Одержано редакцією 05.09.2024 р.
Прийнято до публікації 30.10.2024 р.

УДК 004.415.2:004.5

DOI 10.31651/2076-5886-2024-1-79-90

PACS 07.05.Tp, 89.20.Ff

ВОЙЦІХОВСЬКА Лєна Іванівна
студентка спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького

ДЗЮБА Вікторія Анатоліївна
кандидат технічних наук, старший викладач кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: viktoriya.dzyuba15@vu.cdu.edu.ua
ORCID 0000-0003-1655-0333

РОЗРОБКА ІНФОРМАЦІЙНО-ТЕСТУВАЛЬНОГО ВЕБ-ДОДАТКУ ДЛЯ НАВЧАЛЬНОГО ЗАКЛАДУ

У статті розглядається процес проєктування, розробки та розгортання сучасного веб-застосунку для навчальної платформи. Детально аналізується архітектура системи, включаючи бази даних PostgreSQL, використання ORM Prisma, а також побудова серверної частини на основі Node.js, TypeScript та фреймворку NestJS. Описано принципи організації REST API, механізми аутентифікації та авторизації користувачів із використанням JWT, а також підходи до безпечного зберігання даних і управління доступом.

У роботі також висвітлено розробку клієнтської частини застосунку із використанням React, Vite, Redux та бібліотеки MUI, що дозволило створити інтерактивний і масштабований інтерфейс користувача. Окрему увагу приділено процесу проєктування інтерфейсу у Figma та практичній реалізації системи розгортання за допомогою Docker, MinIO для зберігання файлів і MailHog для тестування електронної пошти. Результатом роботи є комплексне програмне рішення, яке поєднує сучасні технології веб-розробки та забезпечує стабільність, безпеку й зручність використання

.***Ключові слова:*** *веб-застосунок, інформаційна система, клієнт-серверна архітектура, база даних, PostgreSQL, Prisma, Node.js, TypeScript, NestJS, REST API, JWT, аутентифікація, авторизація, React, SPA, Redux, Vite, MUI, Figma, Docker, MinIO, MailHog, розгортання, веб-розробка.*

Вступ

Сучасний ритм життя вносить свої корективи в наші плани та буденність, змушуючи змінювати тимчасово чи на зовсім своє місце проживання, перебувати по декілька годин в укритті, правильно виставляти пріоритети та економити електроенергію. Увесь світ, зокрема Україна, проходить зараз етап цифровізації, щоб будь-яка людина змогла зробити базові речі, не виходячи з дому. Це покупки в магазині, оформлення документів, запис до певного спеціаліста і тому подібне. Таким чином, у деякій мірі, можна забезпечити вдале поєднання безпеки та комфорту людського життя.

Цифровізації зазнає і освіта, адже у такому потоці хаосу потрібно продовжувати навчатись та розвиватись, незалежно від розташування та речей поруч. Для цього

потрібно, щоб все необхідне було в одному місці. Звичайно, технології зараз не замінять вчителя, однак додаткові матеріали, домашні завдання, оцінки - це все, що розміщено на одному сайті з доступом для батьків, учнів та вчителів, спростить життя та додасть мобільності освітньому процесу. Такий підхід буде економією часу та ресурсів у вільному доступі для будь-якого навчального закладу чи організації.

Мета статті – створення інформаційно-тестувального веб-додатку для навчального закладу з базовим та найбільш необхідним функціоналом, який на основі досвіду користувачів, підвищенні власної кваліфікації та розширення команди можна буде вдосконалювати надалі.

Виклад основного матеріалу

1. Аналіз предметної області та постановка задачі

Сучасний етап цифровізації характеризується стрімким проникненням інформаційних технологій у всі сфери суспільного життя, зокрема в освіту. Поширення систем дистанційного та змішаного навчання зумовило появу великої кількості освітніх платформ, що забезпечують організацію навчального процесу в онлайн-середовищі. У межах даного дослідження розглянуто три найбільш поширені рішення: Google Classroom, «Єдина Школа» та освітню платформу Optima.

Google Classroom є безкоштовним сервісом компанії Google, призначеним для організації навчального процесу, створення та перевірки завдань, а також комунікації між викладачами та учнями. Основною перевагою системи є глибока інтеграція з екосистемою Google (Docs, Sheets, Slides, Forms), що забезпечує зручність роботи без використання сторонніх програмних засобів. Платформа має інтуїтивний інтерфейс і не потребує складного налаштування, що робить її доступною для широкого кола користувачів.

«Єдина Школа» – це комплексна інформаційна система для закладів освіти, яка включає електронний журнал, щоденник, засоби комунікації між учасниками освітнього процесу та функціонал дистанційного навчання. Система орієнтована на офіційно зареєстровані навчальні заклади та інтегрується з державними освітніми реєстрами. Її основною особливістю є централізоване управління навчальним процесом, проте недоліком є складність впровадження та необхідність адміністративної інтеграції закладу до системи, а також платна модель використання після пробного періоду.

Optima є комерційною платформою дистанційної освіти для учнів 1-11 класів. Вона забезпечує повноцінний навчальний цикл, включаючи доступ до навчальних матеріалів, відеоуроків, інтерактивних завдань і тестів. Перевагою є високий рівень інтерактивності та методичної структурованості навчального контенту. Водночас суттєвим недоліком є висока вартість навчання, що обмежує доступність платформи для широкої аудиторії.

Порівняльний аналіз зазначених систем показує, що всі вони забезпечують базовий набір функцій, необхідних для організації навчального процесу: створення класів, завдань, тестів та автоматизоване оцінювання. Проте кожна система має власні обмеження, пов’язані або з закритістю екосистеми, або з фінансовою доступністю, або з обмеженою гнучкістю впровадження.

Основними вимогами до сучасної освітньої системи є: зручна та передбачувана навігація, адаптивна архітектура інтерфейсу, розподіл ролей користувачів та чітке розмежування прав доступу. Проєктована система передбачає чотири основні ролі користувачів: учень, вчитель, директор та адміністратор. Для кожної ролі визначено відповідний набір дозволів, що регламентує доступ до функціональних модулів платформи. Окремо виділяється категорія незареєстрованих користувачів (гостей), для

яких передбачено обмежений доступ до інформаційних розділів, зокрема блогу, контактної інформації та загальних відомостей про навчальні заклади.

Розробка програмного продукту розглядається як комплексний процес створення, впровадження та супроводу програмного забезпечення. У загальному вигляді життєвий цикл розробки можна поділити на такі етапи:

1. Аналіз предметної області та дослідження ринку
2. Формування вимог до системи
3. Технічне проєктування
4. Безпосередня розробка програмного забезпечення
5. Тестування та оцінка якості

Перший етап є критично важливим, оскільки визначає архітектуру майбутньої системи та впливає на всі наступні стадії. На основі зібраних вимог формується технічне завдання, яке визначає функціональність системи, вимоги до інтерфейсу та бізнес-логіки.

Технічне проєктування включає розробку дизайну інтерфейсу, структури бази даних та логіки взаємодії компонентів. Етап розробки передбачає налаштування середовища, підключення необхідних технологій та реалізацію функціоналу. Завершальними стадіями є тестування, рефакторинг та оптимізація коду. У процесі розробки програмної системи використовуються сучасні інструменти, що забезпечують ефективність командної роботи та якість кінцевого продукту.

Для управління задачами застосовано Trello, що дозволяє структурувати процес розробки та відстежувати виконання завдань. Проєктування інтерфейсу здійснювалося у середовищі Figma, яке є стандартом у сфері UI/UX дизайну та дозволяє створювати інтерактивні макети.

Основним середовищем розробки було обрано Visual Studio Code, яке забезпечує гнучкість, підтримку розширень та зручну роботу з кодом. Контроль версій реалізовано за допомогою Git, що дозволяє ефективно керувати змінами та організовувати командну розробку.

Для роботи з базою даних використано pgAdmin як інструмент адміністрування PostgreSQL, що забезпечує зручний графічний інтерфейс для управління даними. Для документування та тестування API застосовано Swagger, який дозволяє формалізувати взаємодію між клієнтською та серверною частинами системи.

2. Створення моделей, розробка логіки та серверної частини продукту

Еволюція систем керування базами даних (СКБД) відображає загальний розвиток інформаційних технологій – від файлових структур до сучасних реляційних та гібридних моделей. Первинні підходи до зберігання даних ґрунтувалися на використанні файлів, що призводило до тісного зв’язку між програмним кодом і фізичною організацією даних, ускладнюючи масштабування та повторне використання інформації. Подальший розвиток призвів до появи централізованих систем керування файлами, однак їх обмеження (відсутність гнучкості, дублювання даних, складність спільного доступу) зумовили формування концепції баз даних як централізованого сховища взаємопов’язаних даних із керованим доступом. Важливим етапом стало впровадження реляційної моделі, яка забезпечила формалізований підхід до структурування даних і стала основою сучасних СКБД.

Проєктування бази даних у межах даної роботи здійснювалося на основі комбінованого підходу: з урахуванням як бізнес-вимог системи, так і особливостей предметної області. На етапі логічного моделювання було сформовано набір сутностей, що відображають ключові компоненти освітньої платформи: користувачі, ролі, школи,

класи, предмети, завдання, тести, питання, відповіді, файли, запрошення та публікації (рис. 1).

Для моделювання структури бази даних використано онлайн-інструмент SqlDBM, що дозволив візуалізувати сутності та зв'язки між ними. Такий підхід забезпечив цілісність моделі та спростив подальшу реалізацію фізичної структури бази даних.

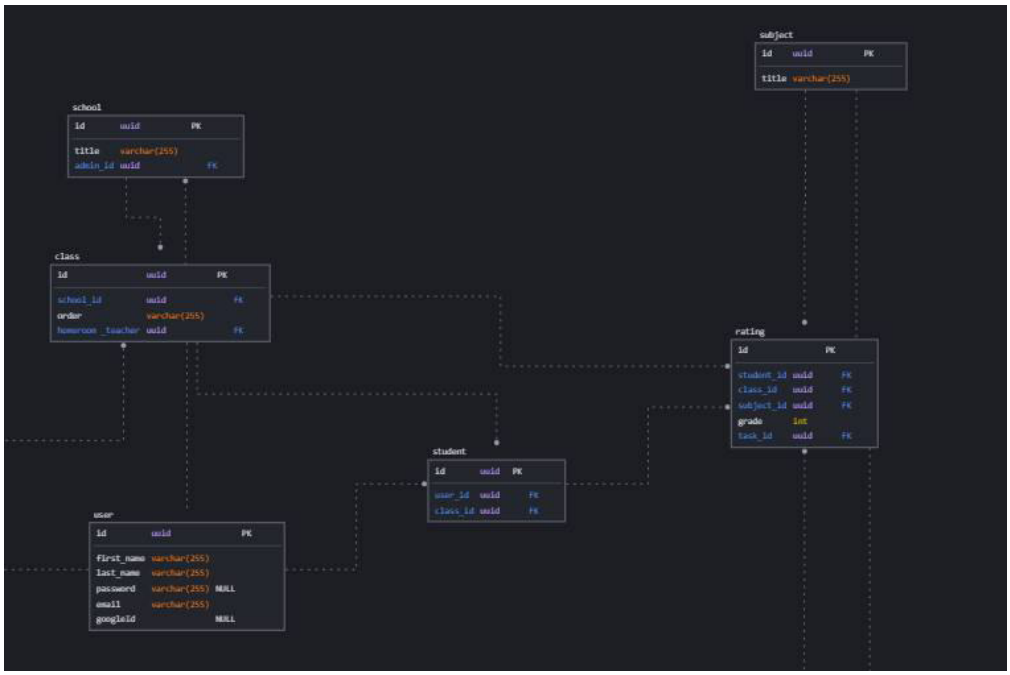

Рис. 1. Проектування моделей бази даних у SqlDBM

З урахуванням характеру системи, яка передбачає значну кількість взаємопов'язаних сутностей і транзакцій, обрано реляційну СКБД PostgreSQL. Вона забезпечує високу надійність, підтримку складних зв'язків, транзакційність та відповідність стандартам SQL.

PostgreSQL є відкритою, кросплатформною системою, що підтримує розширюваність через власні типи даних, функції та модулі (зокрема PostGIS). Її архітектура дозволяє ефективно працювати як з невеликими, так і з масштабними наборами даних, забезпечуючи стабільність і цілісність інформації. Таким чином, вибір PostgreSQL обґрунтовано необхідністю забезпечення складних зв'язків між сутностями та високими вимогами до цілісності даних у навчальній платформі.

Для адміністрування PostgreSQL використано pgAdmin, який надає графічний інтерфейс для створення, редагування та моніторингу об'єктів бази даних (рис. 2). Інструмент дозволяє виконувати SQL-запити, аналізувати продуктивність та керувати користувацькими правами доступу. Використання pgAdmin значно спрощує процес взаємодії з базою даних, особливо на етапі розробки та тестування, забезпечуючи зручність контролю структури та даних.

Для організації взаємодії між серверною частиною та базою даних використано ORM Prisma, який реалізує об'єктно-реляційне відображення даних. Основною перевагою ORM є можливість роботи з даними у вигляді об'єктів, без необхідності прямого написання SQL-запитів. Prisma забезпечує декларативне визначення моделей, автоматичну генерацію запитів, типізований доступ до даних та систему міграцій. Це підвищує надійність коду та зменшує ризик помилок при роботі з базою даних. У межах проєкту Prisma використовується для опису моделей, управління схемою бази даних та генерації типізованого API доступу до даних.

Серверна частина є ключовим компонентом системи, оскільки забезпечує обробку запитів, бізнес-логіку та взаємодію з базою даних. Основна модель роботи базується на принципі клієнт–серверної архітектури, де клієнт надсилає HTTP-запити, а сервер повертає структуровані відповіді.

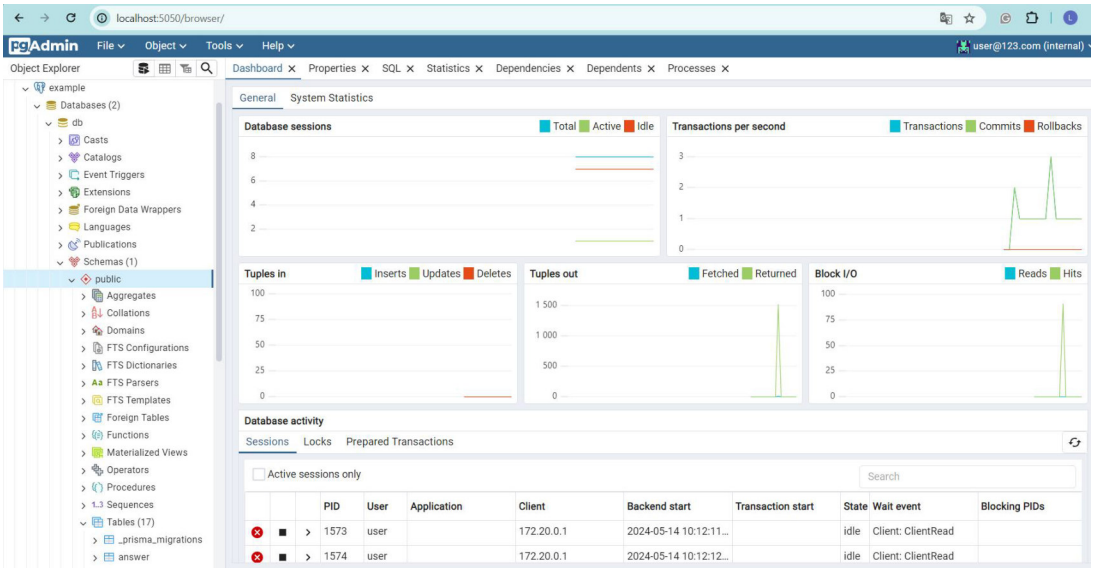

Рис. 2. Інтерфейс середовища PGAdmin

HTTP-протокол передбачає використання основних методів взаємодії з ресурсами: GET, POST, PUT, PATCH, DELETE, кожен з яких відповідає певному типу операцій. Результати обробки запитів визначаються через HTTP-коди стану, які відображають успішність або помилки виконання операцій. Конфігураційні параметри системи (порти, ключі доступу, адреси сервісів) зберігаються у змінних середовища (.env), що підвищує безпеку та гнучкість розгортання.

Для реалізації серверної частини використано Node.js, що дозволяє виконувати JavaScript поза браузером. Його ключовою перевагою є асинхронна неблокуюча модель обробки запитів, що забезпечує високу продуктивність при великій кількості одночасних з'єднань. Node.js широко використовується для побудови REST API, real-time застосунків та масштабованих веб-сервісів, що робить його оптимальним вибором для сучасних серверних рішень.

Для підвищення надійності коду використано TypeScript, який додає статичну типізацію до JavaScript. Це дозволяє зменшити кількість помилок на етапі розробки та забезпечує чітке визначення структур даних. У проєкті реалізовано типізацію моделей та API-запитів, що забезпечує узгодженість між серверною та клієнтською частинами системи.

Основою серверної архітектури виступає NestJS, який забезпечує модульну структуру застосунку. Архітектура NestJS базується на поділі на:

– контролери (обробка HTTP-запитів),
– сервіси (бізнес-логіка),
– модулі (логічне групування компонентів).

Такий підхід забезпечує масштабованість, тестованість та підтримуваність коду. Вхідною точкою системи є файл main.ts, який ініціалізує додаток і підключає всі модулі.

Для тестування API використано Postman, що дозволяє перевіряти роботу серверних маршрутів, параметрів запитів і відповідей. Документування API реалізовано за допомогою Swagger, який автоматично генерує інтерактивну документацію. Це

значно спрощує взаємодію між бекендом і фронтендом та зменшує необхідність ручного аналізу коду.

Безпека користувацьких даних реалізується через механізми аутентифікації та авторизації. Аутентифікація передбачає перевірку облікових даних користувача, тоді як авторизація визначає рівень доступу до ресурсів системи. Основою безпеки виступає модель ролей, відповідно до якої користувач отримує доступ лише до дозволених функцій.

Для реалізації автентифікації використано JWT (JSON Web Token, рис. 3). Токен складається з трьох частин: header, payload і signature. Він використовується для безпечної передачі даних між клієнтом і сервером. JWT зберігається на стороні клієнта у cookies та передається з кожним запитом. Перевірка підпису дозволяє підтвердити цілісність і достовірність даних.

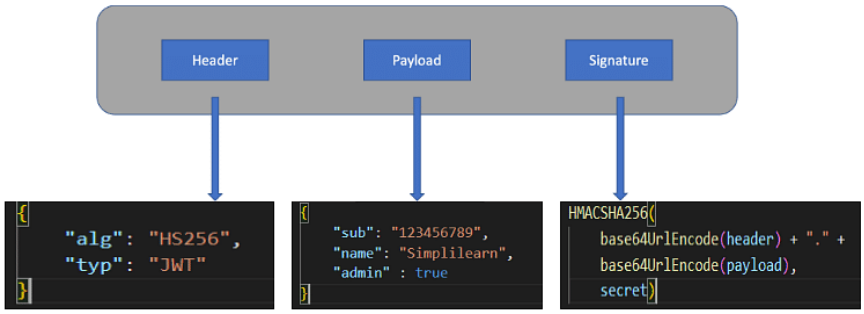

Рис. 3. Структура JSON Web токена.

Додатково для захисту паролів використовується хешування на основі bcrypt, що забезпечує одностороннє шифрування та захист від атак перебором.

Для реалізації автоматичної розсилки електронних повідомлень використано бібліотеку Nodemailer. Вона застосовується для:

– підтвердження реєстрації адміністратора,
– запрошення викладачів до закладу,
– запрошення учнів до класів.

Механізм базується на передачі токенізованих посилань через email, що дозволяє автоматизувати процес підключення користувачів до системи.

3. Дизайн та розробка клієнтської частини проєкту

Процес розробки клієнтської частини веб-застосунку розпочинається з проєктування інтерфейсу користувача. У межах даної роботи використано інструмент Figma, який є сучасним хмарним середовищем для створення UI/UX-дизайну та спільної роботи над макетами.

Figma забезпечує можливість швидкого прототипування інтерфейсів, узгодження дизайну між учасниками проєкту та реалізації компонентного підходу до побудови сторінок. Основними перевагами інструмента є кросплатформеність, підтримка колективної розробки та інтеграція з сучасними процесами UI/UX-дизайну. У межах проєкту в Figma було розроблено структуру інтерфейсу, логотип, іконографіку та головні сторінки системи, що дозволило сформувати цілісну візуальну концепцію майбутнього веб-застосунку (рис. 4).

Для побудови сучасного фронтенд-застосунку використано інструмент збірки Vite, який є одним із найсучасніших рішень для швидкої розробки веб-проєктів. На відміну від традиційних бандлерів (Webpack, Parcel), Vite використовує нативну

підтримку ES-модулів у браузері, що забезпечує значне прискорення запуску проєкту та оновлення коду. Його архітектура включає dev-сервер із підтримкою швидкого Hot Module Replacement (HMR) та оптимізовану production-збірку на основі Rollup. Застосування Vite дозволяє суттєво скоротити час розробки та підвищити продуктивність роботи над інтерфейсом, особливо у великих SPA-застосунках.

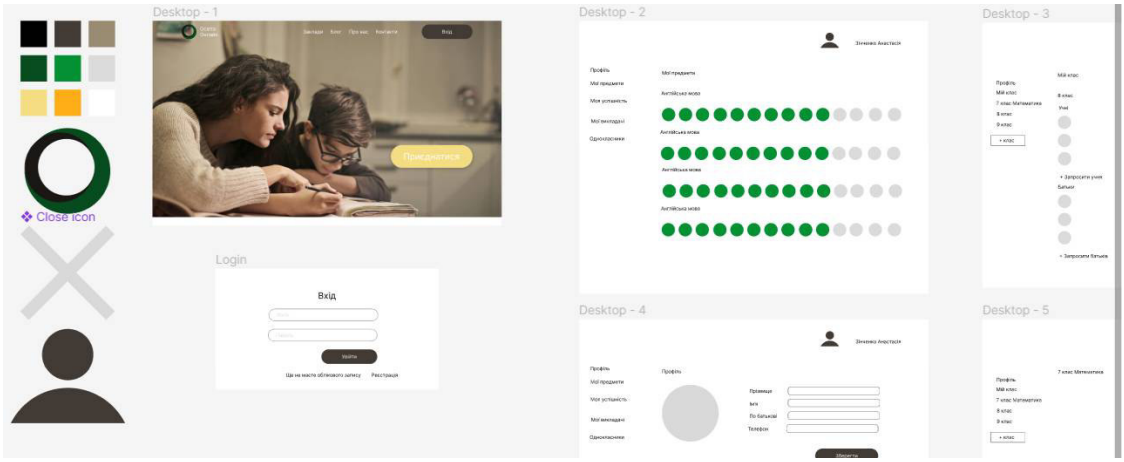

Рис. 4. Створення дизайну в Figma

Клієнтська частина реалізована як SPA (Single Page Application), що передбачає завантаження одного HTML-документа з подальшим динамічним оновленням контенту без повного перезавантаження сторінки. Такий підхід забезпечує:

– високу швидкість взаємодії з користувачем;
– зменшення навантаження на сервер;
– покращений користувацький досвід.

Для реалізації SPA використано бібліотеку ReactJS, яка базується на компонентній архітектурі. Основними перевагами React є:

– декларативний підхід до побудови UI;
– компонентна структура інтерфейсу;
– використання віртуального DOM для ефективного оновлення сторінок.

React дозволяє будувати масштабовані інтерфейси з повторно використовуваних компонентів, що є критично важливим для складних освітніх платформ.

Для опису інтерфейсу використовується JSX (JavaScript XML) – синтаксичне розширення JavaScript, що дозволяє поєднувати логіку та розмітку в межах одного файлу. JSX спрощує створення динамічних інтерфейсів, дозволяючи вбудовувати JavaScript-вирази безпосередньо в розмітку.

Для побудови інтерфейсних компонентів застосовано бібліотеку MUI (Material UI), яка реалізує принципи Material Design. MUI надає готові UI-компоненти, що дозволяє:

– прискорити розробку інтерфейсу;
– забезпечити уніфікований стиль системи;
– реалізувати адаптивність під різні пристрої.

Поєднання JSX і MUI забезпечує гнучкість у розробці складних інтерфейсів та дозволяє швидко формувати сучасний UI без значних витрат на кастомну стилізацію. У складних SPA-застосунках ключовою задачею є керування станом інтерфейсу. Для цього використано бібліотеку Redux, яка забезпечує централізоване управління станом застосунку.

Основні принципи Redux:

– єдине джерело стану (store);
– зміна стану лише через actions;
– використання pure reducers для обробки змін.

Такий підхід забезпечує передбачуваність поведінки застосунку та спрощує налагодження і масштабування. Інтеграція з React здійснюється через react-redux, що дозволяє компонентам отримувати доступ до стану через хуки useSelector і змінювати його через useDispatch. Використання Redux особливо ефективне у системах з великою кількістю взаємопов'язаних компонентів, де необхідна синхронізація даних між різними частинами інтерфейсу.

4. Збірка проєкту

Сучасна розробка веб-застосунків передбачає використання контейнеризації як базового підходу до розгортання програмного забезпечення. У межах даної роботи застосовано технологію Docker, яка дозволяє ізолювати сервіси в окремі контейнери разом із усіма залежностями. Контейнеризація забезпечує відтворюваність середовища, що є критично важливим для командної розробки та перенесення системи між різними середовищами (локальним, тестовим і продакшн). На відміну від віртуальних машин, Docker-контейнери є легшими, швидше запускаються та ефективніше використовують ресурси системи.

У межах проєкту було реалізовано багатоконтейнерну архітектуру, яка включає окремі сервіси для клієнтської частини, серверної логіки, бази даних та допоміжних сервісів. Такий підхід забезпечує модульність системи, спрощує масштабування та підвищує стабільність роботи застосунку.

Функціональність освітньої платформи передбачає роботу з різними типами файлів, включаючи аватари користувачів, вкладення до завдань, навчальні матеріали та медіаконтент. Для реалізації об'єктного сховища даних використано систему MinIO.

MinIO є високопродуктивним об'єктним сховищем, сумісним з API Amazon S3, що забезпечує просту інтеграцію та масштабованість. Його основними перевагами є:
– висока швидкість роботи з файлами;
– сумісність із S3-екосистемою;
– простота розгортання у контейнеризованому середовищі;
– підтримка масштабування та розподіленого зберігання.

У межах проєкту MinIO використовується як централізоване сховище для всіх користувацьких файлів. Взаємодія з ним реалізована через серверний сервіс, який забезпечує завантаження, отримання та видалення об'єктів. Для реалізації функціоналу електронних повідомлень у системі використовується інструмент MailHog, який призначений для тестування SMTP-відправлень у локальному середовищі. MailHog перехоплює всі вихідні електронні листи, не відправляючи їх реальним користувачам, що дозволяє безпечно тестувати функціонал розсилок. Основні можливості інструменту включають:
– запуск локального SMTP-сервера для перехоплення листів;
– веб-інтерфейс для перегляду та аналізу повідомлень;
– зберігання листів у пам'яті для зручного тестування;
– підтримку REST API для автоматизації перевірок;
– сумісність із сучасними мовами програмування та фреймворками.

У межах проєкту MailHog використовується для перевірки сценаріїв відправки листів під час реєстрації користувачів та запрошень до системи, що дозволяє уникнути помилкової відправки реальних повідомлень під час розробки.

Висновки

У роботі було створено інформаційно-тестувальну платформу для навчального закладу. Проведено дослідження та аналіз додатків, технологій, що є актуальними на даний момент у розробці. Розроблено серверну частину за допомогою популярного та потужного фреймворку NestJS та інших бібліотек. У проєкті реалізовано авторизацію та аутентифікацію користувача, встановлена чисельна кількість зв'язків у базі даних, розроблено функціонал обробки відповідей і оцінювання студентів, задокументовано інформацію про запити. Було практично досліджено ORM Prisma для роботи з даними у базі PostgreSQL.

Розроблено клієнтську частину платформи за допомогою бібліотеки React. Реалізовано інтерфейс користувача, зручну навігацію по сайту, реєстрацію користувачів, закладів, класів та предметів. Створено форму для тестування та публікації завдань.

Основні результати проведеної роботи полягають у наступному:

1. Досліджено вже існуючі навчальні проєкти.
2. Розглянуто середовища та засоби створення проєкту.
3. Реалізовано сервер.
4. Розроблено клієнтську сторону платформи.
5. Запропоновано систему інформування та тестування для учнів.
6. Досліджено бібліотеку готових компонентів та їхню стилізацію .
7. Здійснено тестування додатку.

Список використаної літератури:

1. Google Classroom [Електронний ресурс] : вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Google_Classroom. – Назва з екрану.
2. Єдина Школа [Електронний ресурс] : Єдина Школа. – Режим доступу: https://eschool-ua.com/#/home. – Назва з екрану.
3. OptimaSchool [Електронний ресурс] : OptimaSchool. – Режим доступу: https://optima.school/vstup/oplata. – Назва з екрану.
4. Greenfield J., Short K., Cook S., Kent S., Crupi J. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. — Indianapolis: Wiley Publishing, 2004. – 592 p.
5. 60 років базам даних / В.А. Резніченко // Проблеми програмування. – 2021. – № 3. – С. 40-71. – Бібліогр.: 211 назв. – укр.
6. Проектування баз даних [Електронний ресурс] : pidru4niki. – Режим доступу: https://pidru4niki.com/11570718/bankivska_sprava/proektuvannya_baz_danih#616. – Назва з екрану.
7. PostgreSQL [Електронний ресурс] : PostgreSQL. – Режим доступу: https://www.postgresql.org. – Назва з екрану.
8. pgAdmin [Електронний ресурс] : pgAdmin. – Режим доступу: https://www.pgadmin.org. – Назва з екрану.
9. Prisma [Електронний ресурс] : PostgreSQL. – Режим доступу: https://www.prisma.io/docs. – Назва з екрану.
10. Node.js IN ACTION: example-driven tutorial / [Mike Cantelon and others]. – [2-nd edition]. – Manning, 2017. – 36 p.
11. Why does Typescript exist [Електронний ресурс] : typescriptlang.org. – Режим доступу: https://www.typescriptlang.org/why-create-typescript. – Назва з екрану.
12. Про розробку додатків на Nest JS [Електронний ресурс] : foxminded.ua. – Режим доступу: https://foxminded.ua/nest-js. – Назва з екрану.
13. Swagger vs Postman | Top 10 Differences You Should Know [Електронний ресурс] : testsigma.com. – Режим доступу: https://testsigma.com/blog/swagger-vs-postman. – Назва з екрану.
14. JSON Web Token (JWT) [Електронний ресурс] : datatracker.ietf.org. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519. – Назва з екрану.
15. Про токени, JSON Web Tokens (JWT), аутентифікацію и авторизацію. Token-Based Authentication [Електронний ресурс] : https://gist.github.com. – Режим доступу: https://gist.github.com/zmts/802dc9c3510d79fd40f9dc38a12bccfc. – Назва з екрану.
16. What is a JWT? Understanding JSON Web Tokens [Електронний ресурс] : supertokens.com. – Режим доступу: https://supertokens.com/blog/what-is-jwt. – Назва з екрану.

17. Node JS Send an Email [Електронний ресурс] : w3schools.com. – Режим доступу: https://www.w3schools.com/nodejs/nodejs_email.asp. – Назва з екрану.
18. Figma [Електронний ресурс] : вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Figma. – Назва з екрану.
19. Why Vite [Електронний ресурс] : vitejs.dev. – Режим доступу: https://vitejs.dev/guide/why.html. – Назва з екрану.
20. A beginner's guide to create SPA with React JS [Електронний ресурс] : dev.to. – Режим доступу: https://dev.to/hiteshtech/a-beginners-guide-to-create-spa-with-react-js-491c. – Назва з екрану.
21. Material UI - Overview [Електронний ресурс] : mui.com. – Режим доступу: https://mui.com/material-ui/getting-started. – Назва з екрану.
22. Getting Started with Redux [Електронний ресурс] : redux.js.org. – Режим доступу: https://redux.js.org/introduction/getting-started. – Назва з екрану.
23. What is Docker? [Електронний ресурс] : https://aws.amazon.com. – Режим доступу: https://aws.amazon.com/docker/?nc1=h_ls. – Назва з екрану.
24. Introduction to MinIO [Електронний ресурс] : baeldung.com. – Режим доступу: https://www.baeldung.com/minio. – Назва з екрану.
25. MailHog Tutorial [Електронний ресурс] : mailtrap.io. – Режим доступу: https://mailtrap.io/blog/mailhog-explained. – Назва з екрану.

References:

1. Google Classroom [Electronic resource]: Wikipedia. – Access mode: https://uk.wikipedia.org/wiki/Google_Classroom
2. "Unified School" [Electronic resource]: Unified School. – Access mode: https://eschool-ua.com/#/home
3. OptimaSchool [Electronic resource]: OptimaSchool. – Access mode: https://optima.school/vstup/oplata
4. Greenfield J., Short K., Cook S., Kent S., Crupi J. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. — Indianapolis: Wiley Publishing, 2004. – 592 p.
5. 60 years of databases / V.A. Reznichenko // Problems in Programming. – 2021. – No. 3. – P. 40–71. – Bibliography: 211 titles. – Ukrainian.
6. Database design [Electronic resource]: pidru4niki. – Access mode: https://pidru4niki.com/11570718/bankivska_sprava/proektuvannya_baz_danih#616
7. PostgreSQL [Electronic resource]: PostgreSQL. – Access mode: https://www.postgresql.org
8. pgAdmin [Electronic resource]: pgAdmin. – Access mode: https://www.pgadmin.org
9. Prisma [Electronic resource]: Prisma documentation. – Access mode: https://www.prisma.io/docs
10. Node.js in Action: Example-Driven Tutorial / Mike Cantelon et al. – 2nd ed. – Manning, 2017. – 36 p.
11. Why does TypeScript exist [Electronic resource]: typescriptlang.org. – Access mode: https://www.typescriptlang.org/why-create-typescript
12. About NestJS development [Electronic resource]: foxminded.ua. – Access mode: https://foxminded.ua/nest-js
13. Swagger vs Postman | Top 10 Differences You Should Know [Electronic resource]: testsigma.com. – Access mode: https://testsigma.com/blog/swagger-vs-postman
14. JSON Web Token (JWT) [Electronic resource]: datatracker.ietf.org. – Access mode: https://datatracker.ietf.org/doc/html/rfc7519
15. About tokens, JSON Web Tokens (JWT), authentication and authorization. Token-Based Authentication [Electronic resource]: gist.github.com. – Access mode: https://gist.github.com/zmts/802dc9c3510d79fd40f9dc38a12bccfc
16. What is a JWT? Understanding JSON Web Tokens [Electronic resource]: supertokens.com. – Access mode: https://supertokens.com/blog/what-is-jwt
17. Node JS Send an Email [Electronic resource]: w3schools.com. – Access mode: https://www.w3schools.com/nodejs/nodejs_email.asp
18. Figma [Electronic resource]: Wikipedia. – Access mode: https://uk.wikipedia.org/wiki/Figma
19. Why Vite [Electronic resource]: vitejs.dev. – Access mode: https://vitejs.dev/guide/why.html
20. A beginner's guide to create SPA with React JS [Electronic resource]: dev.to. – Access mode: https://dev.to/hiteshtech/a-beginners-guide-to-create-spa-with-react-js-491c
21. Material UI – Overview [Electronic resource]: mui.com. – Access mode: https://mui.com/material-ui/getting-started
22. Getting Started with Redux [Electronic resource]: redux.js.org. – Access mode: https://redux.js.org/introduction/getting-started
23. What is Docker? [Electronic resource]: aws.amazon.com. – Access mode: https://aws.amazon.com/docker/?nc1=h_ls

24. Introduction to MinIO [Electronic resource]: baeldung.com. – Access mode: https://www.baeldung.com/minio
25. MailHog Tutorial [Electronic resource]: mailtrap.io. – Access mode: https://mailtrap.io/blog/mailhog-explained

VOYTSIKHOVSKA Lena,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DZYUBA Viktoria,
Candidate of Technical Sciences, Senior Lecturer in the Department of Applied Mathematics and Computer Science at Bohdan Khmelnytsky National University of Cherkasy, Ukraine

DEVELOPMENT OF AN EDUCATIONAL TESTING WEB APPLICATION FOR AN EDUCATIONAL INSTITUTION

Summary. Introduction. *The fast pace of modern life is forcing us to adjust our plans and daily routines, compelling us to temporarily or permanently relocate, spend several hours in shelters, set priorities wisely, and conserve electricity. The whole world, including Ukraine, is currently undergoing a phase of digitalization so that anyone can handle basic tasks without leaving home. This includes shopping, filing paperwork, scheduling appointments with specialists, and similar activities. In this way, to some extent, we can ensure a successful balance between safety and comfort in people's lives.*

Education is also undergoing digitalization, because in the midst of such chaos, we must continue to learn and grow, regardless of location or the circumstances around us. To do this, everything we need must be in one place. Of course, technology cannot replace a teacher at this point, but supplementary materials, homework assignments, and grades–all hosted on a single website accessible to parents, students, and teachers–will simplify life and add flexibility to the educational process. This approach will save time and resources and be freely available to any educational institution or organization.

Purpose *of this article is to develop an informational and testing web application for an educational institution, equipped with basic and essential functionality, which can be further improved based on user feedback, professional development, and team expansion.*

Results. *The article examines the development of a modern web-based educational platform designed to support interaction between students, teachers, and school administrators. It describes the main stages of system development, including the analysis of existing educational platforms, identification of their advantages and limitations, and formulation of requirements for a unified and scalable learning management system.*

The proposed system integrates core educational processes into a single platform, enabling users to manage schools, classes, subjects, assignments, tests, and communication within one ecosystem. The main features of the system include user registration and authentication, role-based access control (student, teacher, administrator), creation and management of classes, assignment distribution, test creation and evaluation, file sharing, and interaction through posts and notifications. The system also supports invitation mechanisms via email and ensures secure access through token-based authentication.

Special attention is given to the architecture and implementation of the system, which is based on modern technologies such as PostgreSQL, Prisma ORM, Node.js, TypeScript, NestJS, React, Redux, Vite, and Material UI. These technologies provide high performance, modularity, and scalability of the system. The system also integrates additional services such as MinIO for file storage, Docker for containerization, and MailHog for testing email functionality. The client-side and server-side applications are designed as a unified full-stack solution ensuring consistent user experience across different devices.

The testing results confirm the correct operation of all system modules, including authentication, role management, data processing, file handling, and communication features. The developed platform is fully functional and suitable for practical use in educational institutions.

Conclusion. *The article presents a study on the design and implementation of a unified educational management system for schools. The development of such a system addresses the*

fragmentation of existing educational tools and improves the efficiency of communication and data management within educational institutions.

The proposed solution demonstrates the effectiveness of integrating various educational processes into a single digital platform, allowing users to manage academic activities in a structured and convenient way. This approach significantly simplifies interaction between students, teachers, and administrators while improving transparency and organization of learning processes.

The main outcomes of the work include:

- *analysis of existing educational platforms and identification of their limitations;*
- *formulation of functional requirements for a unified system;*
- *design and implementation of a full-stack web application using modern technologies;*
- *integration of authentication, role management, and educational workflows;*
- *testing and validation of system functionality and usability.*

The results of the study confirm that the developed system improves the efficiency of educational process management and can be further extended with additional features such as analytics, mobile applications, and external integrations in the future.

Keywords: *web application, information system, client-server architecture, database, PostgreSQL, Prisma, Node.js, TypeScript, NestJS, REST API, JWT, authentication, authorization, React, SPA, Redux, Vite, MUI, Figma, Docker, MinIO, MailHog, deployment, web development.*

Одержано редакцією 20.09.2024 р.
Прийнято до публікації 30.10.2024 р.

УДК 004.415.2:004.75

DOI 10.31651/2076-5886-2024-1-90-102

PACS 07.05.Tp, 89.20.Ff

ТКАЧЕНКО Олексій Олексійович
студент спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького

ДІДКОВСЬКИЙ Руслан Михайлович,
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

ПІСКУН Олександр Варфоломійович
кандидат технічних наук, доцент, завідувач кафедри прикладної математики та інформатики, Черкаський національний університет ім. Б. Хмельницького
e-mail: piskun@ukr.net
ORCID 0000-0001-5334-6337

РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ МУЛЬТИПЛЕЄРНОГО КЛАВІАТУРНОГО ТРЕНАЖЕРА З ВИКОРИСТАННЯМ REACT ТА WEBSOCKET-ТЕХНОЛОГІЙ

У роботі розглянуто підходи до проектування та розробки клієнтської частини мультиплеєрного веб-застосунку для тренування швидкості та точності клавіатурного