МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально-науковий інститут інформаційних технологій та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавра

на тему: «**Створення веб-застосунку наукового журналу з розширеним пошуком ECJ»**

Виконав: студент 4 курсу, групи КН-41
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Колода Станіслав Геннадійович

Керівник: *викладач кафедри інформаційних технологій та аналітики даних*
Місай Володимир Віталійович

Рецензент: *ФОП в галузі інформаційних технологій, викладач кафедри менеджменту та маркетингу НаУОА*
Шпак Андрій Григорович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри інформаційних технологій та аналітики даних _____
(проф., д.е.н. Кривицька О.Р.)
Протокол №10 від 28 травня 2025 р.

Острог, 2025

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня бакалавра

Тема: Створення веб-застосунку наукового журналу з розширеним пошуком ECJ

Автор: Колода Станіслав Геннадійович

Науковий керівник: викладач кафедри ІТАД, Місай Володимир Віталійович

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *51с., 24 рис., 0 табл., 0 додатків, 16 джерел.*

Ключові слова: *фронтенд, журнал, клієнтська частина, пошук, бекенд, інтерфейс користувача, адаптивний дизайн,*

Короткий зміст праці:
Кваліфікаційна робота присвячена розробці веб-застосунку для наукового журналу "ECJ", який забезпечує ефективну взаємодію між авторами та читачами. У роботі розглянуто побудову системи на основі клієнт-серверної архітектури з використанням React.js для клієнтської частини та Strapi CMS для серверної. Детально описано реалізацію функціонального інтерфейсу з адаптивним дизайном, систему ролей та прав доступу, можливості розширеного пошуку наукових статей. Особливу увагу приділено модульності архітектури та можливості подальшої інтеграції з іншими науковими платформами, що робить систему гнучким і перспективним інструментом для наукової спільноти.

_____

ANNOTATION
of qualification paper
for bachelor's degree

Theme: Development of a Scientific Journal Web Application with Advanced Search (ECJ)

Author: Koloda Stanislav

Scientific supervisor: Lecturer of the Department of ITAD, Misai Volodymyr Vitaliyovych

Defended: «……»…… 20__ year.

Explanatory note to the qualification work: *51 pages, 24 figures, 0 tables, appendices, 16 sources.*

Keywords: *frontend, journal, client-side, search, backend, user interface, responsive design*

Abstract:

The qualification thesis is dedicated to the development of a web application for the scientific journal "ECJ", which ensures effective interaction between authors and readers. The work explores the construction of the system based on a client-server architecture using React.js for the client-side and Strapi CMS for the backend. It describes in detail the implementation of a functional user interface with responsive design, a system of roles and access rights, and advanced scientific article search capabilities. Special attention is given to the modularity of the architecture and the potential for further integration with other scientific platforms, making the system a flexible and promising tool for the academic community.

_____

Зміст

ВСТУП

У сучасному науковому середовищі ефективний обмін знаннями та результатами досліджень є ключовим фактором розвитку науки та освіти. Цифрова трансформація академічних процесів відкриває нові можливості для наукових публікацій, проте значна частина наукових журналів все ще використовує застарілі підходи до організації робочих процесів. Подолання розриву між традиційними методами публікації та сучасними технологічними можливостями стає критично важливим для підвищення ефективності наукової комунікації та поширення знань.

В умовах стрімкого розвитку інформаційних технологій питання доступності та видимості наукових публікацій набуває особливого значення. Традиційні методи публікації наукових статей часто характеризуються високою трудомісткістю процесів рецензування та обмеженою доступністю для широкої аудиторії. Процеси подання матеріалів, їх оцінки, редагування та публікації нерідко залишаються недостатньо автоматизованими, що призводить до значних часових витрат як для авторів, так і для редакційних колегій. Ці обмеження негативно впливають на швидкість поширення наукових результатів та їх інтеграцію у світову наукову спільноту.

Науковий журнал "ECJ" (Economics and Computer Journal) університету, що спеціалізується на публікаціях у сфері економіки стикається з аналогічними викликами. Крім того, обмежені можливості для пошуку та доступу до опублікованих матеріалів зменшують потенційну аудиторію.

Метою даної роботи є розробка веб-застосунку для наукового журналу "ECJ", що забезпечить автоматизацію ключових процесів публікації та спростить взаємодію між усіма учасниками наукової комунікації. Веб-застосунок створить єдину платформу для подання статей, організації процесу рецензування, публікації та доступу до наукових матеріалів.

Реалізація поставленої мети передбачає вирішення наступних завдань:

1. Проведення детального аналізу існуючих процесів публікації та рецензування у журналі "ECJ" та виявлення основних проблемних аспектів;

2. Проектування архітектури системи, що забезпечить надійність, масштабованість та безпеку;

3. Реалізація механізмів автоматизації процесів подання та публікації наукових статей;

4. Впровадження системи пошуку наукових публікацій для забезпечення швидкого доступу до інформації;

Об'єктом дослідження виступає інформаційна система для наукових публікацій та процеси взаємодії між авторами, редакторами, рецензентами та читачами наукового журналу.

Предметом дослідження є процеси автоматизації роботи наукового журналу. Зокрема, досліджуються методи оптимізації редакційних процесів, забезпечення ефективної взаємодії між учасниками та підвищення доступності наукових публікацій за допомогою веб-технологій.

Актуальність дослідження визначається необхідністю модернізації процесів наукової комунікації та підвищення ефективності роботи наукового журналу "ECJ". Сучасні виклики наукового середовища, такі як зростання обсягів інформації та підвищення вимог до якості публікацій, вимагають впровадження інноваційних рішень у сфері наукової комунікації.

Реалізація проекту передбачає використання сучасного технологічного стеку, що включає React.js [1] для розробки інтерфейсу користувача, Strapi [4] CMS для побудови серверної частини, а також додаткові технології для забезпечення, продуктивності та масштабованості системи. Вибір технологій ґрунтується на аналізі їх відповідності функціональним вимогам, можливостей інтеграції та підтримки, а також перспектив довгострокового розвитку системи.

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ

1.1. Постановка проблеми створення веб-застосунку для наукового журналу ECJ

Науковий журнал "Economics and Computer Journal" (ECJ) є важливим елементом наукової екосистеми університету, що забезпечує платформу для публікації наукових досліджень викладачів, аспірантів та студентів. Однак, під час проведення аналізу було виявлено низку критичних проблем, пов'язаних із поточною технічною реалізацією веб-платформи журналу.

На даний момент науковий журнал ECJ функціонує на базі платформи MODX, яка вже не відповідає сучасним стандартам веб-розробки. Проведений аналіз показав, що ця застаріла система створює низку суттєвих обмежень для кінцевих користувачів. Платформа MODX має обмежені можливості для масштабування, недостатню гнучкість у налаштуванні користувацького інтерфейсу та застарілу архітектуру, що ускладнює впровадження нових функціональних можливостей та оптимізацію існуючих процесів.

Під час вивчення поточного стану системи було звернуто особливу увагу на технічні обмеження платформи, які особливо помітні в контексті сучасних вимог до веб-застосунків. Виконавши детальний технічний аналіз, виявлено відсутність підтримки сучасних технологій та фреймворків, обмежені можливості для інтеграції з іншими системами та сервісами, а також проблеми з продуктивністю, які створюють серйозні перешкоди для подальшого розвитку журналу.

Для кращого розуміння потреб користувачів було проведено опитування серед авторів, рецензентів та редакторів журналу. Від користувачів системи надходять регулярні запити щодо необхідності модернізації платформи. Респонденти відзначають складність навігації, недостатню інтуїтивність інтерфейсу та відсутність важливих функціональних можливостей, які є стандартом для сучасних наукових видань. Зокрема, існує потреба в оптимізації процесу подання статей, автоматизації

процедур рецензування, та впровадженні зручних інструментів для відстеження статусу публікацій.

У рамках дослідження було проведено порівняльний аналіз кількох сучасних платформ для наукових журналів, таких як Open Journal Systems (OJS), Scholastica та інші. Це дозволило виокремити ключові функціональні можливості, які повинні бути реалізовані в новому веб-застосунку для ECJ. Також проаналізовано сучасні технологічні стеки, які могли б забезпечити необхідну функціональність, безпеку та масштабованість системи.

Особливу увагу варто приділити тому факту, що журнал ECJ є важливою складовою наукової діяльності економічного факультету університету. Студенти спеціальності "Комп'ютерні науки" мають унікальну можливість застосувати теоретичні знання та практичні навички для створення реального продукту, який принесе користь науковій спільноті. Це не просто навчальний проект - це можливість зробити значний внесок у розвиток університетської наукової інфраструктури.

1.2. Аналіз та вибір інструментів створення веб-застосунку

У процесі проектування веб-застосунку для наукового журналу "ECJ" було проведено ґрунтовний аналіз доступних технологій та інструментів, які могли б забезпечити оптимальне рішення для створення сучасної, функціональної та масштабованої платформи. Вибір правильних інструментів мав критичне значення для успіху проекту, оскільки вони визначають не лише швидкість та ефективність розробки, але й можливості системи в довгостроковій перспективі.

Аналіз та вибір інструментів для фронтенд-розробки

Фронтенд складає основу взаємодії користувача з системою, тому особливу увагу було приділено вибору технологій, які забезпечать інтуїтивно зрозумілий, адаптивний та функціональний інтерфейс. Проведено порівняльний аналіз найпопулярніших технологій та фреймворків:

HTML і CSS

Переваги:

- Основоположні технології для створення веб-сторінок з простим і зрозумілим синтаксисом

- Широка підтримка у всіх браузерах без необхідності додаткових бібліотек

- Відсутність необхідності компіляції або складної конфігурації

- Потужні можливості стилізації з використанням сучасних функцій CSS, таких як Flexbox та Grid

Недоліки:

- Обмежені можливості для створення динамічного та інтерактивного інтерфейсу

- Відсутність вбудованих засобів для управління станом додатка

- Необхідність використання додаткових мов програмування (JavaScript) для забезпечення інтерактивності

- Складність підтримки коду у великих проектах через відсутність компонентного підходу

Angular

Переваги:

- Повноцінний фреймворк з усіма необхідними інструментами для розробки складних веб-додатків

- Чітка структура та архітектура проекту, що забезпечує єдиний підхід до організації коду

- Потужна система залежностей та впровадження (dependency injection)

- Вбудовані інструменти для роботи з формами, HTTP-запитами, анімаціями тощо

- Двостороннє зв'язування даних (two-way data binding), що спрощує роботу з формами

- Підтримка TypeScript, що підвищує якість коду та зменшує кількість помилок

Недоліки:

- Висока складність та круту криву навчання, що потребує значних інвестицій часу
- Надмірна структурованість, яка може бути зайвою для невеликих проектів
- Менша гнучкість у порівнянні з React через більш жорстку архітектуру
- Менша кількість готових компонентів та бібліотек порівняно з React

Vue.js

Переваги:

- Простий та інтуїтивно зрозумілий синтаксис, який полегшує вхід для розробників з різним рівнем досвіду
- Поступова адаптивність - можливість інтегрувати Vue.js в існуючі проекти частково
- Докладна та зрозуміла документація з великою кількістю прикладів
- Ефективна реактивна система для роботи з даними
- Компонентна архітектура, що сприяє повторному використанню коду
- Легка інтеграція з іншими бібліотеками та існуючими проектами

Недоліки:

- Менша екосистема інструментів, плагінів та бібліотек порівняно з React чи Angular
- Менша кількість розробників на ринку, що може ускладнити пошук спеціалістів
- Обмежена підтримка TypeScript порівняно з Angular (хоча і покращується)
- Відносно нижча продуктивність на складних проектах порівняно з React
- Менша кількість успішних кейсів впровадження в масштабних корпоративних проектах

React.js [1]

Переваги:

- Компонентна архітектура, яка забезпечує модульність та повторне використання коду

- Віртуальний DOM, що підвищує продуктивність через оптимізацію оновлення елементів

- Величезна екосистема бібліотек, інструментів та готових рішень

- Гнучкість у виборі додаткових бібліотек та архітектурних рішень

- Хороша сумісність з TypeScript для забезпечення типобезпеки

- Можливість використання хуків (Hooks), що спрощує роботу зі станом компонентів

Недоліки:

- Відносно висока крива навчання, особливо для початківців

- Необхідність використання додаткових бібліотек для функціоналу, який вбудований в Angular

- Часті оновлення та зміни в API, що можуть вимагати додаткових зусиль для підтримки проекту

- Відсутність чітких стандартів організації коду, що може призвести до різних підходів у команді

Вибір фронтенд-інструментів

Після ретельного аналізу всіх варіантів, для розробки клієнтської частини веб-застосунку наукового журналу "ECJ" було обрано React.js [1] з наступних причин:

Компонентний підхід React.js [1] ідеально підходить для створення модульної структури інтерфейсу наукового журналу, де багато елементів (статті, форми подання, тощо) можуть бути реалізовані як окремі компоненти. Висока продуктивність завдяки використанню віртуального DOM є особливо важливою для системи, яка повинна обробляти та відображати значні обсяги наукового контенту. Гнучкість і масштабованість React.js [1] дозволяють поступово розширювати функціонал системи без необхідності повного переписування коду.

Багата екосистема бібліотек та інструментів, таких як React Router для навігації, React Query для роботи з запитами до сервера, значно пришвидшує процес розробки. Важливою перевагою є також можливість інтеграції з іншими технологіями, включаючи Strapi [4] CMS та інші API-сервіси, що планується використовувати на бекенді.

Додатково, для розробки фронтенду було обрано TailwindCSS - фреймворк CSS, який дозволяє швидко створювати адаптивні інтерфейси без написання великої кількості власних стилів. Для покращення користувацького досвіду вирішено використовувати Framer Motion - бібліотеку для створення анімацій.

Аналіз та вибір інструментів для бекенд-розробки

Бекенд-частина веб-застосунку відіграє ключову роль у забезпеченні функціональності системи, роботи з даними та інтеграції з іншими сервісами. Було проаналізовано наступні варіанти:

Node.js [3] з Express

Переваги:

- Єдина мова програмування (JavaScript) для всього стеку розробки
- Асинхронна модель обробки запитів, що забезпечує високу продуктивність при роботі з багатьма одночасними з'єднаннями
- Величезна екосистема пакетів через npm, що дозволяє швидко інтегрувати необхідні функції
- Відмінна сумісність з React.js [1] та іншими фронтенд-фреймворками
- Легка горизонтальна масштабованість
- Активна спільнота розробників та постійні оновлення

Недоліки:

- Необхідність самостійної розробки багатьох стандартних функцій CMS
- Більш тривалий цикл розробки базового функціоналу порівняно з готовими рішеннями

- Можливі проблеми з продуктивністю при обробці завдань, що вимагають інтенсивних обчислень

- Необхідність додаткової конфігурації для забезпечення безпеки

Strapi [4] CMS

Переваги:

- Готова система управління контентом з відкритим кодом на основі Node.js [3]

- Зручний адміністративний інтерфейс для управління контентом без необхідності розробки власного

- Автоматичне створення RESTful і GraphQL API для всіх типів контенту

- Гнучка система ролей та дозволів, що важливо для різних користувачів наукового журналу

- Можливість кастомізації та розширення функціоналу через плагіни

- Вбудована підтримка медіа-файлів та їх обробки

- Проста інтеграція з різними базами даних (MySQL, PostgreSQL, MongoDB)

Недоліки:

- Менша гнучкість порівняно з власноруч розробленим рішенням на Node.js [3]

- Потенційні обмеження у специфічних сценаріях використання

- Залежність від розвитку та підтримки платформи Strapi [4]

- Можливі складнощі при глибокій кастомізації бізнес-логіки

PHP (Laravel)

Переваги:

- Перевірений фреймворк з багатою історією використання у веб-розробці

- Елегантний синтаксис та широкі можливості для розробки складних додатків

- Потужна система маршрутизації, ORM (Eloquent) та шаблонів (Blade)

- Вбудовані інструменти для автентифікації, кешування, черг завдань тощо

- Добре підходить для створення традиційних веб-додатків

Недоліки:

- Менш ефективна інтеграція з сучасними JavaScript-фреймворками порівняно з Node.js [3]
- Нижча продуктивність при обробці одночасних запитів порівняно з Node.js [3]
- Вимагає окремого стеку технологій для бекенду, що ускладнює розробку
- Менш гнучкий для створення сучасних API-орієнтованих застосунків

Вибір бекенд-інструментів

Після аналізу усіх варіантів, для розробки серверної частини веб-застосунку було обрано Strapi [4] CMS на базі Node.js [3] з таких міркувань:

Швидкість розробки є одним із ключових факторів, оскільки готовий функціонал Strapi [4] дозволяє значно скоротити час на реалізацію базових можливостей системи управління контентом, що критично важливо для дотримання термінів проекту. Гнучкість та можливість розширення також відіграли важливу роль у виборі, адже Strapi побудований на Node.js [3] і дозволяє додавати власний код та плагіни, що забезпечує необхідну гнучкість для реалізації специфічних вимог наукового журналу.

Зручний адміністративний інтерфейс Strapi спрощує процес управління контентом для редакторів та адміністраторів журналу без необхідності розробки окремого інтерфейсу. Автоматичне створення API є суттєвою перевагою, оскільки Strapi автоматично генерує RESTful API [5] для всіх типів контенту, що значно спрощує інтеграцію з фронтенд-частиною на React.js

Система ролей та дозволів, вбудована у Strapi [4], дозволяє ефективно реалізувати різні ролі користувачів для наукового журналу.

Для роботи з базою даних було обрано PostgreSQL через її надійність, підтримку складних запитів та добру інтеграцію зі Strapi . Додатково, для забезпечення аутентифікації та авторизації користувачів було вирішено

використовувати JWT (JSON Web Tokens) [6], що забезпечує безпечний та ефективний механізм управління сесіями користувачів.

Висновок

Обрані технології та інструменти для розробки веб-застосунку наукового журналу "ECJ" представляють сучасний, ефективний та збалансований технологічний стек, який відповідає всім вимогам проекту. Використання React.js для фронтенду та Strapi CMS на базі Node.js для бекенду забезпечує оптимальне співвідношення швидкості розробки, гнучкості та продуктивності.

Даний технологічний стек дозволяє створити сучасний, масштабований та функціональний веб-застосунок, який відповідатиме потребам усіх категорій користувачів наукового журналу. Крім того, обрані технології забезпечують можливість подальшого розширення функціоналу системи в разі виникнення нових вимог чи змін у процесах роботи наукового журналу.

У разі необхідності, архітектура системи передбачає можливість доповнення або заміни окремих компонентів без необхідності повного переписування коду, що забезпечує довгострокову життєздатність та розвиток проекту.

Після консультацій з науковим керівником та представниками редакційної колегії журналу, розроблено концепцію нового веб-застосунку, який би відповідав сучасним технологічним стандартам та задовольняв потреби всіх категорій користувачів. Прийнято рішення використати стек технологій на основі React.js для клієнтської частини та Strapi CMS для серверної частини, що забезпечить необхідну гнучкість, масштабованість та можливості для подальшого розвитку.

Таким чином, оновлення сайту ECJ є важливим завданням, яке об'єднує технологічні інновації з потребами наукової спільноти. Успішна реалізація цього проекту зробить внесок у розвиток наукової діяльності університету, створюючи сучасну та ефективну платформу для публікації та поширення наукових досліджень.

1.3 Аналіз аналогічних розробок

У процесі підготовки до розробки веб-застосунку для наукового журналу "ECJ" було проведено детальний аналіз існуючих аналогічних рішень, зокрема платформ, що використовуються іншими науковими виданнями та електронними бібліотеками. Метою аналізу було виявлення типових проблем, обмежень та недоліків, які слід уникнути при створенні нового веб-застосунку, а також визначення ключових функціональних вимог, які забезпечать конкурентні переваги для журналу "ECJ".

Аналіз веб-сайту sporthealth.kubg.edu.ua

При дослідженні веб-сайту sporthealth.kubg.edu.ua було виявлено ряд суттєвих недоліків, пов'язаних із застарілими технологіями та підходами до розробки. Платформа використовує застарілі версії бібліотек, зокрема jQuery 3.3.1, яка, хоча й залишається функціональною, не отримує оновлень безпеки та не підтримує сучасні функції JavaScript. Веб-сервер працює під управлінням Apache 2.4, що при відсутності належного налаштування та оптимізації може призводити до обмежень у продуктивності та масштабуванні.

Технічний аудит показав, що сайт має обмежену підтримку сучасних веб-стандартів, таких як HTML5 та CSS3, що негативно впливає на сумісність з новими браузерами та мобільними пристроями. Це проявляється у неоптимальному відображенні контенту на екранах різного розміру, що погіршує користувацький досвід для значної частини аудиторії.

Структура сайту характеризується складною навігацією, що ускладнює пошук необхідної інформації. Користувачі змушені проходити через кілька рівнів меню для доступу до основних функцій, таких як перегляд статей або інформація про подання рукописів. Відсутні сучасні механізми пошуку та фільтрації контенту, що значно ускладнює роботу з архівом публікацій.

Аналіз веб-сайту elibrary.kubg.edu.ua

Дослідження електронної бібліотеки elibrary.kubg.edu.ua виявило проблеми, пов'язані з серверною інфраструктурою та технологіями, що використовуються для забезпечення функціональності сайту. Як і в попередньому випадку, система побудована на застарілій платформі Apache 2.4 без належної оптимізації, що призводить до повільного завантаження сторінок, особливо при роботі з великими обсягами даних.

Особливу увагу привернула проблема доступності контенту. Сайт активно використовує JavaScript та Ajax для взаємодії з користувачем, але не забезпечує належного дублювання функцій у статичному HTML. Це створює суттєві бар'єри для користувачів з особливими потребами, які використовують спеціалізовані програми для читання екрану, та для пошукових систем, що не завжди можуть коректно індексувати динамічний контент.

Інтерфейс користувача характеризується застарілим дизайном, який не відповідає сучасним тенденціям та очікуванням користувачів. Відсутність адаптивного дизайну робить користування сайтом на мобільних пристроях незручним, що є критичним недоліком у світі, де значна частина веб-трафіку генерується смартфонами та планшетами.

Аналіз веб-сайту журналу nim.media

Аналіз сайту журналу nim.media виявив ряд проблем, пов'язаних з технічною реалізацією та оптимізацією користувацького досвіду. Головним недоліком є надмірне використання сторонніх JavaScript-бібліотек. Це призводить до збільшення часу завантаження сторінок, особливо для користувачів з повільним інтернет-з'єднанням, та збільшує навантаження на пристрої користувачів.

Детальний аналіз показав, що веб-сайт містить велику кількість схем даних та метаданих, які не оптимізовані належним чином для пошукових систем. Це ускладнює індексування контенту та негативно впливає на пошукову оптимізацію, що в результаті зменшує видимість наукових публікацій у пошукових системах.

Дизайн інтерфейсу користувача, хоча й виглядає сучасно, містить елементи, які ускладнюють взаємодію з сайтом. Спливаючі вікна, анімації та інші візуальні ефекти часто відволікають від основного контенту та можуть викликати дискомфорт у користувачів. Навігаційна структура не завжди інтуїтивно зрозуміла, що ускладнює пошук необхідної інформації для нових відвідувачів.

Система управління контентом має обмежені можливості для персоналізації та адаптації під потреби редакторів та авторів. Процес подання статей вимагає багато ручних дій та не автоматизований належним чином, що збільшує навантаження на редакційну колегію та уповільнює процес публікації.

Загальні тенденції та проблеми

Узагальнюючи результати аналізу, можна виділити кілька загальних тенденцій та проблем, характерних для існуючих веб-застосунків наукових журналів:

1. **Застарілі технології та підходи до розробки**: Більшість проаналізованих платформ побудовані на основі застарілих CMS та фреймворків, які не відповідають сучасним вимогам до інтерактивності, безпеки та зручності використання. Ці системи часто мають обмежений функціонал, складний процес управління контентом і застарілий інтерфейс, що не відповідає очікуванням сучасних користувачів.

2. **Проблеми з продуктивністю та оптимізацією**: Відсутність належної оптимізації веб-сторінок та ресурсів призводить до повільного завантаження, що негативно впливає на користувацький досвід.

3. **Обмежена доступність та відсутність адаптивного дизайну:** Багато сайтів наукових журналів не адаптовані для перегляду на мобільних пристроях. Це суттєво обмежує їхню аудиторію та ускладнює доступ до наукових публікацій.

4. **Статичний підхід до побудови інтерфейсу:** З технічної точки зору, більшість проаналізованих веб-сайтів використовують статичний підхід до побудови

інтерфейсу, що ускладнює реалізацію динамічних функцій, таких як автоматизоване подання статей, інтерактивний пошук та персоналізований доступ до матеріалів. Це стає серйозною перешкодою для журналів, які прагнуть розширювати свою аудиторію та відповідати сучасним технологічним стандартам.

5. **Обмежені можливості для автоматизації редакційних процесів:** Більшість систем не забезпечують належного рівня автоматизації для ключових процесів, таких як подання, редагування та публікація. Це збільшує навантаження на редакційну колегію та уповільнює цикл від подання статті до її публікації.

Висновки та напрямки розвитку

На фоні виявлених проблем та обмежень існуючих рішень стає очевидною необхідність створення сучасного веб-застосунку для наукового журналу "ECJ", який відрізнятиметься технологічністю, гнучкістю та орієнтацією на потреби користувачів. Розробка нової платформи повинна враховувати виявлені недоліки.

Ключовими напрямками розвитку для нового веб-застосунку визначено:

1. Впровадження сучасних технологій та архітектурних рішень: Використання React.js [1] для фронтенду та Strapi [4] CMS для бекенду забезпечить високу продуктивність, гнучкість та масштабованість системи. Архітектура, орієнтована на API, дозволить розділити серверну та клієнтську частини, що спростить подальший розвиток та інтеграцію з іншими системами.

2. **Створення інтуїтивно зрозумілого користувацького інтерфейсу:** Розробка адаптивного дизайну з фокусом на зручність використання та доступність для користувачів. Особлива увага буде звернена на оптимізацію мобільного досвіду, враховуючи зростаючу частку користувачів, які отримують доступ до контенту через смартфони та планшети.

3. **Автоматизація редакційних процесів:** Впровадження функціоналу для автоматизації ключових процесів, таких як подання, редагування та публікація

наукових матеріалів. Це дозволить значно скоротити час від подання статті до її публікації.

4. **Впровадження системи пошуку та навігації:** Реалізація інструментів для пошуку та фільтрації наукових публікацій за різними критеріями, що полегшить доступ до інформації для читачів та дослідників.

Висновок

Проведений аналіз існуючих веб-застосунків наукових журналів дозволив виявити ряд критичних проблем та обмежень, які потребують уваги при розробці нового веб-застосунку. Дослідження конкурентів виявило загальні тенденції використання застарілих технологій, проблеми з продуктивністю та обмежену доступність для користувачів.

Ключовим результатом аналізу стало виявлення проблемних аспектів: використання застарілих технологій та підходів до розробки, відсутність адаптивного дизайну, статичний підхід до побудови інтерфейсу та обмежені можливості автоматизації редакційних процесів. Ці виявлені проблеми стали основою для формування ключових напрямків розвитку нового веб-застосунку.

Проведений аналіз підтвердив актуальність та необхідність створення нового веб-застосунку, який буде відповідати сучасним вимогам до веб-розробки та забезпечить ефективну роботу всіх категорій користувачів. Отримані результати дослідження стали основою для формування технічних вимог та архітектурних рішень нового веб-застосунку

РОЗДІЛ 2. ПРАКТИЧНЕ СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ

2.1. Створення середовища для розробки веб-застосунку

Для реалізації веб-застосунку наукового журналу було обрано сучасний стек технологій, що забезпечує оптимальний баланс між швидкістю розробки, продуктивністю та масштабованістю. Концепція розробки базувалася на мікросервісній архітектурі з розділенням функціональних обов'язків між клієнтською та серверною частинами.

2.1.1. Вибір технологічного стеку

Після аналізу існуючих технологій та фреймворків було обрано React для фронтенд-розробки та Strapi як headless CMS для реалізації бекенд-платформи. React обрано через високу продуктивність, компонентний підхід та велику спільноту розробників, що гарантує довгострокову підтримку та розвиток проекту. Strapi як система керування контентом з відкритим кодом надає гнучку структуру для створення API, розширену систему ролей та дозволів, а також інтуїтивно зрозумілу панель адміністрування.

2.1.2. Налаштування локального середовища розробки

Розробку проекту було реалізовано у середовищі Visual Studio Code, що забезпечило оптимальні умови для кодування. Для підвищення ефективності розробки було встановлено додаткові розширення: React DevTools для відлагодження React компонентів та Git Lens для розширеної інтеграції з Git [7].

2.1.3. Встановлення Node.js та npm

Основою середовища виконання JavaScript на сервері було обрано Node.js [3]. Використання npm як менеджера пакетів дозволило ефективно керувати залежностями проекту та підтримувати їх актуальний стан протягом усього циклу розробки.

2.1.4. Створення проекту Strapi

Для реалізації серверної частини веб-застосунку було розгорнуто Strapi. Процес розгортання включав ініціалізацію нового проекту з автоматичним налаштуванням SQLite як бази даних для швидкого розгортання локального середовища.

Strapi надає RESTful API, що слідує принципам REST та використовує стандартні HTTP-методи. API повертає відповіді у форматі JSON з правильними статус-кодами. Для авторизованого доступу використовувались JWT-токени, які генеруються при аутентифікації та передаються у заголовку Authorization для подальших запитів.

Також було налаштовано систему ролей та дозволів для різних типів користувачів: адміністратори з повним доступом, автори з можливістю створювати власні матеріали та анонімні користувачі з доступом тільки для читання опублікованого контенту. Функціональність системи було розширено за допомогою додаткових плагінів для завантаження файлів.

2.1.5. Налаштування React-додатку

Клієнтська частина застосунку базувалася на React. Процес налаштування включав встановлення додаткових необхідних пакетів: axios для HTTP-запитів, react-router-dom для маршрутизації, react-query для керування станом даних, framer-motion для анімацій, jwt-decode для роботи з токенами автентифікації та react-hook-form для керування формами. Для стилізації компонентів було обрано підхід, базований на Tailwind CSS [2].

Структура проекту була організована згідно з принципами чистої архітектури з розподілом коду на компоненти, сторінки, контексти, користувацькі хуки, допоміжні функції, статичні ресурси, типи та сервіси. Така організація забезпечує високу модульність, тестованість та підтримуваність коду. Маршрутизація була налаштована за допомогою React Router з визначенням основних сторінок застосунку.

Висновки

Створене середовище для розробки веб-застосунку наукового журналу забезпечує комплексний підхід до вирішення поставлених завдань. Використання React як фронтенд-фреймворку та Strapi як бекенд-платформи дозволило поєднати гнучкість розробки з можливостями керування контентом. Архітектура проекту дотримується сучасних принципів розробки програмного забезпечення.

2.2 Розробка Frontend частини веб-застосунку наукового журналу ECJ

2.2.1. Розробка головної сторінки

Головна сторінка веб-застосунку (Home Page) є ключовим елементом інтерфейсу, який надає користувачам загальну інформацію про науковий журнал та доступ до основних функцій. Сторінка розроблена з урахуванням принципів user-friendly дизайну та містить декілька основних секцій, які логічно структурують контент та забезпечують зручність користування сайтом.

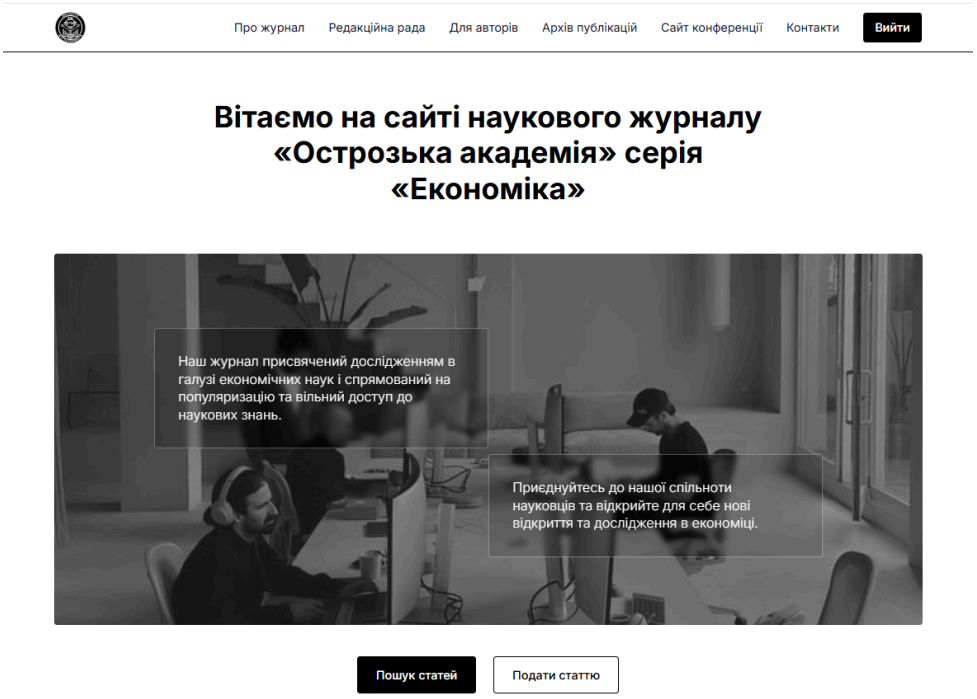

Рис. 2.1. Вітальна секція

Джерело: розроблене автором

2.2.2. Загальна структура та дизайнерські рішення

Розробка головної сторінки включала в себе створення навігаційного меню та сучасного, інтуїтивно зрозумілого інтерфейсу з використанням React.js [1] та Tailwind CSS [2]. При розробці інтерфейсу було використано монохромну кольорову схему з акцентами на контрастних елементах, що підкреслює науковий характер видання та забезпечує високу читабельність контенту.

Загальна структура сторінки складається з п'яти основних блоків:

- Навігаційне меню
- Вітальна секція з основною інформацією
- Блок "Про журнал"
- Секція "Тематичні напрями"
- Інформаційний блок про відкритий доступ
- Секція партнерів журналу
- Футер з контактною інформацією

Верстка сторінки виконана з використанням гнучкої сітки (grid та flex), що забезпечує адаптивність під різні розміри екранів — від мобільних пристроїв до широкоформатних моніторів. Реалізовано підхід "mobile-first", що дозволяє оптимально відображати контент на будь-яких пристроях без втрати функціональності.

2.2.3. Створення навігаційного меню

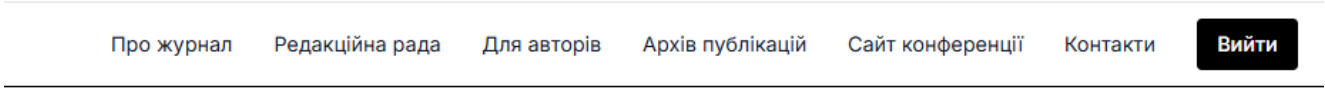

Рис. 2.2. Навігація

Джерело: розроблене автором

Навігаційне меню розташоване у верхній частині сторінки та забезпечує доступ до всіх основних розділів сайту. Структура меню розроблена з урахуванням логіки користування науковим журналом та включає наступні пункти:

- Про журнал
- Редакційна рада
- Для авторів
- Архів публікацій
- Сайт конференції
- Контакти

Для мобільних пристроїв реалізовано згортання меню до іконки "gamburger", при натисканні на яку розкривається повне меню у вигляді вертикального списку. Додатково в правій частині меню розміщено кнопку авторизації, яка змінює свій вигляд залежно від статусу користувача.

2.2.4. Інтерактивні елементи та кнопки навігації

Під вітальною секцією розміщено два основні інтерактивні елементи, які забезпечують швидкий доступ до ключових функцій журналу:

1. Кнопка "Пошук статей" — перенаправляє користувача на сторінку архіву публікацій з можливістю пошуку статей за різними критеріями.

2. Кнопка "Подати статтю" — інтегрована з системою автентифікації та має різну поведінку залежно від статусу користувача:

- Для авторизованих користувачів — перенаправлення на сторінку подання статті
- Для неавторизованих користувачів — перенаправлення на сторінку входу

Обидві кнопки оформлені з використанням контрастних кольорів (чорний та білий) з ефектом інверсії при наведенні курсору. Також реалізовано анімаційні ефекти масштабування та тіні при взаємодії, що покращує користувацький досвід та візуально підкреслює інтерактивність елементів.

Для оптимізації роботи на мобільних пристроях кнопки адаптуються до розміру екрану, змінюючи своє розташування з горизонтального на вертикальне при малих розмірах екрану. Це забезпечує зручність використання на всіх типах пристроїв.

2.2.5. Інформаційні блоки на головній сторінці

Основний контент головної сторінки структурований у вигляді тематичних блоків, кожен з яких присвячений певному аспекту журналу:

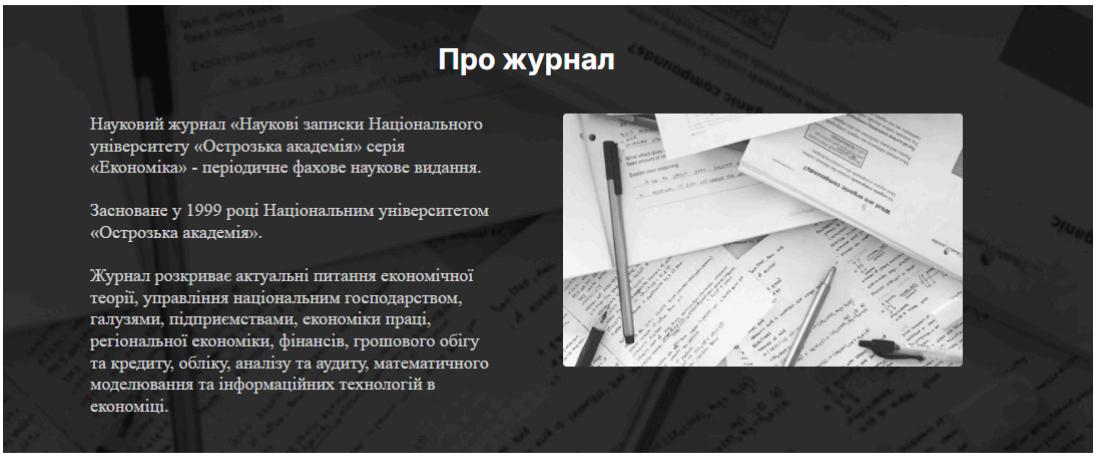

Рис. 2.3. Секція “Про журнал”.

Джерело: розроблене автором

1. Блок "Про журнал" містить коротку історичну довідку про видання, інформацію про наукові напрями та особливості. Оформлений на контрастному темному фоні з використанням зображення наукових публікацій, оброблених у монохромному стилі для створення єдиного візуального стилю.

Рис. 2.4. Секція “Тематичні напрями”.

Джерело: розроблене автором

2. Секція "Тематичні напрями" представляє основні напрями досліджень, які публікуються в журналі. Для візуалізації використано комбінацію текстових блоків з нумерацією та тематичних зображень. Реалізовано інтерактивне з'явлення елементів при прокручуванні сторінки з використанням Intersection Observer API.

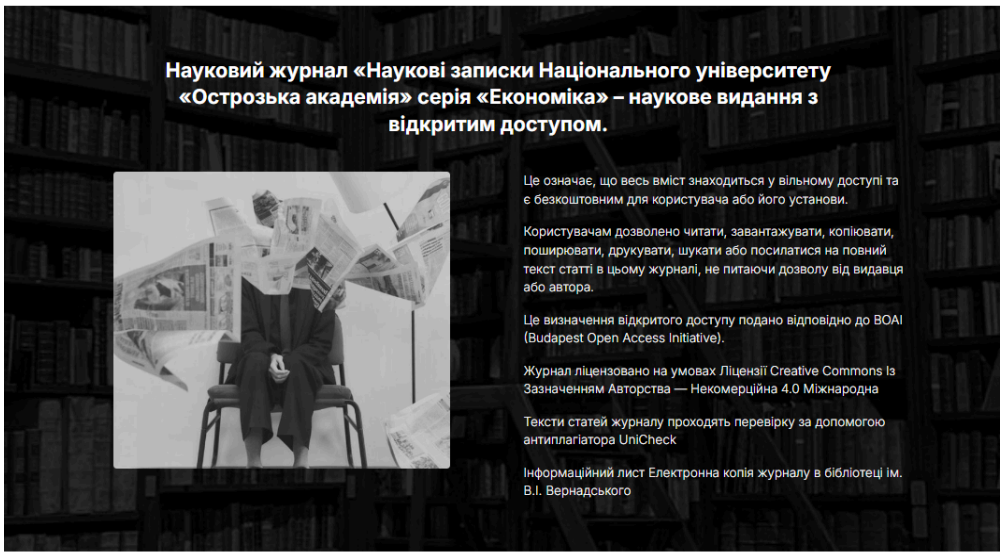

Рис. 2.5. Секція “Політика журналу”.

Джерело: розроблене автором

3. Інформаційний блок про відкритий доступ розкриває політику журналу щодо публікацій, умови використання матеріалів та ліцензування. Ця секція містить важливу інформацію про ліцензію, політику перевірки на плагіат та доступ до архівів.

Рис. 2.6. Секція “Наші партнери”.

Джерело: розроблене автором

4. Секція партнерів відображає логотипи організацій, з якими співпрацює журнал. Реалізовано автоматичну прокрутку логотипів за допомогою CSS-анімації, що дозволяє показати всіх партнерів навіть у обмеженому просторі екрану.

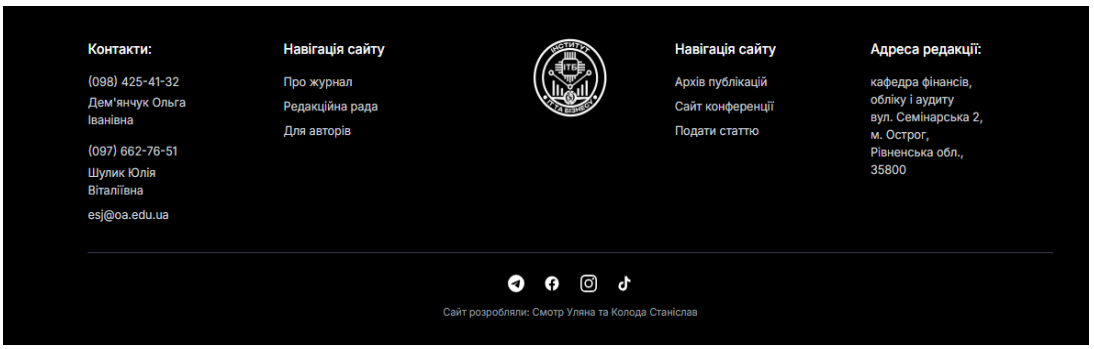

Рис. 2.7. Секція “Footer”.

Джерело: розроблене автором

5. Футер веб-застосунку "ECJ" реалізовано як адаптивний компонент з п'ятиколонковою структурою на десктопі та двоколонковою на мобільних пристроях. Він містить контактну інформацію, навігаційні посилання, логотип журналу, адресу редакції та посилання на соціальні мережі. Дизайн оптимізовано для зручного перегляду на різних пристроях завдяки використанню Flexbox та Grid. Інтерактивні елементи мають плавні переходи, що покращує користувацький досвід.

2.2.6. Система сповіщень

На головній сторінці реалізовано систему динамічних сповіщень, яка інформує користувачів про результати їхніх дій. Сповіщення з'являються у верхній правій частині екрану та мають різний стиль залежно від типу повідомлення:

- Зелений фон для повідомлень про успішні дії

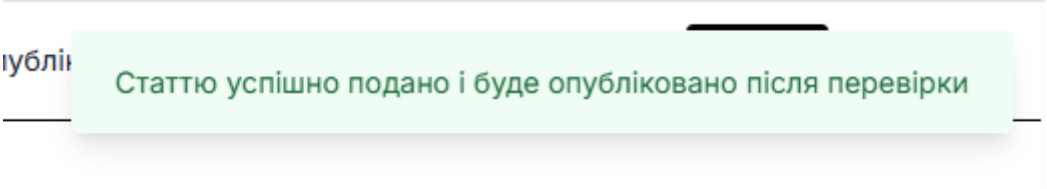

Рис. 2.8. Сповіщення про успішне подання статті.

Джерело: розроблене автором

Сповіщення автоматично зникають після певного часового інтервалу або можуть бути закриті користувачем. Вони з'являються, наприклад, після успішної авторизації або виконання певних дій на сайті, забезпечуючи зворотний зв'язок для користувача.

2.2.7. Розробка сторінки “Для авторів”

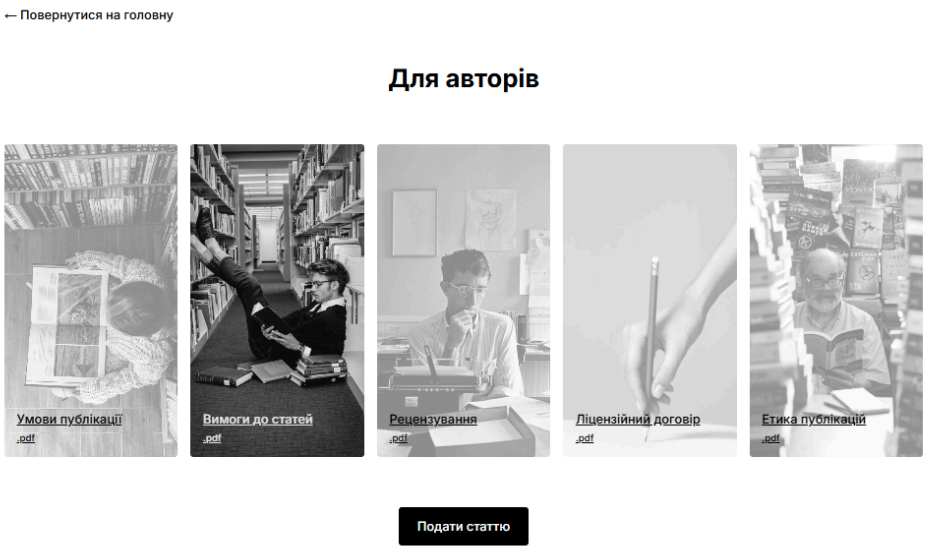

Рис. 2.9. Секція “Для авторів”

Джерело: розроблене автором

Сторінка "Для авторів" створена з метою забезпечення авторів повною та структурованою інформацією щодо публікації в науковому журналі. Вона виконує роль інформаційного хабу та інструменту взаємодії між авторами та редакцією. Ця секція створена за допомогою grid сітки, також на кожний блок додана анімація при наведенні картинка стає більш чіткою для кращого розуміння користувачем що він вибирає.

2.2.8. Розробка сторінки “Пошук статей”

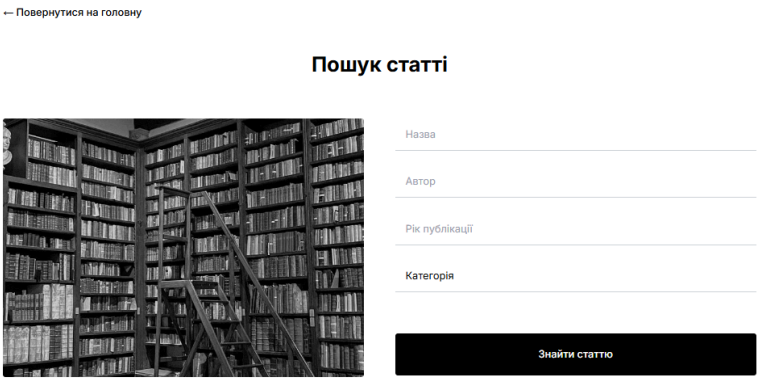

Рис. 2.10. Сторінка “Пошук статей”

Джерело: розроблене автором

Сторінка пошуку статей є важливим компонентом наукового журналу, що надає користувачам можливість здійснювати пошук та перегляд публікацій. Вона реалізована з використанням React.js та забезпечує зручний інтерфейс для фільтрації та відображення результатів.

Основні особливості сторінки:

Сторінка містить форму пошуку, яка дозволяє користувачам фільтрувати статті за різними параметрами: назва, автор, рік публікації та категорія. Для цього використовується стан `searchParams`, що зберігає поточні значення фільтрів. При завантаженні сторінки виконується запит до API для отримання списку категорій, які відображаються у випадаючому меню.

Результати пошуку відображаються у вигляді карток статей, кожна з яких містить основну інформацію: назву, автора, дату публікації та короткий опис.

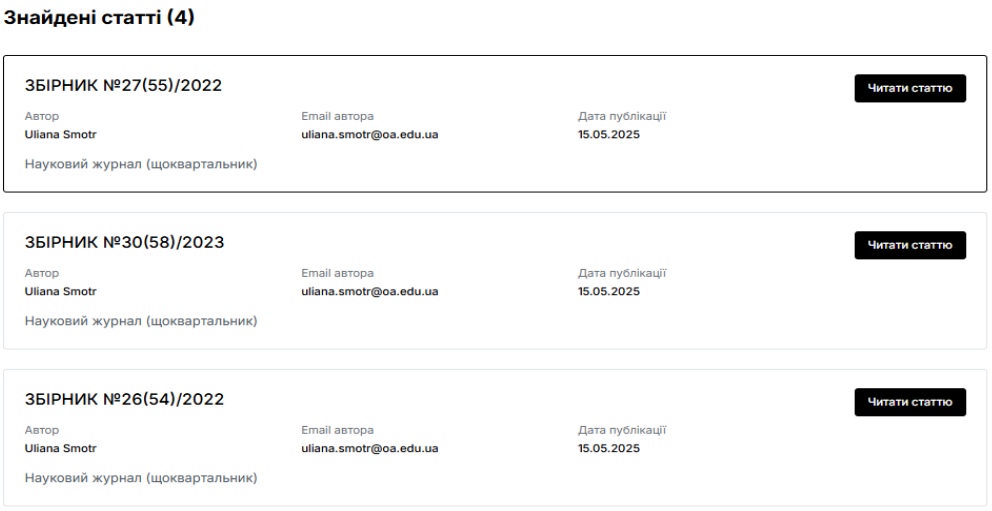

Рис. 2.11. Результат пошуку за автором статті

Джерело: розроблене автором

При натиску кнопки “Читати статтю” відкривається сторінка з більш детальною інформацією про статтю та з можливістю завантажити pdf-файл статті для читання.

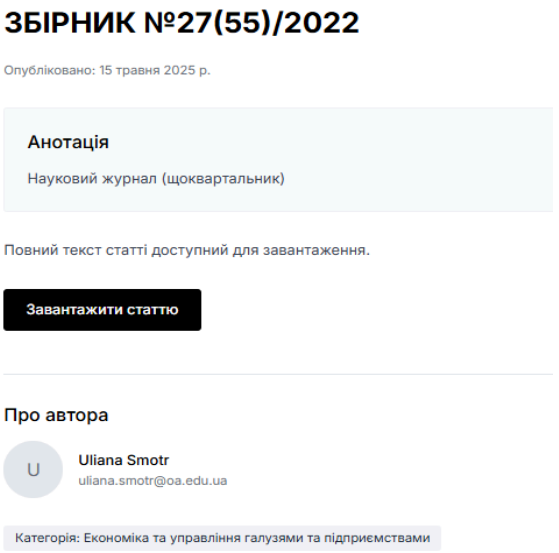

Рис. 2.12. Сторінка статті

Джерело: розроблене автором

Реалізація пошуку:

Пошук реалізовано через асинхронний запит до API, що виконується при відправці форми. Функція `handleSearch` обробляє подію відправки, встановлює стан завантаження `isLoading` та виконує запит до `articlesAPI.searchArticles`, передаючи поточні параметри пошуку. Отримані результати зберігаються у стані `searchResults`.

API-запит для пошуку статей:

```
searchArticles: async (params = {}) => {
  try {
    let url = '/articles?populate=*';
    if (params.title) {
      url += `&filters[title][$containsi]=${params.title}`;
    }
    if (params.author) {
      url += `&filters[author][id]=${params.author}`;
    }
    if (params.year) {
      const year = params.year;
      url +=
`&filters[publishedAt][$gte]=${year}-01-01&filters[publishedAt][$lte]=${year}-12-
31`;
    }
    if (params.category) {
      url += `&filters[category][id]=${params.category}`;
    }
    console.log('Search URL:', url);
    const response = await api.get(url);
    return response.data;
  } catch (error) {
    console.error('Error searching articles:', error);
    throw error;
  }
}
```

Рис. 2.13. Лістинг реалізації API запиту

Джерело: розроблене автором

Цей код формує URL для пошуку статей з різними параметрами фільтрації (назва, автор, рік, категорія) та виконує GET-запит до API. Результати пошуку повертаються у форматі JSON.

Здійснено обробку помилок: Всі запити до API обгорнуті в блок `try-catch`, що дозволяє обробляти можливі помилки та виводити їх у консоль для відлагодження.Якщо пошук не дав результатів, користувач отримує відповідне повідомлення з рекомендаціями щодо зміни параметрів пошуку.

2.2.9. Розробка системи реєстрації та аутентифікації користувачів

У рамках розробки веб-інтерфейсу наукового журналу було реалізовано повноцінну систему реєстрації та аутентифікації користувачів. Ця система є важливою складовою частиною проекту, оскільки забезпечує безпечний доступ до контенту та дозволяє користувачам взаємодіяти з системою на різних рівнях.

Система автентифікації базується на JWT (JSON Web Tokens) [12] технології, що забезпечує безпечну передачу інформації між клієнтом та сервером. При успішній автентифікації користувач отримує JWT токен, який зберігається та використовується для подальших запитів до API.

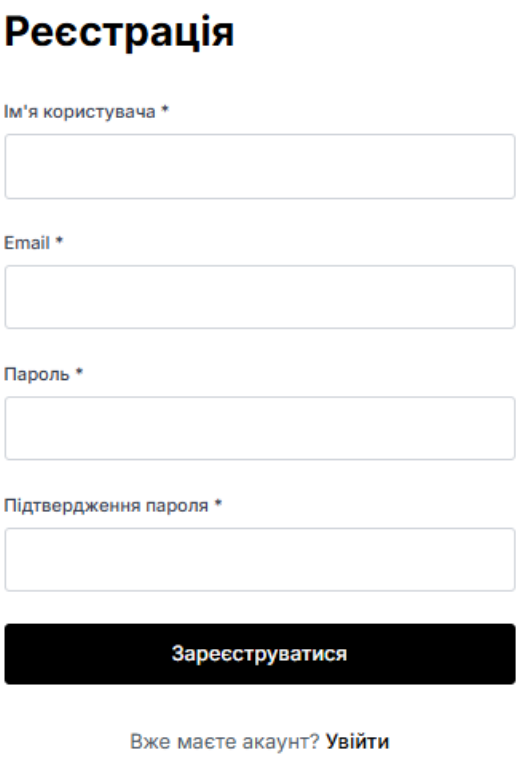

Рис. 2.14. Форма реєстрації користувача

Джерело: розроблене автором

Процес реєстрації нового користувача включає кілька етапів. Спочатку користувач заповнює форму реєстрації, вказуючи своє ім'я, електронну пошту та пароль. Система перевіряє унікальність електронної пошти та імені користувача, а також валідує надійність пароля. Після успішної валідації даних, створюється новий запис у базі даних, а користувач автоматично авторизується в системі.

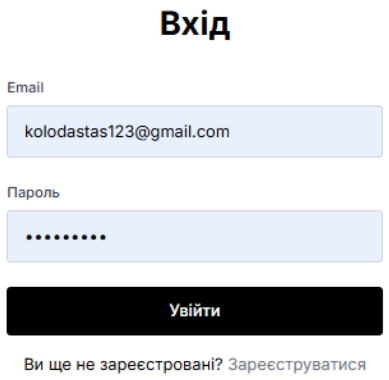

Рис. 2.15. Форма входу користувача

Джерело: розроблене автором

Для входу в систему користувач вказує свої облікові дані (електронну пошту та пароль). Система перевіряє правильність введених даних та, у разі успішної перевірки, генерує JWT токен. Цей токен містить інформацію про користувача та його права доступу, що дозволяє системі ідентифікувати користувача при кожному запиті.

```
// Функція входу користувача
const login = async (email, password) => {
  try {
    const response = await api.post('/auth/local', {
      identifier: email,
      password: password,
    });
    if (response.data.jwt) {
      // Зберігаємо токен та дані користувача в localStorage
      localStorage.setItem('token', response.data.jwt);
      localStorage.setItem('user',
JSON.stringify(response.data.user));
    }
```

```
    return response.data;
  } catch (error) {
    console.error('Помилка входу:', error);
    throw error;
  }
};
```

Рис. 2.16. Лістинг функції входу користувача

Джерело: розроблене автором

Реалізовано систему ролей та прав доступу, яка визначає, які дії може виконувати користувач у системі. Наприклад, авторизовані користувачі можуть переглядати повний текст статей, завантажувати PDF-файли, подавати нові статті. Неавторизовані користувачі мають обмежений доступ до контенту.

Для покращення користувацького досвіду реалізовано:

- Збереження стану автентифікації між сесіями
- Автоматичне перенаправлення на сторінку входу при спробі доступу до захищених ресурсів
- Інформативні повідомлення про помилки автентифікації
- Можливість виходу з системи з будь-якої сторінки

Реалізована система реєстрації та автентифікації забезпечує безпечний та зручний доступ до функціоналу наукового журналу, дозволяючи користувачам ефективно взаємодіяти з системою.

2.2.10. Розробка сторінки “Подати статтю”

Сторінка подачі статті реалізована як інтерактивна форма з використанням React та React Router. Форма містить обов'язкові поля для введення назви статті, опису, вибору категорії та завантаження PDF-файлу. Реалізовано валідацію введених даних, перевірку наявності та формату PDF-файлу, а також обробку помилок завантаження з відображенням відповідних повідомлень.

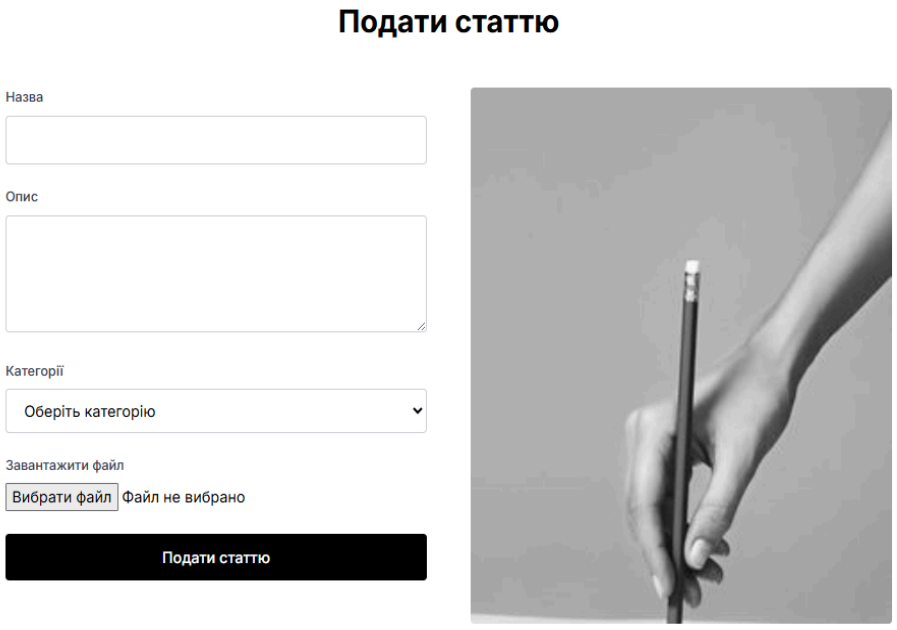

Рис. 2.17. Форма подання статті

Джерело: розроблене автором

Процес подання статті включає завантаження PDF-файлу на сервер, генерацію унікального slug для статті та створення запису в базі даних. Після успішного подання для запобігання повторних відправок користувач перенаправляється на головну сторінку з повідомленням про успіх.

```
const createArticle = async (articleData) => {
  try {
    // Формуємо дані для відправки
    const completeArticleData = {
      data: {
        title: articleData.title,
        description: articleData.description,
        slug: articleData.title
          .toLowerCase()
          .trim()
          .replace(/[^\w\s-]/g, '')
          .replace(/\s+/g, '-')
          .replace(/-+/g, '-'),
        category: articleData.category,
        author: articleData.author,
        file_pdf: articleData.fileId, // ID завантаженого PDF файлу
        publishedAt: null // Стаття створюється як чернетка
```

```
      }
    };
    // Відправляємо запит на створення статті
    const response = await api.post('/articles', completeArticleData);
    return response.data;
  } catch (error) {
    console.error('Помилка при створенні статті:', error);
    throw error;
  }
};
```

Рис. 2.18 Лістинг функції створення статті

Джерело: розроблене автором

2.2.11. Розробка сторінки “Архів публікацій”

Сторінка архіву реалізована як інтерактивний компонент, який забезпечує структурований доступ до архіву публікацій за роками. Основним елементом є система фільтрації публікацій за роками, що дозволяє користувачам швидко знаходити потрібні матеріали. Сторінка має адаптивний дизайн, що забезпечує коректне відображення на різних пристроях.

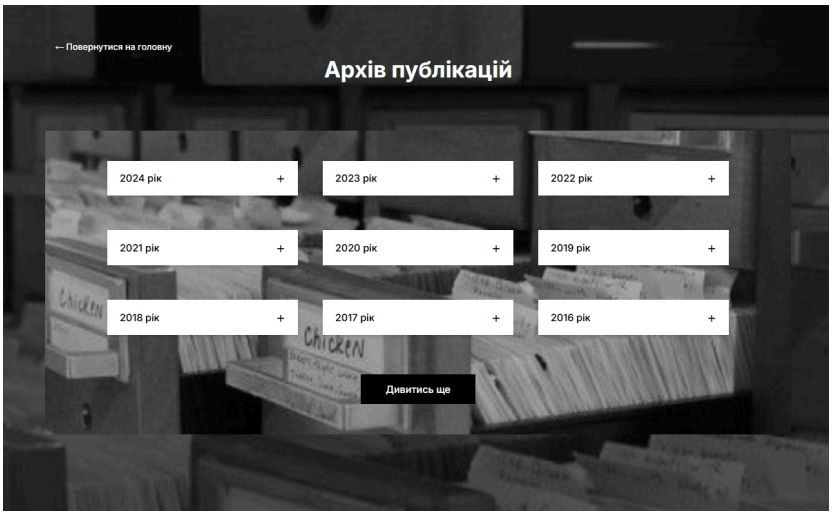

Рис. 2.19. Сторінка “Архів публікацій”

Джерело: розроблене автором

Сторінка має інтерфейс, який включає хронологічний список випусків журналу, фільтрацію за роками публікацій, систему навігації між випусками та інтегрований переглядач PDF-документів.

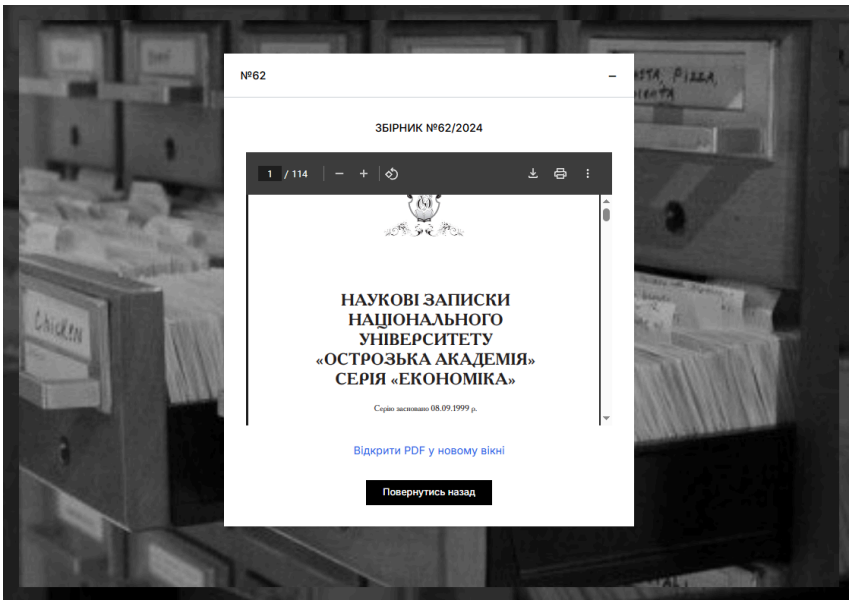

Рис. 2.20. Перегляд pdf-документа безпосередньо на сторінці статті

Джерело: розроблене автором

Користувачі можуть легко переміщатися між різними випусками журналу та переглядати їх вміст без необхідності завантаження файлів це дозволяє переглядати PDF-документи безпосередньо на сторінці, але якщо користувач забажає він може відкрити pdf-документ у новому вікні браузера та повноцінно читати статтю.

2.2.12. Розробка сторінки “Редакційна рада”

Сторінка редакційної ради має адаптивний дизайн, який оптимально відображається на різних пристроях. Основний контент представлений у вигляді сітки карток, де кожна картка містить інформацію про члена редакційної ради. Картки мають ефект наведення, який підсвічує їх та змінює колір фону на темний, що покращує візуальну взаємодію з користувачем.

Рис. 2.21. Сторінка “Редакційна рада”

Джерело: розроблене автором

Фон сторінки реалізовано за допомогою відео, що автоматично відтворюється та зациклюється. Використання `autoPlay` для автоматичного відтворення, параметр `loop` забезпечує безперервне відтворення, CSS-фільтри застосовані для створення ефекту затемнення та чорно-білого відображення

```
<div className="absolute inset-0 z-0 overflow-hidden">
          <video
            autoPlay
            loop
            muted
            className="w-full h-full object-cover"
            style={{
              filter: 'brightness(0.7) grayscale(100%)'
            }}
          >
            <source src="/images/back.mp4" type="video/mp4" />
          </video>
        </div>
```

Рис. 2.22. Лістинг реалізації динамічного фону

Джерело: розроблене автором

2.2.13. Адаптивність та мобільна версія

Особлива увага приділена адаптивності головної сторінки для різних пристроїв. Реалізовано декілька контрольних точок (breakpoints) для адаптації макету:

- До 550px — мобільна версія з вертикальним розташуванням елементів
- 550px-768px — планшетна версія з модифікованим розташуванням блоків
- Від 768px — десктопна версія

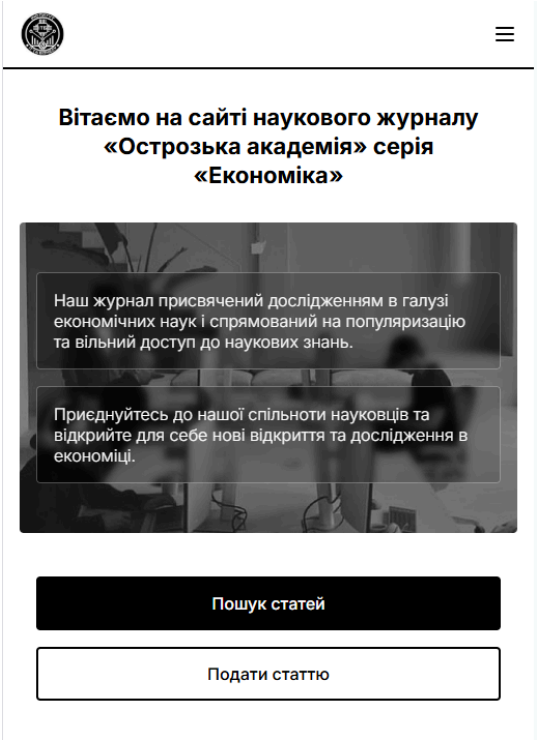

Рис. 2.23. Мобільна версія вітальної секції

Джерело: розроблене автором

Для мобільних пристроїв модифіковано:

- Розмір шрифтів (зменшено для кращої читабельності на малих екранах)
- Розташування елементів (змінено з горизонтального на вертикальне)
- Спрощено складні елементи дизайну (наприклад, секція тематичних напрямів)
- Оптимізовано розміри зображень

Такий підхід забезпечує комфортне використання сайту на будь-яких пристроях без втрати функціональності або важливої інформації.

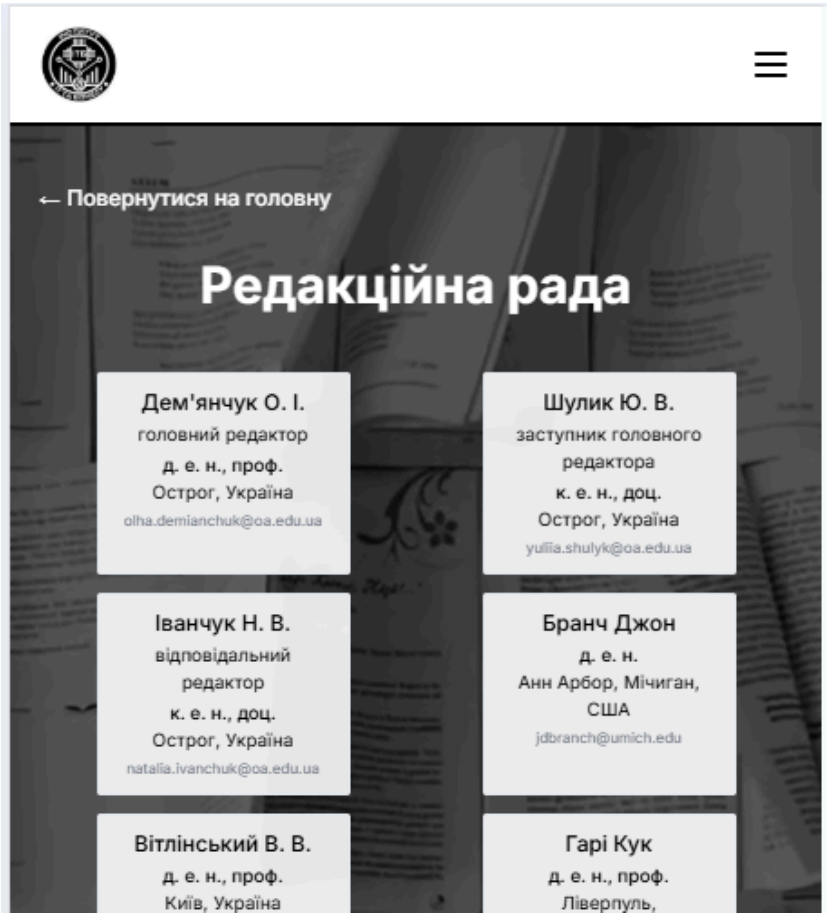

Рис. 2.24. Мобільна версія сторінки “Редакційна рада”

Джерело: розроблене автором

Основний компонент сторінки EditorialPage містить адаптивну сітку карток членів редакційної ради. Кожна картка реалізована як окремий компонент EditorialMemberCard, який відповідає за відображення інформації про члена редакційної ради. Адаптивний дизайн реалізовано через систему брейкпоінтів, яка забезпечує оптимальне відображення контенту на різних пристроях. На мобільних пристроях використовується двоколонкова сітка, на планшетах - три-чотири колонки, а на десктопах - п'ять колонок. Розміри шрифтів, відступи та інші візуальні елементи автоматично адаптуються під розмір екрану, забезпечуючи зручний перегляд інформації.

Рис. 2.25. Мобільна версія сторінки “Для авторів”

Джерело: розроблене автором

Основний компонент сторінки AuthorsPage містить адаптивну структуру, яка включає інформаційні блоки, форми та інтерактивні елементи. Кожен блок реалізовано як окремий компонент, який відповідає за відображення певної інформації для автора. Адаптивний дизайн реалізовано через систему брейкпоінтів, яка забезпечує оптимальне використання простору екрану, збереження функціональності та естетичного вигляду, а також покращену взаємодію з користувачем.

На мобільних пристроях використовується двоколонкова структура, яка забезпечує зручний перегляд інформації. Розміри шрифтів, відступи та інші візуальні

елементи автоматично адаптуються під розмір екрану, забезпечуючи зручний перегляд інформації.

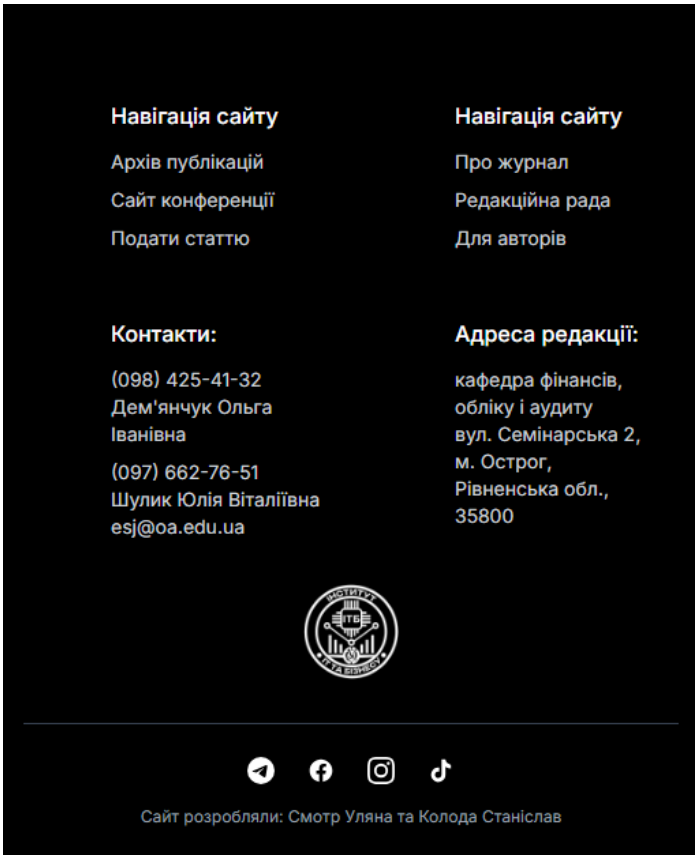

Рис. 2.26. Мобільна версія секції “Footer”

Джерело: розроблене автором

Основний компонент Footer містить адаптивну структуру, яка включає інформаційні блоки, навігаційні посилання та соціальні мережі.

Футер складається з трьох основних частин: контакти, меню та соціальні мережі. На телефоні всі елементи розташовуються в дві колонки, щоб займати менше місця. Текст та відступи автоматично зменшуються для зручного перегляду на малому екрані.

2.2.14. Технічна реалізація

Головна сторінка реалізована як React-компонент з використанням функціонального підходу та хуків для керування станом. Основні технічні аспекти реалізації:

1. Використання React-хуків:

- useState для керування станом видимості елементів
- useEffect для ініціалізації спостерігачів прокрутки
- useLocation для доступу до параметрів маршруту та стану

2. Використання Intersection Observer API для анімації елементів при прокручуванні сторінки, що забезпечує плавну появу секцій при їх входженні у вікно перегляду.

3. Застосування Tailwind CSS [2] для стилізації компонентів з використанням утилітарних класів.

2.3. Реалізація backend за допомогою Strapi

У рамках розробки веб-інтерфейсу наукового журналу було обрано Strapi як основну платформу для реалізації backend частини проекту. Strapi - це сучасна система управління контентом (CMS) з відкритим кодом, яка базується на Node.js та використовує PostgreSQL як систему управління базами даних.

Архітектура backend частини була розроблена з урахуванням специфіки наукового журналу та включає наступні основні компоненти:
Створено основні типи контенту (Content Types):

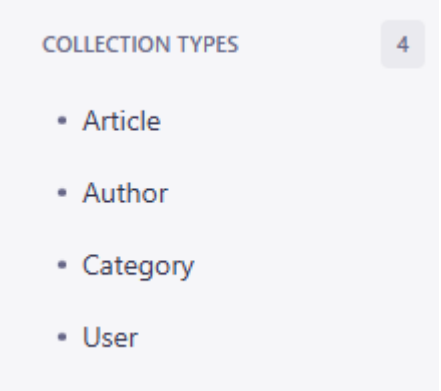

Рис. 2.27. Collection types

Джерело: розроблене автором

Кожен тип має свою структуру полів та зв'язки з іншими типами контенту. Наприклад, стаття пов'язана з авторами через відношення many-to-many, стаття та категорія пов’язана зв’язком many-to-one.

2.3.1. Структура бази даних

Основні моделі даних включають:

- Статті (Articles)
- Автори (Authors)
- Категорії (Categories)
- Файли (Files)

Кожна модель має власну структуру атрибутів та зв'язків з іншими моделями. Наприклад, модель Article містить такі поля:

- Заголовок (title)
- Опис (description)
- Унікальний ідентифікатор (slug)
- PDF-файл (file_pdf)
- Зв'язок з автором
- Зв'язок з категорією

Рис. 2.28. Модель Articles (вигляд у Strapi)

Джерело: розроблене автором

Strapi автоматично генерує REST API [6] для кожної моделі, що дозволяє виконувати всі необхідні операції з даними: отримання, створення, оновлення та видалення записів. Для статей реалізовано додаткові ендпоінти, які забезпечують пошук за різними параметрами, фільтрацію за категоріями, сортування за датою публікації.

2.3.2. Авторизація та права доступу

Реалізовано систему ролей та дозволів для забезпечення безпеки та контролю доступу до контенту. Створено дві основні ролі:

1. Authenticated - для зареєстрованих користувачів
2. Public - для неавторизованих відвідувачів

Кожному з цих користувачів надано різні дозвола для управління контентом.

Реалізовано систему реєстрації та аутентифікації користувачів. Процес реєстрації включає валідацію введених даних, перевірку унікальності email та username, зберігання паролів, генерацію JWT токенів та збереження даних користувача в базі даних.

```
// Реєстрація користувача
  register: async (username, email, password) => {
    try {
      const response = await api.post('/auth/local/register', {
        username,
        email,
        password,
      });
      if (response.data.jwt) {
        localStorage.setItem('token', response.data.jwt);
        localStorage.setItem('user', JSON.stringify(response.data.user));
      }

      return response.data;
```

Рис. 2.29. Реєстрація користувача на сайті

Джерело: розроблене автором

Безпека системи забезпечується через JWT автентифікацію, систему ролей та прав доступу, валідацію вхідних даних. Для ефективної взаємодії з фронтендом оптимізовано запити до бази даних та налаштовано обробку помилок.

Особливу увагу приділено системі управління файлами, особливо PDF-документами. Система забезпечує валідацію типів файлів, оптимізацію розміру файлів, безпечне зберігання та контроль доступу до файлів.

Реалізація backend за допомогою Strapi забезпечила надійну, масштабовану та безпечну основу для веб-інтерфейсу наукового журналу, що дозволяє ефективно керувати контентом та забезпечувати високу продуктивність системи. Це дозволяє легко масштабувати систему та додавати нові функції в майбутньому, забезпечуючи довгострокову підтримку та розвиток проекту.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно розроблено веб-застосунок для наукового журналу "ECJ", який забезпечує ефективну взаємодію між авторами, редакцією та читачами. Розроблений веб-застосунок представляє собою комплексне рішення, яке успішно поєднує сучасні технології з потребами наукової спільноти, вирішуючи ключові проблеми традиційних методів публікації наукових статей.

Система побудована на основі клієнт-серверної архітектури, де клієнтська частина розроблена з використанням React.js, а серверна - на базі Strapi CMS, що забезпечує надійну та масштабовану роботу всього застосунку. Такий підхід дозволив подолати розрив між традиційними методами публікації та сучасними технологічними можливостями, що було однією з ключових цілей дослідження.

Клієнтська частина системи має високу продуктивність та зручність використання завдяки застосуванню сучасних підходів до розробки. Використання React.js дозволило створити динамічний та інтерактивний інтерфейс, а Tailwind CSS забезпечив стильний та адаптивний дизайн, який коректно відображається на будь-яких пристроях. Система включає розвинену навігацію, зручний пошук та фільтрацію контенту, а також оптимізовану роботу з великими об'ємами даних. Це дозволяє подолати проблему обмеженої доступності та видимості наукових публікацій, що є однією з ключових проблем сучасного наукового середовища.

Серверна частина, побудована на Strapi, забезпечує надійне управління контентом та користувачами. Реалізована система ролей та прав доступу дозволяє чітко розмежувати можливості різних категорій користувачів: авторів, редакторів та адміністраторів. Кожна роль має свій набір прав та можливостей, що забезпечує безпеку та ефективність роботи системи. Інтеграція з базою даних гарантує надійне зберігання та швидкий доступ до даних. Всі операції з даними виконуються через API-endpoints з використанням JWT-автентифікації. Це дозволяє автоматизувати трудомісткі процеси подання та редагування статей.

Система пошуку та перегляду наукових статей в веб-застосунку реалізована з урахуванням потреб користувачів та забезпечує зручний доступ до наукового контенту. Розширений пошук дозволяє користувачам знаходити потрібні статті за різними критеріями, включаючи пошук за назвою, анотацією статті, фільтрацію за авторами, роком публікації та категоріями. Це значно підвищує доступність наукових публікацій та сприяє їх інтеграції у світову наукову спільноту.

Користувачі мають можливість переглядати статті двома способами: безпосередньо на веб-сайті, або завантажувати їх у форматі PDF для офлайн-перегляду. Користувачі можуть легко завантажувати PDF файли, зберігати їх локально або відправляти на друк. Це забезпечує гнучкість у роботі з науковими матеріалами та підвищує зручність їх використання.

Модульна архітектура застосунку дозволяє легко додавати нові функції та інтегрувати систему з іншими науковими платформами. Підтримка API дозволяє розробляти додаткові сервіси та розширення, включаючи інтеграцію з науковими базами даних.

Таким чином, розроблений веб-застосунок для наукового журналу "ECJ" є сучасним, функціональним та надійним рішенням, яке відповідає потребам наукової спільноти та має значний потенціал для подальшого розвитку. Система успішно поєднує зручність використання з необхідним рівнем безпеки та функціональності, що робить її ефективним інструментом для публікації та поширення наукових робіт. Модульна архітектура та розширені можливості інтеграції дозволяють системі розвиватися та адаптуватися до змінних потреб наукової спільноти, що особливо важливо в умовах стрімкого розвитку інформаційних технологій та зростання обсягів наукової інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React documentation [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/getting-started.html (дата звернення: 14.01.2025 р.)
2. Tailwind CSS documentation [Електронний ресурс]. – Режим доступу: https://v2.tailwindcss.com/docs(дата звернення: 18.01.2025 р.)
3. Node.js Documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/docs/latest/api/(дата звернення: 20.01.2025 р.)
4. Strapi. Офіційна документація [Електронний ресурс]. – Режим доступу: https://docs.strapi.io/ (дата звернення: 5.02.2025 р.)
5. Strapi plugin documentation [Електронний ресурс]. – Режим доступу: https://market.strapi.io/plugins/@strapi-plugin-documentation (дата звернення: 14.03.2025 р.)
6. Strapi API reference [Електронний ресурс]. – Режим доступу: https://docs.strapi.io/dev-docs/api/rest (дата звернення: 20.03.2025 р.)
7. Git documentation for Strapi [Електронний ресурс]. – Режим доступу: https://github.com/strapi/documentation (дата звернення: 15.04.2025 р.)
8. Material Tailwind documentation [Електронний ресурс]. – Режим доступу: https://www.material-tailwind.com/ (дата звернення: 19.02.2025 р.)
9. Stack Overflow – React.js tag [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/questions/tagged/reactjs (дата звернення: 5.04.2025 р.)
10. W3Schools. Web development tutorials [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/ (дата звернення: 17.02.2025 р.)
11. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/ (дата звернення: 12.02.2025 р.)

12. JWT.IO – JSON Web Tokens Introduction [Електронний ресурс]. – Режим доступу: https://jwt.io/introduction (дата звернення: 8.04.2025 р.)

13. Open Journal Systems (OJS) Documentation [Електронний ресурс]. – Режим доступу: https://docs.pkp.sfu.ca/ (дата звернення: 6.02.2025 р.)

14. RESTful API Design Guidelines – Microsoft [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design (дата звернення: 27.04.2025 р.)

15. Bootstrap documentation [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/ (дата звернення: 24.03.2025 р.)

16. GitHub. Статистика популярності Strapi CMS [Електронний ресурс]. – Режим доступу: https://github.com/strapi/strapi (дата звернення: 10.03.2025 р.)