МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ЕКОНОМІКО-ПРАВНИЧИЙ ФАХОВИЙ КОЛЕДЖ ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ»

Циклова комісія математичних дисциплін та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ ВЕДЕННЯ ЕЛЕКТРОННОГО ЖУРНАЛУ УСПІШНОСТІ СТУДЕНТІВ»

Виконав: здобувач освіти 4 курсу, групи К121-22
Спеціальність 121 Інженерія
програмного забезпечення
(шифр і спеціальність)
Андрій ДМИТРІЄНКО
(ім'я та ПРІЗВИЩЕ)
Керівник Дмитро ДРАЄВСЬКИЙ
(ім'я та ПРІЗВИЩЕ)
Рецензент доцент кафедри програмної інженерії ЗНУ, к.т.н., доцент Віталій МУХІН
(посада, вчене звання, науковий ступінь, ім'я та ПРІЗВИЩЕ)

Запоріжжя
2026

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ВСП «Економіко-правничий фаховий коледж ЗНУ»

Освітньо–кваліфікаційний ступінь фаховий молодший бакалавр

Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ

Голова циклової комісії математичних дисциплін та інформаційних технологій

_____ Т.М. Смолянкова
(підпис)

“ 05 ” червня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ДМИТРІЄНКО Андрію Ігоровичу
(ПРІЗВИЩЕ, ім’я та по– батькові)

1. Тема роботи «Розробка вебсервісу для ведення електронного журналу успішності студентів»

Керівник роботи викладач, ДРАЄВСЬКИЙ Дмитро Сергійович
(науковий ступінь, вчене звання, прізвище, ім’я, по батькові)

Затверджені наказом ВСП ЕПФК ЗНУ від «01» грудня 2025 р. № 1728-с

2. Строк подання студентом роботи 05.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково– пояснювальної записки (перелік питань, які потрібно розробити)
1.Сучасні інформаційні системи.
2. Розробка проекту інформаційної системи.
3. Програмна реалізація інформаційної системи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація до захисту

6. Консультанти розділів роботи

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

7. Дата видачі завдання 01.12.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи:		
	постановка задачі	Грудень 2025	виконано
2.	Збір вихідних даних, обробка методичних		
	та теоретичних джерел	Січень 2026	виконано
3.	Розробка першого розділу:		
	огляд інформаційних систем та технологій для їхнього створення	Лютий 2026	виконано
4.	Розробка другого розділу:		
	розробка проєкту інформаційної системи та вибір технологій	Квітень 2026	виконано
5.	Розробка третього розділу:		
	розробка застосунку	Травень 2026	виконано
6.	Оформлення і нормоконтроль кваліфікаційної роботи та перевірка на плагіат	Червень 2026	виконано
7.	Захист кваліфікаційної роботи	16.06.2026	виконано

Здобувач освіти _____ Андрій ДМИТРІЄНКО
(підпис) (ім'я ПРІЗВИЩЕ)

Керівник роботи _____ Дмитро ДРАЄВСЬКИЙ
(підпис) (ім'я ПРІЗВИЩЕ)

Нормоконтроль пройдено

Нормоконтролер _____ Юлія БОРИСОВСЬКА
(підпис) (ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота: 42 сторінки, 10 рисунків, 13 джерел.

Об'єкт дослідження: процес розробки, впровадження та використання інформаційних вебсистем для обліку академічної успішності в освітньому процесі.

Предмет дослідження: вебзастосунок для автоматизації обліку оцінок студентів.

Мета роботи: розробка та реалізація вебзастосунку, який автоматизує процес виставлення, зберігання та перегляду оцінок студентів з розмежуванням прав доступу.

Метод дослідження: аналітичний, експериментальний, порівняльний аналіз, моделювання баз даних.

У кваліфікаційній роботі проведено детальний аналіз та розроблено цілісний процес створення клієнт-серверного вебзастосунку. Використовуючи теоретичні дані процесу веброзробки, а також середовище Node.js, бібліотеку React та систему управління базами даних PostgreSQL, було розроблено функціональний застосунок з назвою «Електронний журнал успішності».

В рамках дослідження було ретельно спроєктовано та реалізовано інтерфейс застосунку та архітектуру бази даних, що повністю задовольняє вимоги користувачів та забезпечує ефективну і безпечну взаємодію з даними. З використанням сучасних інструментів розробки, фреймворку Express.js, ORM Sequelize та технології JWT-автентифікації було успішно створено систему обліку.

Цей застосунок є результатом глибокого вивчення процесу розробки сучасних вебдодатків, а його функціональність спрямована на надання студентам і викладачам зручного цифрового інструменту для моніторингу та управління академічними результатами. Робота демонструє не лише знання

теоретичних аспектів проєктування вебсистем, але й вміння їх успішно реалізувати в практичному застосуванні.

У результаті проведеного дослідження було досягнуто поставленої мети, а саме розроблено та реалізовано вебзастосунок, який забезпечує автоматизацію обліку успішності. Цей застосунок може бути використаний для підвищення ефективності освітнього процесу, зменшення паперового документообігу та поліпшення контролю за рівнем знань.

REACT, NODE.JS, POSTGRESQL, EXPRESS.JS, JWT, ВЕБЗАСТОСУНОК, ЕЛЕКТРОННИЙ ЖУРНАЛ

SUMMARY

Qualification Work: 42 pages, 10 figures, 13 references.

Object of Study: The process of developing, implementing, and using web-based information systems for academic performance tracking in the educational process.

Subject of Study: Web application for automating student grade tracking.

Purpose of the Work: To develop and implement a web application that automates the process of grading, storing, and reviewing student marks with role-based access control.

Research Methods: Analytical, experimental, comparative analysis, database modeling.

In the qualification work, a detailed analysis and comprehensive process for creating a client-server web application were conducted. Utilizing theoretical data on the web development process, as well as the Node.js environment, React library, and PostgreSQL database management system, a functional application named "Student Journal" was developed.

As part of the study, the interface of the application and the database architecture were carefully designed and implemented to fully meet user requirements and ensure effective and secure data interaction. Using modern development tools, the Express.js framework, Sequelize ORM, and JWT authentication technology, the performance tracking system was successfully created.

This application is the result of an in-depth study of the modern web application development process, and its functionality is aimed at providing students and teachers with a convenient digital tool for monitoring and managing academic results. The work demonstrates not only the theoretical knowledge of web system design but also the ability to successfully implement it in practical use.

As a result of the research, the objective was achieved: a web application that ensures the automation of academic performance tracking was developed and implemented. This application can be used to enhance the efficiency of the educational process, reduce paperwork, and improve the control over the level of knowledge.

REACT, NODE.JS, POSTGRESQL, EXPRESS.JS, JWT, WEB APPLICATION, ELECTRONIC JOURNAL

ЗМІСТ

Завдання на кваліфікаційну роботу студенту 2
Summary 6
Вступ 9
1 Основи розробки освітнього вебзастосунку 11
1.1 Загальні відомості про розробку освітніх платформ 11
1.2 Поняття освітнього вебзастосунку та їх класифікація 12
1.3 Огляд існуючих аналогів 13
1.4 Функціональні можливості існуючих рішень 16
1.5. Підсумки аналітичного дослідження 18
2. Проєктування вебзастосунку «електронний журнал успішності» 19
2.1. Визначення технічних вимог та інструментів розробки 19
2.2. Проєктування архітектури та бази даних 21
2.3. Проєктування функціональної структури застосунку 23
2.4. Результати проєктування архітектури вебзастосунку 25
3. Програмна реалізація та тестування вебзастосунку «Електронний журнал успішності» 26
3.1. Реалізація серверної логіки та взаємодії з базою даних 26
3.2 Розробка клієнтського інтерфейсу та функціональних модулів 32
3.3 Результати розробки та тестування застосунку 35
3.4 Висновки щодо практичної реалізації системи 37
Висновки 39
Перелік використаних джерел 41

ВСТУП

Вебтехнології стали невід’ємною частиною сучасного освітнього процесу, забезпечуючи користувачів широким спектром можливостей: від швидкого доступу до навчальних матеріалів до автоматизації рутинних адміністративних задач. У зв’язку з цим, розробка спеціалізованих вебзастосунків для обліку успішності стає все більш актуальною та затребуваною сферою діяльності. Однією з ключових задач, що стоять перед розробниками освітніх платформ, є створення зручних, ефективних та безпечних інформаційних систем, які б задовольняли потреби як викладачів, так і студентів, та відповідали сучасним стандартам якості.

Дана кваліфікаційна робота присвячена розробці вебзастосунку «Електронний журнал успішності», призначеного для автоматизації процесу виставлення, зберігання та перегляду оцінок у навчальному закладі. Метою цього проєкту є заміна застарілих паперових журналів на надійну цифрову систему, надання зручного інструменту моніторингу успішності та забезпечення суворого розмежування прав доступу для різних ролей користувачів.

Вступна частина роботи охоплює основні етапи та аспекти розробки клієнт-серверного вебзастосунку.

У першому розділі проводиться огляд предметної області, існуючих інформаційних систем для обліку академічної успішності та актуальних технологій їх створення. Здійснюється аналіз потреб учасників освітнього процесу, огляд функціональних можливостей аналогічних рішень та формування загального бачення проєкту. Важливим компонентом є визначення архітектурних концепцій та технічних вимог до системи електронного журналу.

Другий розділ розкриває процес проєктування системи, вибір технологій розробки (таких як середовище виконання Node.js, вебфреймворк

Express.js, бібліотека для побудови інтерфейсів React та реляційна база даних PostgreSQL) та розробку функціональної структури застосунку. Проєктування інтерфейсу користувача охоплює створення гнучкого, сучасного та інтуїтивно зрозумілого дизайну з використанням Tailwind CSS, який забезпечує зручність роботи з даними. Також детально проєктується структура реляційної бази даних (Sequelize ORM) та описується механізм багаторівневого розмежування доступу (Студент, Викладач, Куратор, Адміністратор) на основі JWT-автентифікації.

У третьому розділі представлена програмна реалізація вебзастосунку «Електронний журнал успішності». Описується процес розробки серверної частини, включаючи реалізацію REST API, бізнес-логіки контролерів та middleware для перевірки ролей користувачів. Також розглядається реалізація клієнтської частини, впровадження інструментів управління глобальним станом, інтеграція мультимовності та взаємодія клієнта з сервером. Описується тестування розробленого функціоналу та можливості вдосконалення і масштабування застосунку в майбутньому.

У висновку підводяться підсумки виконаної роботи. Результатом цієї кваліфікаційної роботи є створення повністю функціонального вебзастосунку, який дозволяє ефективно оптимізувати адміністративне навантаження на викладачів та надає студентам прозорий інструмент для моніторингу власних оцінок. Продукт реалізовано на основі передових вебтехнологій, що забезпечує його високу продуктивність, стабільність роботи та надійний захист даних. У підсумку робота демонструє повний цикл створення програмного забезпечення від збору вимог і проєктування БД до розробки повноцінної клієнт-серверної системи.

1 ОСНОВИ РОЗРОБКИ ОСВІТНЬОГО ВЕБЗАСТОСУНКУ

1.1 Загальні відомості про розробку освітніх платформ

Основи розробки вебзастосунків для освітньої сфери зосереджуються на ключових аспектах створення інформаційних систем для закладів освіти, які включають розгляд понять, класифікацію, аналіз інформаційних потреб учасників освітнього процесу, аналіз конкурентів та огляд функціональних можливостей засобів, що використовуються для організації навчального процесу. В умовах стрімкої цифровізації суспільства перехід від паперових методів обліку до автоматизованих цифрових платформ стає не просто перевагою, а необхідним стандартом якості надання освітніх послуг.

Дослідження інформаційних потреб користувачів (студентів, викладачів, адміністрації) розглядає процес аналізу вимог до системи електронного обліку, що є важливим етапом перед безпосередньою розробкою. Цей аналіз дозволяє виявити "вузькі місця" у традиційному документообігу, такі як затримки у передачі інформації, складність ведення статистики та ризики механічних помилок при підрахунку середніх балів. Для розуміння аспектів та подальшої розробки конкретного вебзастосунку з обліку успішності потрібно зробити декілька послідовних кроків.

Першим кроком є проведення детального огляду аналогів майбутнього застосунку, що включає визначення їх основних функціональних можливостей. Розуміння потреб цільової аудиторії та очікувань користувачів дозволить створити продукт, який буде відповідати реаліям сучасного навчального закладу. Це передбачає не лише вивчення програмного коду аналогів, але й аналіз їхньої ергономіки, швидкодії та надійності.

Другим кроком є визначення кращих практик та уникнення поширених помилок при розробці нової інформаційної системи. Аналіз цих аспектів дозволяє створити вебзастосунок з широким спектром функцій, які

підвищують зручність, безпеку та ефективність його використання. Огляд основних аспектів розробки допоміг зрозуміти процес створення електронного журналу та його подальшого практичного застосування, а також визначити перші кроки для проєктування застосунку.

1.2 Поняття освітнього вебзастосунку та їх класифікація

Поняття освітнього вебзастосунку (електронного журналу) охоплює програмне забезпечення, призначене для автоматизації збору, зберігання, обробки та надання інформації про хід навчального процесу та академічну успішність здобувачів освіти через веббраузер [3]. Такі системи базуються на архітектурі клієнт-сервер, де серверна частина забезпечує збереження цілісності даних, а клієнтська - надає зручний інтерфейс для взаємодії [9]. Ці системи забезпечують оптимізацію рутинної роботи викладачів та адміністрації, а також надають студентам інструменти для самоконтролю.

Електронні журнали успішності стають критичною необхідністю для навчальних закладів, оскільки вони забезпечують цілодобовий доступ до інформації, усувають ризики втрати паперових носіїв та сприяють прозорості оцінювання. Завдяки впровадженню таких систем кожен учасник освітнього процесу отримує персоналізований доступ до необхідних йому даних, що значно підвищує рівень довіри та мотивації студентів.

Класифікація освітніх інформаційних систем може бути здійснена за кількома критеріями, що навдені нижче [3].

1. За функціональністю:

– системи управління навчанням (LMS), які охоплюють повний цикл навчання від подачі матеріалу до тестування;

– електронні журнали та щоденники (вузькоспеціалізовані системи обліку), зосереджені саме на фіксації результатів успішності та відвідуваності;

– системи тестування та дистанційного контролю знань, призначені для автоматизованої перевірки рівня засвоєння матеріалу;

– комплексні інформаційно-аналітичні системи управління закладом освіти, що включають фінансові, кадрові та навчальні модулі.

2. За рівнем впровадження та масштабом:

– локальні (в межах однієї кафедри, факультету або окремого навчального закладу);

– регіональні системи, що об'єднують мережу закладів певного регіону;

– загальнонаціональні освітні платформи, які координуються на рівні державних міністерств.

3. За моделлю розгортання (архітектурою):

– хмарні вебзастосунки (SaaS), де програмне забезпечення працює на потужностях провайдера, а навчальний заклад отримує доступ через інтернет;

– системи, що розгортаються на локальних серверах навчального закладу (On-Premise), що дозволяє повністю контролювати фізичне розташування даних.

4. За типом доступу:

– відкриті платформи з публічною реєстрацією для широкої аудиторії;

– закриті корпоративні системи з попередньо згенерованими акаунтами, що забезпечують високий рівень приватності.

Розуміння цих понять і класифікація систем допомагає розробникам краще орієнтуватися у різноманітності програмного забезпечення та обрати оптимальний вектор розробки власного проєкту, враховуючи специфіку спеціальності "Інженерія програмного забезпечення".

1.3 Огляд існуючих аналогів

Аналіз аналогів застосунку дозволив вивчити та порівняти існуючі інформаційні системи, які схожі за функціональністю та цілями, і

відповідають запитам освітніх закладів. Для визначення найбільш ефективних та корисних підходів до розробки клієнт-серверного вебзастосунку «Електронний журнал успішності» було проведено детальний огляд існуючих програмних рішень у цій області.

Спочатку було проведено аналіз ринку вебсистем для обліку успішності. Було переглянуто найпопулярніші спеціалізовані платформи, такі як:

1. Платформа «Нові знання» (nz.ua). Даний програмний продукт орієнтований на заклади загальної середньої освіти та забезпечує синхронізацію даних із центральною базою ЕДЕБО. Система дозволяє вести облік оцінок, відвідуваності та формувати стандартні звіти для адміністрації школи (рис. 1.1). Основним недоліком системи є складна структура користувацького інтерфейсу. Робочий простір перевантажений великою кількістю дрібних табличних форм без належного графічного розподілу. Це створює додаткове навантаження на викладачів під час щоденного внесення великих обсягів інформації та збільшує час на освоєння програми.

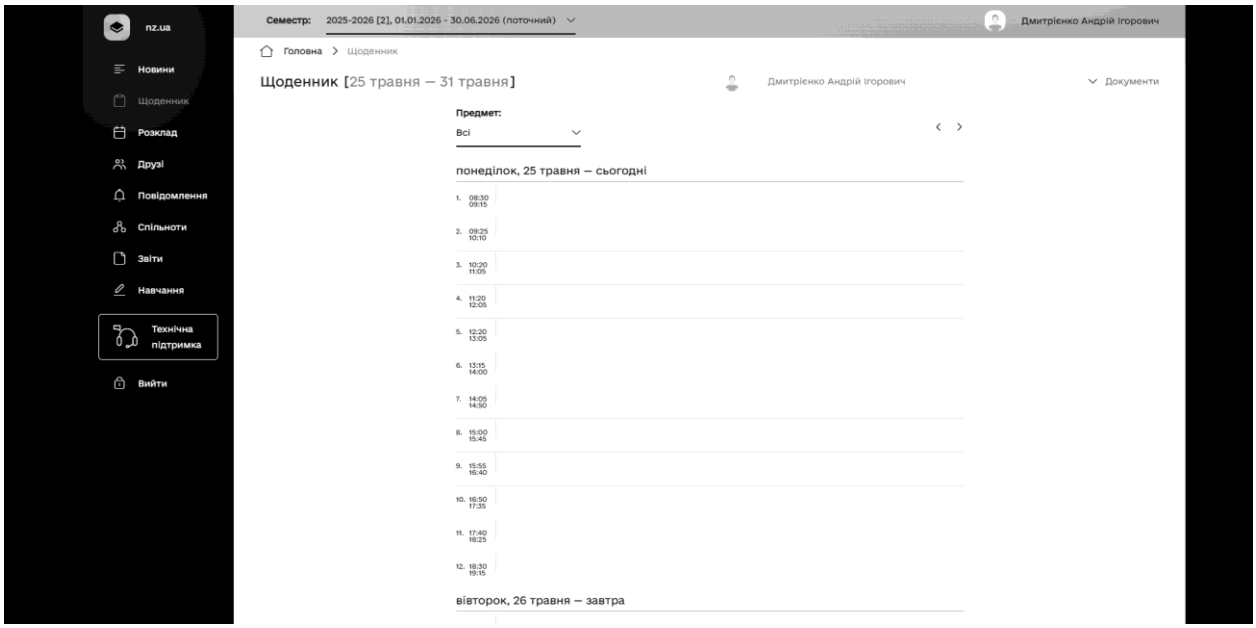


Рисунок 1.1 – Інтерфейс платформи «Нові знання»

2. Комплексна платформа «Єдина школа» (eschool-ua.com). Ця інформаційна система містить вебінтерфейс та додаткові мобільні додатки для

учнів та батьків, що спрощує моніторинг успішності. Платформа має більш сучасний дизайн порівняно з аналогами, проте характеризується низьким рівнем оптимізації серверної частини (рис. 1.2). Практичний досвід використання показує, що в періоди масового виставлення підсумкових оцінок відбуваються збої та тривалі затримки при оновленні даних. Також процес початкового конфігурування бази даних під конкретну структуру закладу є занадто заплутаним.

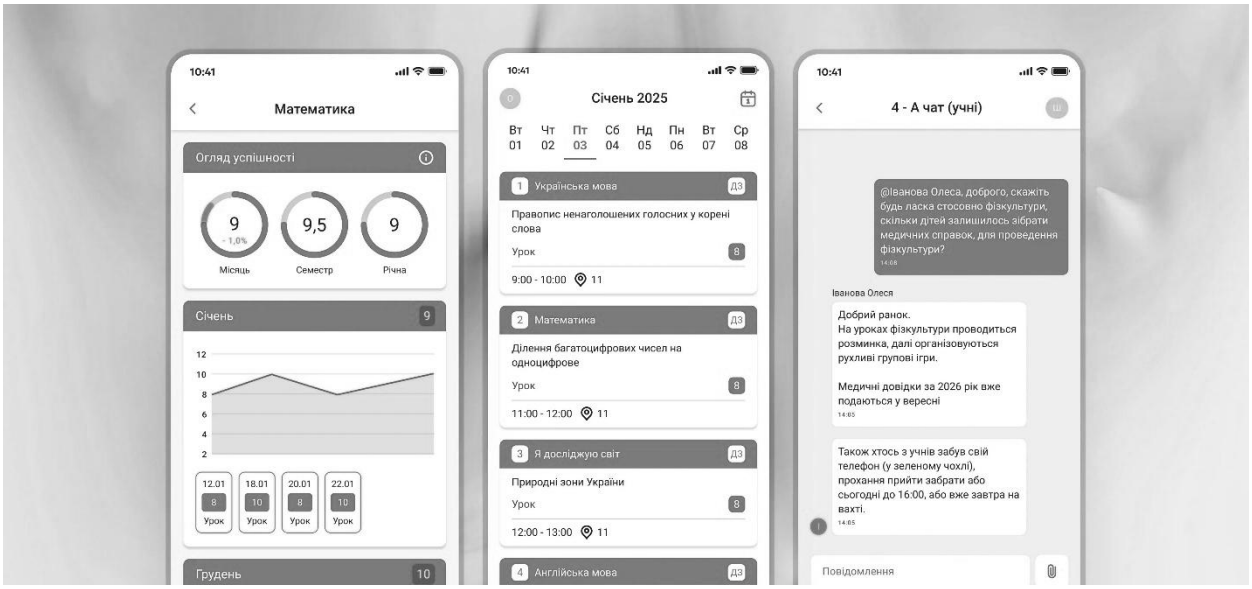


Рисунок 1.2 – Інтерфейс платформи «Єдина школа»

3. Система «АСУ Студент». Вказаний комплекс використовується переважно у вищих навчальних закладах та коледжах для автоматизації роботи відбіркових комісій та деканатів. Програма стабільно працює зі значними масивами даних, але розроблялася без урахування актуальних вимог UI/UX-дизайну (рис. 1.3). Процес виконання стандартних операцій (пошук картки студента чи заповнення відомості) вимагає переходу через багаторівневі текстові меню. Така побудова інтерфейсу уповільнює роботу та призводить до помилок через людський фактор.

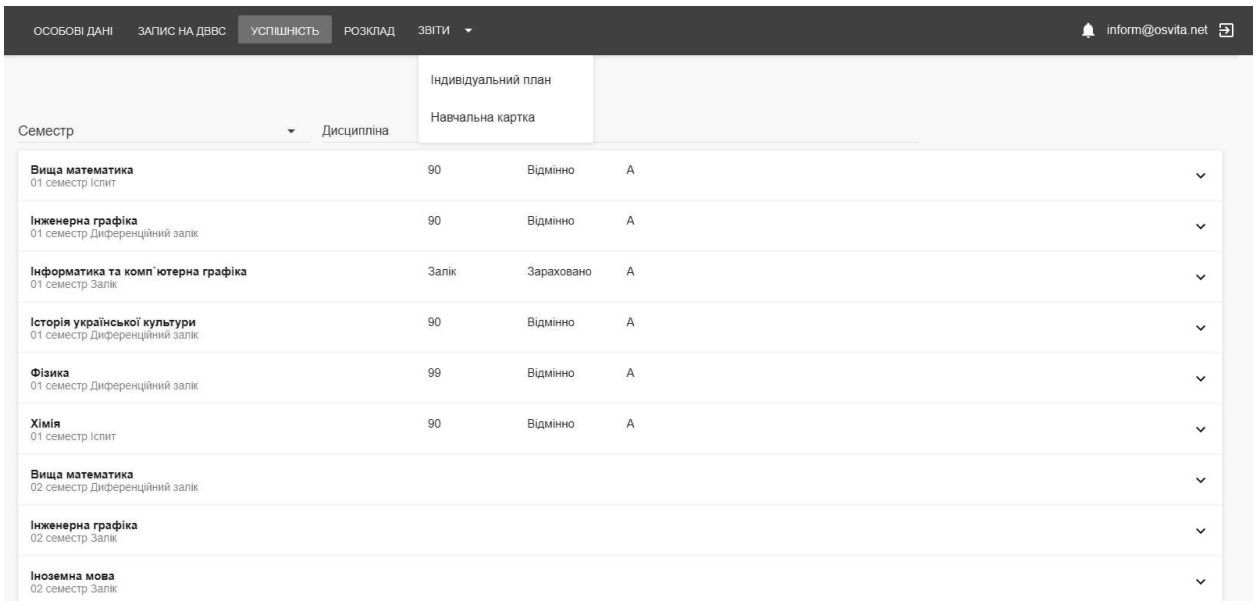


Рисунок 1.3 – Інтерфейс платформи «АСУ студент»

Також були розглянуті комплексні рішення на кшталт Moodle та Google Classroom. Moodle є потужним інструментом з відкритим кодом, що надає величезні можливості для кастомізації журналів, проте його налаштування вимагає високої кваліфікації системного адміністратора. Google Classroom, у свою чергу, є надзвичайно простим у використанні, але його журнал оцінок має обмежений функціонал у частині складної аналітики та створення кастомних типів робіт.

1.4 Функціональні можливості існуючих рішень

Дослідження функціональних можливостей існуючих електронних журналів дозволило з’ясувати різноманітні аспекти, які важливі для користувача під час експлуатації таких систем. Ці можливості формують базу вимог до проектованого застосунку.

Перш за все, були детально проаналізовані механізми реалізації рольових моделей доступу. Якісна система повинна чітко розмежовувати права адміністратора (повний доступ), куратора (управління групами),

викладача (виставлення балів) та студента (перегляд). Важливо, щоб права доступу перевірялися не лише на рівні інтерфейсу, а й на рівні серверних запитів до API.

Також були вивчені можливості прив'язки студентів до академічних груп та навчальних курсів, що дозволяє системі динамічно формувати списки для викладачів. Особлива увага приділена процесу виставлення та редагування оцінок: система повинна підтримувати гнучку шкалу (наприклад, 0-100 балів), дозволяти вказувати тип роботи (лекція, практична, самостійна робота, проєкт) та залишати текстові коментарі для деталізації результату. Це дозволяє студенту розуміти причину отримання певного балу.

Іншим критичним аспектом є наявність зведених електронних таблиць, які не лише відображають бали, але й автоматично підраховують загальні показники успішності студента з певного предмету. Використання кольорового кодування (зелений для високих балів, синій для достатніх, жовтий для середніх та червоний для незадовільних) значно покращує візуальне сприйняття та аналітику, дозволяючи викладачу миттєво ідентифікувати студентів, які мають заборгованості.

Важливою є також функція миттєвої синхронізації даних через технології AJAX або WebSocket, що дозволяє студентам відразу дізнаватися свої результати у власному кабінеті без необхідності оновлення сторінки чи надсилання запитів до деканату. Крім того, сучасні системи підтримують експорт даних у формати PDF або Excel для формування офіційної звітності.

Завдяки детальному аналізу стали зрозумілими очікування користувачів від продукту. Та сформульовано конкретні вимоги до функціональності вебзастосунку «Електронний журнал успішності». Це дозволило визначити ключові функції (JWT-автентифікація, розмежування ролей, управління групами та оцінками, мультимовність), які необхідно реалізувати для створення зручного та безпечного інструменту.

1.5. Підсумки аналітичного дослідження

Проведений аналіз основ розробки освітніх вебзастосунків підтверджує актуальність створення спеціалізованої системи «Електронний журнал успішності». У розділі було досліджено теоретичне підґрунтя функціонування таких систем, їх класифікацію за архітектурними та функціональними ознаками.

Огляд існуючих аналогів, таких як «Нові знання» та інших систем, виявив як сильні сторони (висока функціональність), так і недоліки (складність інтерфейсу, відсутність гнучкого налаштування під конкретний навчальний заклад). Дослідження функціональних можливостей дозволило сформувати перелік необхідних модулів для власної розробки, зокрема: надійна система автентифікації на базі JWT, гнучка база даних на PostgreSQL та динамічний інтерфейс на базі React.

Ретельний аналіз потреб викладачів та студентів став фундаментом для наступного етапу роботи - проектування архітектури застосунку, структури бази даних та визначення логіки взаємодії компонентів системи. Отримані результати дозволяють перейти до розробки технічних специфікацій, які забезпечать створення конкурентоспроможного та ефективного програмного продукту.

2. ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ «ЕЛЕКТРОННИЙ ЖУРНАЛ УСПІШНОСТІ»

2.1. Визначення технічних вимог та інструментів розробки

Проєктування вебзастосунку це критично важливий етап у життєвому циклі розробки програмного забезпечення, що закладає фундамент для всієї подальшої реалізації. Даний процес передбачає не лише вибір інструментарію, а й глибоке опрацювання архітектури системи, структури бази даних, створення ергономічного інтерфейсу користувача та детальну розробку функціональних можливостей, що максимально відповідають потребам цільової аудиторії.

У цьому розділі детально розглядається процес проєктування клієнт-серверної системи «Електронний журнал успішності». На етапі проєктування було враховано необхідність створення масштабованого та безпечного продукту, який забезпечить ефективну взаємодію між усіма групами користувачів: студентами, викладачами, кураторами та адміністраторами закладу. Особлива увага приділена розмежуванню прав доступу та цілісності даних у реляційній моделі.

Визначення технічних вимог включає в себе конкретизацію технологічного стеку, а також формування нефункціональних вимог, таких як безпека, швидкодія та адаптивність. Для розробки було обрано сучасний стек технологій, що базується на мові JavaScript як для серверної, так і для клієнтської частини.

Обґрунтування вибору інструментів серверної частини:

– Node.js та Express.js. Серверну частину побудовано на базі середовища виконання Node.js (версії 18+) [2]. Використання асинхронної моделі введення-виведення Node.js дозволяє системі обробляти велику кількість одночасних запитів без втрати продуктивності [1]. Вебфреймворк

Express.js обрано через його мінімалістичність та гнучкість у налаштуванні маршрутизації REST API;

– PostgreSQL та Sequelize ORM. Для зберігання даних обрано реляційну СУБД PostgreSQL, яка відома своєю надійністю та підтримкою складних запитів [11]. Взаємодія з БД здійснюється через Sequelize ORM [10]. Це дозволяє абстрагуватися від прямого написання SQL-запитів, прискорює розробку та забезпечує безпеку від SQL-ін'єкцій;

– JWT та bcryptjs. Безпека системи базується на використанні JSON Web Tokens для автентифікації без збереження стану на сервері [12]. Для захисту конфіденційних даних застосовується алгоритм хешування паролів bcrypt, що робить неможливим їх дешифрування у разі витоку бази даних.

Обґрунтування вибору інструментів клієнтської частини:

– React та Vite. Для побудови динамічного інтерфейсу обрано бібліотеку React. Вона дозволяє реалізувати концепцію SPA (Single Page Application), що забезпечує миттєвий відгук інтерфейсу [5]. Інструмент збірки Vite використовується для оптимізації процесу розробки та отримання максимально полегшених файлів для кінцевого користувача;

– Zustand та React Router DOM. Управління станом авторизації та глобальними даними реалізовано через Zustand - сучасну альтернативу Redux, яка має значно менший обсяг коду та високу швидкість роботи [7]. Маршрутизація забезпечує логічний поділ застосунку на сторінки без перезавантаження браузера;

– Tailwind CSS. Стилізація виконана за допомогою utility-first фреймворку, що дозволяє створювати унікальний та адаптивний дизайн. Це гарантує коректне відображення журналу як на великих моніторах, так і на мобільних пристроях студентів (рис. 2.1).;

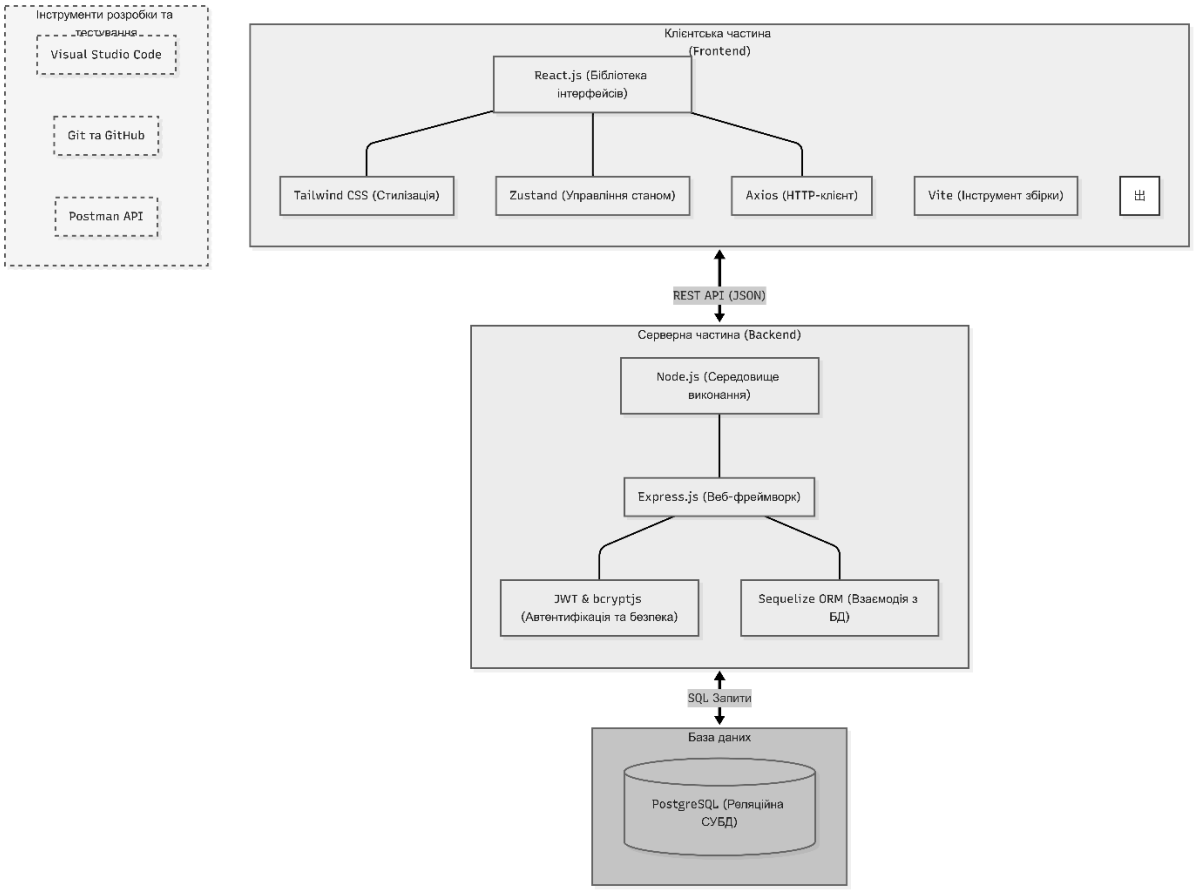


Рисунок 2.1 – Схема технологічного стеку проєкту

2.2. Проєктування архітектури та бази даних

Застосунок побудований за трирівневою архітектурою: рівень представлення (React), рівень бізнес-логіки (Express) та рівень даних (PostgreSQL) [9]. Клієнт взаємодіє з сервером через захищені HTTP-запити, отримуючи відповіді у форматі JSON.

Важливим етапом проєктування стало створення логічної структури бази даних. Реляційна модель розроблена з дотриманням нормальних форм для усунення надмірності даних [11].

Деталізація структури таблиць та асоціацій (рис. 2.2):

1. Таблиця users. Є центральною таблицею системи. Окрім базових полів (id, email, password), вона містить дискримінатор role, який визначає

рівень доступу. Поле group_id є зовнішнім ключем, що вказує на академічну групу студента.

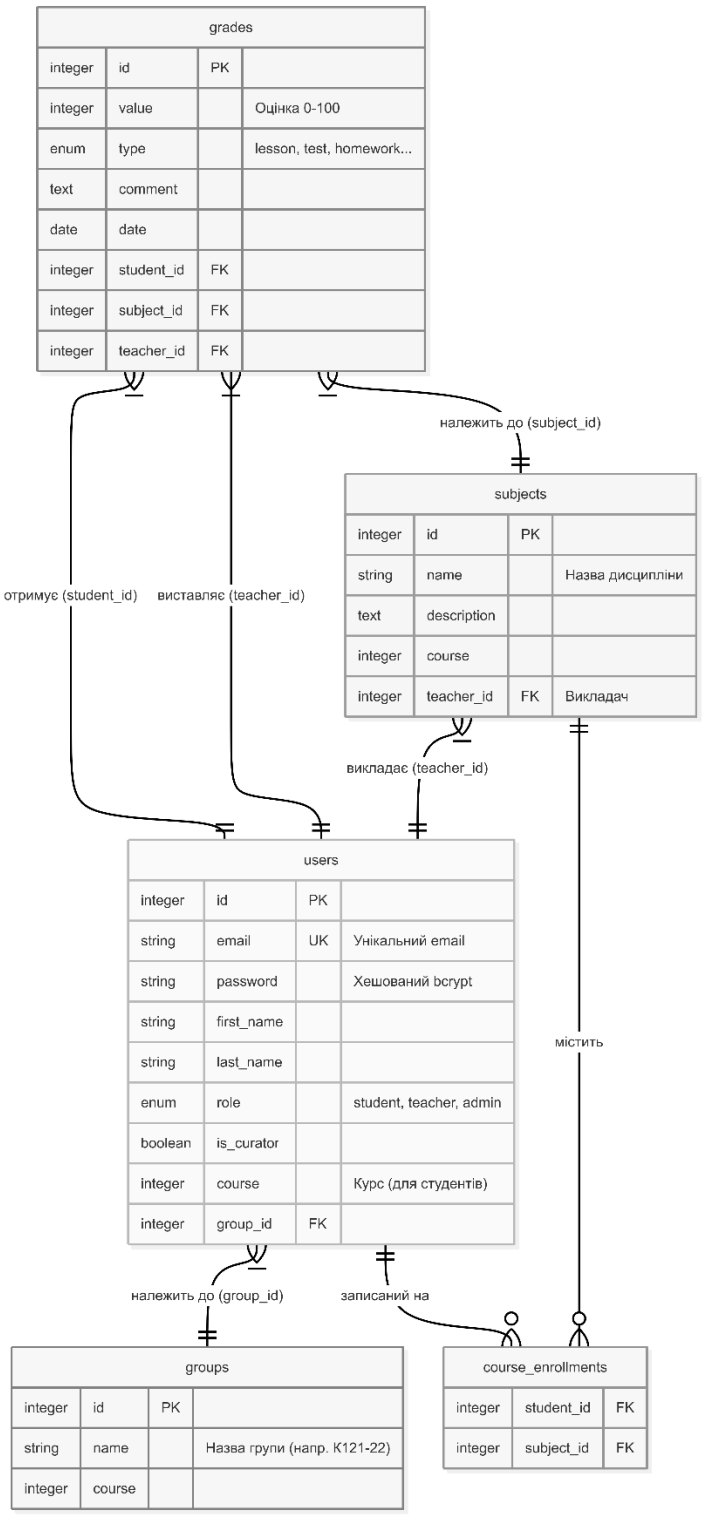


Рисунок 2.2 – ER-діаграма бази даних (схема зв'язків)

2. Таблиця subjects. Містить назву та опис дисципліни. Поле teacher_id створює зв'язок "один-до-багатьох", закріплюючи предмет за конкретним викладачем.

3. Таблиця grades. Це найбільш завантажена таблиця, що фіксує кожен навчальний результат. Окрім балу (0-100), зберігається поле type, що дозволяє проводити гнучку аналітику успішності. Таблиця має зв'язки з users (студент та викладач) та subjects.

4. Таблиця groups. Використовується для організаційної структури. Кожна група має унікальну назву та закріплена за певним курсом навчання.

5. Таблиця course_enrollments. Виконує роль сполучної таблиці для реалізації зв'язку "багато-до-багатьох" між студентами та предметами. Це дозволяє одному студенту бути записаним на багато предметів, а одному предмету - містити багато студентів.

2.3. Проєктування функціональної структури застосунку

Функціональна структура веб-системи базується на рольовій моделі доступу, де кожна роль має власний робочий простір та обмежений набір доступних API-ендпоінтів.

Модуль студента орієнтований на надання користувачеві прозорої інформації про його навчальний прогрес. Базовим елементом модуля є інтерактивний журнал, який забезпечує відображення всіх отриманих оцінок у хронологічному порядку або згрупованих за предметами. Для зручності сприйняття впроваджено кольорову індикацію, що візуально розподіляє бали за рівнем успішності: зелений колір (90-100 балів), синій (75-89), жовтий (60-74) та червоний (менше 60). Додатково реалізовано аналітичний блок, який автоматично розраховує середній бал з кожної дисципліни та формує загальний рейтинг здобувача освіти.

Модуль викладача розроблено з урахуванням вимог щодо зручності ведення академічного обліку. Через панель управління викладач отримує доступ виключно до тих дисциплін, які за ним закріплені. Інструмент електронної відомості дозволяє переглядати повний список студентів обраної групи разом із їхніми поточними оцінками. Також функціонал модуля підтримує повний спектр операцій з даними, що дає змогу створювати нові записи, редагувати існуючі бали у випадку апеляції або видаляти помилково виставлені оцінки.

Модуль куратора поєднує в собі розширені функції викладача та адміністратора окремої групи. До його завдань входить менеджмент контингенту, що включає додавання нових студентів до груп та актуалізацію їхніх персональних даних. Крім того, куратор має права на формування навчального плану, створення нових предметів та закріплення їх за відповідними курсами й академічними групами.

Модуль адміністратора призначений для глобального налаштування та управління системою. Його основний функціонал охоплює управління персоналом, зокрема створення облікових записів для викладачів та призначення їм статусу куратора. Також адміністратор здійснює аудит системи, переглядає активність користувачів та керує глобальними параметрами навчального закладу.

Технічна логіка процесу виставлення оцінки передбачає сувору перевірку даних на кожному етапі взаємодії компонентів та складається з наступних кроків:

– валідація на стороні клієнта (фреймворк React перевіряє цілісність введеного числа та його належність до діапазону від 0 до 100) [5];

– формування POST-запиту, який містить JWT-токен викладача;

– авторизація на сервері (функція authenticate перевіряє валідність токена, а middleware-компонент isTeacher підтверджує наявність відповідних прав доступу);

– бізнес-валідація (сервер перевіряє факт викладання даного предмета конкретним викладачем та наявність студента на цьому курсі);

– запис даних до бази PostgreSQL через ORM Sequelize та оновлення стану клієнтської частини за допомогою менеджера Zustand для миттєвого відображення змін.

2.4. Результати проєктування архітектури вебзастосунку

У ході проєктування було повністю сформовано архітектурний вигляд майбутнього вебзастосунку «Електронний журнал успішності». Обґрунтовано вибір сучасного стеку технологій (Node.js, Express, PostgreSQL, React), який дозволить системі працювати швидко та надійно в умовах реального навчального процесу [2].

Розроблена структура бази даних забезпечує цілісність інформації та дозволяє зберігати складні зв'язки між студентами, викладачами та предметами. Проєктування функціональної структури та рольової моделі доступу гарантує, що кожен користувач матиме доступ лише до дозволених йому даних, що є критичним для систем освітнього обліку.

Ретельне опрацювання всіх аспектів проєктування на цьому етапі дозволяє мінімізувати ризики при подальшій програмній реалізації та створює передумови для розробки якісного програмного продукту, що відповідає стандартам спеціальності «Інженерія програмного забезпечення». Спроектована архітектура є гнучкою, що дає змогу в майбутньому легко додавати новий функціонал без докорінної зміни структури системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ «ЕЛЕКТРОННИЙ ЖУРНАЛ УСПІШНОСТІ»

3.1. Реалізація серверної логіки та взаємодії з базою даних

Процес програмної реалізації вебзастосунку є найбільш трудомістким та відповідальним етапом розробки, який перетворює теоретичні архітектурні схеми у функціональний продукт. Даний етап включає безпосереднє написання програмного коду, налаштування серверного середовища, конфігурацію бази даних та інтеграцію всіх компонентів системи у єдину екосистему. Основна увага під час розробки була приділена забезпеченню високої швидкодії застосунку, безпеці персональних даних користувачів та стабільності обробки REST-запитів.

Під час реалізації проєкту основним інструментом розробки було обрано інтегроване середовище Visual Studio Code. Вибір обґрунтований наявністю великої кількості розширень для роботи з Node.js та React, вбудованим терміналом та зручними засобами відлагодження коду [2].

Серверна частина системи базується на платформі Node.js, яка забезпечує виконання JavaScript-коду на стороні сервера [2]. Клієнтський інтерфейс побудований на базі сучасної бібліотеки React, що дозволяє створювати динамічні компоненти [5]. Взаємодія між цими частинами здійснюється через архітектурний стиль REST API, де обмін даними відбувається у форматі JSON [9]. Це забезпечує високу гнучкість системи та можливість її подальшого розширення (рис. 3.1).

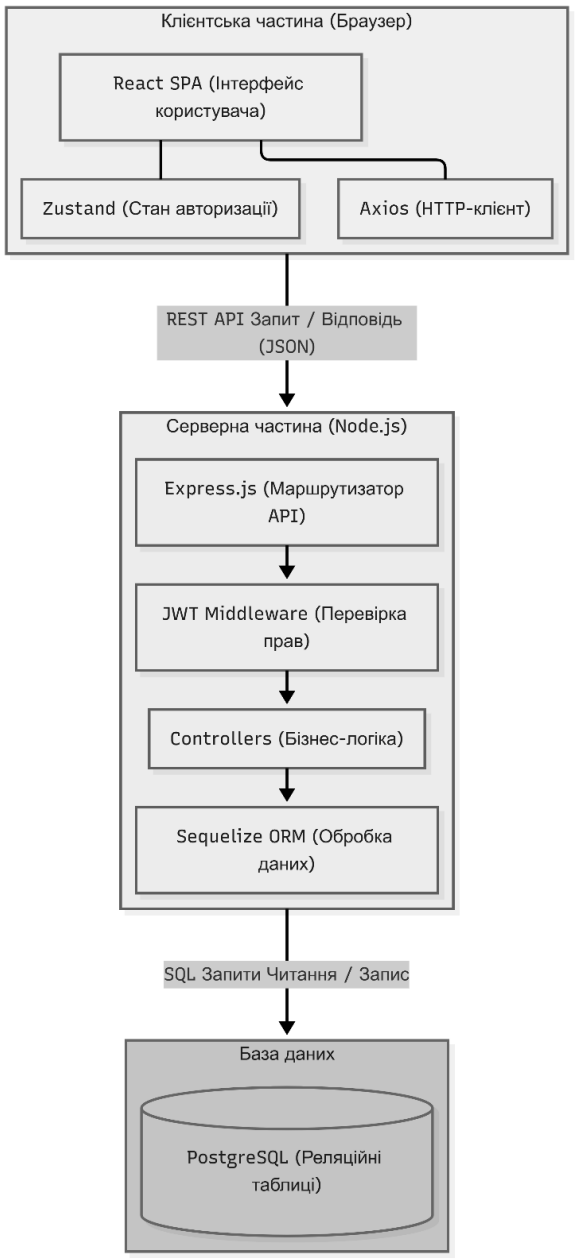


Рисунок 3.1 – Схема архітектурної взаємодії компонентів вебзастосунку

Реалізація серверної частини стала фундаментом для функціонування всього проєкту. Вона відповідає за обробку запитів, виконання бізнес-логіки, взаємодію з базою даних PostgreSQL та забезпечення безпеки. Весь бекенд побудований за модульним принципом, що дозволяє легко модифікувати окремі частини системи без впливу на загальну стабільність [6].

Порядок кроків та деталізація реалізації бекенд-частини:

1. Ініціалізація та налаштування середовища. Процес розпочався з формування структури проєкту та створення файлу package.json. Було

встановлено базовий набір залежностей: express для створення вебсервера, sequelize та pg для роботи з PostgreSQL, jsonwebtoken для автентифікації та bcryptjs для безпечного хешування паролів. Також було налаштовано змінні оточення у файлі .env, де зберігаються конфіденційні дані, такі як пароль до бази даних та секретний ключ для підпису токенів.

2. Конфігурація підключення до БД. У файлі config/database.js було реалізовано логіку підключення до СУБД PostgreSQL через екземпляр Sequelize [11]. Це дозволило використовувати сучасний підхід до управління даними. Було налаштовано пул з'єднань для оптимізації навантаження на сервер та механізм автоматичної перевірки зв'язку з базою при запуску сервера:

```
import { Sequelize } from 'sequelize';
import dotenv from 'dotenv';

dotenv.config();

const sequelize = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASSWORD,
  {
    host: process.env.DB_HOST,
    port: process.env.DB_PORT,
    dialect: 'postgres',
    logging: false,
    pool: {
      max: 5,
      min: 0,
      acquire: 30000,
      idle: 10000
    }
  }
);

export default sequelize;
```

Розробка моделей даних та асоціацій. У папці models було створено опис структур таблиць у вигляді JavaScript-класів. Модель User містить дані користувачів, Subject - навчальні предмети, Grade - оцінки, а Group - академічні групи. Важливим етапом стало налаштування складних асоціацій між ними у файлі models/index.js. Зокрема, було реалізовано зв'язок "багато до багатьох" між студентами та предметами через проміжну таблицю CourseEnrollment, а також ієрархічні зв'язки між викладачами та виставленими ними оцінками [10].

Реалізація механізму автентифікації. Для управління доступом було створено контролер authController.js. Він містить дві ключові функції:

– register: приймає дані нового користувача, перевіряє унікальність email, хешує пароль за допомогою bcryptjs (з використанням солі 10 раундів) та зберігає запис у базу.

– login: перевіряє наявність користувача, порівнює хеш введеного пароля з тим, що зберігається в БД, та у разі успіху генерує JWT-токен, що містить ID користувача та його роль [13].

Розробка системи маршрутизації та Middleware. Серверна логіка організована за принципом розділення відповідальності. Маршрути лише приймають запити та передають їх відповідним контролерам. Було розроблено систему захисту маршрутів у папці middleware. Функція authenticate перевіряє наявність та валідність JWT-токена у заголовку запиту. Якщо токен вірний, дані користувача додаються до об'єкта запиту req.user, що дозволяє наступним функціям знати, хто саме робить запит [13]:

```
import jwt from 'jsonwebtoken';
import { User } from '../models/index.js';

export const authenticate = async (req, res, next) => {
  try {
    const authHeader = req.headers.authorization;

    if (!authHeader || !authHeader.startsWith('Bearer ')) {
      return res.status(401).json({
```

```
      success: false,
      message: 'Токен не надано. Будь ласка, увійдіть у систему.'
    });
  }

  const token = authHeader.split(' ')[1];
  const decoded = jwt.verify(token, process.env.JWT_SECRET);
  const user = await User.findByPk(decoded.id);

  if (!user) {
    return res.status(401).json({
      success: false,
      message: 'Користувача не знайдено'
    });
  }

  req.user = user;
  next();
 } catch (error) {
  console.error('Auth middleware error:', error);
  return res.status(401).json({
    success: false,
    message: 'Недійсний або прострочений токен'
  });
 }
};

export const isStudent = (req, res, next) => {
 if (req.user.role !== 'student') {
  return res.status(403).json({
    success: false,
    message: 'Доступ лише для студентів'
  });
 }
 next();
};
```

Рольова авторизація та бізнес-логіка. Для кожного типу користувача реалізовано окремі рівні доступу. Наприклад, викладачі мають доступ до

методів створення оцінок, тоді як студенти - лише до методу перегляду. Це реалізовано через функції isTeacher, isStudent та isCurator. Контролери, такі як teacherController.js, містять складну логіку перевірок: наприклад, при виставленні оцінки сервер перевіряє, чи дійсно даний викладач закріплений за цим предметом, і чи дійсно обраний студент записаний на цей курс (рис. 3.2).

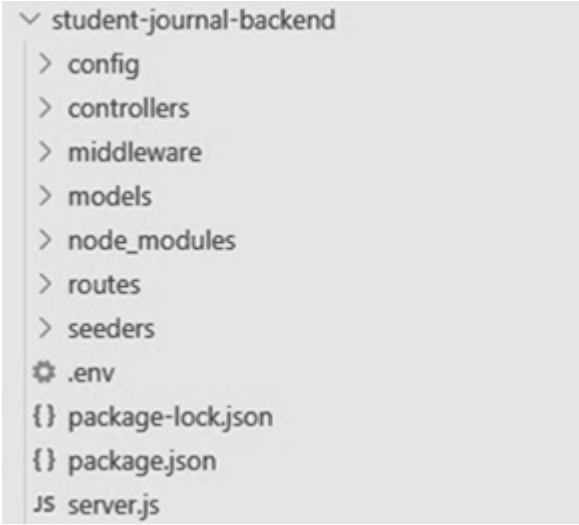


Рисунок 3.2 - Модульна структура серверної частини проєкту

Обробка помилок та валідація. На сервері впроваджено централізовану систему обробки помилок. Всі операції з базою даних обгорнуті у блоки try-catch [8]. У разі виникнення помилки (наприклад, невалідні дані або відсутність зв'язку з БД), сервер повертає структуровану відповідь із відповідним HTTP-статусом (400, 403, 500) та повідомленням про помилку у форматі JSON, що дозволяє фронтенду коректно відобразити попередження користувачу.

Така детальна реалізація серверної логіки дозволила створити надійний та захищений фундамент для вебзастосунку «Електронний журнал успішності», забезпечивши цілісність даних та швидку обробку запитів від багатьох користувачів одночасно.

3.2 Розробка клієнтського інтерфейсу та функціональних модулів

Клієнтська частина вебзастосунку розроблена як Single Page Application, що є сучасним стандартом побудови динамічних вебресурсів [4]. Використання бібліотеки React дозволило реалізувати компонентний підхід, де кожен елемент інтерфейсу (таблиця, кнопка, модальне вікно) є незалежною одиницею коду [13]. Це не лише спрощує розробку, але й забезпечує високу швидкість роботи, оскільки браузер завантажує основну логіку лише один раз, а далі лише оновлює дані через асинхронні запити.

Ключові аспекти та технічні рішення при реалізації фронтенд-частини: Управління глобальним станом через Zustand. Для великого застосунку з різними ролями користувачів критично важливо мати централізоване сховище даних. Було реалізовано файл store/authStore.js, який керує станом авторизації. Вибір Zustand замість Redux обґрунтований його легковажністю та відсутністю зайвого шаблонного коду [7]. Сховище зберігає об’єкт користувача та JWT-токен, що дозволяє будь-якому компоненту системи миттєво дізнатися роль поточного користувача та змінити свій вигляд. Наприклад, бічна панель навігації автоматично приховує розділ "Управління групами", якщо в системі авторизований звичайний студент:

```
import { create } from 'zustand';
import { persist } from 'zustand/middleware';

const useAuthStore = create(
  persist(
    (set) => ({
      user: null,
      isAuthenticated: false,

      login: (userData) => set({
        user: userData,
        isAuthenticated: true
      }),

      logout: () => set({
```

```
      user: null,
      isAuthenticated: false
    }),

    updateUser: (data) => set((state) => ({
      user: { ...state.user, ...data }
    })),
  }),
  {
    name: 'auth-storage', // ім'я ключа в localStorage
  }
 )
);
export default useAuthStore;
```

Взаємодія з сервером через Axios. Для здійснення HTTP-запитів до REST API використано бібліотеку Axios. У файлі services/api.js було створено екземпляр клієнта з базовими налаштуваннями URL. Найважливішою частиною стали перехоплювачі. Перехоплювач запиту автоматично зчитує токен із локального сховища та додає його до заголовка Authorization: Bearer <token> [13]. Перехоплювач відповіді обробляє помилки: якщо сервер повертає статус 401, застосунок автоматично скидає стан авторизації та перенаправляє користувача на сторінку входу.

Адаптивна верстка з Tailwind CSS. Дизайн системи побудований за принципом Mobile First. Використання Tailwind CSS дозволило відмовитися від написання тисяч рядків кастомного CSS-коду на користь службових класів безпосередньо в розмітці. Це забезпечило ідеальну адаптивність: на великих моніторах викладач бачить розгорнуту таблицю успішності групи, а на смартфонах ця таблиця трансформується у зручні картки студентів, що дозволяє працювати з журналом навіть у дорозі або під час аудиторних занять (рис. 3.3).

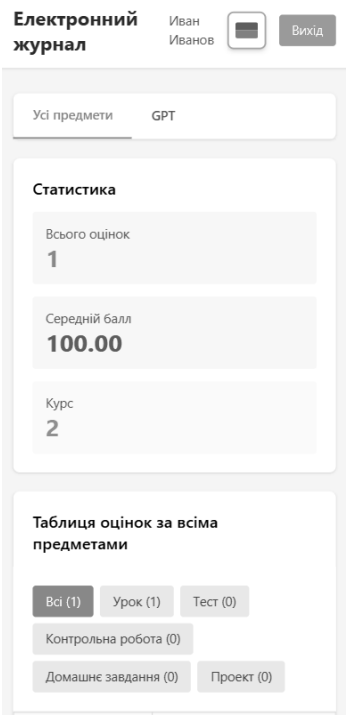


Рисунок 3.3 – Демонстрація адаптивності інтерфейсу за допомогою Tailwind CSS

Реалізація мультимовності (i18next). Оскільки застосунок орієнтований на широке коло користувачів, впроваджено підтримку декількох мов. Конфігурація i18n.js дозволяє динамічно змінювати мову без перезавантаження сторінки. Усі текстові константи винесені в JSON-файли у папці locales. Це значно полегшує підтримку проєкту: для додавання нової мови достатньо лише додати новий файл перекладу без втручання в логіку компонентів [8].

Деталізація функціональних модулів:

– Модуль студента. Основний акцент зроблено на компоненті GradesTable.jsx. Він отримує масив оцінок від API та виконує їх рендеринг. Реалізовано функцію клієнтської фільтрації: студент може миттєво відсіяти оцінки за певним предметом або за типом роботи (наприклад, переглянути лише результати контрольних робіт), не роблячи нових запитів до сервера.

– Модуль викладача. Тут центральним елементом є складний компонент управління журналом. Реалізовано систему модальних вікон (наприклад, AddGradeModal.jsx), які використовують бібліотеку react-hook-

form для валідації введених балів [5]. Викладач може клікнути на клітинку з оцінкою, щоб відкрити вікно редагування або переглянути коментар до роботи (рис. 3.4).

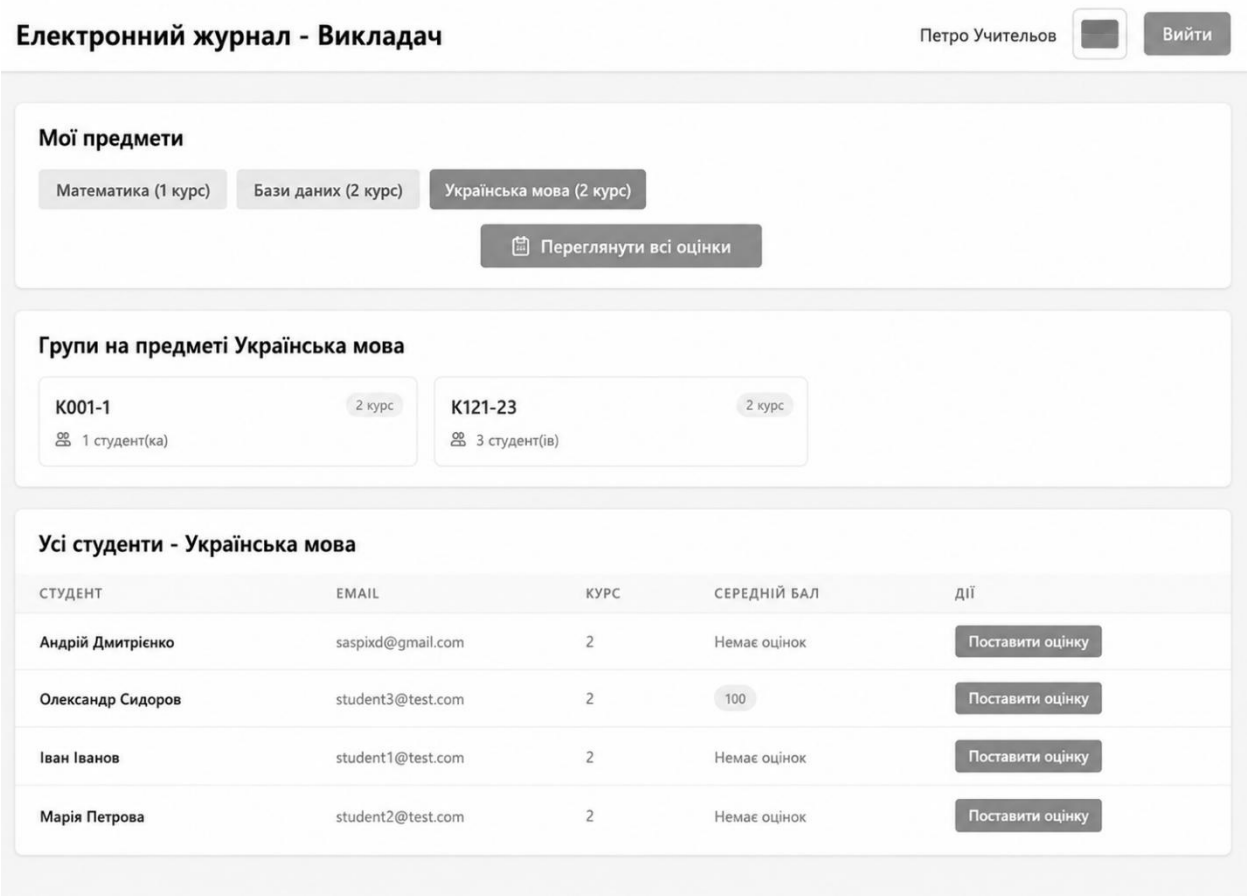


Рисунок 3.4 – Інтерфейс кабінету викладача для виставлення оцінок

3.3 Результати розробки та тестування застосунку

Завершальним етапом реалізації стала комплексна перевірка працездатності всіх систем. Тестування проводилося ітераційно: спочатку перевірялися окремі функції, потім взаємодія між фронтендом і бекендом, і наприкінці - поведінка всієї системи в цілому. Це дозволило отримати стабільний продукт, готовий до експлуатації.

Детальний опис процесів тестування:

1. Unit-тестування та валідація даних. На рівні серверної частини було перевірено всі моделі Sequelize. Тести підтвердили, що валідатори в моделі User коректно відхиляють реєстрацію з дубльованим email, а модель Grade не дозволяє зберігати значення поза діапазоном 0-100. Це гарантує цілісність бази даних навіть у разі спроби прямого маніпулювання запитами.

– Тестування API через Postman. Це був критичний етап перевірки безпеки. Було створено колекцію запитів для кожної ролі.

– Сценарій 1 (Позитивний): викладач з валідним токеном успішно створює оцінку. Результат: статус 201 Created.

– Сценарій 2 (Негативний): студент намагається відправити запит на видалення предмета. Результат: статус 403 Forbidden, оскільки middleware isCurator заблокувала доступ.

– Сценарій 3 (Авторизація): запит без токена. Результат: статус 401 Unauthorized.

2. UI/UX та інтеграційне тестування. На боці клієнта перевірялася логіка відображення компонентів. Було протестовано швидкість оновлення інтерфейсу: після виставлення оцінки викладачем, дані у базі оновлюються за 150-200 мс, а фронтенд миттєво відображає новий рядок у таблиці без миготіння сторінки. Окремо тестувалася робота "Захищених маршрутів": при спробі неавторизованого користувача ввести в адресному рядку /teacher-dashboard, система миттєво перенаправляє його на сторінку логіну.

3. Стійкість та безпека. Проведено перевірку на стійкість до некоректного введення. Форми реєстрації та входу обладнані регулярними виразами для перевірки формату пошти та складності пароля. Паролі зберігаються у базі виключно у вигляді хешів (bcrypt), що гарантує їх захист навіть у разі фізичного витоку файлів бази даних (рис. 3.5).

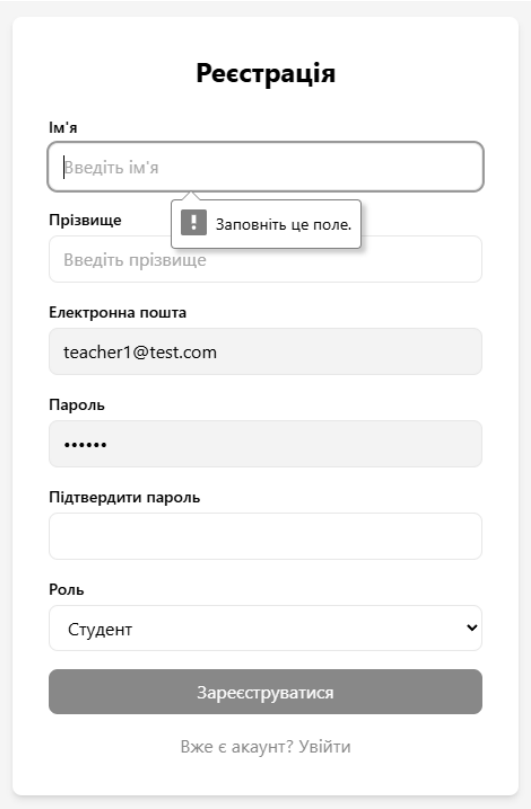


Рисунок 3.5 - Робота системи валідації введених даних на фронтенді

Додаток продемонстрував стабільну роботу при симуляції одночасної роботи декількох десятків користувачів. Миттєва синхронізація станів через Zustand та Axios дозволяє викладачам та студентам бачити актуальну інформацію без затримок.

3.4 Висновки щодо практичної реалізації системи

В даному розділі було детально описано процес практичної реалізації вебзастосунку «Електронний журнал успішності». Програмна розробка охопила створення потужного бекенду на базі Node.js та Express, а також розробку сучасного, адаптивного клієнтського інтерфейсу на React [2, 6].

Використання сучасних інструментів, таких як Sequelize ORM для роботи з базою даних PostgreSQL та JWT для автентифікації, дозволило створити надійну систему з високим рівнем безпеки [11]. Розподіл логіки на

незалежні контролери та маршрути забезпечив легкість підтримки та масштабування коду. Реалізація фронтенд-частини з використанням Zustand та Tailwind CSS дозволила досягти високої продуктивності та зручності користування на будь-яких типах пристроїв.

Результати проведеного тестування (unit, API, UI/UX) підтвердили повну відповідність розробленого продукту технічному завданню. Система коректно обробляє ролі користувачів, захищає конфіденційні дані та забезпечує стабільну взаємодію між викладачем і студентом. Створена архітектура є відкритою для майбутніх розширень, таких як інтеграція з іншими освітніми платформами або додавання модулів автоматичної генерації PDF-звітів, що робить цей проєкт актуальним та перспективним для впровадження у навчальний процес.

ВИСНОВКИ

Результатом даної кваліфікаційної роботи стало створення вебзастосунку «Електронний журнал успішності», призначеного для автоматизації обліку оцінок студентів у навчальному закладі. У процесі роботи були ретельно проаналізовані потреби цільової аудиторії, існуючі аналоги та сучасні підходи до розробки клієнт-серверних систем, що дозволило визначити оптимальні технології та інструменти для реалізації проєкту.

Завдяки продуманому проєктуванню, правильному вибору технологій та грамотній реалізації, вдалося створити повноцінний вебзастосунок, який відповідає всім поставленим вимогам та задовольняє потреби користувачів у зручному інструменті для виставлення, зберігання та моніторингу балів. Застосунок «Електронний журнал успішності» має інтуїтивно зрозумілий інтерфейс та надає швидкий доступ до результатів успішності в реальному часі.

Додаток працює стабільно, пройшов тестування серверної частини (API) та клієнтського інтерфейсу в різних сучасних браузерах, і всі виявлені помилки були своєчасно виправлені.

Вебзастосунок «Електронний журнал успішності» має значний потенціал для подальшого розвитку та може бути легко розширений завдяки модульній структурі. Це відкриває нові можливості для вдосконалення функціоналу та адаптації під інші адміністративні потреби навчального закладу. Дана робота не лише демонструє знання теоретичних аспектів розробки вебзастосунків, але й показує вміння їх успішно реалізувати на практиці, що сприяє підвищенню професійного рівня та відкриває нові перспективи для подальшого розвитку та кар'єрного зростання в сфері веброзробки.

Отримання практичного досвіду та нових знань у процесі проєктування та розробки клієнт-серверного застосунку «Електронний журнал успішності» стало важливим етапом мого професійного становлення. Цей проєкт дозволив мені оволодіти основними принципами веброзробки та успішно втілити їх на практиці.

Під час роботи над додатком, я здобув безцінний досвід у використанні середовища Node.js, вебфреймворку Express.js, реляційної бази даних PostgreSQL та бібліотеки React для побудови інтерфейсів [2, 6]. Ці знання дозволили створити функціональний застосунок, який відповідає вимогам безпеки (через JWT-автентифікацію) та впроваджується в контекст цифровізації освітнього процесу.

Отриманий досвід у розробці застосунку «Електронний журнал успішності» став важливим етапом у моєму професійному зростанні, дозволяючи відкрити нові горизонти для подальшого вдосконалення навичок та реалізації інноваційних ідей у сфері Full-Stack розробки. Цей проєкт зміцнив мою впевненість у здатності досягати професійних цілей та успішно розвиватися у сфері інформаційних технологій.

Під час розробки застосунку, я зміг глибоко оцінити важливість усіх етапів навчання за спеціальністю 121 «Інженерія програмного забезпечення». Це навчання сприяє підготовці висококваліфікованих фахівців, які можуть ефективно використовувати сучасні технології та методи розробки інформаційних систем. Весь процес навчання відкрив широкі перспективи для моєї майбутньої кар'єри в сфері програмування.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бенкс Алекс, Порчелло Євгеній. React: Патерни розробки вебзастосунків.: Діалектика, 2018. 336 с. URL: https://github.com/Kvadeck/React-i-Redux-Funkcional-naya-veb-rrazrabotka-Aleks-Benks-Eva-Porsello/blob/master/Benx_A__Porsello_E_React_I_Redux_Funktsionalnaya_Veb-Razrabotka_2018.pdf.

2. Хавербеке Марейн. Виразний JavaScript.: 3-тє видання : Символ-Плюс, 2020. 400 с. URL: https://dokumen.pub/qdownload/javascript-3nbsped-978-5-4461-1226-5.html.

3. Фленаган Девід. JavaScript: Вичерпне керівництво.: 7-ме видання: Діалектика, 2021. 720 с. URL: https://scribd.com/document/503342295/Javascript-Pol-Nr-Uk-7.

4. Чернишина Г.Г., Юдкова К.В. КЛАСИФІКАЦІЯ ІНФОРМАЦІЙНИХ СИСТЕМ, 2015. 7 с. URL: https://ippi.org.ua/sites/default/files/yudkova.pdf.

5. Douglas Crockford. JavaScript : The Good Parts. O'Reilly Media, 2008. 170 p. URL: https://andersonguelphjs.github.io/OReilly_JavaScript_The_Good_Parts_May_2008.pdf.

6. Martin Fowler: Patterns of Enterprise Application Architecture.: Addison-Wesley, 2002. 560 p. URL: https://sar.ac.id/stmik_ebook/prog_file_file/EFCofwzsj0.pdf.

7. Robin Wieruch: The Road to React. 2nd edition: CreateSpace Independent Publishing Platform, 2018. 200p. URL: https://wbpms.in/Tender/Download/TENDER/Robin%20Wieruch%20-%20The%20Road%20to%20React.pdf.

8. Документація Sequelize ORM. URL: https://sequelize.org/ (дата звернення : 25.01.2026).

9. Велика спільнота розробників. URL: https://stackoverflow.com (дата звернення : 05.04.2026).

10. Офіційна документація Node.js, посібник до серверної платформи. URL : https://nodejs.org/uk/docs (дата звернення : 12.01.2026).

11. Офіційна документація PostgreSQL. URL: https://www.postgresql.org/docs/ (дата звернення : 18.01.2026).

12. Офіційна документація React. URL: https://react.dev/learn (дата звернення : 14.01.2026).

13. Стандарт JWT (JSON Web Tokens). URL: https://jwt.io/introduction (дата звернення : 20.01.2026).

Декларація

академічної доброчесності

здобувача освіти ВСП «Економіко-правничий фаховий коледж ЗНУ»

Я, Дмитрієнко Андрій Ігорович, здобувач освіти 4 курсу, спеціальності 121 Інженерія програмного забезпечення, освітньо-професійної програми Розробка програмного забезпечення, групи К121-22, адреса електронної пошти dmytriienko.andrii@gmail.com

- підтверджую, що написана мною кваліфікаційна робота на тему «Розробка вебсервісу для ведення електронного журналу успішності студентів» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений;

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 26.05.2026 Підпис  Андрій ДМИТРІЄНКО

Дата _____ Підпис _____ Дмитро ДРАЄВСЬКИЙ