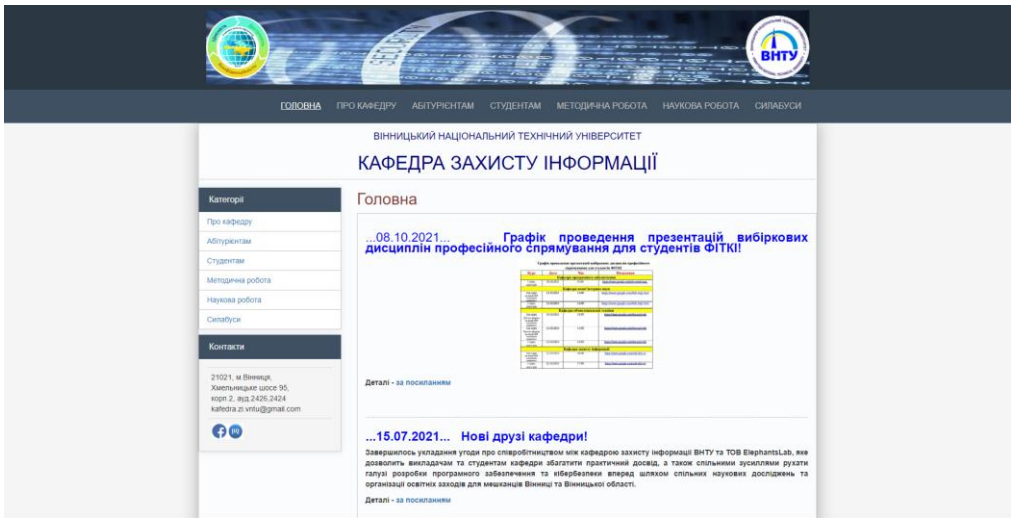

Рисунок 1.4 — Вигляд головної сторінки кафедри захисту інформації

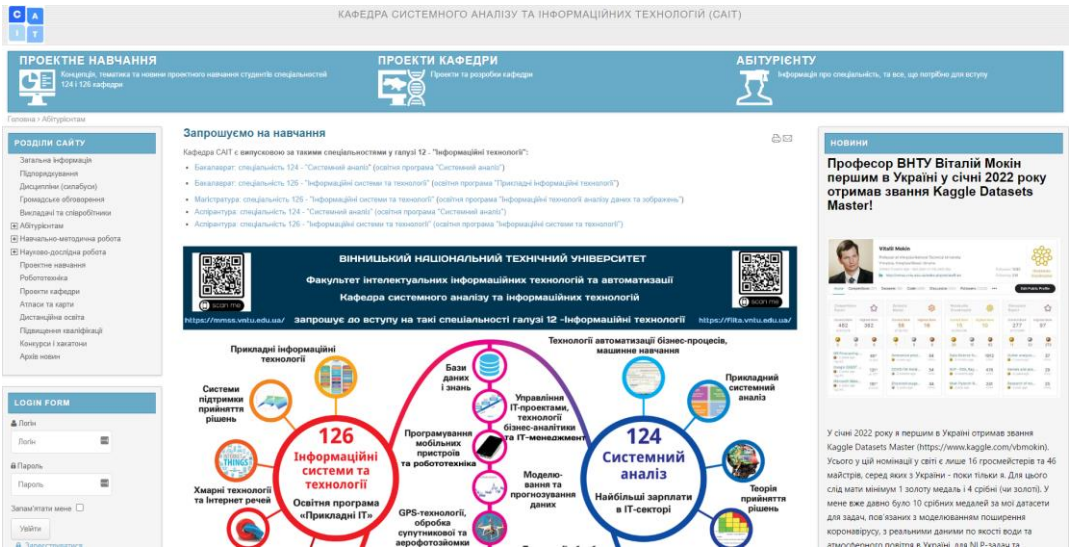

Рисунок 1.5 — Вигляд головної сторінки кафедри системного аналізу та інформаційних технологій

Веб-сайт кафедри автоматизації та інтелектуальних інформаційних технологій ВНТУ в цілому має добре підібрану колірну палітру та зрозуміле розташування потрібної інформації (рисунок 1.6). Недоліком такого веб-інтерфейсу є навігація, яка розділена на два рядки і неправильно підібрані розміри шрифтів в головному заголовку і основній навігації.

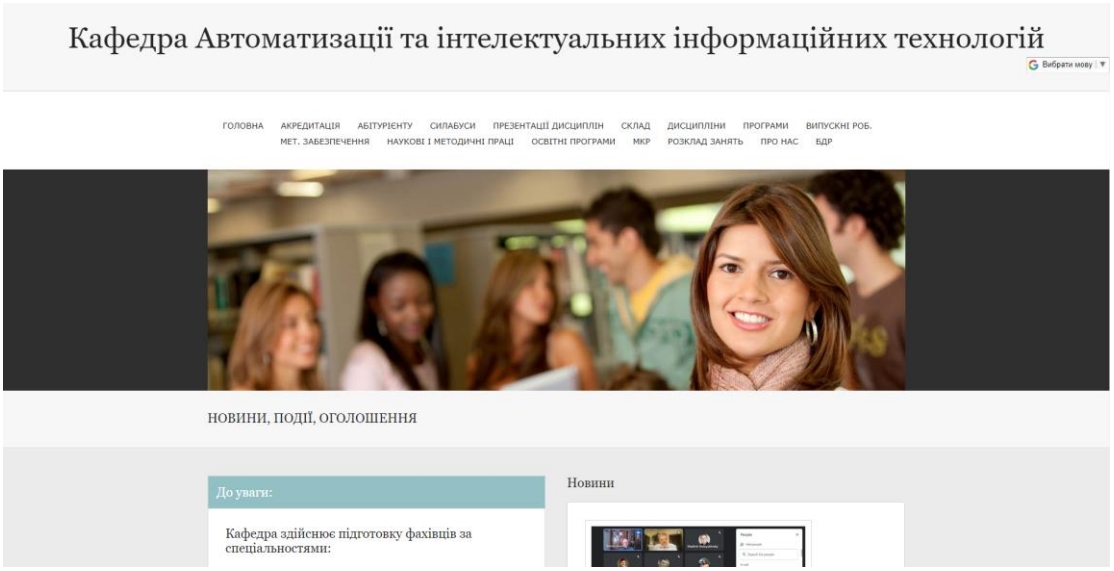

Рисунок 1.6 — Вигляд головної сторінки кафедри автоматизації та інтелектуальних інформаційних технологій

Отже, проаналізувавши аналогічні ресурси, можна зробити висновок про необхідність застосування правильно підібраної колірної палітри, розмірів шрифтів в дизайні та на сайті. Веб-інтерфейс повинен бути інтуїтивно-зрозумілим і відповідати сучасним вимогам, а основні елементи веб-сайту повинні розташовуватись в швидкому доступі.

2 РОЗРОБКА ВЕБ-ІНТЕРФЕЙСУ АДМІНІСТРАТИВНОЇ ПАНЕЛІ

2.1 Розробка макету адміністративної панелі структурного підрозділу навчального закладу

2.1.1 Загальні налаштування фрейму сторінки

Для створення дизайну веб-додатку використовувався веб-сервіс Figma, що дозволяє швидко та зручно, використовуючи наявні інструменти, створювати інтерфейс.

Після створення проекту в Figma виконується перехід в робочу зону для роботи над дизайном, де представлені основні інструменти.

Спочатку було додано фрейм, який є місцем для дизайну сторінки управління новинами.

На рисунку 2.1 представлені характеристики головного фрейму дизайну у вигляді CSS стилів.

```css
/* Головна сторінка */

position: relative;
width: 1920px;
height: 3207px;

/* grayWhite */
background: #FBFBFB;
```

Рисунок 2.1 — CSS стилі фрейму головної сторінки

З рисунку 2.1 видно, що задано ширину фрейму 1920 пікселів, тому що на даний час це найпопулярніший розмір монітору, тому дизайн потрібно виконувати саме з таким розміром. Також, як фоновий колір використано колірну змінну, що відповідає кольору #FBFBFB та виглядає приємнішим для сприйняття за звичайний білий колір.

2.1.2 Використання колірних змінних в Figma для створення макету інформаційної системи

Колірні змінні в Figma це дуже зручна можливість використання кольорів в дизайні. Замість того, щоб змінювати колір кожного окремого елементу, можна створити колірну змінну, назва якої буде відповідати кольору, та у всіх місцях застосування використовувати її. Це дасть змогу за кілька секунд змінювати палітру всього сайту для тестування різних колірних схем та підбору найбільш красивої чи відповідної тематиці [13].

Колірна палітра підібрана таким чином, щоб основні кольори та кольори акценту доповнювали один одного та в цілому створювали приємне враження від дизайну без зайвого дискомфорту при використанні інформаційної системи.

Колірні змінні, що використовуються в дизайні представлені на рисунку 2.2.

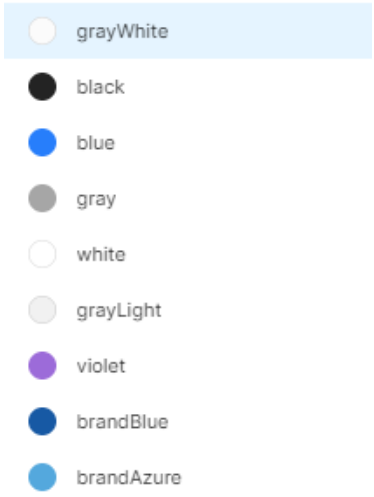

Рисунок 2.2 — Перелік створених колірних змінних

На основі колірної палітри та використовуючи колірні змінні, що показані на рисунку 2.2, був створений макет адміністративної панелі інформаційної системи (рисунок 2.3) структурного підрозділу навчального закладу.

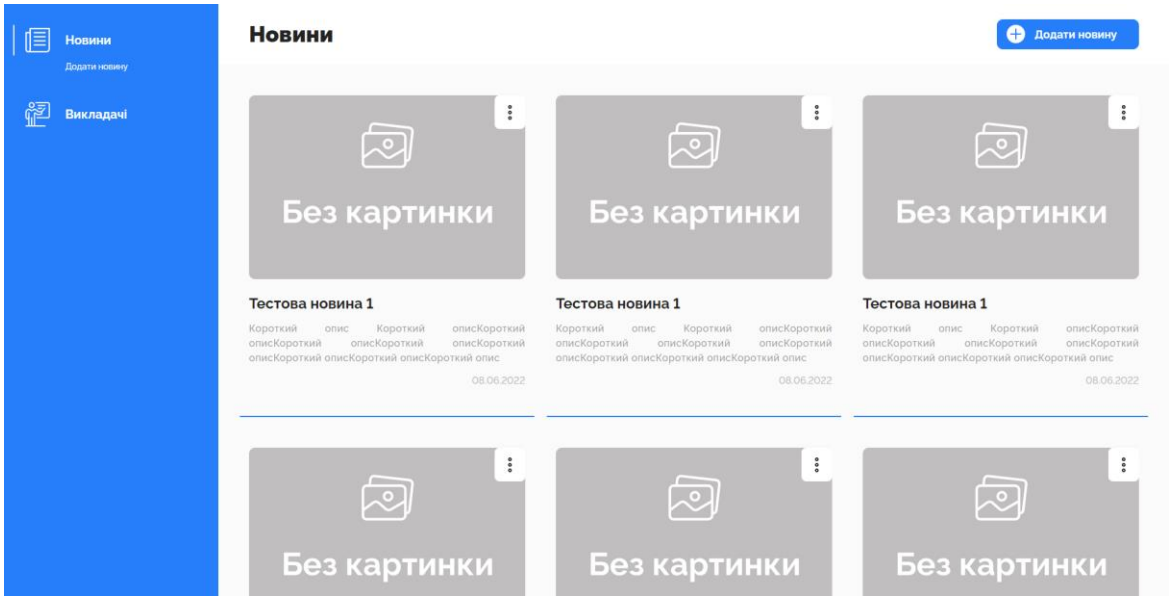

Рисунок 2.3 — Макет сторінки управління новинами

2.2 Обґрунтування вибору середовища розробки для створення адміністративної панелі

Перед початком розробки був проведений аналіз найпопулярніших рішеннь для розробки веб-сайтів, зокрема редактору VSCode та середовища JetBrains Webstorm.

Різниця в поняттях редактора і середовища досить велика. Редактор VSCode може конкурувати великою кількістю плагінів, за допомогою яких набуває всього необхідного функціоналу, якого йому не вистачає за замовчуванням. Зі встановленими плагінами різниця між VSCode та JetBrains Webstorm буде невеликою. Як і в середовищі, так і в редакторі є можливість використання вбудованих терміналів, що також дозволяє виконувати потрібні консольні команди [15].

Але для розробки панелі адміністратора було обрано інтегроване середовище розробки JetBrains Webstorm. Воно створене на основі платформи IntelliJ IDEA та зазвичай використовується для написання коду на JavaScript чи верстки за допомогою HTML та CSS. Такий вибір був зроблений на основі

наявності великої кількості готових рішень, які є в даному середовищі, доволі зручним Intellisense та широкими можливостями налаштування.

2.3 Програмна реалізація адміністративної панелі для веб-клієнта підрозділу навчального закладу

Використовуючи створений дизайн адміністративної панелі, розроблені такі веб-сторінки, як управління новинами та управління викладачами. Також розроблений необхідні функціональні можливості для зручного додавання нового інструментарію по керуванню іншими даними інформаційної системи, наприклад, стейкхолдерами і методичними розробками.

Програмну реалізацію веб-додатку показано на прикладі управління новинами.

2.3.1 Розробка фізичної структури проекту

Важливу роль в проекті відіграє його структура. Її задача полягає в тому, щоб новий розробник, який почав працювати з створеним проектом, інтуїтивно розумів розташування ключових та інших файлів та міг без проблем і необхідної консультації почати роботу.

Фізична структура проекту складається з каталогів (рисунок 2.4), кожен з яких грає важливу роль в масштабованості та взаємодії між React компонентами:

— node_modules, каталог з великою кількістю бібліотек, які використовуються для налаштування React-додатку, запуску проекту, використання самого React в коді;

— public, каталог, в якому знаходяться статичні файли веб-сайту, такі як index.html чи іконка сайту favicon.ico;

— src, головний робочий каталог, в якому знаходяться файли розробки;

— assets, каталог, в якому зберігаються файли оформлення веб-додатку: загальні стилі, картинки та іконки, шрифти;

— core, каталог для зберігання основних для веб-додатку компонентів чи функцій;

— functions, каталог, що потрібен для зберігання часто повторюваних функцій;

— hooks, каталог, в якому зберігаються власноруч написані додаткові React хуки;

— ui, каталог, який призначений для компонентів сайту, що використовуються багато або декілька раз;

— modules, каталог з загальними компонентами сторінок панелі адміністратора;

— routes, каталог з файлом MainRoutes.js, в якому описана реалізація React-роутингу і перенаправлення в випадку неіснуючої сторінки;

— schemas, каталог для зберігання файлів GraphQL схем;

— store, каталог з файлами MobX, що дозволяють створити та керувати глобальним сховищем;

— utils, каталог, в якому знаходяться допоміжні функції.

2.3.2 Налаштування Apollo Client для бібліотеки React

Перед початком безпосередньої роботи над панеллю адміністратора, потрібно налаштувати Apollo Client для забезпечення взаємодії між клієнтом та сервером.

Бібліотеку завантажено за допомогою команди npm install @apollo/client graphql, де graphql — пакет, що забезпечує логіку парсингу GraphQL запитів.

Після завантаження бібліотеки та додаткового пакету в файлі index.js імпортуються необхідні залежності (лістинг 2.1) та ініціалізується Apollo Client, після передачі його конструктору об’єкта з конфігураційними полями (лістинг 2.2).

Лістинг 2.1 — Імпортування необхідних залежностей

```
import { ApolloClient, InMemoryCache, ApolloProvider } from
'@apollo/client'
```

Лістинг 2.2 — Ініціалізація Apollo Client

```
const client = new ApolloClient({
    uri: 'https://graphql.slivki-cat.com',
    cache: new InMemoryCache(),
})
```

В лістингу 2.2 використовуються параметри, як uri та cache, де:

— uri визначає url-адресу сервера GraphQL;

— cache є екземпляром класу InMemoryCache, який Apollo Client використовує для кешування результатів запитів після їх безпосереднього отримання.

Це достатньо для стандартного налаштування Apollo Client для можливості роботи з GraphQL запитами. Після цього компонент App обгорнуто компонентом ApolloProvider та створений раніше екземпляр client переданий в цей компонент за допомогою props, як показано в лістингу 2.3.

Лістинг 2.3 — Обгортка для компоненту App

```
<React.StrictMode>
    <Router>
        <ApolloProvider client={client}>
            <App/>
        </ApolloProvider>
    </Router>
</React.StrictMode>
```

2.3.3 Реалізація авторизації та додавання ролі адміністратора

Можливість використання адміністративної панелі є тільки у ролі адміністратора. Така роль спершу визначається розробником для створеного спеціального користувача. Після цього користувач набуває ролі адміністратора і при додаванні інструментарію з управління ролями інших зареєстрованих викладачів або студентів буде мати можливість додавати інших адміністраторів.

На сторінці входу створені два поля для введення електронної пошти та паролю. При неправильному введенні даних одного з полів буде висвітлена помилка з інформацією про невдалий вхід.

2.3.4 Створення сторінки керування новинами та додавання роутингу

Для створення сторінки керування новинами потрібно для початку створити відповідний компонент за допомогою скорочення rsc (рисунок 2.4) та додати для нього роутинг.

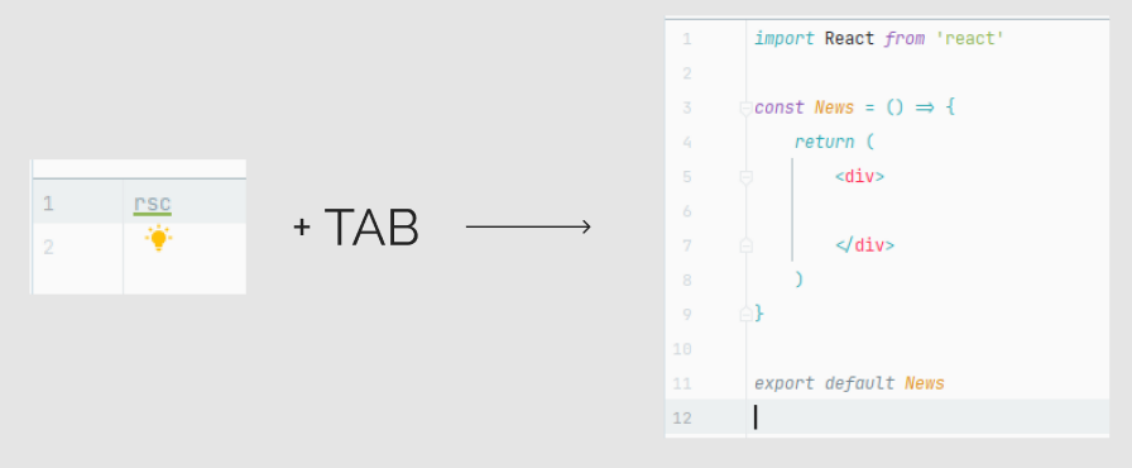

Рисунок 2.4 — Створення компонента сторінки за допомогою скорочення rsc

В файлі MainRoutes.js, який був створений при проектуванні структури сайту, додається стрічка, що буде відповідати за відображення сторінки керування новинами. Як параметр path записано «/news» для відображення компоненту керування новинами, якщо в адресну стрічку буде доданий такий шлях. Також в параметр element передається компонент, який буде відображуватись, якщо шлях співпаде з параметром path.

Для створення перенаправлення на конкретну сторінку, якщо в адресну стрічку був введений неіснуючий шлях, використовується ще одна стрічка з параметром path, який дорівнює символу зірочки та в параметр element передано імпортований з бібліотеки react-router-dom компонент Navigate [13].

Код цієї реалізації показано в лістингу 2.4:

Лістинг 2.4 — Базове налаштування роутингу

```
<Routes>
    <Route path='/news' element={<News/>}/>
    <Route path='*' element={<Navigate to='/news'
replace/>}/>
</Routes>
```

Цей компонент може приймати параметр to, в якому вказується шлях, на який буде створено перенаправлення. Параметр replace приймає лише true або false і потрібен для того, щоб очистити історію від лишнього неправильного переходу, після чого перехід на минулу сторінку (використовуючи стрілку браузеру) відбудеться без зайвого перенаправлення. Якщо записати тільки назву параметра без значення, то за замовчуванням він буде дорівнювати true.

2.3.5 Отримання новин з бази даних за допомогою схеми GraphQL

Для того, щоб отримати дані з бази даних, створено файл зі схемою GraphQL в каталозі schemas (лістинг 2.5).

Лістинг 2.5 — Схема для отримання новин

```
export const GET_NEWS = gql`
    query {
        getNews {id
            title
            description
            content
            dateTime
            imageStorageUrl
            imageName
    }
`
```

В лістингу 2.5 використовується спеціальний літерал gql, в який записується сама схема, вказавши тип запиту query, тому що потрібно тільки отримати дані. Аналогом query в REST API є GET запит. В даному випадку не потрібно вказувати ніяких параметрів, а просто написати назву потрібної схеми — getNews та поля, які потрібно отримати:

— id, ідентифікатор новини;

— title, заголовок;

— description, стислий опис новини;

— content, розгорнутий опис, який буде відображатись на сторінці окремої новини;

— dateTime, дата створення новини;

— imageStorageUrl, посилання на картинку, що збережена у хмарному сховищі;

— imageName, назва зображення, щоб записати її в атрибут alt тегу img.

2.3.6 Використання в проекті хука useQuery

Для використання GraphQL схеми, описаної вище, використовується хук useQuery, що імпортується з бібліотеки @apollo/client та аргументом передається імпортована схема getNews. За допомогою деструктуризації з об’єкту, що повертає даний хук, отримуємо поля:

— loading, потрібно для асинхронного відстеження процесу виконання запиту, повертає true або false;

— error, необхідно для відстеження помилки в запиті, та відображення її тексту;

— data, дані, які отримуються з бази даних.

Для коректного використання поля data, перед поверненням розмітки (рендером) задані дві умови.

Першою є перевірка на те, чи запит виконався, чи ще в процесі виконання. Для цього використано поле loading, яке отримано раніше. Якщо запит ще не виконався, то необхідно рендерити текст завантаження, або ж візуальний завантажувальник, який є кращим рішенням.

Друга перевірка стосується обробки помилки, якщо така виникне при виконанні запиту. В цьому випадку також повертається відповідний блок з

текстом помилки, або ж в іншому вигляді, якщо є необхідність унікального відображення.

Тільки після двох перевірок описується основний рендер компонента, як показано на рисунку 2.5.

```
7   export const News = () => {
8       const {data, loading, error} = useQuery(GET_NEWS)
9
10      if (loading) return <p>Loading...</p>
11      if (error) return <p>Error: {error}</p>
12
13      return (
14          <div>
15
16          </div>
17      )
18  }
```

Рисунок 2.5 — Використання хука useQuery в коді компонента

2.3.7 Виведення отриманих новин в панелі адміністратора

Отримавши дані в полі data, що являє собою об’єкт, в якому є масив новин getNews, необхідно їх відобразити на сайті.

Спершу створено новий компонент за шаблоном, що описувався вище, який названо NewsItem. Щоб використовувати його в основному компоненті сторінки, необхідно імпортувати цей компонент, та написати його в повернутому JSX (розширення синтаксису для JavaScript).

Так як необхідні дані описані в вигляді масиву, використано JSX синтаксис для того, щоб вивести на сайт всі новини, які отримано з бази даних. Цю задачу вирішує один з методів масивів, map.

Щоб передати всі дані конкретної новини в компонент NewsItem для подальшого їх використання, написано параметр data, в який передається об’єкт ітерації (newsItem). Для параметру key використано поле id, що є в

кожній новині. Код компонента сторінки керування новинами поданий на рисунку 2.6.

```
return (
    <PageWrapper title={'Новини'} addNew>
        {
            loadingData ? <img src={loader} alt='loader'/> : (
                NewsStore.news.length > 0 ? (
                    <div className={styles.news}>
                        {
                            NewsStore.news.map(newsItem => (
                                <NewsItem data={newsItem} key={newsItem.id}/>
                            ))
                        }
                    </div>
                ) : (
                    <div className={styles.newsNoItems}>
                        -- немає новин --
                    </div>
                )
            )
        }
        <Pagination info={paginationInfo} />
    </PageWrapper>
)
```

Рисунок 2.6 — Код компонента керування новинами

2.3.8 Додавання нової новини в базу даних

Для додавання нової новини потрібно натиснути на кнопку «Додати новину». Після цієї дії виконається перенаправлення на сторінку додавання новини зі зручним та інтуїтивно зрозумілим інтерфейсом. Програмна реалізація даної сторінки показана на рисунку 2.7.

Кнопка «Відмінити» потрібна для відміни додавання новини та повернення на сторінку назад.

При завантаженні фотографії, з’явиться її попередній перегляд, для цього використовується додатковий компонент UploadFile (рисунок 2.8). Якщо була вибрана не та фотографія, то замінити її можливо повторним завантаженням.

```
return (
    <PageWrapper title={'Додати новину'}>
        <div className={styles.addNew}>
            <Form className={styles.addNewForm} onSubmit={checkInputs}>
                <Input type={'text'} label={'Назва новини'} id={'newName'} {...inputName} error={errors.newName}/>
                <Input type={'text'} label={'Короткий опис'} id={'shortDesc'} {...inputDesc}
                       error={errors.shortDesc}/>
                <Input label={'Детальний опис'} id={'fullDesc'} textarea {...textareaDesc} error={errors.fullDesc}/>
                <UploadFile className={styles.addNewUpload} label={'Прикріпити фото'} setFile={setFile}/>
                <div className={styles.addNewActions}>
                    <Btn
                        className={styles.addNewActionsBtn}
                        type={'button'}
                        onClick={() => navigate('/news')}
                        red
                        disable={addingNew}
                    >
                        Відмінити
                    </Btn>
                    <Btn className={styles.addNewActionsBtn} type={'submit'} disable={addingNew}>
                        Додати
                    </Btn>
                </div>
            </Form>
        </div>
    </PageWrapper>
)
```

Рисунок 2.7 — Код компонента додавання новини

```
return (
    <div className={cn( args: {
        [styles.upload]: true,
        [className]: className,
    })}>
        <div className={styles.uploadTitle}>{label}</div>
        <div className={styles.uploadInner}>
            <label className={styles.uploadWrapper}>
                <div>Завантажити</div>
                <input className={styles.uploadInp} type='file' onChange={handleUpload}
                       accept='image/png, image/jpeg'/>
            </label>
            {
                src && (
                    <button className={styles.uploadRemove} type={'button'} onClick={removeImage}>
                        <img src={remove} alt='remove'/>
                    </button>
                )
            }
        </div>
        {src && <img className={styles.uploadPreview} src={src} alt='preview'/>}
    </div>
)
```

Рисунок 2.8 — Код компонента завантаження зображення

Для кодового виконання функції додавання необхідно використовувати хук useMutation. Якщо useQuery потрібен для отримання даних з серверу та є аналогом до методу GET в REST API, то даний хук призначений для змінення даних, та може виступати як методом POST, так і методом DELETE, в залежності від конкретної задачі.

Для використання хука також потрібна створена GraphQL схема з відповідними відмінностями від схеми для query запиту (лістинг 2.7). Створено дві аналогічні схеми, що будуть мати різні назви, тобто modifyNew та addNew, але будуть розділені по місцю використання та своєму сенсу. Ще одна відмінність схеми modifyNew в тому, що параметри, які будуть передаватись в об’єкті, не є обов’язковими і через це з’являється можливість змінювати лише потрібні поля: змінити лише картинку, чи заголовок, чи повністю всю новину.

Лістинг 2.7 — GraphQL схема додавання новини

```
import gql from 'graphql-tag'
export const ADD_NEW = gql`
    mutation ($title: String!, $description: String!,
$content: String!, $image_url: String) {
        add_new(objects:{
            title: $title,
            description: $description,
            content: $content,
            image_url: $image_url,
        }){
            id
        }
    }
`
```

В лістингу 2.7 показано, що аргументами для mutation є поля з назвами параметрів в яких дотримується аналогія з назвами при передачі параметрів в хуці useMutation.

Використання useMutation в компоненті подано на рисунку 2.9.

По аналогії з хуком useQuery, аргументом хука передається GraphQL схема. Після рендеру компонента useMutation повертає кортеж, серед якого є функція для мутації даних, яка використовується в будь-який час та в будь-якому місці цього компонента. Це є перевагою даного хука над useQuery, хоча і виконують вони різні операції. Другим елементом кортежу є об’єкт, що містить ті ж поля, що і попередній хук. Тобто можливо виконувати перевірку на час виконання запиту, а також обробку помилок при необхідності.

```
const [addNewHook, {loading: addingNewLoading}] = useMutation(ADD_NEW)

const [newData, setNewData] = useState( initialState: {})

const addNew = async () => {
    await addNewHook( options: {
        variables: {
            title: newData.title,
            description: newData.description,
            content: newData.content,
            image_url: newData.image_url,
        }
    })
}
```

Рисунок 2.9 — Додавання новини через хук useMutation

Для зберігання та зміни стану компонента, використано React хук useState. В даному випадку він потрібен для зберігання та оновлення даних, які були введені адміністратором для додавання новини в веб-інтерфейсі.

При натисканні кнопки «Додати» та обробки помилок (якщо було введено не всі дані) викликається функція addNew, в якій викликається потрібна функція, повернена з useMutation. Її аргументом є об’єкт параметрів, одним з яких є поле variables, яке, в свою чергу, теж приймає ще один об’єкт з аргументами, необхідними для відправлення схеми і додавання новини в базу даних.

Функція є асинхронною, тому після її виконання є можливість маніпулювати з повернутими даними, якщо є така необхідність в задачі. В нашому випадку res — це об’єкт, в якому є поле id, яке було згенеровано після створення новини.

Якщо все було зроблено вірно та не виникла помилка, отримаємо візуальне повідомлення в верхньому правому кутку, яке проінформує про успішність додавання новини.

Отже, в даному розділі був описаний процес створення проекту, налаштування його структури та використання кожного з каталогів. На

прикладі кількох сторінок управління новинами був показаний процес розробки, взаємодія з серверною частиною за допомогою GraphQL запитів та налаштування Apollo Client для коректної їх роботи, використання хуків useQuery та useMutation.

3 ОПИС ОСНОВНИХ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ВЕБ-ІНТЕРФЕЙСУ АДМІНІСТРАТИВНОЇ ПАНЕЛІ

Основними інструментами адміністративної панелі є додавання даних, їх редагування, перегляд та видалення. Програмно розроблено можливість додавання сторінок для керування іншими структурами даних, таких як ролі, що видаються при авторизації, методичні розробки та стейкхолдери.

Основний інструментарій розглянуто на прикладі сторінки управління новинами.

3.1 Інструментарій на сторінці управління новинами

В адміністративній панелі структурного підрозділу навчального закладу було вирішено зробити сторінкою за замовчуванням сторінку управління новинами (рисунок 3.1).

Для розуміння, на якій сторінці знаходиться адміністратор веб-додатку — додано динамічний заголовок, який змінюється при переході на різні сторінки. Крім того на цій сторінці для зручності додана кнопка для добавлення нової новини (також є можливість використати посилання в меню).

В кожній новині представлена загальна інформація про неї: зображення (якщо вона є), заголовок, короткий опис новини і дата її добавлення в базу даних або дата редагування. Детальний опис новини знаходиться на сторінці її перегляду. Також в верхньому правому кутку знаходиться кнопка, при натисканні на яку з’являються опції для маніпулювання конкретною новиною (рисунок 3.2):

— кнопка для переходу на сторінку перегляду новини;

— посилання на сторінку редагування новини;

— кнопка видалення новини (при натисканні з’явиться просте меню підтвердження);

— кнопка для закриття опцій.

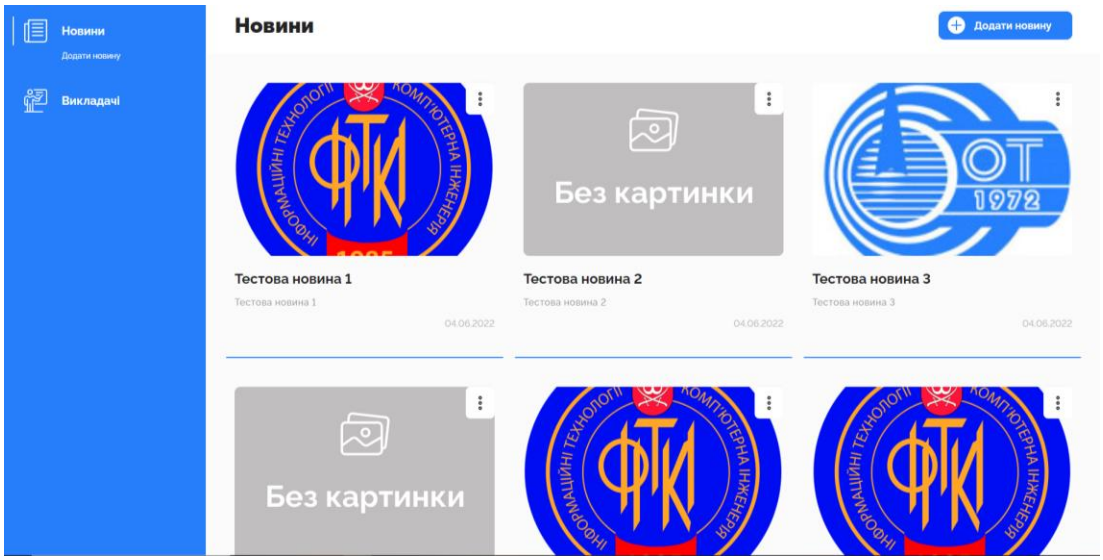

Рисунок 3.1 — Візуальний інтерфейс сторінки перегляду новин

Рисунок 3.2 — Представлення опцій для керування новиною

Відкриття, закриття та інші дії супроводжуються анімаціями для зручності роботи.

3.2 Сторінка додавання новини

Сторінка додавання новини складається з декількох полів для введення назви новини, інформації, завантаження картинки і кнопками для додавання новини або відміни (рисунок 3.3).

Поля «Назва новини», «Короткий опис» та «Детальний опис» є обов’язковими. Задля запобігання помилкам створена перевірка на введені дані і якщо їх не ввести та натиснути на кнопку «Додати» — новина не буде додана, але з’являться помилки з допоміжним текстом опису причини цих помилок (рисунок 3.4).

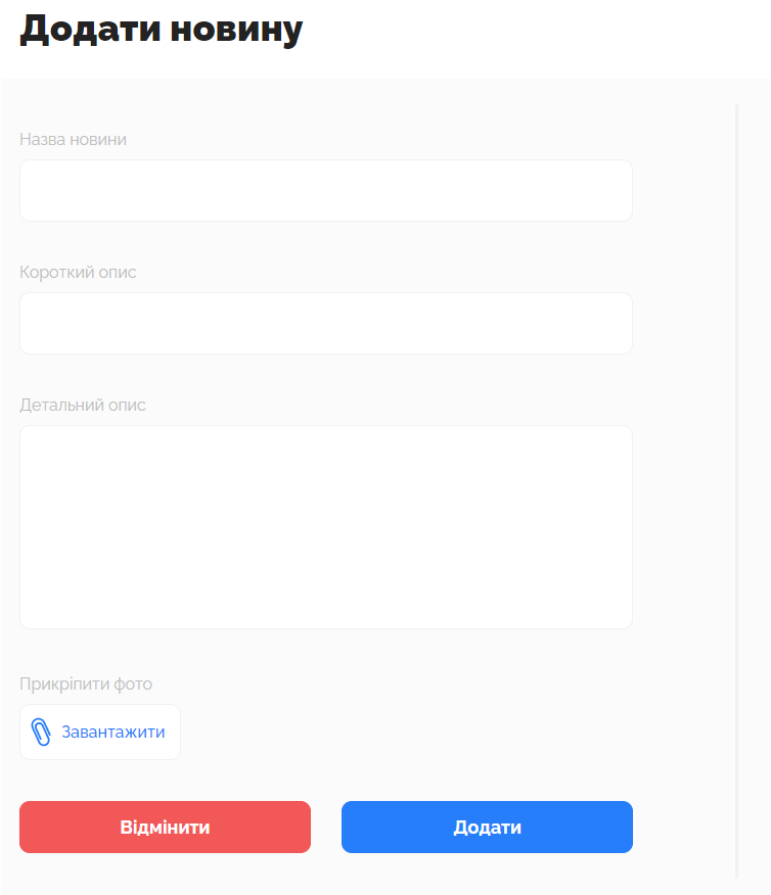

Рисунок 3.3 — Візуальний інтерфейс форми додавання новини

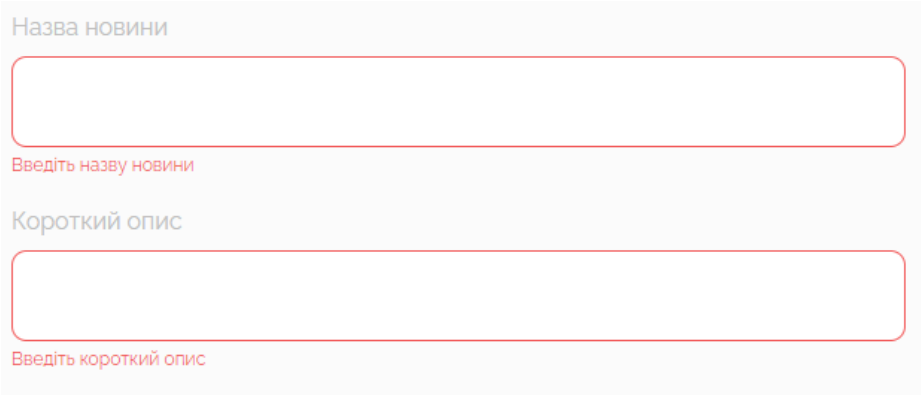

Рисунок 3.4 — Вигляд помилок при пустих полях

При натисканні на кнопку «Завантажити» можна додати фотографію до новини. За допомогою атрибута accept для текстового поля з файловим типом вибрані розширення файлів, які можливо загрузити, а саме png та jpeg/jpg.

Після завантаження вибраної фотографії з’являється її попередній перегляд, як показано на рисунку 3.5, та кнопка видалення цієї фотографії.

Якщо після успішного завантаження зображення вибрати нову фотографію, зображення заміниться.

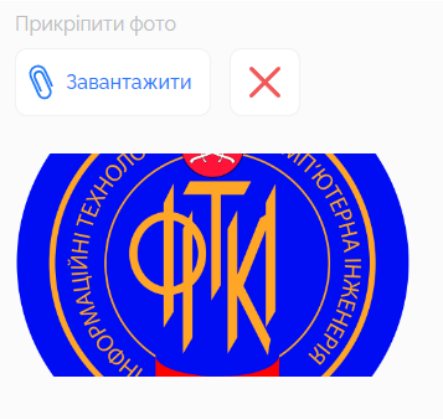

Рисунок 3.5 — Попередній перегляд завантаженої фотографії

Коли без помилок введені необхідні дані для новини та завантажене зображення, можна натискати на кнопку «Додати». Відразу після натискання кнопки стають неактивними до моменту остаточного виконання запиту. Якщо запит проходить без проблем та новина додається — всі текстові поля очищуються.

3.3 Сторінка редагування новини

Після натискання на кнопку редагування новини в меню її опцій (на сторінці перегляду новин) виконується перехід на форму редагування (рисунок 3.6). Основними відмінностями від сторінки додавання є заголовок і заповнені поля форми поточними даними. В заголовку вказується ідентифікаційний номер новини, що редагується. Це зроблено для кращого розуміння, яка саме новина редагується і, в разі необхідності, для можливості порівняння ідентифікаційного номер з новиною, що знаходиться в базі даних.

Для редагування тексту в відповідному полі вводиться його нове значення. Щоб змінити зображення, необхідно загрузити нове, або видалити в випадку його непотрібності.

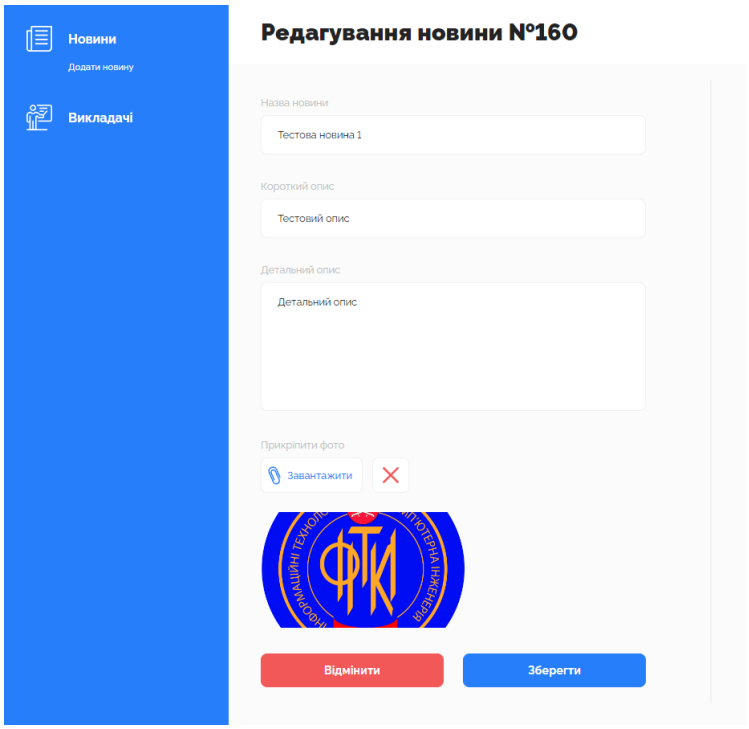

Рисунок 3.6 — Візуальне представлення сторінки редагування новини

Після редагування також виконується перевірка полів форми на помилки і, як відмінність від сторінки додавання, автоматичний перехід на сторінку перегляду новин.

3.4 Реалізація додаткових функціональних можливостей адміністративної панелі

3.4.1 Застосування пагінації на сторінці керування новинами

Пагінація — можливість розділяти велику кількість однотипної інформації на різні сторінки. Це робиться для оптимізації додатку та зручнішого пошуку необхідної інформації, в даному випадку — новин.

В адміністративній панелі пагінація представляється у вигляді посилань на номер конкретної сторінки.

Також в API додана можливість за допомогою параметра вказувати кількість однотипних елементів на одній сторінці, що дає можливість зручно маніпулювати відображенням новин. Це робиться в коді компонента сторінки новин, при надсиланні запиту для їх отримання з бази даних.

Крім того, пагінація надає можливість використання посилань на конкретну сторінку новин.

3.4.2 Демонстрація сповіщень при інтерактивності з веб-додатком

Для кращої взаємодії веб-додатку з користувачем або адміністратором було вирішено створити різні типи сповіщень, які з'являються при деякій інтерактивності.

Якщо на сторінці перегляду новин в відкритому меню опцій конкретної новини натиснути кнопку видалення та підтвердити його — новина видалиться зі сторінки та з бази даних, використовуючи REST API. Також на 3 секунди в верхньому правому кутку з'явиться повідомлення про успішне видалення цієї новини, щоб дати адміністратору зрозуміти, що видалення відбулось (рисунок 3.7). У разі виникнення помилки в параметрах запиту або ж помилки на сервері сповіщення буде іншого типу, а саме помилкового (рисунок 3.8).

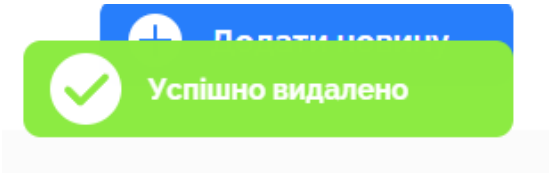

Рисунок 3.7 — Сповіщення успішного видалення новини

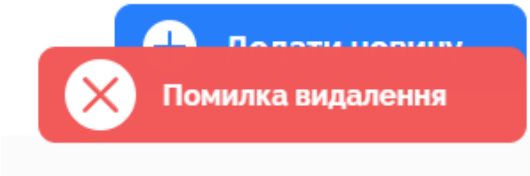

Рисунок 3.8 — Сповіщення виникнення помилки при видаленні новини

Ще один тип сповіщення, який інформує адміністратора про те, що відбувається завантаження, яке потребує часу (рисунок 3.9). Таке сповіщення виникає при додаванні новини в базу даних, якщо була завантажена картинка.

Завантаження зображеня в хмару займає деякий час, тому таке сповіщення необхідне для розуміння процесу виконання цього завантаження.

Програмно додана можливість вписати час демонстрації конкретного сповіщення. Наприклад, якщо вказати 3 секунди, то сповіщення з’явиться і після того, як пройдуть ці 3 секунди, воно зникне. В випадку сповіщення про завантаження, час не вказується, воно зникне лише тоді, коли завантаження пройде успішно або виникне помилка.

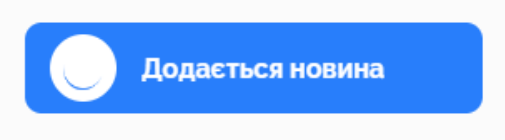

Рисунок 3.9 — Сповіщення, що з’являється при завантаженні зображення

На рисунку 3.10 продемонстровано ситуацію, коли з’являються декілька сповіщень одночасно, вони вишиковуються в колонку та з’являються з простою анімацією, яка не дуже відволікає адміністратора.

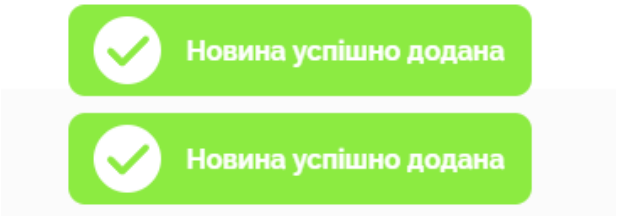

Рисунок 3.10 — Візуальне представлення декількох сповіщень одночасно

3.5 Аналіз необхідного апаратного та програмного забезпечення для оптимальної роботи адміністративної панелі

Проведено аналіз апаратного та програмного забезпечення, які не тільки зручно використовувати при розробці, а і забезпечують хорошу оптимізацію роботи інформаційної системи.

Для оптимальної роботи адміністративної панелі комп’ютер повинен мати не менше 512Мб оперативної пам’яті. При виборі процесора потрібно

звернути увагу на як мінімум 2 ядра та 2 потоки, тобто, наприклад, Intel Pentium Gold і вище.

Як операційну систему для функціонування розробленої інформаційної системи та адміністративної панелі зокрема, можна використовувати MacOS, Windows, Linux.

Для забезпечення комфортної швидкості виконання запитів швидкість Wi-Fi або кабельного підключення повинна бути не менше 5Мбіт/сек.

ВИСНОВКИ

У комплексному бакалаврському проекті було реалізовано адміністративну панель інформаційної системи структурного підрозділу навчального закладу та реалізовано клієнт-серверну взаємодію зі сторони клієнтської частини.

Проведений огляд додатків і сервісів для створення прототипу дизайну веб-інтерфейсу та кінцевого макету веб-додатку, наведена порівняльна таблиця. Проаналізовано сучасні технології для організації клієнт-серверної взаємодії та обґрунтовано вибір оптимального стеку технологій для розробки адміністративної панелі інформаційної системи структурного підрозділу навчального закладу.

Клієнт-серверну взаємодію реалізовано з використанням технологій Apollo Client та REST API. Технологія Apollo Client дозволила використання GraphQL схем, які застосовано для отримання інформації з серверу та гнучкого налаштування запрошених даних. Також за допомогою стандартної технології REST API виконані запити на додавання інформації в базу даних, її редагування та видалення.

В веб-додатку розроблено авторизацію, за допомогою якої забезпечено безпечне керування веб-сайтом інформаційної системи структурного підрозділу навчального закладу тільки при наявності ролі адміністратора.

Для оптимізації швидкодії веб-додатку було використано пагінацію — розділення великої кількості повторюваних даних на різні сторінки, що забезпечило відображення однотипних елементів на одній сторінці в кількості, яка програмно задається при GET запиті.

При інтерактивності з веб-додатком розроблено просту систему сповіщень. Після виконання запитів на отримання, редагування чи видалення даних, з’являються сповіщення, які інформують адміністратора про результат виконаної дії. Якщо виконується довготривалий запит на додавання

зображення в хмарне середовище, то з’являється завантажувальник, який зникає після вдалого виконання дії або виникнення помилки.

Комплексна бакалаврська дипломна робота розроблена з наданням можливостей зручного та зрозумілого розширення інструментарію адміністративної панелі веб-додатку. Реалізована інтуїтивно зрозуміла фізична структура адміністративної панелі, що дозволить новим розробникам масштабувати веб-клієнт.

Результати комплексної бакалаврської дипломної роботи впроваджено мережі ВНТУ як сайт кафедри обчислювальної техніки (додаток П).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Аналіз методів клієнт-серверної взаємодії при розробці панелі адміністратора за допомогою реактивного фреймворка React [Електронний ресурс] / В.О. Никитюк, О.В. Войцеховська — 2022. — https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15104.

2. Що таке UI та UX дизайн? [Електронний ресурс]. Режим доступу: https://te.itstep.org/blog/ui-and-ux-design.

3. Introduction to Apollo Client [Електронний ресурс]. Режим доступу: https://www.apollographql.com/docs/react/.

4. The benefits of ReactJS [Електронний ресурс]. Режим доступу: https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html.

5. Fetch API [Електронний ресурс]. Режим доступу: https://www.jscamp.app/docs/javascript27/.

6. React і Redux: функціональна веб-розробка. Алекс Бенкс, Єва Порселло 2018р. 336с.

7. Flux in-depth overview [Електронний ресурс]. Режим доступу: https://facebook.github.io/flux/docs/in-depth-overview/.

8. MobX introduction [Електронний ресурс]. Режим доступу: https://mobx.js.org/README.html.

9. Патерн Спостерігач [Електронний ресурс]. Режим доступу: https://refactoring.guru/uk/design-patterns/observer.

10. Redux vs MobX [Електронний ресурс]. Режим доступу: https://www.educba.com/mobx-vs-redux/.

11. Photoshop vs. Sketch vs. Adobe XD vs. Figma [Електронний ресурс]. Режим доступу: https://belovdigital.agency/blog/photoshop-vs-sketch-vs-adobe-xd-vs-figma-design-apps-comparison/.

12. Кафедра захисту інформації ВНТУ [Електронний ресурс]. Режим доступу: https://zi.vntu.edu.ua/.

13. Create color, text, effect and layout grid styles [Електронний ресурс]. Режим доступу: https://help.figma.com/hc/en-us/articles/360038746534-Create-color-text-effect-and-layout-grid-styles.

14. VS Code vs. WebStorm — A Detailed Comparison [Електронний ресурс]. Режим доступу: https://swimm.io/blog/vscode-vs-webstorm-a-detailed-comparison/.

15. React Router Docs [Електронний ресурс]. Режим доступу: https://reactrouter.com/docs/en/v6.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
_____ проф., д.т.н. О.Д. Азаров
«__» _____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання комплексної бакалаврської дипломної роботи на тему:
«Клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 3. «Розробка панелі адміністратора за допомогою реактивного фреймворка React»
08-23.КБДР.013.00.000 ПЗ

Науковий керівник: к.т.н.,доц. каф. ОТ
_____ Семеренко В.П.

Виконав: студент групи КІ-20мсз
_____ Никитюк В.О.

Вінниця 2022 р.

1 Підстава для виконання дипломної роботи

Підстава для виконання дипломної роботи (ДР): актуальність досліджень полягає у необхідності розробки інтерактивної адміністративної панелі інформаційної системи структурного підрозділу навчального закладу, створенні нового дизайну веб-інтерфейсу та реалізації керування даними та наказ про затвердження теми дипломної роботи.

2 Мета ДР і призначення розробки

Мета ДР — розробка адміністративної панелі клієнт-серверної інформаційної системи структурного підрозділу навчального закладу. Призначення ДР полякає в виконанні комплексної бакалаврської дипломної роботи для подальшого використання.

3 Вихідні дані для виконання ДР

Вихідними даними для виконання ДР є веб-сайт кафедри обчислювальної техніки.

4 Технічні вимоги до виконання ДР

Технічними вимогами до виконання ДР є контент для наповнення веб-додатку, дані по технологіям розробки клієнтської частини та дані для створення макету веб-інтерфейсу.

5 Етапи ДР та очікувані результати приведені в таблиці А.1.

Таблиця А.1 — Етапи ДР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз веб-інтерфейсів, виділення плюсів та мінусів	09.02.22	24.02.22	Розділ 1
2	Аналіз технологій для розробки веб-додатку	25.02.22	05.03.22	Розділ 1

3	Вибір колірної палітри та створення прототипу веб-сайту	06.03.22	20.03.22	Розділ 2
4	Створення дизайну веб-інтерфейсу	21.03.22	05.04.22	Розділ 2
5	Розробка фізичної структури веб-додатку	06.04.22	10.04.22	Розділ 2
6	Верстка сторінок веб-сайту	11.04.22	02.05.22	Розділ 3
7	Програмна реалізація та налаштування клієнт-серверної взаємодії	03.05.22	25.05.22	Кінцевий веб-додаток
8	Оформлення пояснювальної записки та ілюстративного матеріалу	26.05.22	10.06.22	ПЗ, презентація

6 Матеріали, що подаються до захисту ДР

Пояснювальна записка ДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відзив наукового керівника, рецензія, анотації до МКР українською та іноземною мовами, нормоконтроль про відповідність оформлення ДР діючим вимогам.

7 Порядок контролю виконання та захисту ДР

Виконання етапів графічної та розрахункової документації ДР контролюється науковим керівником згідно зі встановленими термінами. Захист ДР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення ДР

Вимоги викладені в методичних вказівках до дипломного проектування, ДСТУ_ 3008-95, ДСТУ 3974-2000 «Правила виконання дослідно-конструкторських робіт. Загальні положення» та діючого ГОСТ 2.114-95 ЕСКД.

Технічне завдання до виконання прийняв ___ Никитюк В’ячеслав

ДОДАТОК Б

Структурна схема ідентифікації користувача

Рисунок Б.1 — Структурна схема ідентифікації користувача

ДОДАТОК В

Структурна схема додавання новини

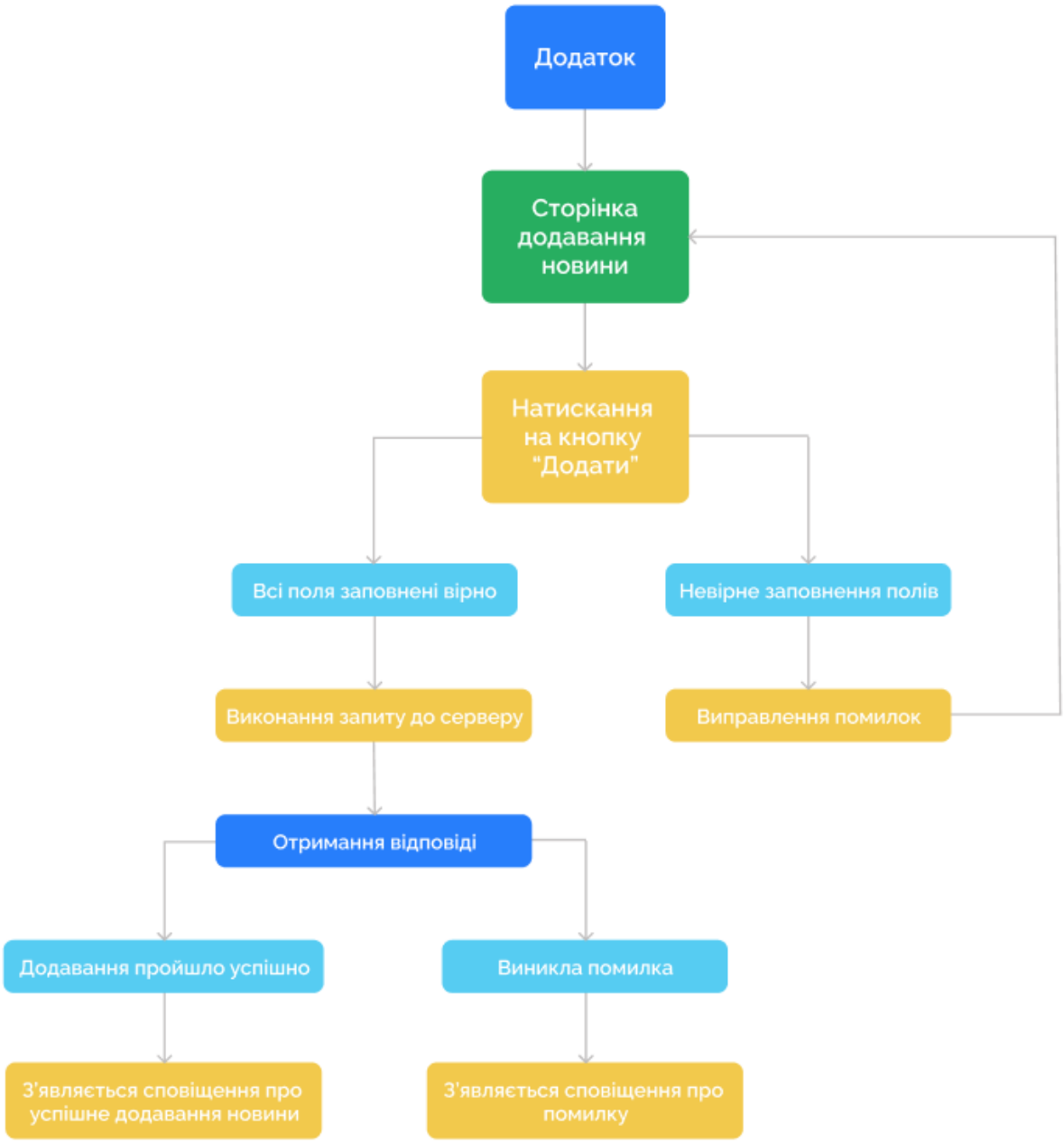

Рисунок В.1 — Структурна схема додавання новини

ДОДАТОК Г

Дизайн сторінок адміністративної панелі

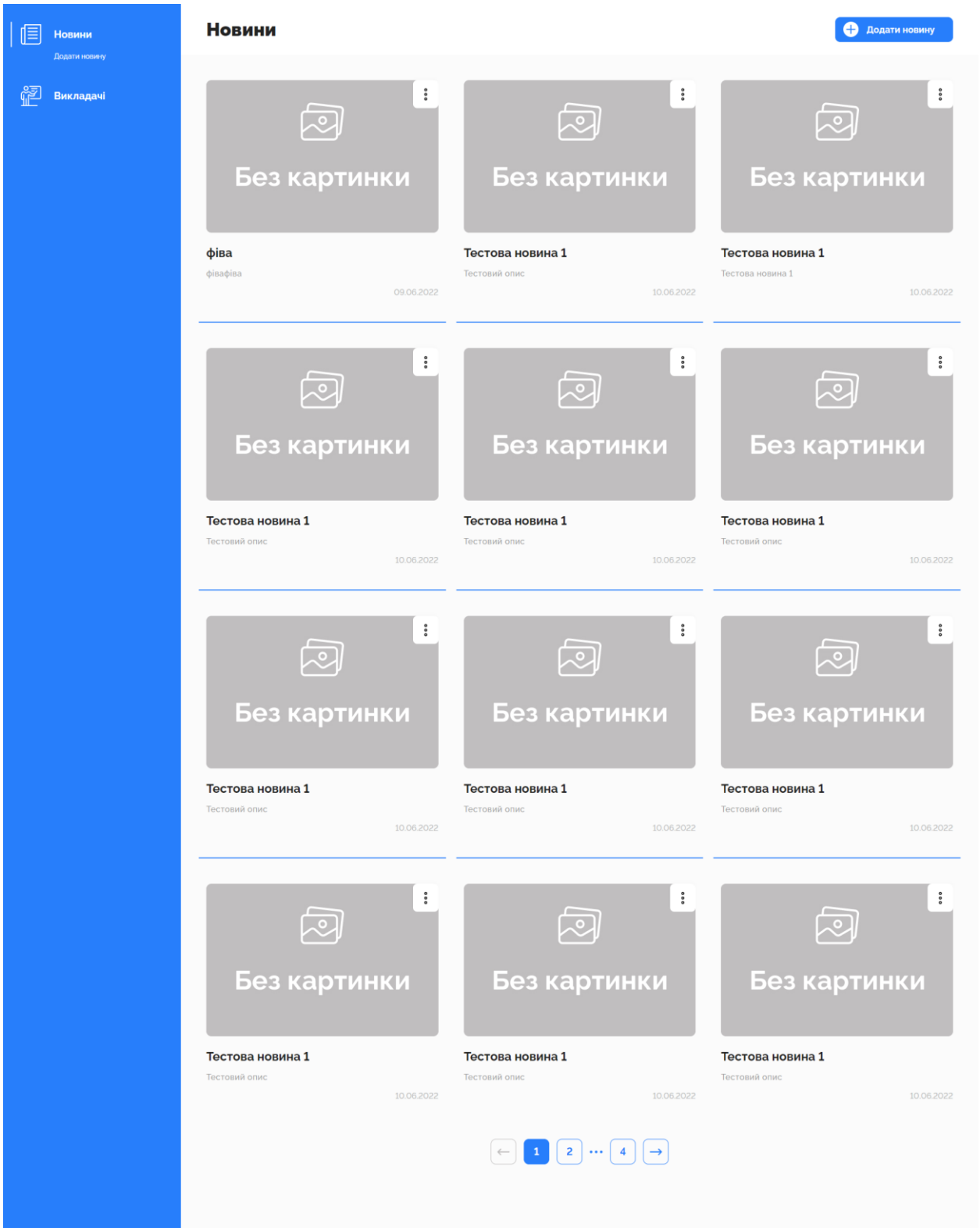

Рисунок Г.1 — Дизайн сторінки керування новинами

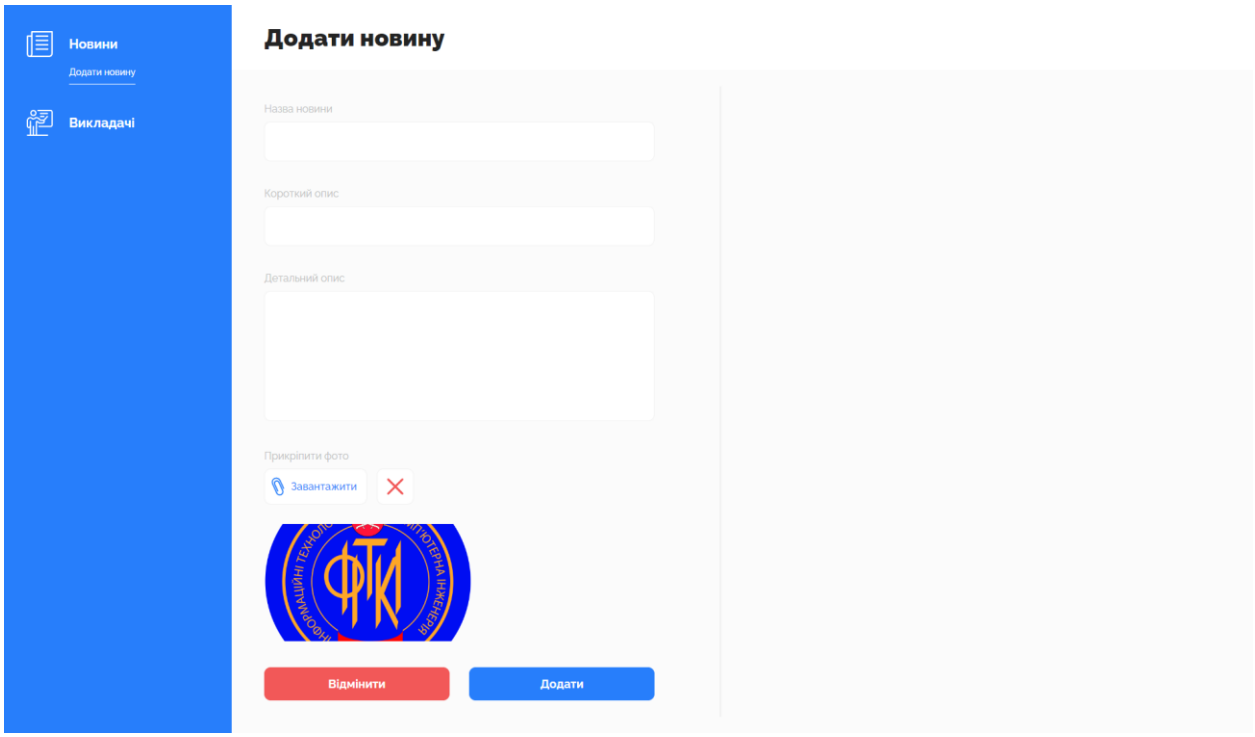

Рисунок Г.2 — Дизайн сторінки додавання новини

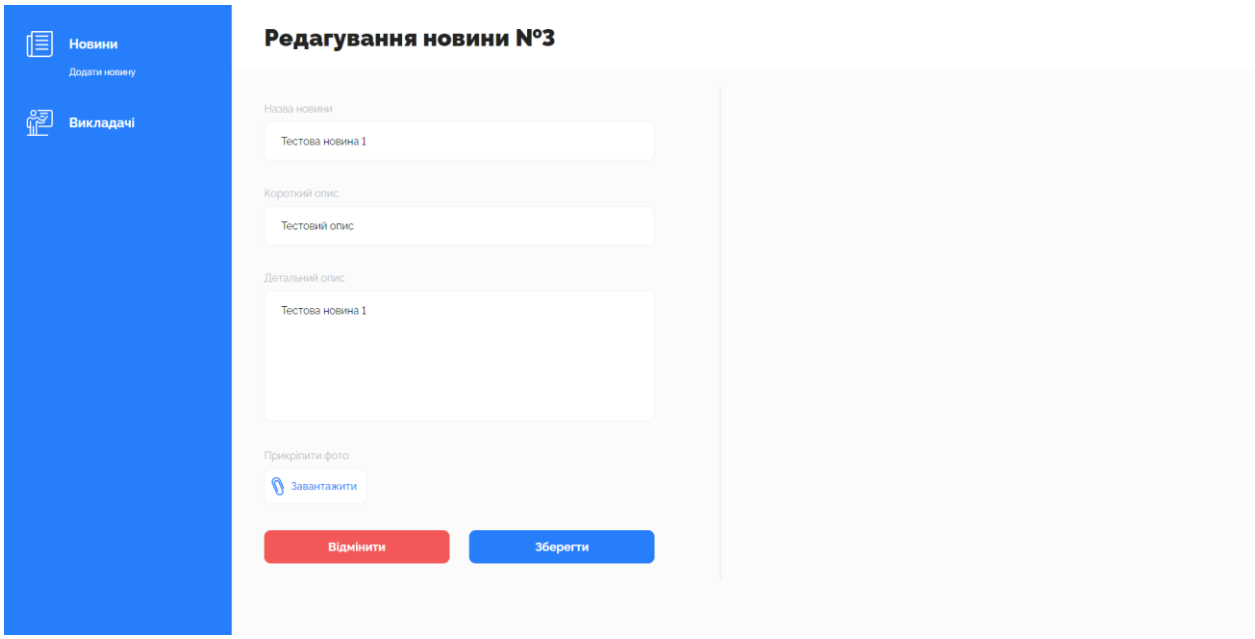

Рисунок Г.3 — Дизайн сторінки редагування новини

ДОДАТОК Д

Лістинг компонента сповіщення

Файл Notifications.js

```
import React from 'react'
import cn from 'classnames'
import { observer } from 'mobx-react-lite'

// Styles
import styles from './index.module.scss'

// Store
import {NotificationsStore} from '../../../store'

// Images
import loader from '../../../assets/images/loader.gif'
import { SvgSprite } from '../SvgSprite/SvgSprite'

export const Notifications = observer(() => {
   return NotificationsStore.notifications.length > 0 ? (
      <div className={styles.notifications}>
         {
            NotificationsStore.notifications.map(notificationItem => (
               <div className={cn({
                  [styles.notificationsItem]: true,
                  [styles.notificationsItemLoading]: notificationItem.type ===
'loading',
                  [styles.notificationsItemGreen]: notificationItem.type === 'success',
                  [styles.notificationsItemRed]: notificationItem.type ===  'error',
```

```
                })} key={notificationItem.id}>
                    <span className={styles.notificationsItemIcon}>
                        {
                            notificationItem.type === 'loading' && <img src={loader}
alt='loader'/>
                        }
                        {
                            notificationItem.type === 'success' && <SvgSprite
spriteID={'success'} />
                        }
                        {
                            notificationItem.type === 'error' && (
                                <SvgSprite
                                    className={styles.notificationsIconError}
                                    spriteID={'error'}
                                />
                            )
                        }
                    </span>
                    {notificationItem.text}
                </div>
            ))
        }
    </div>
  ) : null
})
```

ДОДАТОК Е

Лістинг компонента поля вводу інформації

Файл Input.js

```
import React from 'react'
import cn from 'classnames'

// Styles
import styles from './index.module.scss'

export const Input = ({placeholder, error, textarea, type, id, label, ...typeHandle})
=> {
    return (
        <div className={cn({
            [styles.inputWrapper]: true,
            [styles.inputWrapperError]: error,
        })}>
            <label className={styles.inputLabel} htmlFor={id}>{label}</label>
            {
                textarea ? (
                    <textarea
                        className={styles.inputTextarea}
                        value={typeHandle.value}
                        onChange={typeHandle.onChange}
                        id={id}
                    />
                ) : (
                    <input
                        className={styles.input}
```

```
                type={type}
                placeholder={placeholder}
                value={typeHandle.value}
                onChange={typeHandle.onChange}
                id={id}
              />
            )
          }
          {error && <span className={styles.inputError}>{error}</span>}
        </div>
    )
}
```

ДОДАТОК Ж

Лістинг файлу ініціалізації react-додатку

Файл index.js

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import { BrowserRouter as Router } from "react-router-dom"
import App from './App'
import { ApolloClient, InMemoryCache, ApolloProvider } from '@apollo/client'

// Styles
import './assets/styles/general.scss'
import './assets/styles/app.scss'

const client = new ApolloClient({
   uri: 'https://departament.slivki-cat.com/graphql',
   cache: new InMemoryCache(),
})

const root = ReactDOM.createRoot(document.getElementById('root'))
root.render(
   <React.StrictMode>
      <Router>
         <ApolloProvider client={client}>
            <App/>
         </ApolloProvider>
      </Router>
   </React.StrictMode>
)
```

ДОДАТОК И

Лістинг файлу класу глобального сховища сповіщень

Файл Notifications.js в каталозі store

```
import { makeAutoObservable } from 'mobx'

class Notifications {
   notifications = []

   constructor() {
      makeAutoObservable(this)
   }

   addNotification (id, type, text) {
      this.notifications.push({
         id, type, text,
      })
   }

   removeNotification (timeout, id) {
      setTimeout(() => {
         this.notifications.splice(this.notifications.findIndex(notificationItem =>
notificationItem.id === id), 1)
      }, timeout)
   }
}

export const NotificationsStore = new Notifications()
```

ДОДАТОК Н
ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 3. «Розробка панелі адміністратора за допомогою реактивного фреймворка React»

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності
Unicheck

Оригінальність 90,3% Схожість 9,7%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Никитюк В.О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Семеренко В.П.
(підпис) (прізвище, ініціали)

ДОДАТОК П

Акт впровадження

УЗГОДЖЕНО
Декан факультету інформаційних технологій та комп'ютерної інженерії, к.пед.н., доцент
_____ Світлана КИРИЛАЩУК
«___» _____ 2022 р.

ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи та організації освітнього процесу ВНТУ, к.т.н., доцент
_____ Олександр ПЕТРОВ
«___» _____ 2022 р.

АКТ ВПРОВАДЖЕННЯ № ____

результатів комплексної бакалаврської дипломної роботи

Замовник Вінницький національний технічний університет
(найменування організації)

Цим актом підтверджується, що результати роботи – «Клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 3. «Розробка панелі адміністратора за допомогою реактивного фреймворка React»,
що виконана студентом гр. КІ – 20мсз, ВНТУ, Никитюком В.О.
(виконавець)
впроваджено у Вінницькому національному технічному університеті
(найменування організації, де здійснювалося впровадження)

1.Вид впроваджених результатів сайт кафедри обчислювальної техніки ВНТУ
(експлуатація виробу, роботи, технології)

2. Характеристика масштабу впровадження одиничне
(унікальне, одиничне, партія, масове, серійне)

3. Форма впровадження програмний продукт

4. Новизна результатів роботи модернізація старих розробок
(піонерські, принципово нові, якісно нові, модифікації, модернізація старих розробок)

5. Впроваджені: в мережі ВНТУ

6. Річний економічний ефект____-_____

Від виконавця:
Студент групи КІ-20мсз
_____ В'ячеслав НИКИТЮК

Від організації:
Завідувач кафедри обчислювальної техніки ВНТУ, д.т.н., професор
_____ Олексій АЗАРОВ

Науковий керівник
_____ доц. Василь СЕМЕРЕНКО