Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

. Кафедра обчислювальної техніки

Пояснювальна записка

до бакалаврської дипломної роботи на тему:

«Клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 4. «Розробка веб-клієнту за допомогою реактивного фреймворка React»

Виконав: студент 2 курсу, групи 1КІ-20мс
спеціальності 123 — Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Іванов В.М.
(прізвище та ініціали)
Керівник к.т.н., доцент каф. ОТ
Войцеховська О.В.
(прізвище та ініціали)
«13» 06 2022 р.
Рецензент к.т.н., ст.викладач каф. ЗІ
Лукічов В.В.
(прізвище та ініціали)
«14» 06 2022 р.

Допущено до захисту
Зав. кафедри ОТ 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

Іванов Владислав Миколайович

1 Тема роботи — клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 4. «Розробка веб-клієнту за допомогою реактивного фреймворка React», керівник роботи Войцеховська Олена Валеріївна,затверджені наказом вищого навчального закладу від 24.03.2022 № 66

2 Термін подання студентом роботи 14.06.2022.

3 Вихідні дані до роботи: Макет інтерфейсу користувача, інформація по технологіям для створення веб-клієнту, дані для наповнення веб-клієнту

4 Зміст текстової частини: огляд та обгрунтування вибору технологій для розробки веб-клієнту інформаційної системи підрозділу навчального закладую. Розробка та програмна реалізація веб-клієнту клієнт-серверної інформаційної системи структурного підрозділу навчального закладу. Проектування веб-клієнтського інтерфейсу інформаційної системи.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): Мапа веб-клієнту, вигляд головної сторінки, вигляд сторінки з новинами, вигляд навігаційного блоку бургер-меню.

6 Консультанти розділів роботи приведено в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

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

7 Дата видачі завдання 08.02.2022.

8 Календарний план приведено в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання		Примітка
		початок	закінчення	
1	Отримання завдання.	08.02.22		Виконав
2	Аналіз завдання.	09.02.22	12.02.22	Виконав
3	Процес проектування плану створення веб-клієнтської частини.	12.02.22	20.02.22	Виконав
4	Огляд технологій створення та оптимізації веб-клієнтських інтерфейсів.	21.02.22	28.02.22	Виконав
5	Аналіз макету та вибір підходу його створення.	29.02.22	14.03.22	Виконав
6	Проектування файлової структури для веб-клієнту.	15.03.22	21.03.22	Виконав
7	Створення та програмна реалізація окремних компонентів для веб-клієнту	22.03.22	20.04.22	Виконав
8	Створення основних сторінок веб-клієнтської частини та отримання інформації з серверної частини.	21.04.22	15.05.22	Виконав
9	Перевірка працездатності клієнтської частини веб-додатку	16.05.22	30.05.22	Виконав
10	Оформлення пояснювальної записки та ілюстративного матеріалу	01.06.22	14.06.22	Виконав
11	Перевірка якості виконання бакалаврської роботи та усунення недоліків	14.06.22		Виконав

Студент Іванов В. М.

Керівник роботи Войцеховська О. В.

АНОТАЦІЯ

Комплексна бакалаврська дипломна робота складається з 73 сторінок формату А4, на яких є 22 рисунки, перелік джерел посилання містить 16 найменувань.

В бакалаврській дипломній роботі проаналізовано технології для створення веб-клієнтських інтерфейсів. Проведено порівняння функціональних можливостей фреймворків React та Angular, також проведено аналіз та порівняння комплектувальників webpack та gulp.

Використовуючи компонентний підхід було спроектовано та програмно реалізовано веб-клієнт інформаційної системи підрозділу навчального закладу. Спроектовано фізичну структуру веб-клієнту з можливістю подальшого масштабування.

Описано функціональні можливості веб-клієнтської частини проекту. Доданий основний інструментарій для клієнт-серверної взаємодії.

Ключові слова: React, фреймворк, Angular, Webpack, Gulp.

ABSTRACT

The complex bachelor's thesis consists of 73 A4 pages, which contain 22 figures, the list of reference sources contains 16 titles.

In the bachelor's thesis the technologies for creating web client interfaces are analyzed. The functionality of the React and Angular frameworks was compared, and the webpack and gulp components were analyzed and compared.

Using the component approach, a web client of the information system of the educational unit of the educational institution was designed and implemented. The physical structure of the web client with the possibility of further scaling is designed.

The functionality of the web client part of the project is described. Added basic tools for client-server interaction.

Keywords: React, framework, Angular, Webpack, Gulp.

ЗМІСТ

ВСТУП

В світі існує багато різних способів передачі інформації, але на сьогодні сайти є одним з найпопулярніших та ефективніших інформаційних джерел. У наш час за будь-якої необхідності людина одразу звертається в інтернет за пошуком інформації. Якщо під час пошуку інформації людина потрапляє на сайт із заплутаним інтерфейсом або низькою продуктивністю, то користувач просто звернеться до іншого ресурсу. Тому важливо приділяти увагу таким речам, як UI(User Interface) та UX(User Experience). Це допоможе створити сайт який буде зручний та зрозумілий для користувача.

Зазвичай доступ до інтернет ресурсів можна отримати майже з будь-якого пристрою, як з телефона з невеликим екраном так і стаціонарних комп'ютерів. Тому під час розробки веб-додатків важливо враховувати його адаптивність під різні пристрої.

Також важливу роль грає швидкодія сайту. Яким би сайт не був зручним та привабливим для користувача, якщо він буде довго завантажуватися, то людина скоріш за все просто звернеться до іншого ресурсу. Тому часто для покращення швидкодії та продуктивності веб-додатку використовують підхід Single Page Application.

Отже, розробка клієнт-серверної інформаційної системи підрозділу навчального закладу з можливістю розгортання в хмарному середовищі є актуальною.

Метою роботи є розробка веб-клієнту для веб-сайту інформаційної системи підрозділу навчального закладу, який надасть можливість швидко та зручно знаходити та переглядати інформацію на сайті.

Задачі дослідження бакалаврської роботи:

— проаналізувати сучасні технології та інструменти для розробки веб-інтерфейсів користувача, технології комплектування модулів веб-додатків;

— розробити структуру та програмну реалізацію веб-клієнту для інформаційної системи підрозділу навчального закладу, використовуючи компонентний підхід;

— спроектувати веб-клієнтський інтерфейс інформаційної системи.

Об’єкт дослідження — процес створення клієнтського інтерфейсу за допомогою популярних технологій, що дасть можливість підвищити продуктивність та захищеність інформаційної системи.

Предметом дослідження є сучасні технології, фреймворки та методології для створення інтерфейсу користувача.

Методами дослідження є використання фреймворку React, препроцесору Sass та методології БЕМ.

Апробація результатів бакалаврської роботи: зроблено доповідь на LI науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії [1].

Публікація за темою роботи: Іванов В.М. — Аналіз технологій створення веб-клієнту за допомогою реактивного фреймворка React / В.М. Іванов, О.В. Войцеховська. Матеріали LI наукової-технічної конференції підрозділів Вінницького національного технічного університету (Вінниця, 2022 р.) [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/author/submission/15736.

1 ОГЛЯД ТА ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-КЛІЄНТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДРОЗДІЛУ НАВЧАЛЬНОГО ЗАКЛАДУ

Розробка клієнтської частини інтерфейсу користувача або фронт-енд розробка в даний час є дуже популярною. Велика кількість веб-сайтів, сукупність яких тільки зростає з року в рік, потребує великої кількості розробників для їх підтримання та розробки нових сайтів. Бек-енд розробник відповідає за розробку серверної частини веб-додатку на якій відбуваються основні обрахунки та робота з даними. Бек-енд розробник працює разом з розробником баз даних, як результат їхньої співправці є можливість обмінюватись даними між клієнтською частиною та базою даних.

1.1 Базові технології розробки веб-клієнтів

Задача front-end розробника — розробити зовнішній вигляд сайту за допомогою мови розмітки гіпертексту HTML (Hyper Text Markup Language), за допомогою якої браузер розташовує різні типи елементів на сторінці, а каскадні таблиці стилів CSS (Cascading Style Sheets) дозволяють розробнику присвоювати стилі елементам, що розміщені на сторінці. Також для підтримки інтерактивності сторінки потрібно використовувати мову програмування JavaScript, за допомогою якої сторінка набуває деякої динаміки та інтерактивності. Тобто дає можливість задавати логіку елементам HTML-розмітки для взаємодії користувача з ними.

1.1.1 Особливості мови гіпертекстової розмітки HTML

Інформація, яка зберігається у форматі HTML-документа виглядає як звичайний текстовий файл. Але є одна відмінність, вона полягає в тому, що деякі символи в них інтерпретуються як розмітка. Ці символи називаються тегами (tag). Ці теги допомагають документу стати структурованим, а саме розділити його на такі елементи як: параграфи, розділи, колонтитулі, малюнки,

таблиці, абзаци, списки, індекси, зміст тощо. Також кожному блоку можна надавати різне стильове оформлення: змінювати розмір символів, шрифт, колір тексту, виділяти текст напівжирним та/або робити його курсивним [2].

Головною особливістю HTML є здатність використовувати гіперзв'язки (links), завдяки яким можна створювати посилання та переходи з поточної веб-сторінки на інші документи, як локальні (документи які знаходяться на поточному комп'ютері або сервері), так і такі, що розміщені на серверах у будь-якій точці земного кулі.

1.1.2 Каскадні таблиці стилів CSS та препроцесори CSS

CSS використовується для представлення зовнішнього вигляду документу написаного на HTML.

Використовуючи CSS можна керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром і розташуванням компонентів, надавати фонові зображення або кольори, створювати варіантами відображення компонентів на сторінці для різних типів пристроїв і розмірів екрана [3].

CSS-препроцесори (наприклад, Less, Sass і Stylus) надають можливість розширити функціональні можливості стандартного CSS, дають змогу використовувати mixins, вкладені правила, змінні, вбудований імпорт, тощо. Це дозволить зробити CSS-стилі більш організованими, скоротити вміст css файлів, що збільшить зручність їх читання та зробить їх більш компактними. CSS-препроцесори ще називають динамічними мовами стилів [4].

Браузер створений таким чином, що не зможе інтерпретувати синтаксис препроцесорів, тому вихідний код повинен бути перетворений в CSS. Для компіляції стилів в оригінальний CSS існує три способи:

— в браузері за допомогою JavaScript-інструменту;

— на стороні сервера з використанням мови програмування;

— локально на комп'ютері за допомогою програми.

Sass використовує розширення .scss (напр., style.scss), Less — .less, а Stylus —.styl.

LESS — це препроцесор CSS, який прискорює оформлення HTML сторінок сайту за допомогою таблиці стилів CSS. Препроцесор LESS — це мова (частково схожа на мову програмування, але набагато простіше), яка значно розширить можливості при написанні CSS коду.

Використання препроцесора LESS дасть ряд можливостей, основні з яких: створення змінних, використання вкладеності та імпортування.

Stylus є найменш популярним серед інших препрцесорів, але деякі його можливості не поступаються іншим.

Як і інші препроцесори, Stylus має багато переваг над звичайним CSS — змінні, міксини, оператори, функції, імпорти тощо.

Ще один препроцесор — Sass, який має ряд переваг [5]:

— Sass повністю сумісний з усіма версіями CSS, так як розробники приділяють велику увагу сумісності, що дозволяє легко використовувати будь-які бібліотеки CSS;

— більша кількість можливостей ніж в любому іншому препроцесорі;

— Sass активно підтримується і розробляється високотехнологічними компаніями та декількома сотнями розробників.

1.1.3 Мова програмування JavaScript

JavaScript — це мова програмування, яка дає можливість розроблювати динамічні та інтерактивні веб-сторінки. Дозволяє створювати та налаштовувати оновлення окремих блоків веб-сторінки, інтерактивні карти, анімації графіки, прокрутка відео в медіапрогравачі, тощо.

JavaScript є мовою, що інтерпретується (англ. interpreted programming language). Це означає, що їй потрібен інтерпретатор для роботи. Такими інтерпретаторами є веб-браузери, якими люди користуємося з дня в день.

Будь-яка веб-розробка не можлива без використання мови JavaScript. Кожна компанія, яка спеціалізується на розробці веб-додатків, повинна мати розробника, який має знання з даної мови програмування [6].

Також за допомогою мови JavaScript можна легко працювати з об'єктною моделлю документа. JavaScript DOM містить в собі багато функцій, які допомагають зручно взаємодія з елементами веб-сторінки, а саме знаходити, створювати, видаляти, перезаписувати їх вміст, доступатися до стилів тих чи інших елементів та змінювати їх, якщо потрібно. Також за допомогою мови JavaScript можна надавати логіку елементам, наприклад змінювати їх вміст, стилі або поведінку при певній взаємодії користувача з сторінкою.

1.2 Огляд сучасних технології для ефективної розробки веб-інтерфейсів користувача

1.2.1 Особливості технології React

Сучасна технологія React дозволяє створювати оптимізовані та інтерактивні користувацькі інтерфейси. React спроектований таким чином що він не залежить від браузера. Він призначений для створення користувацьких інтерфейсів, а за рендерінг та відображення React-компонентів в браузері відповідає бібліотека React-DOM. Фреймворк React спроектований таким чином, що можна окрім браузерних додатків, за допомогою React Native, створювати також додатки на Android, IOS та навіть Windows. Таким чином ядро React містить велику кількість концепцій, а за візуалізацію в тому чи іншому оточені відповідають інші бібліотеки.

Функціональність React розширюється бібліотеками залежно від того, яким має бути веб-додаток або веб-сторінка. Це дає змогу власноруч структурувати проект, що робить дану технологію гнучкою та не великою за об’ємом.

Також однією з основних переваг React є те що він дозволяє розділяти сторінку на компоненти. Це дозволяє зручно використовувати ті самі блоки

коду в інших частинах проекту, що значно зменшує час та ресурси, витрачені на розробку веб-сайту [7].

1.2.1.1 Virtual DOM

DOM (Document Object Model) — це структуроване представлення елементів HTML, які присутні на веб-сторінці або додатку. При кожній зміні інтерфейсу, DOM також оновлюється для відображення цих змін, але часті маніпуляції із DOM негативно впливають на продуктивність [8].

DOM представлений як деревоподібна структура даних, тому зміни та оновлення самого DOM є досить швидкими. Але після зміни оновлений елемент та всі його дочірні елементи повинні бути повторно завантажені (відрендерені) для оновлення UI. Повторний рендеринг — дуже повільний процес. Таким чином, чим більше компонентів UI, тим більш дорогими з точки зору продуктивності є оновлення DOM.

В React замість взаємодії з DOM безпосередньо, розробник працює з його копією. Розробник може вносити зміни в копію, виходячи з потреб, а потім застосовувати зміни до реального DOM.

При цьому відбувається порівняння DOM-дерева з його віртуальною копією, визначається різниця та запускається перезавантаження того, що було змінено.

Такий підхід працює швидше, тому що не включає всі ресурсозатратні частини реального DOM.

1.2.1.2 Переваги застосування функціональних компонентів технології React

При створенні сторінки одразу можна виділити такі блоки як: хедер, навбар, сайдбар, блок з основним контентом та футер. Кожен з цих блоків має свої блоки. Наприклад хедер може містити в собі логотип, меню, контакти, тощо.

В React кожен з цих блоків називається компонентом. Кожен компонент може містити в собі більш дрібні компоненти, а ті в свою чергу ще більш дрібніші.

Ці компоненти можуть бути створенні за допомогою класів або функцій.

У даний момент часу функціональні компоненти використовувати краще, вони мають ряд переваг над класовими компонентами.

По-перше функціональні та класові компоненти мають різний синтаксис написання. У класових компонентах синтаксис написання більш складний (рисунок 1.1) та з часом може накопичувати в собі велику кількість коду, що робить його складним для рефакторингу коду.

Рисунок 1.1 — Приклад створення класового компоненту

У класовому компоненті всі властивості передаються через конструктор. А доступ до властивостей у компоненту класі надається таким виразом: this.props. Іноді це може викликати помилки так, як властивості React не змінюються, а this являється мутабельним за своєю суттю. Також класовий компонент повинен містити метод render(), який повертає React-елемент.

В той час, як функціональні компоненти мають більш простий синтаксис (рисунок 1.2) в порівняні з класовими.

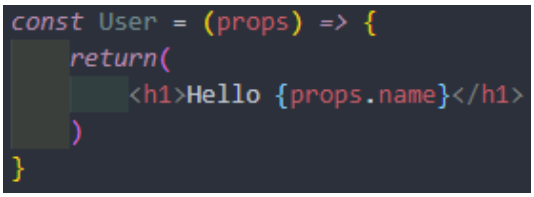


Рисунок 1.2 — Приклад створення функціонального компоненту

Функціональні компоненти приймають props як єдиний аргумент, тому можна завжди передавати властивості як props. Компонент приймає об'єкт props, де міститься значення, які передані від батьківського компонента. У props можна передавати будь-які типи навіть інші компоненти [9].

На відміну від класового компоненту функціональний компонент повертає React-елемент. Він має бути тільки один, якщо їх більше, вони мають бути обгорнуті батьківським елементом.

1.2.1.3 Аналіз відповідності життєвих циклів до функцій hooks

React Hooks — відносно нова технологія React, яка дозволяє використовувати стан та інші можливості компонентів без використання класів. За допомогою hooks (хуків) ми можемо отримувати стан компонента, щоб його можна було тестувати і перевикористовувати. Хуки дозволяють повторно використовувати логіку стану без зміни ієрархії компонентів. Це полегшує обмін посиланнями між багатьма компонентами або всією системою в цілому.

Для створення життєвого циклу в класовому компоненті існують функції componentDidMount(), componentDidUpdate() та componentWillUnmount().

Класові компоненти реалізують такі методи життєвого циклу:

— componentDidMount() — метод спрацьовує одразу після рендеру компонента та ідеально підходить для налаштування початкового стану, проте згодом, після видалення компонента, метод необхідно скасувати;

— componentDidUpdate() — метод не викликається на етапі початкового рендеру, проте він викликається щоразу, коли компонент

оновлюється, а стан чи властивості змінюються, також логіка методу завжди обертається умовою if для порівняння оновлених властивостей з попередніми;

— componentWillUnmount() — метод спрацьовує безпосередньо перед відключенням від віртуального DOM та видаленням компонента, зокрема він очищає будь-яке "сміття", наприклад, анулює таймери, скасовує мережні запити або будь-які підписки.

Зазвичай, класові компоненти доволі добре реалізують детальний контроль стану, але вони мають значні недоліки в порівняні з функціональними компонентами.

По-перше, класи стають не зручними для читання, тобто в процесі розробки класи можуть розростатися та зберігати в собі різну логіку. Наприклад, метод componentDidMount() може задавати стиль якомусь блоку і в той же час підключатись до сокету та змінювати title сторінки. І все це буде відбуватись в одному життєвому циклі.

По-друге, часто при використані класових компонентів доводиться витрачати час на оптимізацію, проводити рефакторинг коду, змінювати структуру конкретного компонента і навіть інших компонентів , які пов’язані з ним.

Для уникнення проблем, описаних вище, використовують функціональні компоненти, в яких є можливість використовувати технологію hooks. Дана технологія дозволяє працювати з локальними станами та методами життєвого циклу, не використовуючи класів.

За додавання стану до функціонального компоненту відповідає React хук useState (рисунок 1.3).

Хук useState імпортується з React і складається з трьох основних елементів:

— змінна стану;

— функція, що оновлює стан;

— початкова змінна стану.

```
import React, { useState } from 'react';
function Counter() {
    const [count, setCount] = useState(0);
    return (
        <div>
            <p>Кількість натискань: {count} </p>
            <button onClick={() => setCount(count + 1)}>
                Click
            </button>
        </div>
    );
}
```

Рисунок 1.3 — Приклад застосування useState

При монтуванні даного компонента в змінній count буде зберігатися пуста стрічка, а за допомогою функції setCount, яка викликається при натисканні на кнопку, в змінну count буде записано одинцю і з подальшими натисканнями на кнопку дана зміна буде збільшуватися на 1.

Отже, проведений аналіз показав, що для створення життєвого циклу в класовому компоненті існують функції componentDidMount(), componentDidUpdate() та componentWillUnmount(), а в функціональному компоненті всі ці функції заміняються одною функцією useEffect. Використання функції useEffect дасть можливість зробити код більш читабельним та запобігти появленню помилок при подальшій розробці додатку [10].

1.2.1.4 Модуль маршрутизації React Router

Кожен багатосторінковий веб-сайт повинен мати своє налаштування маршрутів для того щоб користувач, який відвідує сайт, бачив де він знаходиться в будь-який момент часу. Своє теперішнє місцезнаходження користувач може побачити в адресній стрічці браузера. Тому веб-сайт повинен вміти зпівставляти певну адресу URL з сторінкою, яка їй відповідає [11].

Також на веб-сайті повинна працювати історія, що дасть можливість користувачу переходити на попередню сторінку веб-клієнта.

Даний модуль маршрутизації дає можливість надати сторінкам відповідні URL адреси та за замовчуванням зможе використовувати історію користувача.

Цей модуль популярний та доволі простий у використанні. Він надає такі основні можливості як:

— навігація по кліку (компонент <Link>);
— перенаправлення (компонент <Redirect>);
— маршрутизація (компонент Route);
— історія (властивість history).

Дана технологія використовується для того, щоб переключатись між сторінками без перезавантаження додатку. Технологія React Router використовує динамічну маршрутизацію, що дозволить змінювати елементи UI інтерфейсу без повторного завантаження статичних файлів.

1.2.2 Огляд технології Angular та порівняння її з React

Angular є фреймворком, який створила компанія Google. Angular так само, як і React потрібен для створення клієнтських інтерфейсів та програм. В основному він призначений для розробки SPA-рішень (Single Page Application), тобто односторінкових додатків. В загальному, Angular та React мають однакові функціональні можливості, але різний підхід їх використання.

Також є певні відмінності між цими двома технологіям, які мають ключову роль при виборі однієї з цих технологій, в залежності від того який має бути проект.

Перша та одна з основних відмінностей Angular від React це те, що перший працює безпосередньо з DOM, а другий має власний віртуальний DOM. Віртуальний DOM дає змогу значно пришвидшити роботу веб-додатку.

Друга відмінність це прив'язування даних (Data Binding). Тут дві технології сильно відрізняються одна від одної. Angular надає таку функціональність як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому.

Проте React має лише односторонній зв'язок. Спочатку стан моделі оновлюється, а потім він відображає зміну елемента інтерфейсу. Однак якщо ви змінюєте елемент інтерфейсу, стан моделі не змінюється.

Ще одною значною відмінністю цих технологій є те, що Angular має каркасну структуру і надає розробнику всі можливості програмування «з коробки», тобто Angular потрібно тільки встановити і розробник буде мати весь необхідний функціонал, а саме: шаблони, що базуються на HTML; впровадження залежності (Dependency injection); роутінг; інкапсуляція компонентів CSS; утиліти для тестування компонентів; можливості для створення форм тощо. На відміну від Angular React є фреймворком який надає тільки представлення(View). Але його функціональні можливості можна розширювати в залежності від того яким повинен бути проект. Тобто можна зробити висновок, що технологія Angular має великий функціонал, в якому деякі функції використовуватися не будуть. В деяких випадках це може значно сповільнити роботу веб-додатку. А React має можливість власноруч розширювати свій функціонал, що робить цю технологію гнучкою та не великою за об’ємом.

1.3 Огляд сучасної методології створення веб-ресурсів

Як відомо, підтримувати стислість коду, робити його повторно використовуваним і обслуговуваним досить важко. Якщо нехтувати будь-якою послідовністю в написанні коду, то код може вийти з-під контролю, як в малих, так і середніх і великих проектах, в яких задіяно понад одного розробника.

Якщо код великий, з часом в нього вносяться зміни, а організація відсутня, то це призводить до того, що команди розробників починають додавати нові стилі в кінець документа. Розробники перестають видаляти шматки коду і змінювати вже існуючі. Основна причина в тому, що найчастіше видалення і редагування стилів може призвести до несподіваних наслідків, а також до поломки дизайну в окремих місцях. Це програшна стратегія, яка

може привести до дублювання коду, проблемам специфічності, де перевизначення правил перетворюється в цілу битву і загальне роздування коду.

Найчастіше вибір методології — це інтерактивний процес, що починається з ознайомлення з тим, що вже є в мережі [12].

1.3.1 БЕМ методологія

БЕМ розшифровується як Блок-Елемент-Модифікатор. Ця методологія допомагає створювати, розширювати та повторно використовувати компоненти інтерфейсу.

ЇЇ основна концепція — легка підтримка проектів з часом і повторне використання компонентів.

Головна стратегія БЕМ полягає в організації CSS-коду в повторно використовуваних модулях за допомогою розумної системи іменування.

Знаходження блоків - вирішальний фактор застосування методології БЕМ. Блок — це функціонально незалежний компонент сторінки, який може бути повторно використаний. В HTML блоки представлені атрибутом class.

Блоки можна вкладати один в одного. Наприклад, в блок хедера можна помістити блок меню.

Блоки можна використовувати в будь-якому місці на сторінці, тому в CSS-коді для блоків не повинно бути ніяких зовнішніх відступів і правил позиціонування.

І нарешті, при виборі імені потрібно вказувати призначення блоку, а не його зовнішній вигляд або стан. Іншими словами, ім'я має відповідати на питання: що це? (наприклад, хедер, меню і т.д.). Питання не повинно бути типу «як це виглядає» (наприклад, фіксований хедер, маленьке меню і т.п.).

За методологією БЕМ елемент — це складова частина блоку, яка не може використовуватися у відриві від нього.

Принципи застосування елементів:

— елементи знаходяться тільки в блоках;

— елементи не можуть належати іншим елементам, вони можуть лише бути в блоках;

— можна створювати вкладені елементи;

— імена елементів описують їх призначення, а не зовнішній вигляд;

— при іменуванні елементів необхідно дотримуватися стилю block__element.

Модифікатор — це сутність, що визначає зовнішній вигляд, стан або поведінку блоку або елементу. Наприклад, блок хедера може бути зафіксований у верхній частині сторінки, блок з випадючим списком може відкриватися і закриватися, блок кнопки може бути відключений і т.д [13].

Стиль іменування модифікаторів в БЕМ: block__element_modifier.

Це ядро методології БЕМ. Також БЕМ пропонує принципи організації структури файлів та зручний набір інструментів.

Переваги методології БЕМ:

— нові розробники можуть швидко зрозуміти зв'язок між компонентами в розмітці і CSS;

— методологія сприяє підвищенню продуктивності в команді. Переваги особливо помітні у великих проектах;

— система іменування знижує ризики колізій з класами і витік стилів;

— CSS несильно прив'язаний до розмітки в певному місці на сторінці;

— CSS стає повторно використовуваним.

1.4 Аналіз засобів для комплектування модулів програмного продукту

Для кращого зрозуміння переваг використання комплектувальників у проектах, можна уявити наступну ситуацію. Є один великий за обсягами проект з безліччю складовими, над яким працює декілька розробників і у кожного з розробників є власні js та css файли. У розробницькій збірці знаходиться велика кількість різних файлів, не стиснутих картинок та css-файли на тисячі рядків коду.

Тут знадобиться технологія комплектування, яка об’єднає всі .css, .scss, .styl в один єдиний мінімізований css-файл. Те саме зробить і зі скриптами, стисне картинки та згрупує їх у потрібну папку. І на виході отримується досить оптимізований проект значно менший за об’ємом. Далі даний проект завантажується на сервер. У результаті на сервері зберігається готовий та оптимізований продукт, а кінцевий користувач не отримує нічого зайвого.

1.4.1 Огляд технології комплектування Webpack

В своєму роді Webpack є комплектувальником (bundler) модулів сучасних додатків JavaScript. Він аналізує всі модулі у додатку, аналізує залежності, а потім об’єднує всі модулі разом в один або кілька бандлерів(bundle), на які зможе посилатися файл index.html [14].

Зазвичай при створенні додатку в основі якого лежить JavaScript, код розділяють на декілька частин (Modules). Далі у фалі index.html потрібно вказати посилання на кожен з цих модулів. При такому підході важливо не тільки не забути про якийсь фрагмент коду, але й розташувати їх в правильному порядку. Якщо загрузити код, який залежить від React, до завантаження самого React, то додаток зламається. Webpack вирішує ці задачі. Не потрібно думати про послідовність підключення модулів.

Але збірка модулів це лише один із можливостей webpack. При необхідності webpack може здійснювати деякі перетворення модулів перед їх додаванням у бандл. Наприклад, якщо у проекті використовувати файли з розширенням SCSS, які потрібно перетворити на CSS. Налаштування таких перетворень у webpack здійснюється дещо легше ніж в інших технологіях комплектування додатків .

1.4.2 Огляд технології комплектування Gulp

Альтернативною технологією Webpack є Gulp. Gulp — це спеціальний інструмент за допомогою якого можна автоматизувати рутині задачі. Рутині

задачі — це звичайна мінімізація коду CSS або JS, компіляція файлів SASS або LESS, перетворення значків SVG у шрифти, перегляд файлів на предмет змін та інших дій тощо. Також за допомогою технології Gulp можна комплектувати проекти, але для цього потрібно встановлювати додаткові плагіни [15].

Ці задачі набагато зручніше та швидше робити за допомогою Webpack так, як налаштування його відбувається в рази швидше та більше варіантів для реалізації потрібної задачі.

Отже, для створення інтерактивного та оптимізованого додатку було проаналізовано декілька найпопулярніших технологій створення веб-клієнтських інтерфейсів, а саме React та Angular. Був проведений аналіз основних можливостей обох технологій та розглянуті їх основні відмінності. Проведений аналіз показав, що використання технології React дасть можливість створити більш швидку та оптимізовану інформаційну систему у порівняні з Angular.

Також була розглянута методологія розробки веб-додатків, що дозволить значно зменшити час та ресурси витрачені на створення інформаційної системи.

Для подальшої оптимізації додатку було проаналізовано та порівняно декілька комплектувальників. Після проведення підсумків було вирішено, що комплектувальник Webpack дещо краще ніж Gulp так, як його використання та налаштування значно простіше ніж у Gulp. Також Webpack має більші функціональні можливості, використання яких допоможе значно пришвидшити роботу розробника.

2 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-КЛІЄНТУ КЛІЄНТ-СЕРВЕРНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ СТРУКТУРНОГО ПІДРОЗДІЛУ НАВЧАЛЬНОГО ЗАКЛАДУ

2.1 Створення проекту React для розробки веб-кліенту інформаційної системи

Для того щоб створити React проект для початку потрібно створити пустий проект у якому потрібно відкрити термінал. Щоб створити пустий проект достатньо створити папку та відкрити її за допомогою редактору VS Code. Далі потрібно відкрити термінал та прописати команду npx create-react-app. Після команди потрібно ввести назву створюваного проекту (рисунок 2.1).

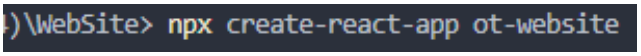


Рисунок 2.1 — Команда створення React проекту

Далі створюється базова структура будь-якого Front-end додатку (рисунок 2.2).

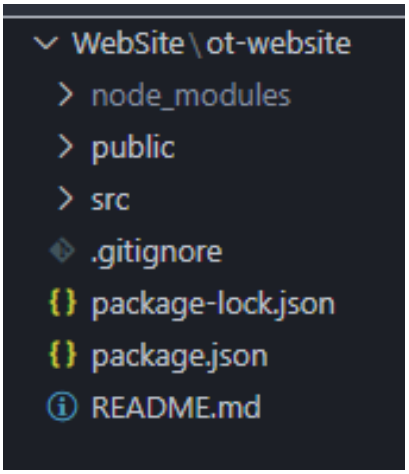


Рисунок 2.2 — Структура базового React проекту

Базова структура складається с папки node_modules в якій зберігаються різні бібліотеки, які використовуються для роботи React та різних комплектувальників проектів.

Файл .gitignore (рисунок 2.3) існує для того, щоб деякі не потрібні файли не потрапляли в систему контролю версій git, наприклад папка node_modules.

```
# dependencies
/node_modules
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*
```

Рисунок 2.3 — Вміст файлу .gitignore

Також в структурі є дві основні папки public та src. В папці public розміщений файли index.html, який є кореневим файлом та запускає весь додаток.

Після видалення з файлу index.html лишніх коментарів, тегів та мета тегів файл буде мати такий вигляд (рисунок 2.4).

index.html M

WebSite > ot-website > public > index.html > html > body

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>React App</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Рисунок 2.4 — Вміст файлу index.html

Після видалення всіх не потрібних елементів файл index.html має базову структуру звичайного файлу html. Всередині тегу body знаходиться єдиний блок div в якого є id root. Саме в цей блок буде вмонтовуватися весь додаток.

Папка src містить в собі два основних файла. Це файли App.js та index.js. З файлу index.js запускається весь додаток. Вміст файлу показано на рисунку 2.5.

```
JS index.js > ...
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'

const root = ReactDOM.createRoot(document.getElementById('root'))
root.render(
    <React.StrictMode>
        <App/>
    </React.StrictMode>
)
```

Рисунок 2.5 — Вміст файлу index.js

В даному файлі спочатку відбується імпорт React та ReactDOM, також імпортується компонент App.

Для того, щоб запустити весь додаток потрібно звернутися до бібліотеки ReactDOM та викликати у неї метод render, де відбувається відображення компоненту App. А далі потрібно імпортувати та викликати цей компонент в тому самому div з id root, який був описаний вище. Тобто весь додаток який представлений в компоненті App буде складатися в div, який знаходиться в кореневому файлі index.html. Файл App.js це самий звичайний React компонент, який створений через ключове слово function. Тобто це звичайна функція, яка повертає синтаксис JSX. В даному файлі будуть зібрані усі основні компоненти інформаційної системи, а саме: header, navbar, footer та main.

Далі розглянемо компонент App (рисунок 2.6).

```jsx
import react  from "react";
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

Рисунок 2.6 — Вміст файлу App.js

2.2 Проектування фізичної структури інформаційної системи структурного підрозділу навчального закладу

Правильно розроблена структура - це важлива частина розробки будь-якого проекту, адже інтуїтивно зрозуміла структура значно пришвидшить процес розробки додатку. Це дасть можливість швидко стартувати з розробкою не відволікаючись на пошук потрібних файлів. Також з правильною структурою проекту вірогідність помилки під час комплектування проекту значно зменшується.

В каталозі Assets розташовані папки fonts, images та style. Папки fonts та images містять в собі шрифти та зображення відповідно, які будуть використовуватися у проекті.

У папці styles розташовані загальні файли стилів всього проекту

Фізична структура каталогів проекту подана на рисунку 2.7.

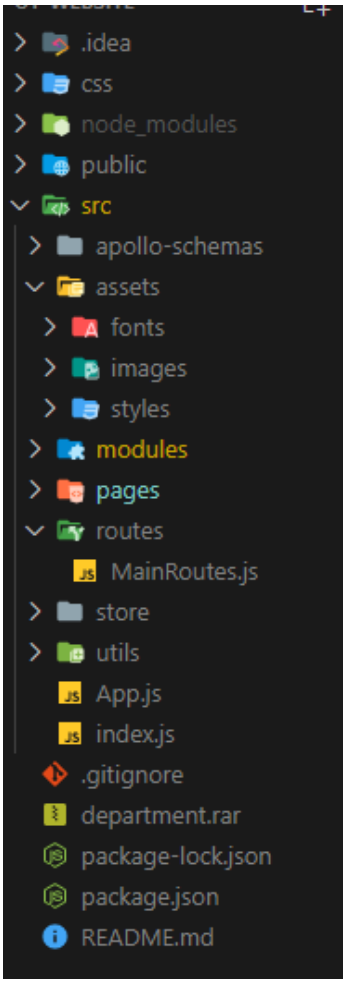


Рисунок 2.7 — Структура каталогів проекту

Короткий опис кожного з цих файлів:

— _fonts служить для того, щоб через даний файл підключати шрифти;

— _mixins в даному файлі будуть міститися заготовлені стилі, які можна буде потім підключати до відповідних блоків;

— _vars - тут зберігаються змінні, наприклад зміна в якій міститься певній колір;

— app зберігає стилі для загального компонента App;

— general містить загальні стилі для всього сайту.

В папці modules зберігаються ті компоненти коду, які будуть використані ще не один раз. Наприклад кнопки, футер, хедер та інші.

В папці pages зберігаються всі сторінки які є на сайті. Наприклад головна, новини, викладачі та інші.

Папка Routes містить в собі один файл, який має назву MainRoutes.js. В даному файлі буде налаштована навігація сайту для того, щоб користувач в будь-який момент часу бачив де він знаходиться

2.3 Розробка веб-клієнту інформаційної системи підрозділу навчального закладу

2.3.1 Аналіз макету веб-клієнту структурного підрозділу навчального закладу

Під час аналізу макету потрібно уявно розділяти макет на компоненти, уважно продивлятися кожен елемент макету, звертати увагу на однакові розміри шрифтів, відступи між блоками, їх стилізацію, знаходити однакові блоки та стилі. В подальшому такий підхід дозволить значно пришвидшити верстку макету та не писати один і той самий код декілька разів. Достатньо буде підключити один створений компонент в структуру додатку, де він потрібен.

Макетне представлення веб-сайту було розроблено у веб-сервісі Figma.

Figma — це зручний веб-сервіс в якому можна переглядати та аналізувати макети, які в подальшому будуть верстатися. В даному інструменті зручно переглядати різноманітні характеристики: стилі, відступи, розміри шрифтів їх сімейство, розміри блоків їх обведення та розташування, тощо. Для цього достатньо вибрати потрібний блок та з права на панелі будуть відображатися всі стилі вибраного блоку (рисунок 2.8). Всі ці базові характеристики, які можна зручно переглядати в Figma, покращують продуктивність розробки інтерфейсів користувача [16].

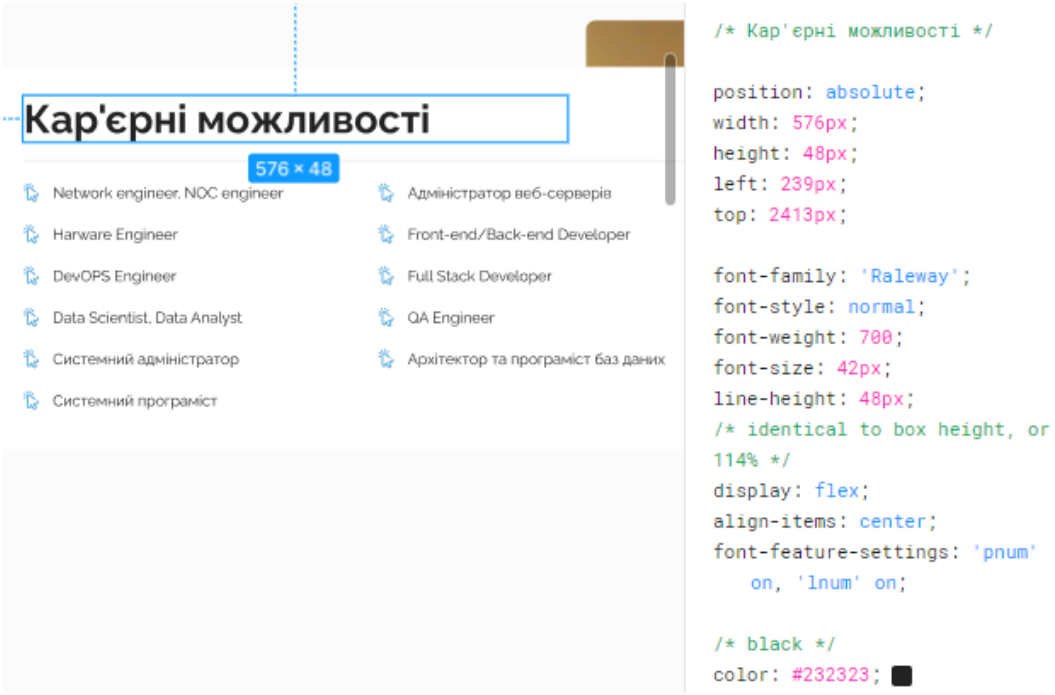


Рисунок 2.8 — Відображення основних стилів блоку у сервісі Figma

Після проведення аналізу макету було знайдено декілька однакових блоків та компонентів, а саме: хедер, навбар, бургер-меню, заголовки та футер. Ці компоненти будуть використовуватися декілька разів, тому зберігаються у каталозі modules.

Такі компоненти, як хедер, навбар, бургер-меню та футер будуть присутні на кожній сторінці. Тому дані блоки будуть створенні як окремий компонент. Який в подальшому буде імпортований в файл app.js. Це дозволить не створювати декілька разів однакові блоки на кожній з сторінок.

Також під час аналізу макету було знайдено блоки заголовків, які мають однакову стилізацію, але відрізняються наповненням. Тому буде створений окремий компонент, який буде відображати заголовок на будь якій сторінці. Для цього достатньо тільки викликати даний компонент і передати в нього текст з яким він з’явиться на сторінці.

2.3.2 Створення та програмна реалізація окремих компонентів для веб-клієнту

Підключення іконок з форматом SVG відбувається за допомогою тегу svg, таке підключення робить код дуже громістким. Задля уникнення цієї проблеми в каталозі modules було створено додатковий компонент SvgSprite в якому будуть зберігатися всі іконки (рисунок 2.9). Кожній з цих іконок було присвоєна ID, за допомогою, якої в подальшому її можна буде знайти та підключити у файл іншого компонента.

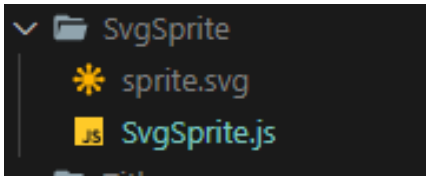


Рисунок 2.9 — Вміст папки SvgSprite

В папці SvgSprite знаходяться два файли sprite.svg та SvgSprite.js. В першому файлі безпосередньо зберігаються усі іконки які використовуються на сторінках. А файл SvgSprite.js містить логіку яка знаходить, вибирає потрібну іконку та повертає вибраний svg файл (лістинг 2.1).

Лістинг 2.1 — Імпорт зображень, стилів та модулів

```
const SvgSprite = ({className, id}) => {
    return (
        <svg className={className}>
            <use href={`${sprite}#${id}`} />
        </svg>
    )
}
```

2.3.2.1 Створення компоненту Header

Верхня частина головної сторінки буде називатися header. При переході на інші сторінки даний блок змінюватися не буде. Тому в каталозі

modules створюємо папку з назвою Header. В ній описані два файли, необхідних для відображення компонента та його стилізації (рисунок 2.10).

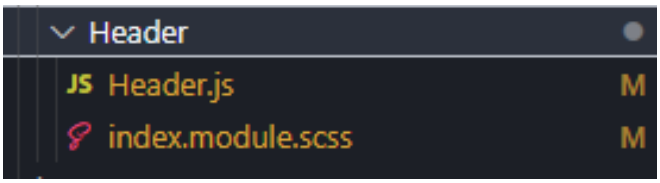


Рисунок 2.10 — Вміст папки Header

Перш за все потрібно проаналізувати цей блок та експортувати зображення та іконки. Також потрібно завантажити шрифт та підключити його у проект.

Зображення будуть зберігатися в папці images, а шрифти в папці Fonts. Обидві ці папки розміщені в каталозі додаткових файлів assets.

Другим кроком буде імпортування всіх зображення та стилів в header.js для того щоб їх було зручно підключати (лістинг 2.2).

Лістинг 2.2 — Імпорт зображень. стилів та модулів

```
// Images
import logoVntu from '../../assets/images/logoVntu.png'
import logoOt from '../../assets/images/logoOt.png'
// Styles
import styles from './index.module.scss'
// Components
import BurgerMenu from '../BurgerMenu/BurgerMenu'
import HeaderLangs from './HeaderLangs'
import SvgSprite from '../SvgSprite/SvgSprite'
```

Згідно з макетом максимальна ширина головної сторінки 1600px, тому було створено обмежувальний блок div з класом container. Далі в файлі index.module.scss описуємо стилі для класу container.

В подальшу вся розробка даного блоку буде відбуватися в середині обмежувального блоку.

Далі потрібно розбити блок header (шапка сайту) на підблоки для зручного маніпулювання розташуванням блоків, а в подальшому і його адаптації під різні пристрої (рисунок 2.11). Даний блок буде містити в собі блок з логотипами ФІТКІ та ОТ, блок з пошуковою стрічкою та кнопкою пошуку, блок зі зміною мови на сторінках, блок з контактами та блок з реалізацією бургер-меню.

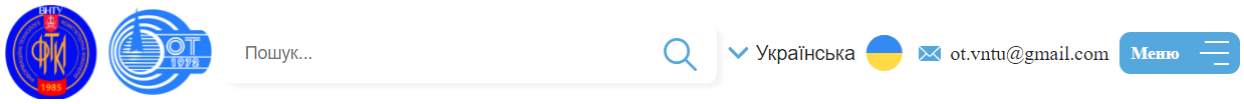


Рисунок 2.11 — Header веб-кліента

Але перед тим, як створювати блоки конкретних елементів шапки потрібно створити ще один загальний блок з класом mainHeader. Цей блок існує для того, щоб відобразити усі елементи блоку header в рядок та відцентрувати їх.

Логотипи ФІТКІ та ОТ будуть знаходитися в одному спільному блоці. В цьому блоці кожен з логотипів буде обгорнутий ще одним блоком. Це робиться для того, щоб можна було зручно налаштувати правильне відображення зображення на сторінці.

Далі потрібно зверстати пошукову стрічку. Даний блок буде складатися з тега form в якому буде створено пошукову стрічку та кнопку для пошуку (Лістинг 2.3).

Наступним кроком буде створення списку з двома розділами, в якому можна буде вибрати українську або англійську мову в інформаційній системі. Для цього було створено файл HeaderLangs.js в папці Header так, як даний блок потребує певної логіки та налаштувань.

У даному блоці буде можливість обрати українську чи англійську мову яка буде відображатися у інтерфейсі. Тому перш за все у файлі HeaderLangs.js створюємо масив з двома об’єктами (лістинг 2.4). Кожен з об’єктів буде мати два поля - назву та ID іконки.

Лістинг 2.3 — Блок з пошуковою стрічкою та кнопкою

```
<div className={style.headerSearch}>
 <form>
  <input type="search" name="name" placeholder="Пошук..." />
  <button type='submit'><img src={searchBtn} /></button>
 </form>
```

Лістинг 2.4 — Масив з об’єктами

```
const langs = [
        {name: 'Українська', iconID: 'ukraine'},
        {name: 'English', iconID: 'england'},
]
```

За допомогою хука useState створюємо два стани для даного блоку. Перший стан буде відповідати за відображення вибраного об’єкту, тобто вибраної мови. Другий стан буде відповідати за висвітлення списку в якому можна обрати іншу мову. Також створено інструкцію яка буде змінювати мову при виборі іншого елементу масиву (лістинг 2.5).

Перший стан має змінну activeLangID та функцію setActiveLangID. Змінна activeLangID за замовчуванням буде містити в собі ID першого об’єкта масиву langs. А функція setActiveLangID буде змінювати стан.

Другий стан має булеву зміну openedDropdown, яка по замовчуванню буде мати значення false та функцію setOpenedDropdown, яка буде змінювати стан.

Лістинг 2.5 — Стани блоку вибору мови

```
const [activeLangID, setActiveLangID] =
useState(langs[0].iconID)
    const [openedDropdown, setOpenedDropdown] =
useState(false)
    const changeLang = id => {
        setActiveLangID(id)
        setOpenedDropdown(false)
    }
```

Далі тегом div створюється звичайний блок, в якому буде створена логіка поведінки блоку з відображенням мов. В створеному блоці створюємо ще один блок на який підключаємо слухача подій onClick, який дозволить викликати функцію setOpenedDropdown в яку потрібно передати змінну openedDropdown в інверсії (лістинг 2.6).

Далі в цьому блоці викликається іконка та до неї підключаються класи з стилями. Класс headerLangsArrow буде підключений завжди, а клас headerLangsArrowActive тільки тоді коли на цей блок натиснуть. Дана частина коду відповідає за інтерактивне відображення іконки.

Після стрілочки на екран потрібно вивести назву та іконку вибраної мови. Для цього було використано функцію map, яка буде перевіряти кожен об'єкт масиву поки не знайде потрібний. Якщо функція map знаходить об'єкт, який підпадає під вираз activeLangID === lang.iconID, то створюється тег React.Fragment, який являється спеціальним тегом, що не відображається у веб-інтерфейсі. При натисканні на даний блок повинен з'являтися список в якому можна буде обрати потрібну мову.

Лістинг 2.6 — Створення блоку вибору мови

```
<div className={styles.headerLangsItem} onClick={() =>
setOpenedDropdown(!openedDropdown)}>
      <SvgSprite className={cn({
            [styles.headerLangsArrow]: true,
            [styles.headerLangsArrowActive]: openedDropdown,
                  })} id={'arrow'}/>
      {
            langs.map((lang) => activeLangID === lang.iconID
&& (<React.Fragment key={lang.iconID}>
                  {lang.name}
       <SvgSprite className={styles.headerLangsFlag}
id={lang.iconID}/>
                        </React.Fragment>))}
 </div>
```

Для того щоб створити такі функціональні можливості звертаємося до змінної openedDropdown і якщо зміна буде мати значення true, то створиться блок в якому реалізовано відображення та можливість вибору потрібної мови (лістинг 2.7).

В даному блоці коду знову реалізована функція map яка проходить по всьому масиву та шукає об'єкт який задовільнить логічний вираз activeLangID !== lang.iconID. Після знаходження такого об'єкту, на екрані з'явиться список з іншою мовою. Також на цей блок підключаємо слухача подій, який при натиску на даний блок викличе функцію changeLang, яка приймає значення lang.iconID та передає його в функцію setActiveLangID.

Далі був створений та стилізований блок в якому відображаються контакти для зворотного зв'язку. Даний блок складається з тегу <a>, а в середині цього тегу створені ще два тега. Перший тег SvgSprite для підключення іконки, а другий span в якому буде відображена електрона адреса.

Лістинг 2.7 — Реалізація можливості вибору мови

```
openedDropdown && (
     <div className={styles.headerLangsDropdown}>
      {
        langs.map(lang => activeLangID !== lang.iconID && (
                  <div className={styles.headerLangsItem}
onClick={() => changeLang(lang.iconID)} key={lang.iconID}>
                       {lang.name}
                       <SvgSprite className=
{styles.headerLangsFlag} id={lang.iconID}/>
                  </div>
                            ))
      }
     </div>
)
```

Останній блок в header буде викликати меню-бургер, тобто спеціальну панель, яка буде слугувати картою сайту. Для того, щоб даний блок відкривав

меню було створено хук useState, який містить в собі булеву зміну menuActive, яка має значення false та функцію setMenuActive, яка буде змінювати стан. Далі на цей блок було підключено слухача подій onClick. При натисканні на цей блок викликається функцію setMenuActive, яка приймає значення !menuActive.

Далі для реалізації відкриття меню викликаємо зміну menuActive (лістинг 2.8).

Лістинг 2.8 — Умова для відображення бургер-меню

```
{menuActive && <BurgerMenu setActive={setMenuActive}/>}
```

Якщо ця зміна буде мати значення true, то на екран буде виведено компонент BurgerMenu.

2.3.2.2 Створення BurgerMenu

BurgerMenu складається зі списку в якому містяться всі необхідні посилальні пункти. Кожен такий пункт буде складатися з тега Link, який імпортується з react-router-dom та іконки (лістинг 2.9).

Також в бургер-меню реалізована кнопка для його закриття, а точніше зміни стану даного компонента.

Лістинг 2.9 — Створення одного з елементів списку бургер-меню

```
<li className={styles.menuListItems}>
        <Link className={styles.menuListItem} to='/news'>
              <SvgSprite id={'menuIcon2'}/>
               Новини
        </Link>
</li>
```

Ця кнопка складається з блоку div та містить в собі два тега span за допомогою яких буде створений хрестик. При натисканні на який спрацьовує слухач подій onClick, який передасть в зміну setActive значення false, що змінить стан блоку и меню зникне.

2.3.2.3 Створення компоненту Navbar

Для зручного створення компоненту Navbar було створено масив з об'єктами. Кожен з цих об'єктів має поля name та link. В навбарі присутні випадні списки, тому деякі об'єкти масиву будуть мати масиви nestedMenu, в яких теж будуть знаходитися об'єкти з полями name та link (лістинг 2.10).

Далі розроблюється логіка виводу елементів навбару на екран. Для цього необхідно створити список в якому викликається масив menuItems та функція map, яка пройде по кожному елементу масиву.

Лістинг 2.10 — Створення об'єктів масиву

```
const menuItems = [
    {name: 'Головна', link: '/'},
    {
       name: 'Про кафедру',
       nestedMenu: [
           {name: 'Історія кафедри', link: '/history'},
           {name: 'Викладачі', link: '/teachers'},
           {name: 'Випускники', link: '/graduates'},
           {name: 'Стейкхолдери', link: '/stakeholders'},
           ],
    }
```

Об'єкти масиву, які мають масив будуть виводитися по іншому, тому для цього створюємо перевірку, яка буде перевіряти чи в об'єкті є поле з масивом.

За умови, якщо об'єкт не має масиву, елемент буде виведено, як показано в лістингу 2.11.

Тег NavLink підключається з бібліотеки react-router-dom. Цей тег використано для того, щоб налаштувати навігацію. Для кожного посилання за допомогою атрибута to визначається шлях переходу. Особливістю тега NavLink є те, що він дозволяє використовувати стан посилання.

Лістинг 2.11 — Створення одного з елементів списку бургер-меню

```
<NavLink className={({isActive}) => isActive ?
styles.navbarMenuLinkActive : styles.navbarMenuLink}
to={menuItem.link}>
      {menuItem.name}
</NavLink>
```

Тобто за допомогою булевої змінної isActive можна надавати активному елементу списку іншу стилізацію. Змінна isActive перевіряє вміст атрибута to та зрівнює його з тим посиланням, яке відображається в URL-адресі. Якщо значення змінної буде мати значення false, то до елементу буде підключено клас navbarMenuLink, в іншому випадку буде підключено клас navbarMenuLinkActive.

При умові, що об’єкт має масив буде створений елемент списку в який буде підключено два класи зі стилями navbarMenuItem та navbarMenuItemDropdown. Далі всередині елементу створюються два блока div. В першому блоці реалізовано виведення назви випадного списку та іконки стрілочки. В другому блоці реалізовано виведення на екран елементів випадних списків (лістинг 2.12).

Лістинг 2.12 — Реалізація блоку з випадним списком

```
menuItem.nestedMenu.map(nestedItem =>
       !nestedItem.anotherSite ? (
          <NavLink to={nestedItem.link}
key={nestedItem.name}>
      {nestedItem.name}
          </NavLink>
                    ) : (
          <a href={nestedItem.link} target={'_blank'}
key={nestedItem.name}>
              {nestedItem.name}
          </a>)
)
```

У випадному списку деякі посилання ведуть на внутрішні сторінки інформаційної системи, використовуючи React Router. Також є посилання які ведуть на інші веб-ресурси. Тому в масиві menuItems для посилань на інші веб-ресурси було створено ще одне поле anotherSite, який буде мати значення true (лістинг 2.13). За допомогою цієї змінної буде відбуватися пошук елементів масиву, які ведуть на інший ресурс.

Лістинг 2.13 — Вміст об’єкту з посиланням на інший ресурс

```
{
    name: 'Ми в JetIQ',
    link: 'https://vntu.edu.ua/',
    anotherSite: true
}
```

Далі в блоці з випадним списком реалізовано перевірку змінної anotherSite. Створюється умова в якій реалізована інверсія даної змінної. Тобто, якщо значення змінної дорівнює true, то воно перетвориться на false і виведеться посилальний елемент тегом <a>. Елементи масиву де поля anotherSite немає виведуться через тег NavLink так, як значення змінної буде false, а за допомоги інверсії даний вираз буде мати протилежне значення.

2.3.2.4 Створення пагінації

Пагінація — це розбиття великої кількості інформації на не великі за об’ємом сторінки з відображенням їх нумерації.

Пагінація буде використовуватися на декількох сторінках, тому вона буде створена як окремий компонент. В каталозі modules створюємо папку Paganition, в якому буде зберігатися 3 файли, а саме: index.module.scss, Pagination.js, PaginationItem.js. В файлі index.module.scss будуть розміщені стилі для пагінації, а в файлах Pagination.js та PaginationItem.js буде розроблена логіка поведінки пагінації.

У файлі PaginationItem.js буде реалізовано виведення елементу відповідно до результатів обчислень які відбуваються в файлі в файлі Pagination.js.

Для створення логіки виведення нумерації для пагінації в файл Pagination.js потрібно передати інформацію про кількість сторінок, номер активної сторінки, інформація чи є в активної сторінки наступна та попередня сторінка.

Після отриманих даних створюється блок, в якому першим кроком буде перевірка того чи дані про сторінки передалися. Якщо дані передалися, то створюється список, в якому будуть викликатися компонент PaginationItem.js та проходити певну логіку для правильного відображення блоків. Крайніми боками даного списку будуть стрілочки за допомогою яких можна переходити на наступну або попередню сторінку, якщо такі існують (лістинг 2.14).

Лістинг 2.14 — Логіка для стрілочки пагінації

```
<PaginationItem
        link={`/news/${info.pageNumber - 1}`}
        prevArrow
        disabled={info.pageNumber === 1}
/>
```

В компонент PaginationItem передається три параметри. Параметр link містить посилання на попередню сторінку, компонент prevArrow створений для того, щоб компонент PaginationItem розумів, що в даний блок потрібно вивести стрічку направлену в ліво. Третій параметр є булевим значенням, яке створене для перевірки того чи активна сторінка має номер 1, якщо це значення буде true, то стрілочка буде мати інші стилі.

Код виведення блоків нумерації сторінок подано у додатку Д.

Для присвоєння відповідних класів блокам нумерації сторінок у файл PaginationItem.js передаються такі параметри (лістинг 2.15).

Параметри prevArrow, nextArrow відповідають за знаходження блоків з стрілочками, яким підключають відповідну іконку стрілочки.

Лістинг 2.15 — Параметри компоненту PaginationItem

```
PaginationItem = ({prevArrow, nextArrow, disabled, link, children, active, separator})
```

Параметри disable та active будуть змінювати стилі крайніх блоків списку, якщо користувач знаходиться на першій або останній сторінці відповідно.

Параметр link містить в собі адресу сторінки на яку буде перенаправлений користувач при натисканні на блок з цим посиланням.

Параметр children відповідає за вміст тегу компонента.

При передачі параметру separator буде виведено блок з трьома точками.

Даний блок реалізує наступну логіку (лістинг 2.16):

Лістинг 2.16 — Реалізація логіки компонента PaginationItem

```
{
  separator ? (
     <div className={styles.paginationItemSeparator}>
           <span />
           <span />
            <span />
     </div>
               ) : (
     <Link className={cn({
         [styles.paginationLink]: true,
         [styles.paginationLinkDisabled]: disabled,
         [styles.paginationLinkActive]: active,
                    })} to={link}>
         {prevArrow && <SvgSprite spriteID={'prevArrow'}/>}
         {nextArrow && <SvgSprite spriteID={'nextArrow'}/>}
         {children}
     </Link>)}
```

Якщо компонент буде мати параметр separator, то виведеться блок з трьома точками. В іншому випадку тегом Link буде створенно посилання, яке буде мати різні класи в залежності від переданих в нього параметрів.

2.3.3 Реалізація сторінок інформаційної системи

Для створення сторінки з новинами спочатку потрібно їх отримати з серверної частини. Для реалізації отримання та виведення на екран новин потрібно створити функцію getNews, яка буде викликатися в функції useEffect для того, щоб оновлювати новини при переході між сторінками (лістинг 2.17).

Лістинг 2.17 — Використання хука useEffect

```
useEffect(() => {
        getNews()
    }, [page])
```

В середині функції getNews створюємо дві змінні response та news. Змінна response буде приймати відповідь з сервера, яку потрібно конвертувати в об’єкт. Даний об’єкт буде записаний в змінну news. Далі створюється об’єкт setPaginationInfo. В ньому буде зберігатися 4 поля, які будуть містити інформацію про кількість сторінок, активну сторінку, інформацію про те чи є у активної сторінки попередня та наступна сторінка (лістинг 2.18).

Лістинг 2.18 — реалізація функції getNews

```
const getNews = async () => {
    const response = await fetch(`https://departament.slivki-cat.com/PagedNews?page=${page}&PageSize=${pageSize}`)
        const news = await response.json()
        setPaginationInfo({
            pageNumber: news.pageNumber,
            totalPages: news.totalPages,
            hasPreviousPage: news.hasPreviousPage,
```

Продовження лістингу 2.18

```
                hasNextPage: news.hasNextPage,})

        setNews(news.itemsAtPage)
```

Реалізацію виведення новин на сторінку представлено на додатку Е.

2.3.4 Вимоги до системних можливостей для функціонування веб-клієнту

Інформаційна система навчального закладу реалізована у вигляді веб-сайту. Тому для її використання необхідно лише мати персональний комп'ютер, ноутбук або смартфон з мінімальними апаратними можливостями, на які можна встановити веб-браузер.

Системні вимоги:

— операційна система: Windows XP/Vista/7/8/10, Mac OS X 10.6 або вище;

— процесор: Intel Pentium 4/Athlon 64 або пізнішої версії з підтримкою SSE2.

— вільного місця на диску: не менше 350 Мб;

— оперативна пам'ять: не менше 512 Мб;

— відеокарта: інтегрована.

Також дана інформаційна система буде мати певні системні вимоги до хостингу. Для цієї системи мінімальних характеристик хостингу буде достатньо.

Основні мінімальні характеристики хостингу:

— обсяг SSD-диску – 30 ГБ;

— трафік 100 ГБ;

— доступ до GIT;

— близько 10000 візитів на місяць.

3 ПРОЕКТУВАННЯ ВЕБ-КЛІЄНТСЬКОГО ІНТЕРФЕЙСУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Мапа клієнтського інтерфейсу

Основною метою інформаційної системи є надання користувачу зручного та швидкого доступу до інформації навчального відділу. Тому було розроблено інтерфейс, в якому перехід між сторінками відбувається дуже зручно та зрозуміло.

Після розробки веб-клієнту інформаційної системи було створено мапу інтерфейсу користувача (рисунок 3.1).

Рисунок 3.1 — Мапа інтерфейсу користувача

Особливістю даного інтернет ресурсу є те, що на сторінках розміщено два навігаційні блоки — навігаційне меню (навбар) та бургер-меню. Вони

дозволяють легко переміщатися по сторінкам ресурсу з будь-якої іншої сторінки інформаційної системи.

Навбар — це горизонтальна навігація, яка слугує для швидкого розуміння наповнення веб-ресурсу та тих можливостей, що в ньому передбачені.

Бургер-меню призначене для розташування більшої кількості посилань, кращого їх стилістичного оформлення та широких можливостей для адаптування під різні види пристроїв.

За рахунок використання бібліотеки React Router в адресній стрічці браузера можна буде побачити назву сторінки, на якій знаходиться користувач в даний момент часу. Це зроблено для зручності користувача, щоб він міг бачити де він знаходиться. Далі наведено список усіх кінцевих сторінок інформаційної системи:

— /news;
— /history;
— /teachers;
— /graduates;
— /stakeholders;
— /entrant;
— /materials;
— /opp;
— /postgraduate;
— /journal;
— /methodical
— /developments

Вказана сторінка з новинами (/news) може містити в собі багато новин, тому було прийнято рішення розбити сторінку на частини за допомогою пагінації (рисунок 3.2).

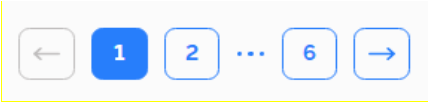


Рисунок 3.2 — Вигляд вікна пагінації на сторінці

Пагінація — це розбивка великого масиву даних на невеликі за об’ємом сторінки з відображенням нумерованої навігації по ним.

3.2 Детальний огляд основних сторінок веб-додатку

Практична реалізація веб-клієнту виконувалась для кафедри обчислювальної техніки факультету інформаційних технологій та комп’ютерної інженерії ВНТУ.

Перша та основна сторінка розробленого веб-клієнту — це головна сторінка. На ній розміщена загальна інформація, а саме інформація про освітні програми, основні відомості про спрямування спеціальностей, випускників, актуальні новини та кар’єрні можливості після закінчення навчання.

Вигляд головної сторінки веб-клієнту показано на рисунку 3.3.

На кожній сторінці розроблюваного веб-клієнту будуть знаходитися статичні блоки — header, footer та navbar. Вони не будуть змінюватися при переході між сторінками тому, що користувач завжди повинен мати доступ до основних елементів, навігації та контактної інформації інформаційної системи.

В шапці знаходяться певні функціональні елементи, які допоможуть краще зорієнтуватися користувачу в інформаційній системі, а саме пошукова стрічка, блок з можливістю змінювати мову та бургер-меню.

За допомогою пошукової стрічки користувач зможе знаходити потрібні новини. Користувач вписує в пошукову стрічку ключові слова, які будуть використовуватися для пошуку потрібної йому новини. Далі робиться запит на серверну частину в якому передається інформація, яку ввів користувач. В серверній частині реалізований пошук новин за їх заголовком. Після пошуку серверна частина відправляє знайдені новини користувачу.

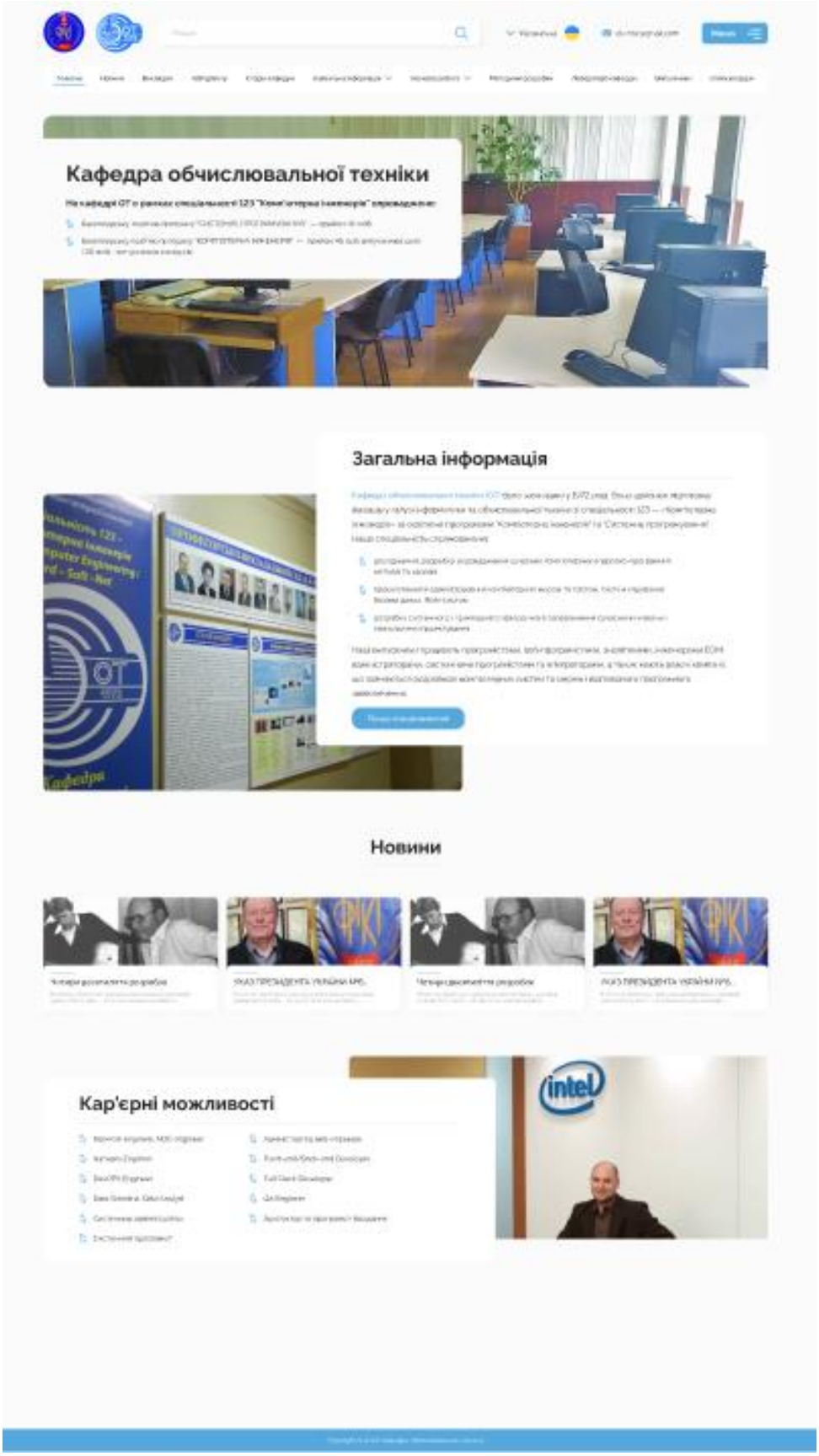


Рисунок 3.3 — Вигляд головної сторінки веб-клієнту

Також користувач завжди повинен мати можливість переключити мову на сторінці (рисунок 3.4), тому блок який за це відповідає знаходиться в шапці.

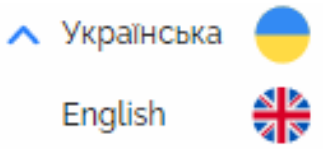


Рисунок 3.4 — Блок з вибором мови

Бургер-меню, вигляд якого показано на (рисунку 3.5), дозволить користувачу зручно доступатися до навігації веб-ресурсу, з якого б пристрою користувач не знаходився в інформаційній системі.

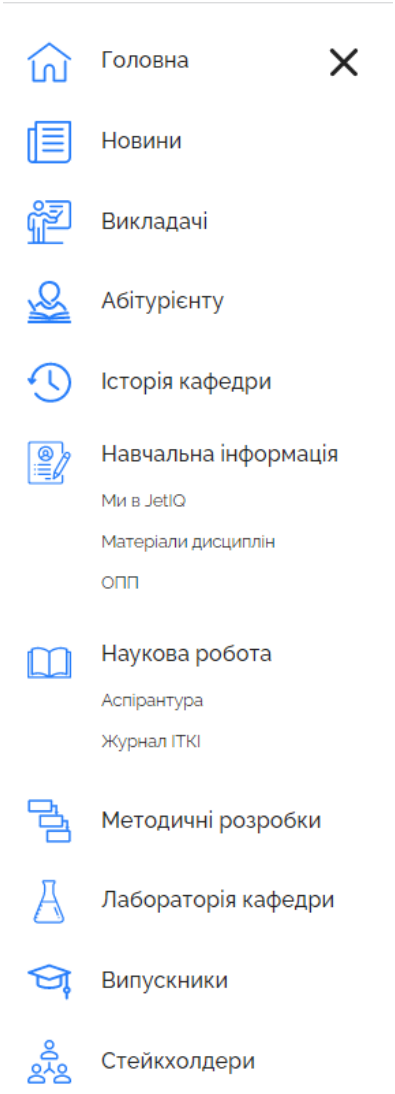


Рисунок 3.5 — Навігаційний блок бургер-меню

Використання такого компоненту значно розширює можливості адаптації інтерфейсу під різні типи пристроїв.

Бургер-меню буде відображатися на екрані будь-якого розміру так, як цей блок має не великий об’єм. Це дозволить завжди мати доступ до навігації сайту — достатньо тільки натиснути на кнопку відкриття і меню відкриється.

При створені сторінки з новинами було реалізовано динамічне виведення новин. Для цього використовувалась технологія Rest API. За допомогою функції fetch API виконується запит на сервер та передається інформація про кількість новин, які мають бути на одній сторінці, а також номер сторінки, на якій знаходиться користувач. Далі сервер відправляє всю необхідну інформацію, а саме новини які знаходяться на відповідній сторінці.

Вигляд тестової сторінки з новинами представлено на рисунку 3.6.

Динамічне додавання новин було реалізовано за допомогою функції fetch (лістинг 3.1).

Лістинг 3.1 — Код функції для отримання новини з серверу

```
const getNews = async () => {
          const      response      =      await      fetch
(`http://localhost:5000/PagedNews?page=${page}&PageSize=${pa
geSize}`)
        const news = await response.json()

        setPaginationInfo({
            pageNumber: news.pageNumber,
            totalPages: news.totalPages,
            hasPreviousPage: news.hasPreviousPage,
            hasNextPage: news.hasNextPage,
        })
        setNews(news.itemsAtPage)
    }
```

Рисунок 3.6 — Вигляд сторінки з новинами

Для отримання новин з серверу створюється функція getNews. В даній функції створюється дві змінні — response та news. В змінну response передається вся інформація з новинами, а в зміну news буде записана та сама інформація але кожна новина буде перетворена на об’єкт.

Об’єкт setPaginationInfo створено для того, щоб передавати інформацію про активну сторінку, максимальну кількість сторінок, попередню та наступну сторінку в модульний блок пагінації.

Об’єкт setNews використовується для зміни стану сторінки, тобто при переході на іншу сторінку масив з новинами буде оновлюватися.

На сторінці з новинами відображаються новини, які відповідають виділеному номеру на модульному блоці з пагінацією. Також, як і на всіх інших сторінках, відображаються блоки header та navbar, що дає можливість зручніше використовувати інформаційну систему та краще орієнтуватися в її структурі.

Кожну новину можна детальніше переглянути. Для цього достатньо тільки натиснути на неї та завантажиться вся інформація, яка відповідає обраній новині.

Приклад відображення новини на сторінці веб-клієнту інформаційної системи представлено на рисунку 3.7.

Також було реалізовано сторінку з відомостями про історію створення кафедри обчислювальної техніки (рисунок 3.8).

Інформації на сторінці історія кафедри реалізована статично. Тобто інформація виводиться на сторінку не з серверу, а безпосередньо розробником представницького рівня. Інформація, яка відображена на статичній сторінці, вводиться під час створення елементів сторінки.

Рисунок 3.7 — Представлення сторінки з новиною

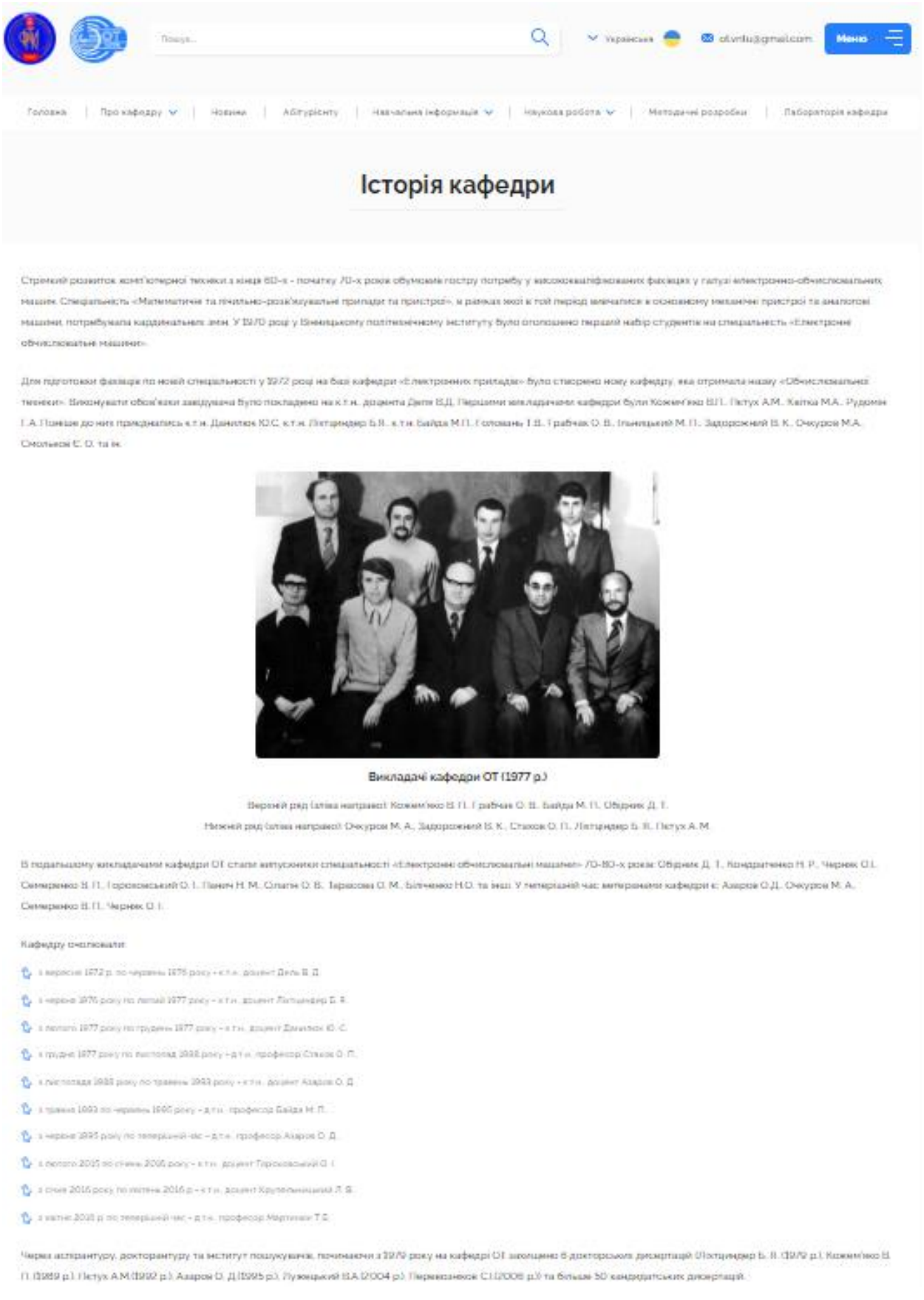


Рисунок 3.7 — Представлення сторінки з історією кафедри

ВИСНОВКИ

У даній комплексній бакалаврській дипломній роботі розроблено та програмно реалізовано веб-клієнтську частину клієнт-серверної інформаційної системи структурного підрозділу навчального закладу з використанням мови програмування JavaScrip та реактивного фреймворку React. Веб-клієнт розроблявся для кафедри обчислювальної техніки ВНТУ.

В роботі проаналізовано найпопулярніші технології для розробки веб-додатків. Розглянуто та порівняно такі популярні на сьогодні фреймворки, як React та Angular, їх функціональні можливості, основні відмінності, переваги та недоліки. Після порівняння та аналізу цих фрейморків було вирішено використовувати саме React, оскільки використання React дасть можливість розробити більш швидку та оптимізовану інформаційну систему у порівняні з Angular. Також було розглянуто БЕМ методологію для створення веб-ресурсів та сучасні засоби для комплектування модулів програмного продукту.

Спроектовано фізичну структуру клієнтської частини інформаційної системи структурного підрозділу навчального закладу з можливістю подальшого масштабування веб-клієнту. Під час розробки та програмної реалізації веб-клієнту було використано компонентний підхід. Це дало можливість зручно та швидко розробляти сторінки та блоки веб-клієнту шляхом підключення компонентів саме там, де вони необхідні. В подальшому це значно спростить адміністрування та можливість масштабування інтерфейсу інформаційної системи. Також за допомогою бібліотеки React Router був налаштований швидкий перехід між сторінками без перезавантаження сторінки. Описано функціональні можливості веб-клієнту, а саме пошук новин, зміна мови на сторінці, пагінація та динамічне додавання інформації, отриманої з серверу. Отже, задачі, поставлені в комплексній бакалаврській дипломній роботі, виконано повністю.

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

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Аналіз технологій створення веб-клієнту за допомогою реактивного фреймворка React / В.М. Іванов, О.В. Войцеховська. Матеріали LI наукової-технічної конференції підрозділів Вінницького національного технічного університету (Вінниця, 2022 р.) [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/author/submission/15736.

2. Гіпертекстова розмітка HTML. [Електронний ресурс]. Режим доступу: http://www.znannya.org/?view=html_teach

3. Каскадні таблиці стилів. [Електронний ресурс]. Режим доступу: https://naurok.com.ua/urok-kaskadni-tablici-stiliv-css-90218.html

4. Препроцесори CSS. [Електронний ресурс]. Режим доступу: https://vyspiansky.gitbook.io/introduction-to-web-development/css/preprocessors

5. Препроцесор Sass. [Електронний ресурс]. Режим доступу: https://sass-scss.ru/documentation/

6. Мова JavaScript та її можливості. [Електронний ресурс]. Режим доступу: https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti

7. Фреймворк React. [Електронний ресурс]. Режим доступу: https://uk.reactjs.org/docs/getting-started.html

8. Virtual DOM. [Електронний ресурс]. Режим доступу: https://www.codecademy.com/article/react-virtual-dom

9. React Components. [Електронний ресурс]. Режим доступу: https://www.xenonstack.com/blog/functional-vs-class-components

10. Hooks API Reference. [Електронний ресурс]. Режим доступу: https://reactjs.org/docs/hooks-reference.html?no-cache=1#usestate

11. React Router. [Електронний ресурс]. Режим доступу: https://o7planning.org/12137/undertanding-react-router-with-example-on-the-client-side

12. Методології CSS. [Електронний ресурс]. Режим доступу: https://webformyself.com/css-metodologii-css-bem-smacss-ecss/

13. Методологія БЕМ [Електронний ресурс]. Режим доступу: https://wiki.cuspu.edu.ua/index.php/Методологія_БЕМ

14. Основи Webpack [Електронний ресурс]. Режим доступу: https://codeguida.com/post/454

15. Gulp [Електронний ресурс]. Режим доступу: https://gulpjs.com/

16. Figma [Електронний ресурс]. Режим доступу: https://wezom.academy/ua/chto-takoe-figma-funktsii-instrumenty-ipreimuschestva/

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 бакалаврської дипломної роботи

«Клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 4. «Розробка веб-клієнту за допомогою реактивного фреймворка React»

08-23.КБДР.047.00.000 ТЗ

Керівник роботи: к.т.н.,доц. каф. ОТ.

_____Войцеховська О.В.

Виконав: студент гр. 1КІ-20мс

_____Іванов В. М.

2022

1 Підстава для виконання дипломної роботи (ДР):

— актуальність розробки полягає у необідності забезпечення структурного підрозділу навчального закладу веб-клієнтською частиною шляхом розробки інтерфейсу користувача для неї з використанням сучасних технологій та методологій розробки веб-інтерфейсів;

— наказ про затвердження теми бакалаврської дипломної роботи.

2 Мета і призначення ДР:

— метою роботи є розробка веб-клієнтської частини веб-додатку інформаційної системи структурного підрозділу навчального закладу, яка буде складатись з навігаційних блоків та сторінок веб-сайту, які будуть отримувати відповідну інформацію з серверної частини веб-додатку;

— призначення розробки — виконання комплексної бакалаврської дипломної роботи із подальшою підтримкою та розвитком.

3 Джерела розробки:

— макет з дизайном для створення функціонального інтерфейсу користувача;

— дені для наповнення веб-клієнту;

— інформація по технологіям розробки веб-клієнтської частини.

4 Технічні вимоги до виконання ДР:

— реалізувати фізичну структуру веб-клієнту для подальшого його масштабування;

— створення функціонального інтерфейсу згідно макету;

— наповнення компонентів певним функціоналом для кращої взаємодії з користувачем;

— реалізувати динамічне відображення інформації на сторінках.

5 Етапи ДР та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання		Очікувані результати
		початок	закінчення	
1	Аналіз завдання	08.02.22		Задачі дослідження
2	Процес проектування плану створення веб-клієнтської частини.	09.02.22	12.02.22	Аналітичний огляд джерел
3	Огляд технологій створення та оптимізації веб-клієнтських інтерфейсів.	12.02.22	20.02.22	Розділ 1
4	Аналіз макету та вибір підходу його створення.	20.02.22	28.02.22	Розділ 1
5	Проектування файлової структури для веб-клієнту.	29.02.22	14.03.22	Розділ 2
6	Створення та програмна реалізація окремних компонентів для веб-клієнту	15.03.22	21.03.22	Розділ 2
7	Створення основних сторінок веб-клієнтської частини та отримання інформації з серверної частини.	22.03.22	20.04.22	Розділ 2
8	Перевірка працездатності клієнтської частини веб-додатку	21.04.22	15.05.22	Розділ 2
9	Перевірка працездатності серверної частини веб-додатку	16.05.22	30.05.22	Розділ 3
10	Оформлення пояснювальної записки та ілюстративного матеріалу	01.06.22	13.06.22	ПЗ, додатки, презентація

6 Матеріали, що подаються до захисту ДР: пояснювальна записка КБДР, графічні та ілюстративні матеріали, протокол попереднього захисту роботи на кафедрі, відзив наукового керівника, рецензія опонента, анотації до КБДР українською та іноземною мовами, нормоконтроль про відповідність оформлення КБДР 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.

8 При оформлені використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

Технічне завдання до виконання прийняв _____ Владислав Івано

ДОДАТОК Б

Мапа інтерфейсу користувача

Рисунок Б.1 — Мапа інтерфейсу користувача

ДОДАТОК В

Головна сторінка

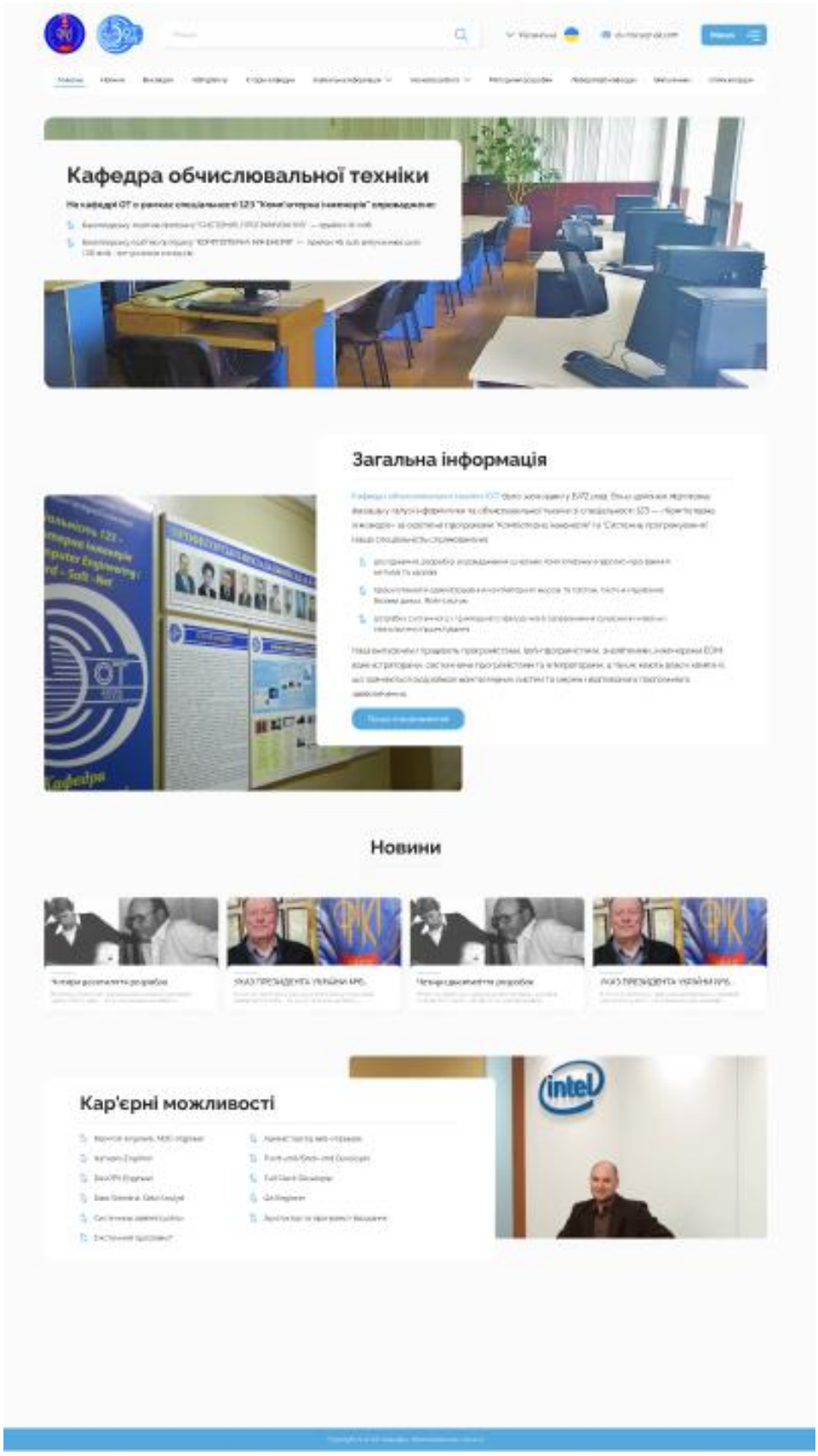


Рисунок В.1 — Вигляд головної сторінки

ДОДАТОК Г

Сторінка з новинами

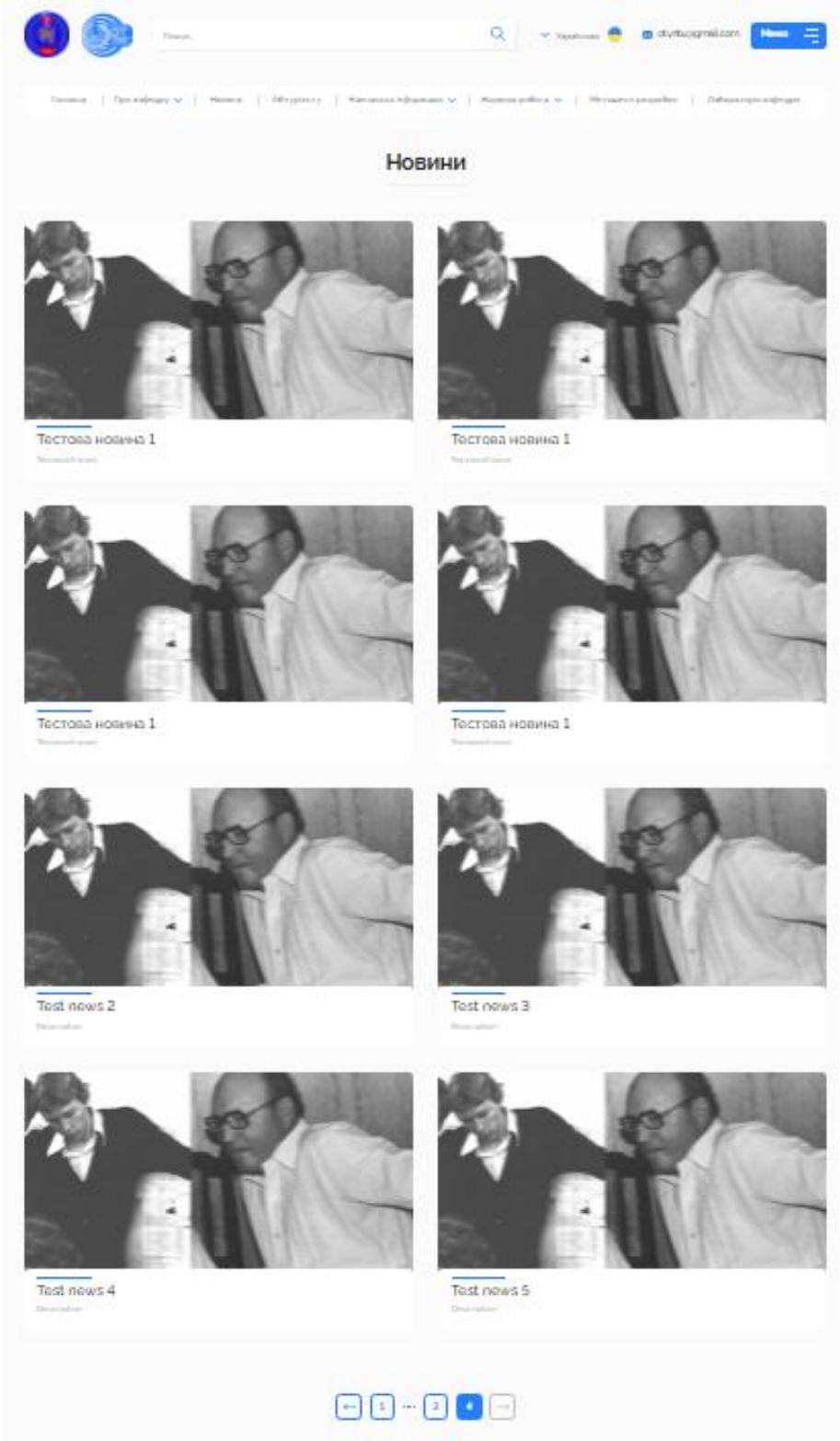


Рисунок Г.1 — Вигляд сторінки з новинами

ДОДАТОК Д

Навігаційний блок бургер-меню

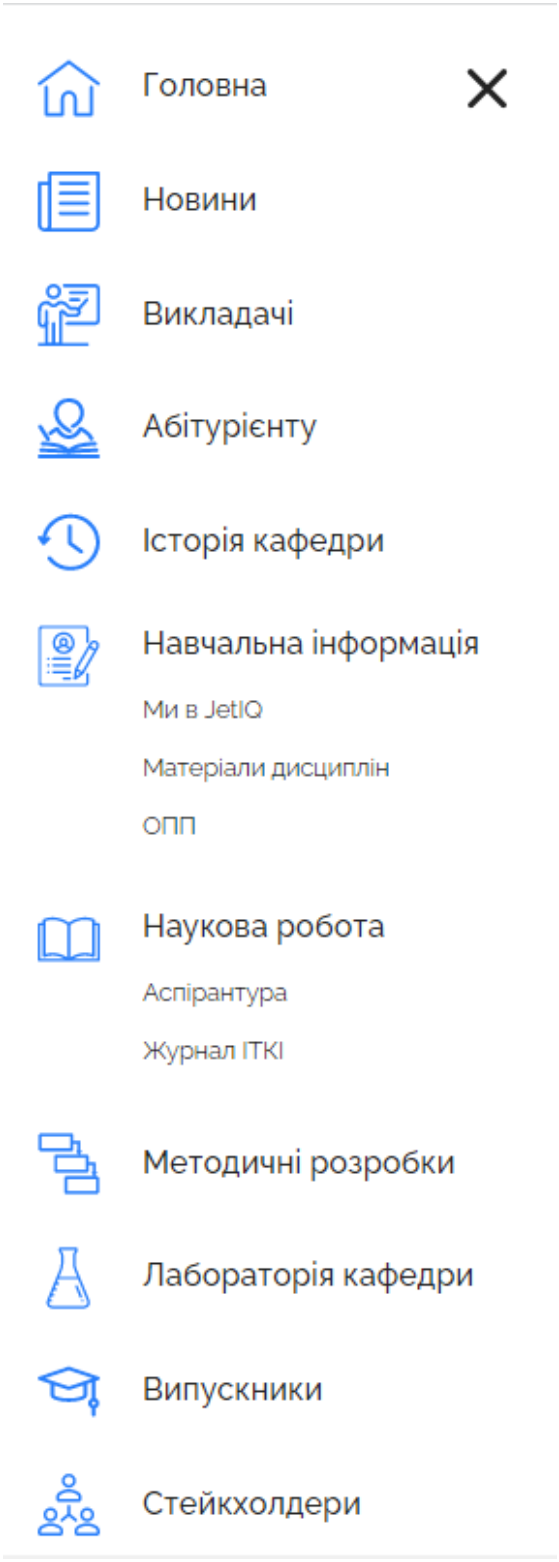


Рисунок Д.1 — Навігаційний блок бургер-меню

ДОДАТОК Е

Програмний код для блоку виведення нумерації сторінок

```
const Pagination = ({info}) => {
  return (
    <div className={styles.pagination}>
      {
        info && (
          <ul className={styles.paginationList}>
            <PaginationItem
              link={`/news/${info.pageNumber - 1}`}
              prevArrow
              disabled={info.pageNumber === 1}
            />
            {info.pageNumber === 3 && <PaginationItem link={`/news/1`}>1</PaginationItem>}
            {
              info.pageNumber > 3 && (<>
                <PaginationItem link={`/news/1`}>1</PaginationItem>
                <PaginationItem separator />
              </>)
            }
            {
              info.pageNumber > 1 && (
                <PaginationItem link={`/news/${info.pageNumber - 1}`}>
                  {info.pageNumber - 1}
                </PaginationItem>
              )
            }
            <PaginationItem link={`/news/${info.pageNumber}`} active>
              {info.pageNumber}
```

```
                </PaginationItem>
                {
                  info.pageNumber < info.totalPages - 1 && (
                    <PaginationItem link={`/news/${info.pageNumber + 1}`}>
                      {info.pageNumber + 1}
                    </PaginationItem>
                  )
                }
                {
                  info.pageNumber < info.totalPages - 2 && (<>
                    <PaginationItem separator />
                    <PaginationItem link={`/news/${info.totalPages}`}>
                      {info.totalPages}
                    </PaginationItem>
                  </>)
                }
                {
                   info.pageNumber < info.totalPages && info.pageNumber >
info.totalPages - 3 && (
                    <PaginationItem link={`/news/${info.totalPages}`}>
                      {info.totalPages}
                    </PaginationItem>
                  )
                }
                <PaginationItem
                  link={`/news/${info.pageNumber + 1}`}
                  nextArrow
                  disabled={info.pageNumber === info.totalPages}
                />
              </ul>
```

```
            )
        }
    </div>
  )
}
```

ДОДАТОК Ж

Програмний код блоку виведення новин

```
return (
    <div className={styles.news}>
      <div className={styles.newsContent}>
        <div className={styles.container}>
          <div className={styles.newsTitleWrapper}>
            <Title className={styles.newsTitle} contents='Новини'/>
          </div>
          <div className={styles.homeNewsItems}>
            {
              news.length > 0 && news.map(newItem => (
                <div className={styles.homeNewsItem} key={newItem.id}>
                  <img
                    src={newItem.imageStorageUrl ? newItem.imageStorageUrl : HomeNewsFirstPicture}
                    alt='HomeNewsFirstPicture'
                  />
                  <div className={styles.homeNewsItemContent}>
                    <h4 className={styles.homeNewsItemTitle}>
                      {newItem.title}
                    </h4>
                    <p className={styles.homeNewsItemText}>
                      {
                        newItem.description.length > 400
                          ? newItem.description.slice(0, 400) + '...'
                          : newItem.description
                      }
                    </p>
                  </div>
```

```
                        </div>
                    ))
                }
            </div>
            <Pagination info={paginationInfo} />
        </div>
    </div>
</div>
)
}
```

ДОДАТОК К

Протокол перевірки кваліфікаційної роботи на наявність текстови запозичень

Назва роботи Клієнт-серверна інформаційна система підрозділу навчального закладу з можливістю розгортання в хмарному середовищі. Частина 4. «Розробка веб-клієнту за допомогою реактивного фреймворка React»

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 92,8% Схожість 7,2%

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)

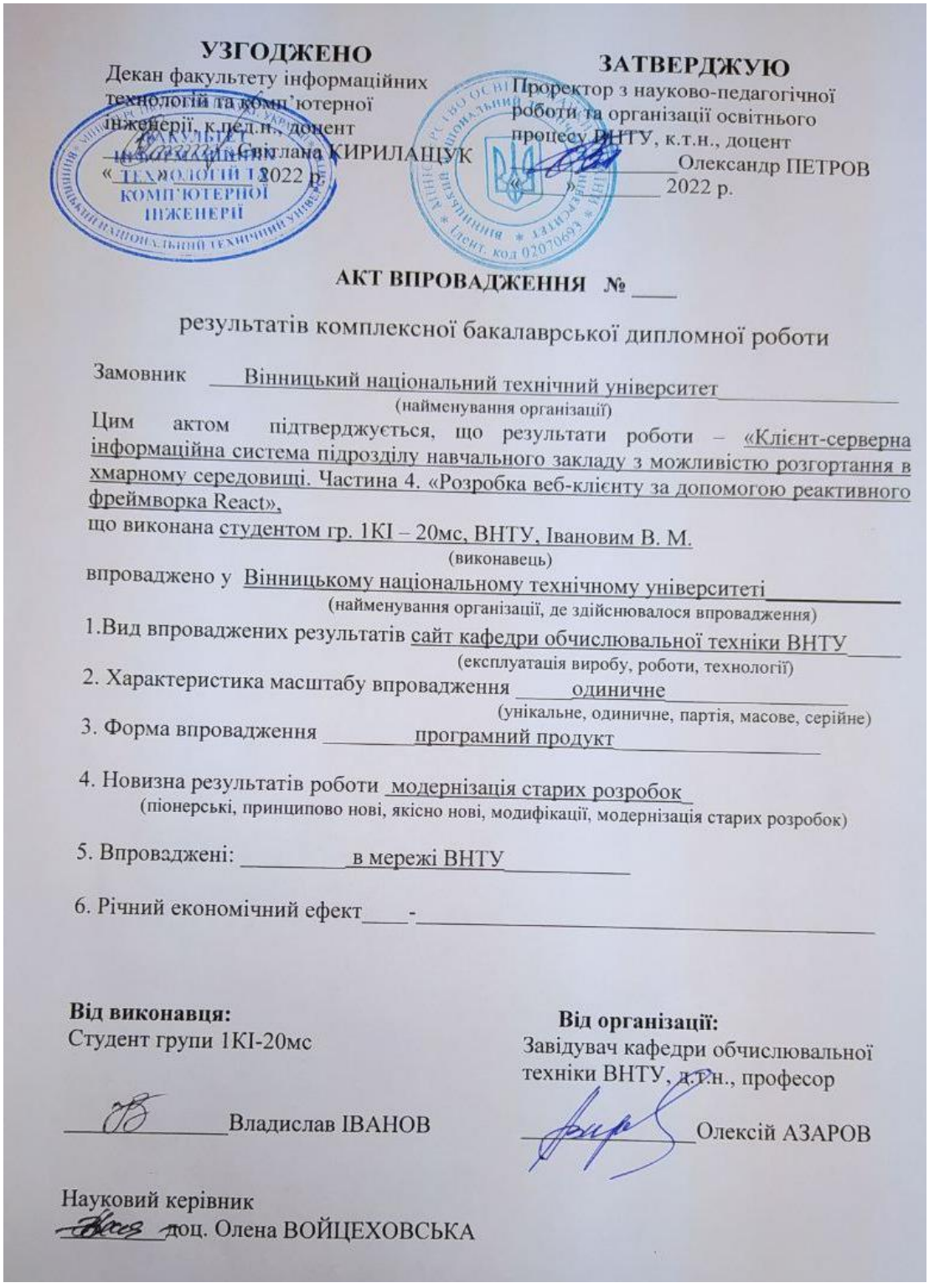
Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Іванов В.М.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Войцеховська О.В.
(підпис) (прізвище, ініціали)

ДОДАТОК Н

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 4. «Розробка веб-клієнту за допомогою реактивного фреймворка React»,
що виконана студентом гр. 1КІ – 20мс, ВНТУ, Івановим В. М.
(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 1КІ-20мс
_____Владислав ІВАНОВ

Від організації:
Завідувач кафедри обчислювальної техніки ВНТУ, к.т.н., професор
_____Олексій АЗАРОВ

Науковий керівник
_____доц. Олена ВОЙЦЕХОВСЬКА