Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

«Розробка веб-застосунку з використанням фреймворку Vue»

Текстова частина до курсової роботи

за спеціальністю «Комп'ютерні науки» 122

Керівник курсової роботи
ст. викладач Борозенний С.О.

_____
(підпис)
"_____" _____ 2023 р.

Виконав студент Біловербенко І.І.
"_15__" __травня__ 2023 р.

Київ 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри мультимедійних систем
доцент, к.ф.-м.н.
О. П. Жежерун

_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту _____Біловербенку Іллі Ігоровичу____

3-го курсу факультету інформатики

ТЕМА: Розробка веб-застосунку з використанням фреймворку Vue

Вихідні дані:

Зміст ТЧ до курсової роботи:
Індивідуальне завдання
Вступ

Дата видачі "_____" _____ 2023 р. Керівник_____
(підпис)
Завдання отримав_____
(підпис)

Зміст

Вступ

У сучасному світі все більше людей створюють власний онлайн-контент. Це стосується різних категорій людей, таких як блогери, журналісти та науковці. Сервіси для публікації відео, фото, аудіо контенту є дуже популярними нині, проте текстовий контент відходить на другий план. Звісно, попит на текстову інформацію зменшився, але не зник і не зникне, мабуть, ніколи. Саме тому розробка зручного веб-застосунку для публікації статей є актуальним рішенням цієї проблеми.

Результатом розробки став веб-застосунок, який може забезпечити широку аудиторію користувачів зручною та простою платформою для обміну інформацією.

Метою розробки веб-застосунку є створення платформи для обміну текстовою інформацією між користувачами. Застосунок дозволяє користувачам не тільки публікувати власні статті, а й читати та коментувати статті інших користувачів, а також підписуватись на їхні оновлення.

Можливості, які надає веб-застосунок, включають у себе:

- Публікацію та редагування статей
- Перегляд статей інших користувачів
- Коментування статей інших користувачів
- Підписку на інших користувачів та отримання оновлень їхніх статей
- Редагування власного профілю

Усі ці можливості дозволяють користувачам ефективно обмінюватись інформацією один з одним та мати позитивний досвід користування додатком.

Додатковою метою розробки було дослідження можливостей фреймворку Vue для створення подібних веб-застосунків. У наші дні веб-розробка стає все складнішою та різноманітнішою, тому використання потужних інструментів є дуже важливим. Vue та Nuxt 3 - це сучасні фреймворки, які дозволяють розробляти веб-додатки швидко та ефективно. Дослідження можливостей цих інструментів дозволить краще зрозуміти, як вони працюють та як їх можна використовувати для розвитку більш складних веб-додатків у майбутньому. Таким чином, створення програми не тільки дозволило створити корисний сервіс, а й дослідити можливості та переваги одного з найпопулярніших фреймворків веб-розробки.

Анотація

Ця курсова робота зосереджена на розробці веб-додатків для публікації статей і взаємодії користувачів. Метою цієї роботи є дослідження можливостей фреймворку Nuxt 3 та NodeJS, а також створення повноцінного веб-застосунку з використанням вищезгаданих технологій та засобів.

У роботі буде проведено аналіз предметної області, сформульовано вимоги до кінцевого застосунку та обрано необхідні технології для розробки.

У якості середовища розробки був використаний Visual Studio Code. Для клієнтської частини - Vue та CSS-фреймворк Tailwind. Для бекенду був використаний NodeJS з фреймворком Express. База даних - MongoDB. Також були використані технології та фреймворки для полегшення розробки, зокрема Nuxt 3, Mongoose та Axios, для тестування - Postman.

У роботі буде описано реалізацію веб-застосунку, зокрема:

- створення сутностей у базі даних
- архітектуру серверу та клієнтської частини
- розробку компонентів
- функціоналу веб-застосунку.

Окремо буде описано інтерфейс користувача та його можливості.

Перевагами цього застосунку є:

- інтуїтивно зрозумілий і легкий інтерфейс
- не тільки публікувати свої статті, але й коментувати статті інших користувачів, підписуватись на інших користувачів та бачити оновлення їхніх статей

- забезпечує захист даних користувачів та надійність системи
- швидкий доступ до статей та коментарів за допомогою використання новітніх технологій та оптимізованих алгоритмів

Всі ці переваги роблять застосунок привабливим для використання та подальшої розробки.

Отже, результатом роботи є повноцінний веб-застосунок, розроблений з використанням сучасних технологій та засобів, який дозволяє користувачам публікувати та редагувати статті, коментувати статті інших користувачів, підписуватись на інших користувачів та взаємодія

У роботі передбачено 3 головні розділи:

1. аналіз предметної області:
   - постановки головних цілей та задач
2. вибір технологій та засобів для розробки:
   - середовища розробки
   - клієнтської частини
   - серверної частини
   - тестування
3. реалізація веб-застосунку:
   - демонстрація архітектурних рішень у програмі
   - демонстрація кінцевого продукту в ролі користувача

1. Аналіз предметної області

1.1 Постановка цілі

Метою курсової роботи є розробка веб-застосунку з використанням сучасних технологій та фреймворків, який дозволить користувачам обмінюватися інформацією у вигляді статей і коментарів до них. Крім того, ця курсова робота має на меті дослідження можливостей фреймворку Vue (разом з Nuxt 3) та інших технологій для реалізації клієнтської частини, серверу і бази даних.

1.2. Вимоги до кінцевого застосунку

Згідно вимог до кінцевого застосунку, перш за все, він має бути доступним з будь-якого пристрою з підтримкою браузера та Інтернету, що дозволить користувачам зручно і ефективно використовувати сервіс. Крім того, користувачі очікують від застосунку інтуїтивного та легкого у використанні інтерфейсу, що дозволить їм з легкістю публікувати свої статті, редагувати їх та коментувати статті інших користувачів.

Для того, щоб забезпечити безпеку персональних даних користувачів, необхідно, щоб у додатку була система автентифікації та авторизації. Це дозволить користувачам входити в систему, використовуючи свої особисті дані, наприклад, пароль, і отримувати доступ до власної панелі зі статтями, налаштуваннями та іншими функціями. Крім того, мають бути створені облікові записи з розширеними можливостями - ці особи виконуватимуть роль модераторів у додатку, які можуть модерувати застосунок, видаляти статті і коментарі інших користувачів.

Однією з вимог є також функціонал підписки на інших користувачів, що дозволить відстежувати активність інших користувачів та бути в курсі їхніх останніх публікацій та дій.

Щоб забезпечити комфортну та безпечну взаємодію користувачів з сервісом, а також зробити його зручним у використанні та привабливим для потенційних користувачів, додаток повинен відповідати цим вимогам.

2. Вибір технологій та засобів для розробки

2.1. Вибір середовища розробки. Visual Studio Code.

Visual Studio Code - одне з найпопулярніших і найзручніших середовищ розробки на сьогоднішній день. Воно безкоштовне, має відкритий вихідний код, активну спільноту, підтримує різні мови програмування та технології, включаючи й ті, що використовуються у даному проекті.

Рішення обрати Visual Studio Code для розробки цього веб-додатку є дуже виправданим, оскільки воно надає багато зручних інструментів та розширень, які допомагають підвищити продуктивність, читабельність та якість коду.

Одним з таких розширень є Prettier. Воно дозволяє автоматично форматувати код відповідно до заданих правил і зберігати його у відповідному стилі, що допомагає зменшити кількість помилок та підвищити читабельність коду.

Ще одне розширення - ESLint, допомагає виявляти та виправляти помилки в коді на підставі заданих правил, що допомагає підвищити якість коду та зменшити час на виявлення помилок.

Розширення Auto Rename Tag дозволяє автоматично змінювати назву HTML -тегів. Це допомагає зменшити кількість помилок та підвищує швидкість розробки.

Git Graph допомагає зрозуміти історію комітів і гілок у Git-репозиторії та керувати комітами, що дозволяє більш ефективно працювати з Git.

Розширення Tailwind CSS IntelliSense дозволяє працювати з фреймворком CSS Tailwind і надає підказки назв класів, що дозволяє швидше та зручніше додавати необхідні стилі.

Розширення Vue Language Features (Volar) допомагає покращити продуктивність розробки Vue-додатків. Воно надає додаткові функції для розробки Vue-додатків, такі як автодоповнення та підказки для Vue-компонентів.

2.2 База даних: MongoDB

MongoDB - це документо-орієнтована база даних, яка забезпечує гнучку і швидку роботу з даними. Вона зберігає та опрацьовує документи у форматі JSON. Ця база даних є однією з найбільш популярних баз даних серед розробників, оскільки вона має кілька переваг, які є важливими для розробки даного застосунку:

- *Гнучкість*. MongoDB дозволяє гнучко працювати з даними, що у свою чергу забезпечує легкість і швидкість розробки та спрощує масштабованість. MongoDB не потребує жорстко структурованих схем таблиць. Таким чином, користувачі можуть легко реагувати та адаптуватися до змін у функціональних вимогах доволі швидко.
- *Швидкість*. MongoDB є швидкою базою даних, оскільки вона дозволяє просте та швидке збереження даних у форматі JSON.
- *Масштабованість*. Бази даних MongoDB легко масштабуються, що дозволяє розробникам збільшувати її обсяг без значних затримок у роботі додатку.

У моєму застосунку, MongoDB забезпечує гнучку та швидку відповідь на запити користувачів, а також зберігання і редагування інформації у базі.

Також тут MongoDB використовуються в парі з NodeJS та Express для забезпечення повної функціональності додатку та високої продуктивності. Ця комбінація є доволі популярною у розробці веб-застосунків, тому що є

зручною у використанні розробниками. Це стало одною з причин вибору цієї бази даних.

2.3. Серверна частина: NodeJS, Express

Серверна частина є ключовою складовою будь-якого веб-додатку, який взаємодіє з базою даних та забезпечує обробку запитів користувачів. У моєму застосунку я використовую NodeJS та Express для реалізації серверної логіки.

NodeJS - одне з найпопулярніших та найефективніших рішень для серверного програмування. Це середовище дозволяє розробникам створювати високопродуктивні та масштабовані веб-додатки, пропонуючи широкий спектр різноманітних пакетів, які можуть значно скоротити час розробки. Асинхронна модель виконання забезпечує швидку реакцію на запити користувачів.

Express - це фреймворк для NodeJS, який дозволяє легко створювати веб-сервери та API. Express забезпечує швидку та ефективну обробку HTTP-запитів, підтримує middleware для роботи з різними протоколами, форматами даних та базами даних. Використання Express значно спрощує розробку серверної логіки та підвищує продуктивність додатку.

Використання NodeJS та Express дозволяє реалізувати весь необхідний функціонал для роботи з базою даних, аутентифікації користувачів, роутингу запитів тощо.

2.4. Клієнтська частина застосунку: Vue, Nuxt 3, CSS-фреймворк Tailwind.

Vue.js - це прогресивний фреймворк для створення веб-додатків та інтерфейсів. Він дозволяє створювати реактивні компоненти, які можуть

бути повторно використані, що дозволяє прискорити і полегшити процес розробки.

Nuxt 3 - це фреймворк, який дозволяє розробникам створювати універсальні веб-додатки на основі Vue.js. За допомогою Nuxt 3 можна розробляти високопродуктивні веб-додатки, які працюють як на стороні клієнта, так і на стороні сервера. Фреймворк пропонує різні корисні функції, такі як автоматична генерація маршрутів, підтримка модулів та рендеринг на стороні сервера.

Основна перевага Vue та Nuxt 3 полягає в тому, що вони дозволяють створювати високопродуктивні та масштабовані веб-додатки швидко та ефективно. Крім того, ці фреймворки можуть похвалитися великою кількістю бібліотек та інших інструментів, що полегшують розробку складних проектів. Завдяки декларативному підходу Vue та широким функціям Nuxt 3, розробники можуть зосередитися на функціональності та взаємодії компонентів або інших елементів архітектури додатку, і не приділяти занадто багато часу деталям реалізації.

Мало того, Vue та Nuxt 3 також підтримують TypeScript, що дозволяє розробникам писати код з меншою кількістю помилок, який краще налагоджується. Використання мови TypeScript зменшує кількість помилок у коді та робить його більш зрозумілим для інших розробників завдяки використанню чітко визначених типів. Відсутність цих типів у Jasvascript розглядається як недолік цієї мови багатьма розробниками. Тому використання TypeScript у поєднанні з Vue або Nuxt 3 може помітно покращити процес розробки, зменшивши при цьому кількість потенційних помилок.

CSS фреймворк Tailwind став досить популярним інструментом для швидкої та ефективної розробки користувацьких інтерфейсів. Він надає великий вибір вбудованих класів, що дозволяють легко створювати різноманітні елементи інтерфейсу, такі як кнопки, форми, меню, таблиці та багато іншого.

Однією з головних переваг Tailwind є його підхід до стилів, який ґрунтується на використанні класів, а не на ручному написанні коду CSS. Це дозволяє швидко створювати та змінювати стилі, забезпечуючи при цьому їхню узгодженість та одноманітність.

Іншою перевагою Tailwind є його налаштування та можливості налаштування власних стилів. Розробники можуть змінювати існуючі та додавати власні класи.

Крім того, з Tailwind не потрібно писати багато CSS коду, щоб швидко впроваджувати анімацію та інші ефекти. Загалом, Tailwind є потужним і зручним інструментом для швидкої розробки користувацьких інтерфейсів. Особливо це стосується використання у поєднанні з такими фреймворками, як Vue та Nuxt 3.

2.5. Технології для полегшення розробки. Mongoose, Axios, Passport.js.

Mongoose - це бібліотека, яка спрощує та покращує роботу з базами даних MongoDB. Вона дозволяє створювати моделі даних і виконувати CRUD-операції, полегшує навігацію і розуміння бази даних. Інтерфейс для виконання цих операцій простий та інтуїтивно зрозумілий, що робить управління базами даних більш приємним.

Axios - це бібліотека, яка використовується для створення HTTP-запитів з боку клієнта до сервера. За допомогою цієї бібліотеки можна робити асинхронні запити та отримувати відповіді у форматі JSON. Завдяки Axios

взаємодія з API стає простою, навіть коли мова йде, наприклад, про аутентифікацію користувачів.

Passport.js - це бібліотека, яка використовується для аутентифікації користувачів в NodeJS. По суті, Passport.js забезпечує систему автентифікації, яка гарантує безпеку та конфіденційність користувачів. Passport.js має підтримку різних стратегій аутентифікації, таких як логін та пароль, OAuth і т.д. У моєму застосунку, зокрема, використано підхід логіну та паролю.

Разом з Vue, Nuxt 3, Typescript, CSS-фреймворком Tailwind та NodeJS з Express, ці технології допомагають створювати масштабований та безпечний веб-додаток зі зрозумілим інтерфейсом користувача та швидким доступом до даних з бази MongoDB.

2.6. Тестування: Postman

Ручне та автоматизоване тестування є важливими етапами процесу розробки будь-якого додатку. Це гарантує, що функції програми працюють коректно і не містять помилок.

Для тестування API можна використовувати різні інструменти. Одним з найпопулярніших є Postman. Postman - це платформа для розробки, тестування та документування API. Завдяки зручному користувацькому інтерфейсу Postman дозволяє легко створювати та виконувати запити до API для швидкого тестування різних сценаріїв.

Ручне тестування дозволяє перевірити окремі функції застосунку вручну. Наприклад, можна перевірити, чи можна додати нову статтю, редагувати існуючу статтю, коментувати статті та інші функції.

З іншого боку, автоматизоване тестування ґрунтується на створенні тестів, які можна запускати автоматично. Зокрема, для тестування API можна використовувати Postman у якості інструменту для написання автоматизованих тестів. Наприклад, можна написати тест, який перевіряє, чи можна отримати список всіх статей, чи можна додати нову статтю та перевірити, чи вона була додана до бази даних.

Програма Postman дозволяє автоматично проводити тести, підтверджуючи успішне завершення кожного з них. Цей спосіб допомагає виявити помилки та баги, що в кінцевому результаті покращує якість додатків та зменшує ймовірність появи помилок під час розробки. Автоматизація тестування за допомогою Postman також спрощує цей важливий процес, заощаджуючи час і зусилля.

Отже, використання Postman для тестування застосунку дозволяє забезпечити якість коду та функціональності, знижує ризик виникнення помилок і неочікуваної поведінки програми.

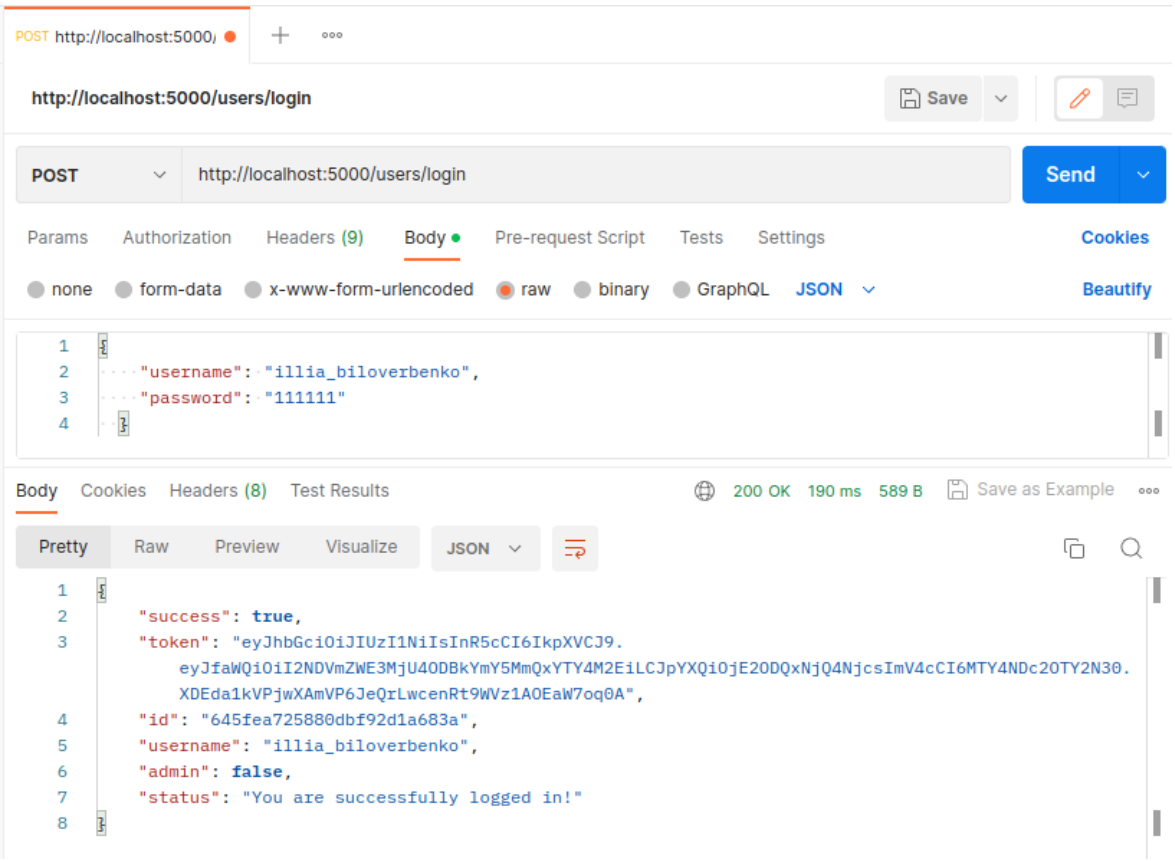

Рисунок 2.6.1 - Ручне тестування на прикладі запиту автентифікації

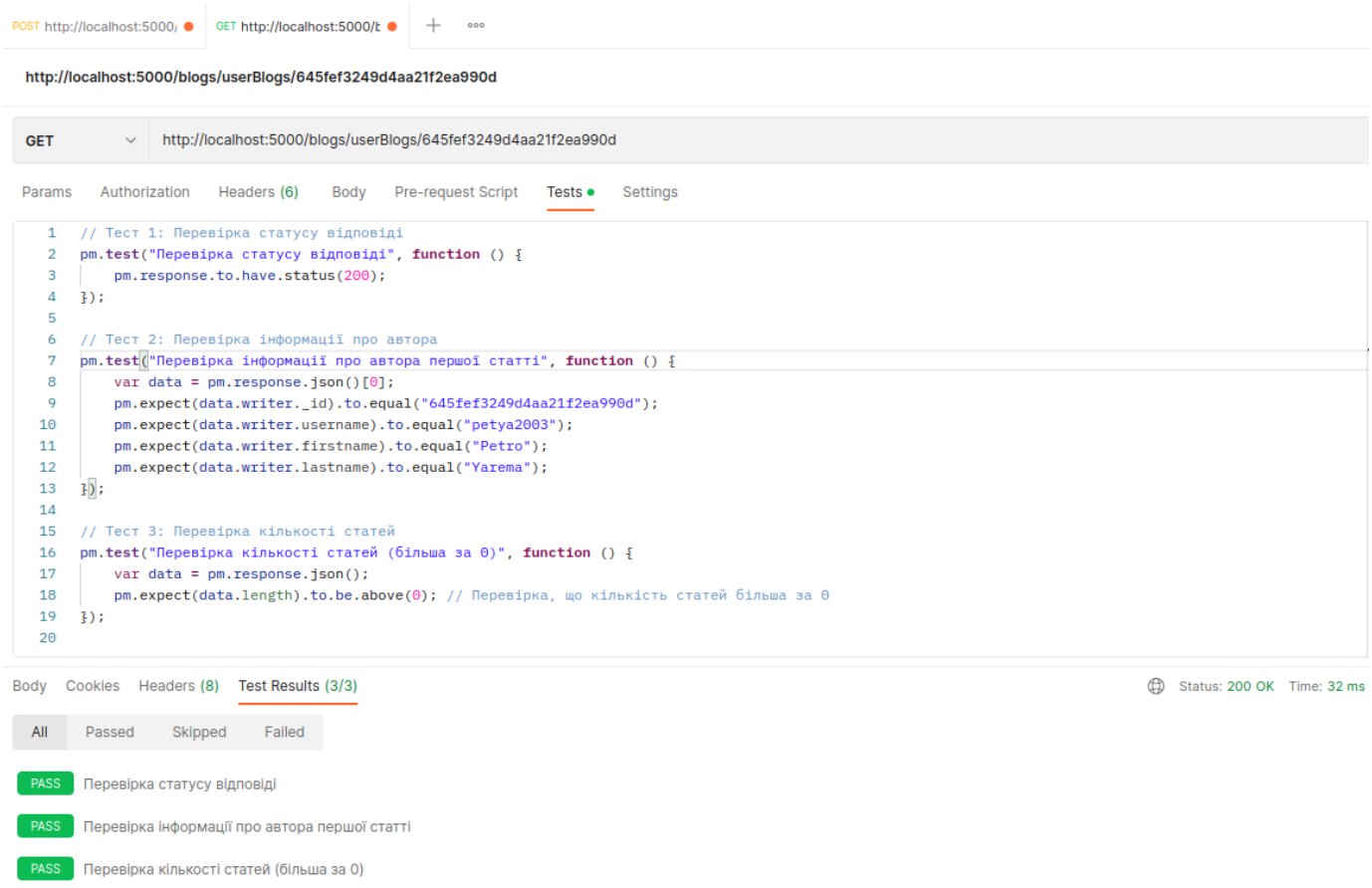

Рисунок 2.6.2 - Автоматизоване тестування на прикладі запиту отримання статей користувача з певним id

3. Реалізація веб-застоснку

3.1. Сутності у базі даних

У базі даних є три сутності: "user", "blog" та "comments", які відповідають схемам *User*, *blogSchema* та *commentSchema* відповідно.

User відображає користувача та містить наступні поля:

- username: ім'я користувача, тип String, обов'язкове поле, унікальне значення; використовується для аутентифікації користувача.
- firstname: ім'я користувача, тип String, за замовчуванням порожній рядок;
- lastname: прізвище користувача, тип String, за замовчуванням порожній рядок;
- admin: прапорець, що вказує, чи є користувач адміністратором, тип Boolean, за замовчуванням - false;
- following: масив користувачів, на які користувач підписаний, тип [String] (масив значень типу String).

blogSchema відображає блог та містить наступні поля:

- header: заголовок блогу, тип String, обов'язкове поле, унікальне значення;
- article: текст блогу, тип String, обов'язкове поле;
- writer: id автора блогу, тип mongoose.Schema.Types.ObjectId, посилання на колекцію *User*, обов'язкове поле;
- label: поле для надання додаткової інформації про блог (його категорія, тематика), тип String, необов'язкове поле;
- comments: масив коментарів до блогу, тип [commentSchema](масив значень типу *commentSchema*);

- timestamps: параметр колекції у Mongoose, який додає час створення та оновлення запису в колекції; не є полем сутності;

commentSchema описує коментар до блогу. Він містить наступні поля:

- review: текст коментаря, тип String, обов'язкове поле;
- author: id автора коментаря, тип mongoose.Schema.Types.ObjectId, посилання на колекцію *User*, обов'язкове поле;
- timestamps: параметр колекції у Mongoose, який додає час створення та оновлення запису в колекції; не є полем сутності;

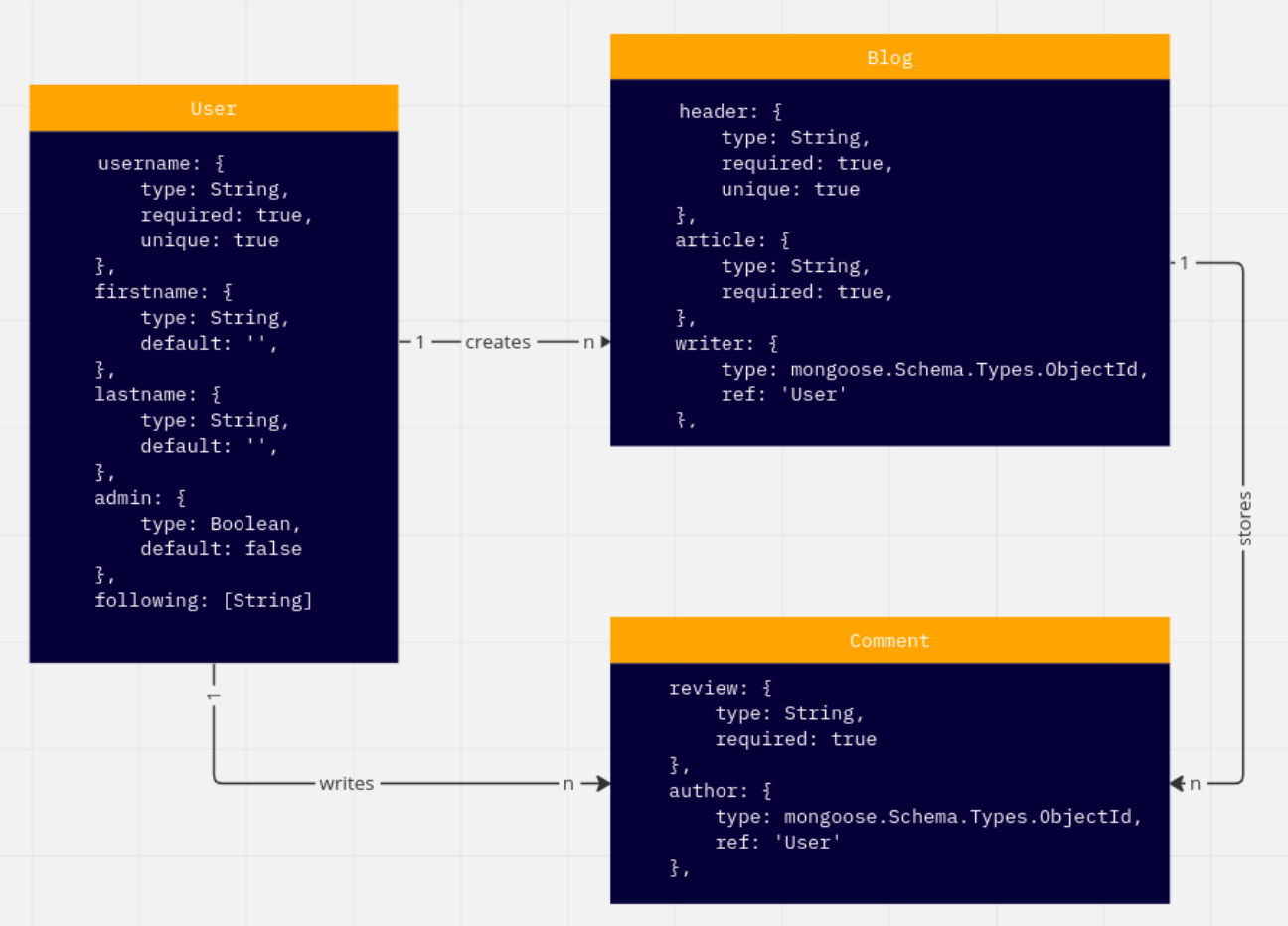

Рисунок 3.1.1 – Колекції, які використовуються для зберігання інформації у базі даних

3.2. Архітектура серверу

Весь код серверу (бекенд частини) знаходиться у директорії “/backend”.

Корінь серверу - *app.js*. Цей файл створює новий екземпляр додатку Express та встановлює різні middleware для обробки запитів. Ось деякі з них:

- createError і logger використовуються для відображення помилок у логах.
- express - фреймворк для побудови серверної частини додатку на Node.js.
- cors - middleware, який дозволяє обмінюватися даними між доменами.
- passport - middleware для автентифікації.
- config - модуль, який містить конфігураційні дані для додатку.
- indexRouter, usersRouter та blogRouter - маршрутизатори, які оброблюють запити для відповідних сторінок додатку.

Крім того, у корені серверу ми здійснюємо з'єднання з базою даних MongoDB за допомогою пакету Mongoose. Після підключення до бази даних ми створюємо користувача з роллю адміністратора, дані про якого знаходяться у файлі *config.js.*

Загалом, app.js забезпечує з'єднання з базою даних, налаштування middleware та маршрутизацію запитів.

Файл *authenticate.js* містить декілька middleware, які використовуються при авторизації та автентифікації користувача. Серед них налаштування passport-стратегії для перевірки даних для автентифікації на основі JWT, перевірка користувача на автентифікованість, перевірка його ролі (адміністратор чи ні).

У директорії “/routes” знаходяться маршрутизатори indexRouter, usersRouter і blogRouter, які обробляють запити, які надходять до кореневої

директорії, до директорії “/users” (стосуються користувачів) та до директорії “/blogs” (стосуються статей) відповідно.

cors.js налаштовує CORS (Cross-Origin Resource Sharing) на сервері з використанням пакету cors. exports.cors - це middleware, який дозволяє запити з будь-якого джерела, а exports.corsWithOptions - це middleware, який дозволяє запити лише з вказаного джерела (в даному випадку з http://localhost:3000).

docker-compose.yaml містить конфігурацію для запуску сервісу MongoDB в контейнері Docker. Він описує один сервіс з назвою "mongodb", який використовує останню версію образу MongoDB. Контейнер має ім'я "mongodb" і завжди перезапускається у випадку будь-яких помилок. У ньому також відкритий порт 27017, який можна використовувати для підключення до бази даних MongoDB з іншого контейнера або зовнішнього середовища. Крім того, визначається змінна оточення MONGO_INITDB_DATABASE зі значенням "blogsDb", яка визначає ім'я бази даних, що буде створена під час запуску контейнера.

Для того, щоб запустити сервер і базу даних, потрібно запустити скрипт у файлі run_backend.sh за допомогою команди `sh run_backend.sh`.

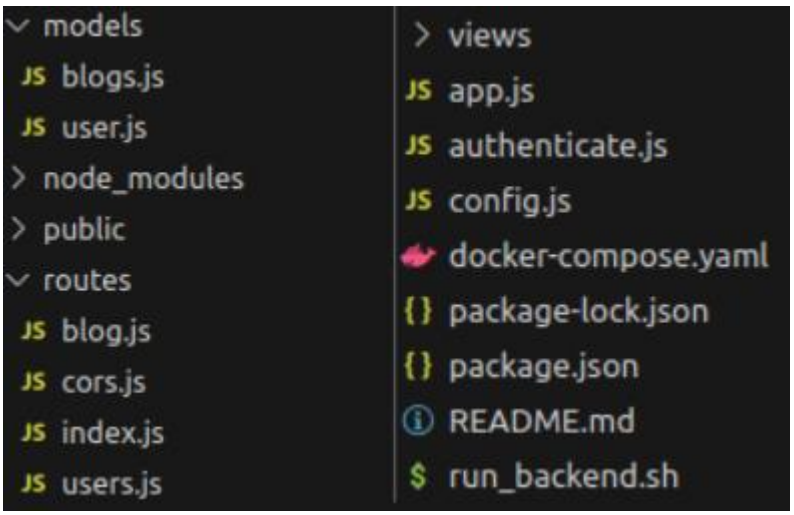

Рисунок 3.2.1 - Структура серверної частини застосунку

3.3. Архітектура клієнтської частини

Усі файли, які стосуються клієнтської частини, знаходяться у директорії “/nuxt-frontend”.

Кореневим файлом у клієнтській частині є app.vue.

Каталог "/*layouts*" містить файл стандартного шаблону, який використовується для всіх сторінок, якщо вони не мають власних шаблонів. Так, файл default.vue додає єдині для всіх сторінок заголовок і меню до всіх сторінок, які не є сторінками входу та реєстрації. Це гарантує, що ці компоненти не потрібно перезавантажувати щоразу, коли користувач переходить на іншу сторінку; натомість вони завантажуються лише при першому зверненні до сторінки.

Директорія “*/composables”* містить функції, які можна використовувати в різних компонентах і сервісах застосунку. Серед них, наприклад:

- dbActions.ts - містить функції, які використовуються для шаблонізації запитів до серверу
- getUser.js - функція для отримання інформації із бази даних про користувача, використовуючи його ім’я
- validateUserData.ts - перевіряє коректність введених при реєстрації або авторизації даних користувача. Дозволяє уникнути надсилання запитів, які гарантовано призведуть до помилки

Директорія “*/components”* містить Vue.js компоненти, які використовуються на різних сторінках додатку та дозволяють уникнути повторення коду. Деякі з них:

- the-header.vue - компонент header’у (“шапки” веб-сторінки)
- the-menu.vue - компонент меню (навігації по сайту)

- the-comment.vue - компонент для відображення коментарів

У директорії "/*middleware*" міститься функція, розташована у файлі "*auth.global.ts*", яка застосовується до усіх сторінок перед їхнім завантаженням задля унеможливлення користування застосунком неавторизованими користувачами.

Директорія "/*pages"* містить Vue.js компоненти, які відображають різні сторінки застосунку. Компоненти, розташовані у цій директорії автоматично перетворюються на HTML-сторінки під час рендерингу сторінок з відповідними до назв файлів шляхами.

Іншими директоріями є:

- assets - директорія для зберігання статичних файлів. У даному проекті використовується для зберігання стилів, які застосовуються в усіх компонентах застосунку
- types - містить TypeScript типи для додатку
- public - містить статичні файли, які будуть доступні на всіх сторінках додатку, такі як зображення. Файли в цій директорії не оптимізуються, та доступні через HTTP-запити без обробки на сервері

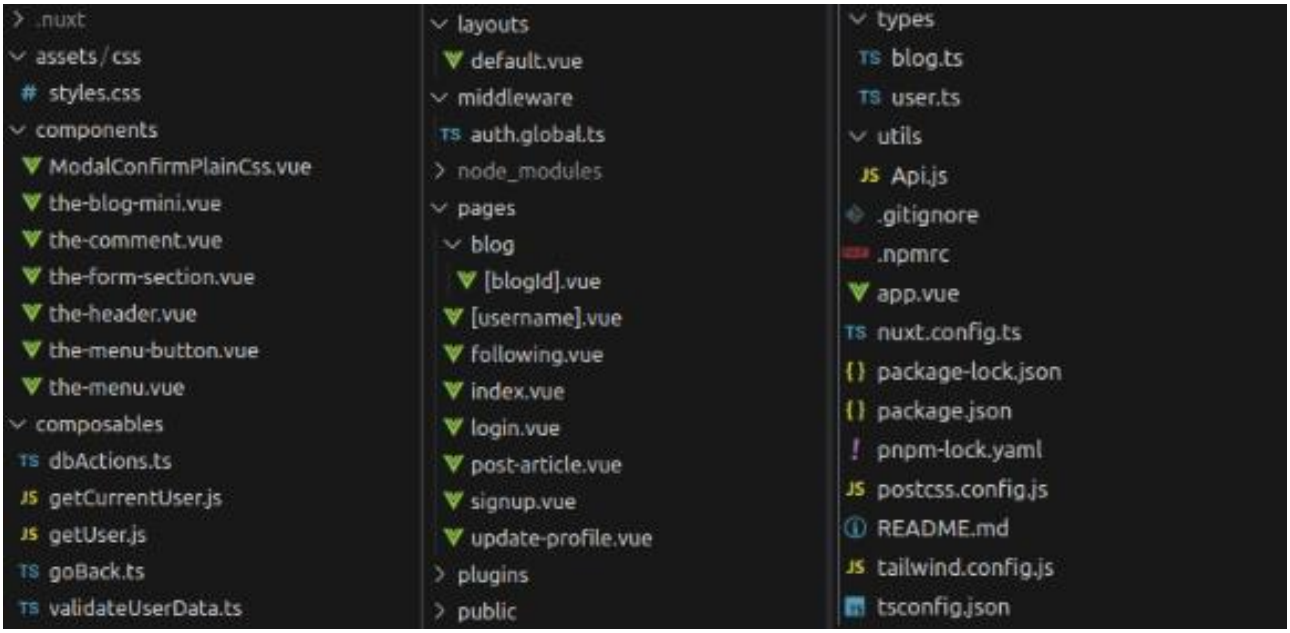

Рисунок 3.3.1 - Структура клієнтської частини застосунку

3.4. Функціонал веб-застосунку

Функціонал веб-застосунку включає наступне:

1. Автентифікація

- Sign up — новий користувач може створити обліковий запис, вказавши своє ім'я, електронну адресу та пароль. Після успішної реєстрації користувач автоматично проходить процес автентифікації та перенаправляється на домашню сторінку.
- Login — користувач може увійти в систему за допомогою свого імені користувача та пароля. Після успішного входу користувач перенаправляється на домашню сторінку.
- Logout — користувач може вийти з системи та очистити всі свої сеанси. Після виконання цієї дії користувач перенаправляється на сторінку автентифікації.

2. CRUD-операції для користувача

- Створення користувача (Sign up , описаний вище).

- Отримання даних користувача за його іменем (username).
- Отримання даних усіх користувачів: адміністратор може отримати дані з бази про усіх зареєстрованих користувачів.
- Редагування користувача: користувач може змінювати будь-які дані про себе, які він вводив.
- Видалення користувача: користувач може видалити свій обліковий запис разом з усіма статтями, але коментарями до статей інших користувачів.

3. CRUD-операції для колекції статей (доступні лише для автентифікованих користувачів)

- Створення статті: користувач може створити нову статтю з назвою, описом та текстом статті.
- Отримання списку статей: користувач може переглянути список всіх статей.
- Отримання списку статей, написаних певним користувачем.
- Отримання детальної інформації про статтю: користувач може переглянути детальну інформацію про окрему статтю: її назву, опис, зміст та коментарі.
- Отримання списку статей користувачів, на яких підписаний користувач: користувач може підписуватися на певних користувачів. Таким чином, їхні статті будуть відображатися на окремій сторінці.
- Редагування статті: користувач може редагувати інформацію про статтю, написану ним.
- Видалення статті: користувач може видалити статтю, написану ним.

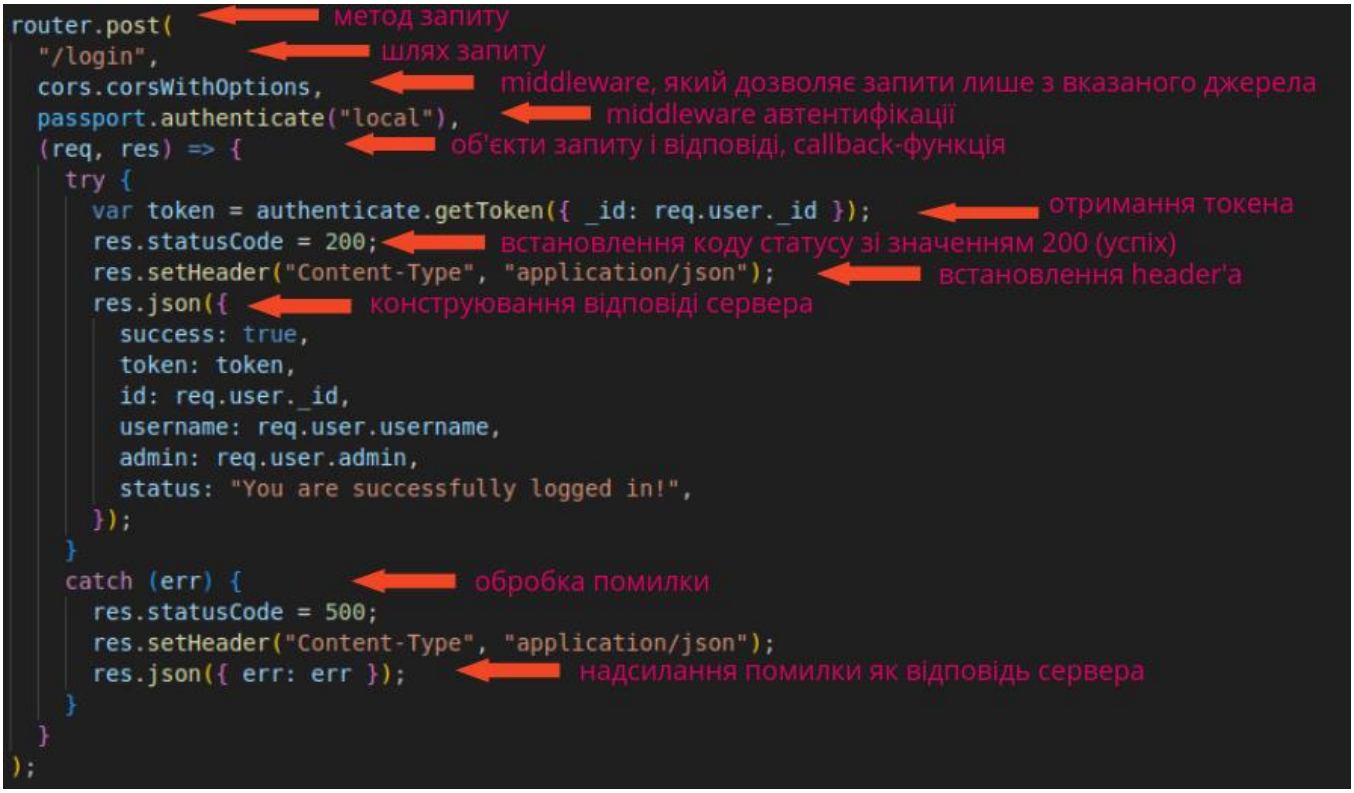

Рисунок 3.4.1 - приклад обробки запиту на сервері

3.5. Права доступу

У застосунку реалізована авторизація. Це означає, що не всі дії, доступні для адміна, доступні і для звичайного користувача. Наприклад, видалення будь-якого коментаря або статті вимагає використання проміжного програмного забезпечення, яке виконується перед обробкою запиту і відмовляє в доступі до запиту, повертаючи помилку, якщо не виконуються певні умови програмного забезпечення.

У застосунку є два middleware, розташованих у директорії “/backend/authenticate.js”, які забезпечують авторизацію:

- verifyUser - використовується для верифікації користувача. Його функціонал включає перевірку правильності пароля та логіна, перевірку наявності користувача в базі даних. Він застосовується до усіх запитів, окрім тих, що реєструють або автентифікують

користувача.

- verifyAdmin - використовується для перевірки, чи є користувач адміном. Це дозволяє надавати доступ до певних ресурсів тільки адміністраторам. Застосовується для запитів видалення статей, отримання інформації про усіх зареєстрованих користувачів.

```
exports.verifyUser = passport.authenticate('jwt', {session: false});

exports.verifyAdmin = function(req, res, next) {
    if (!req.user.admin) {
        let err = new Error('You are not allowed to perform this operation!');
        err.status = 403;
        return next(err);
    }
    next();
};
```

Рисунок 3.5.1 - реалізація проміжного програмного забезпечення автентифікації і авторизації

3.6. Інтерфейс користувача

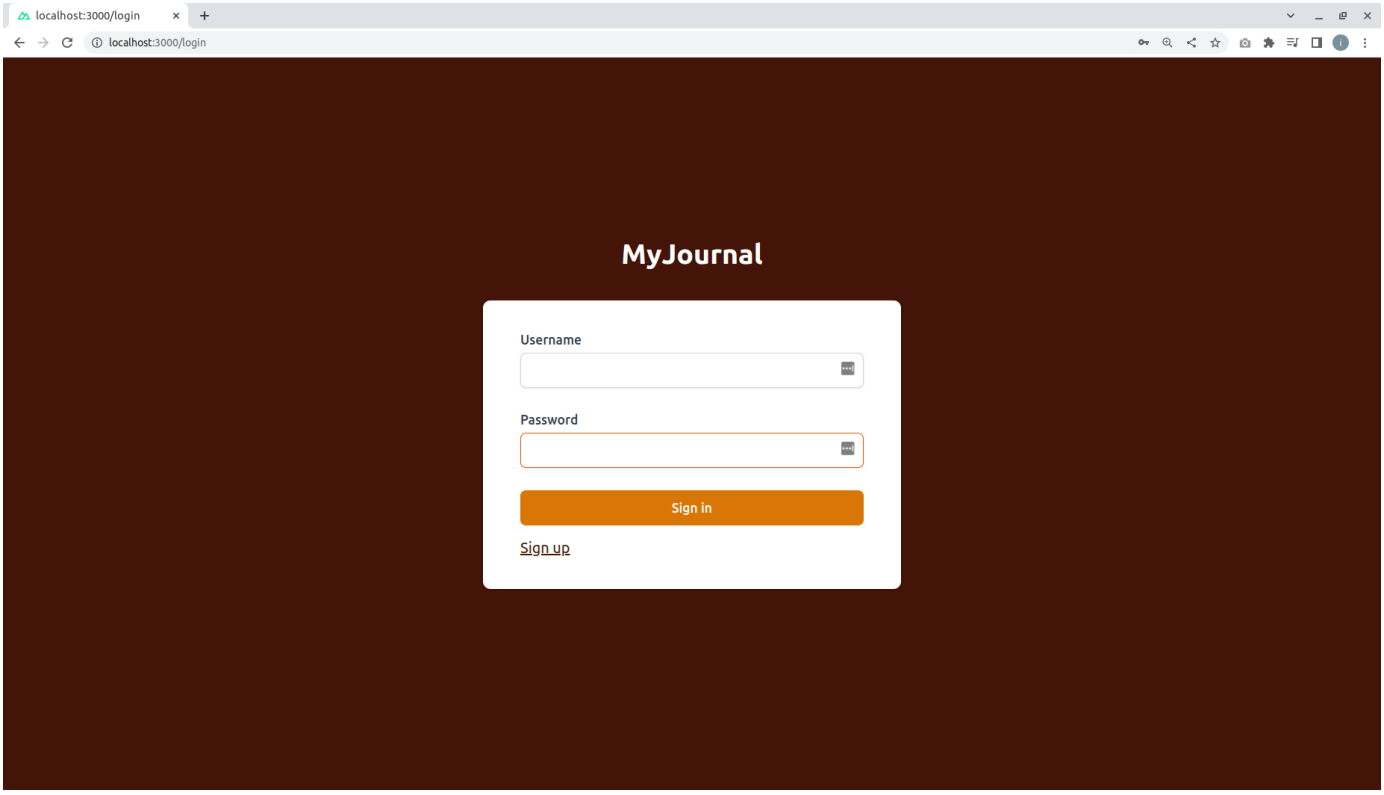

Рисунок 3.6.1 — сторінка автентифікації користувача

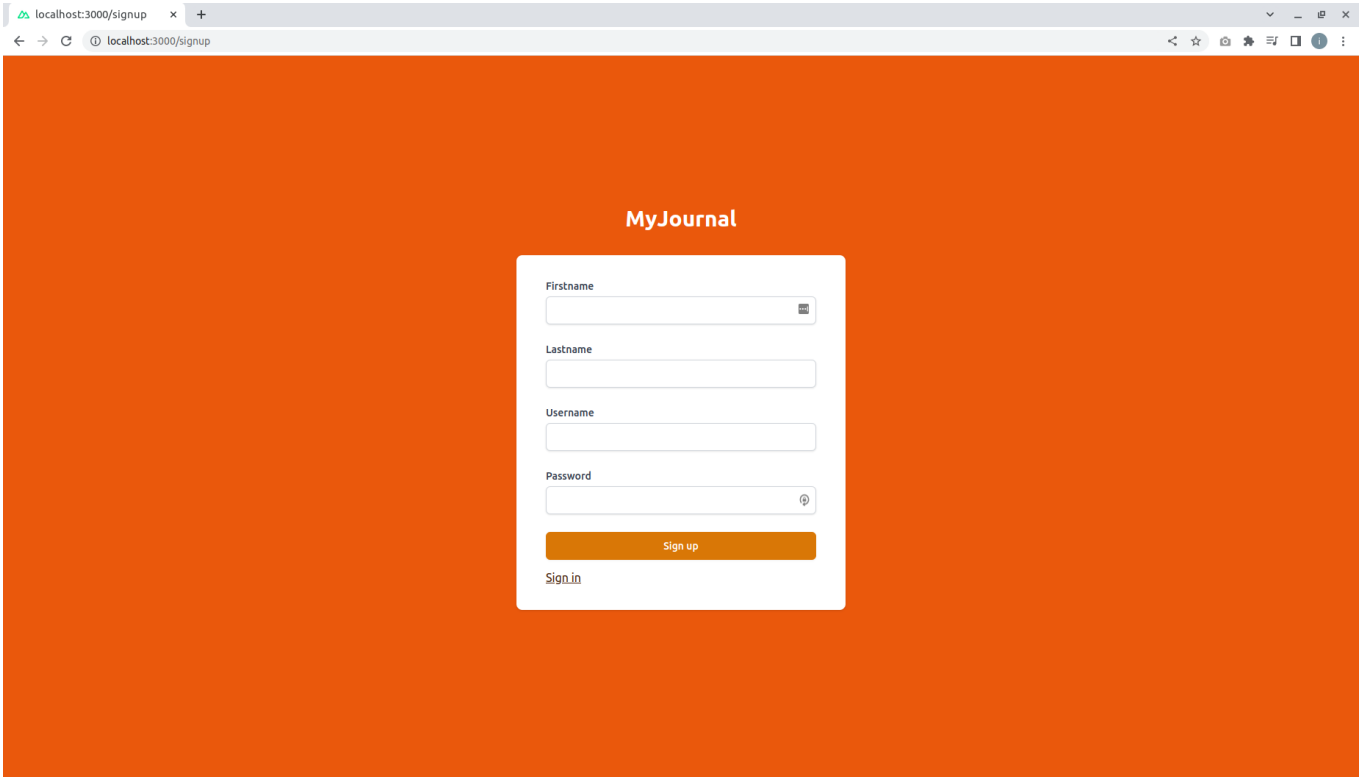

Рисунок 3.6.2 — сторінка реєстрації користувача

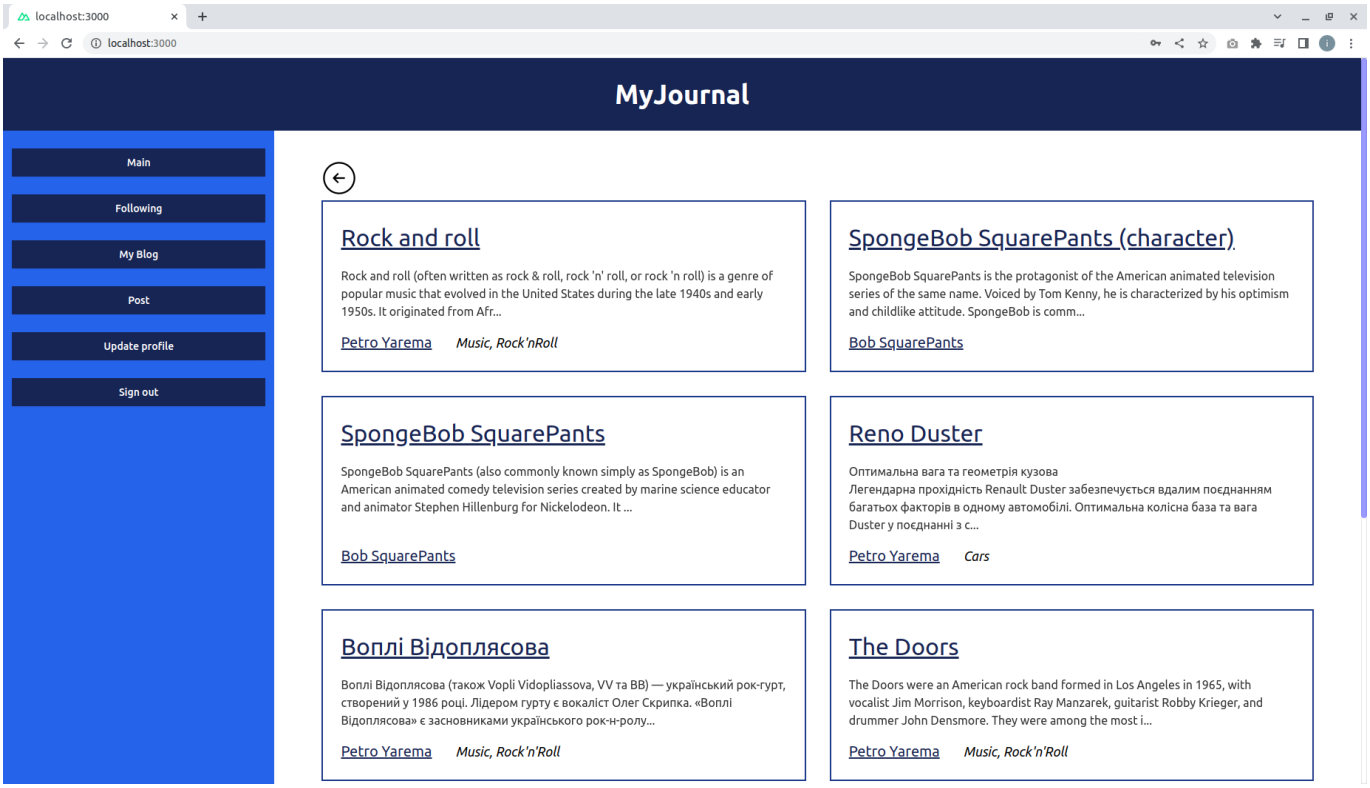

Рисунок 3.6.3 — головна сторінка з усіма статтями посортованих за датою по спаданню

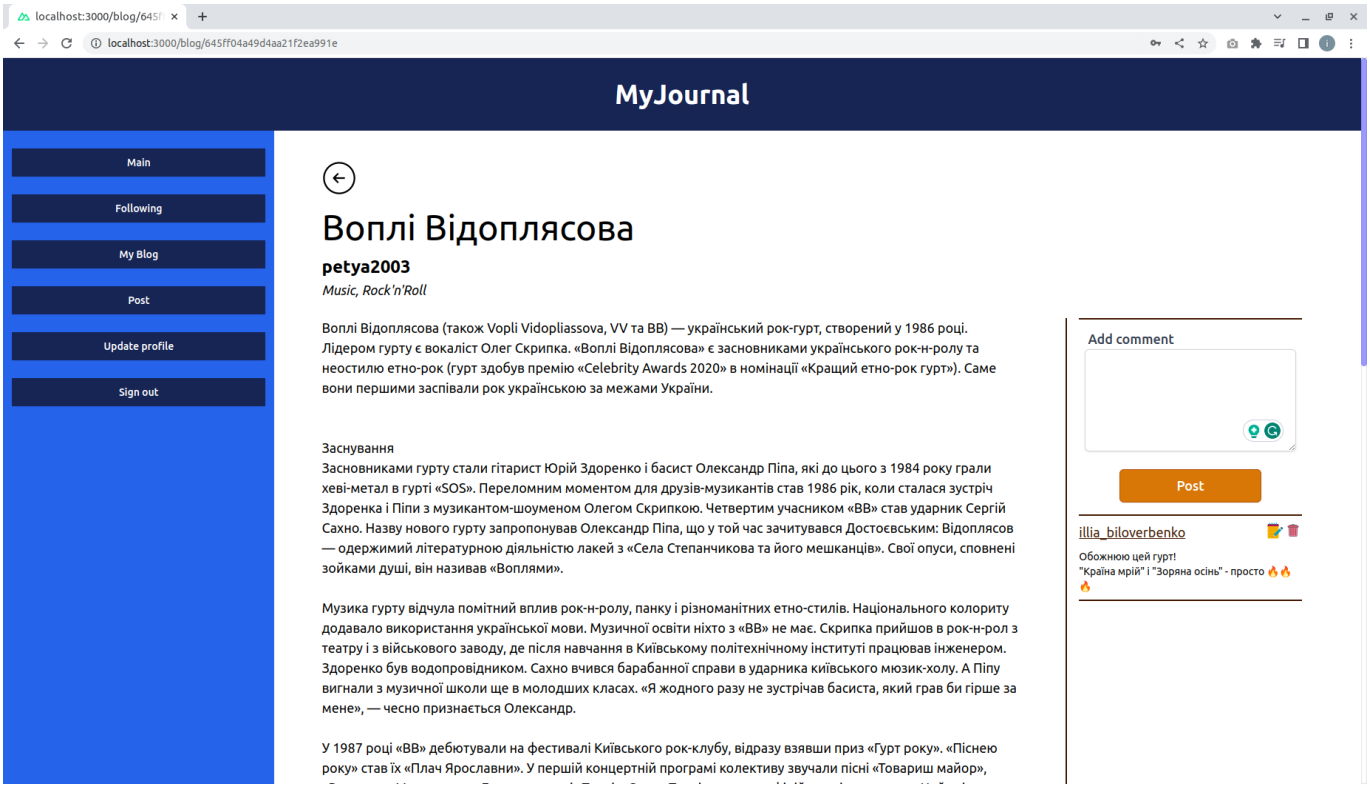

Рисунок 3.6.4 — сторінка статті

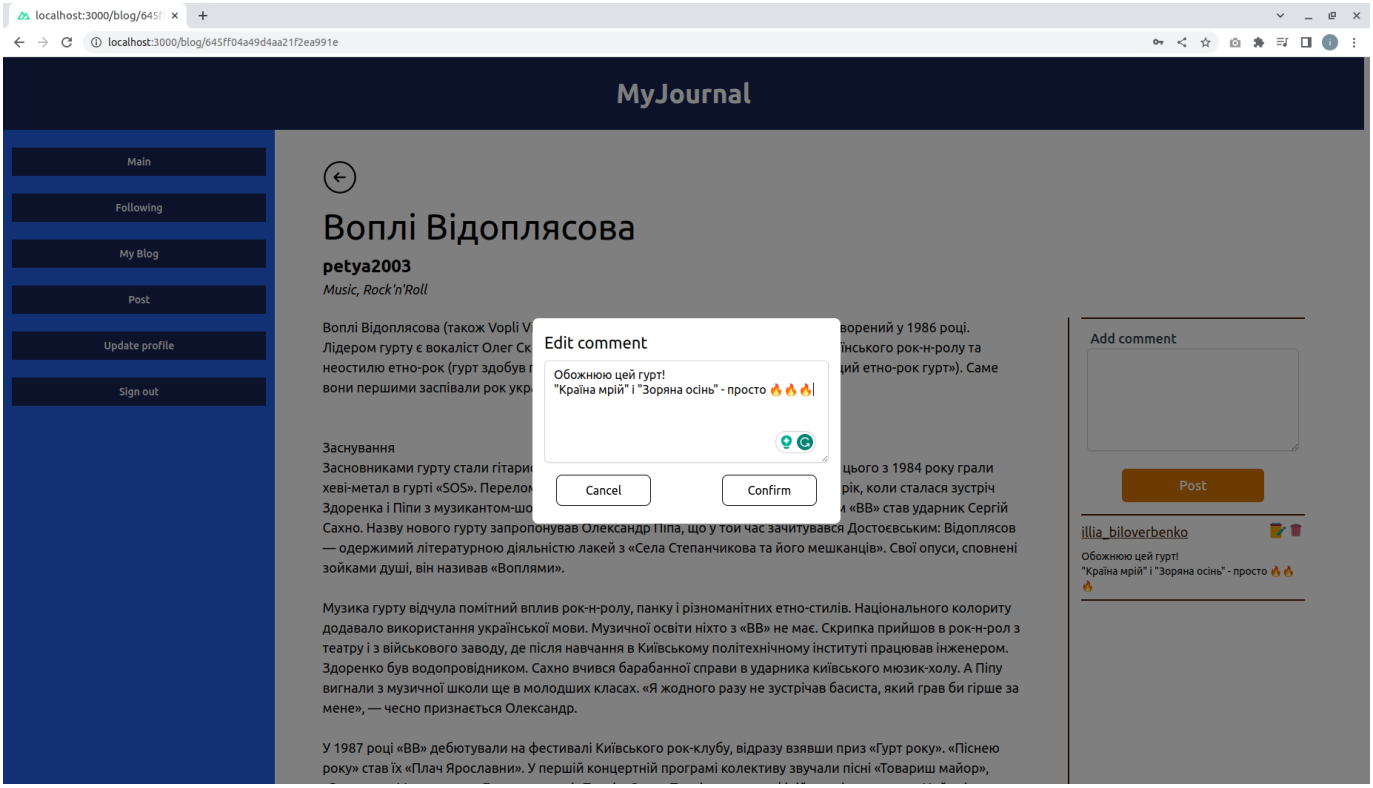

Рисунок 3.6.5 — редагування коментаря

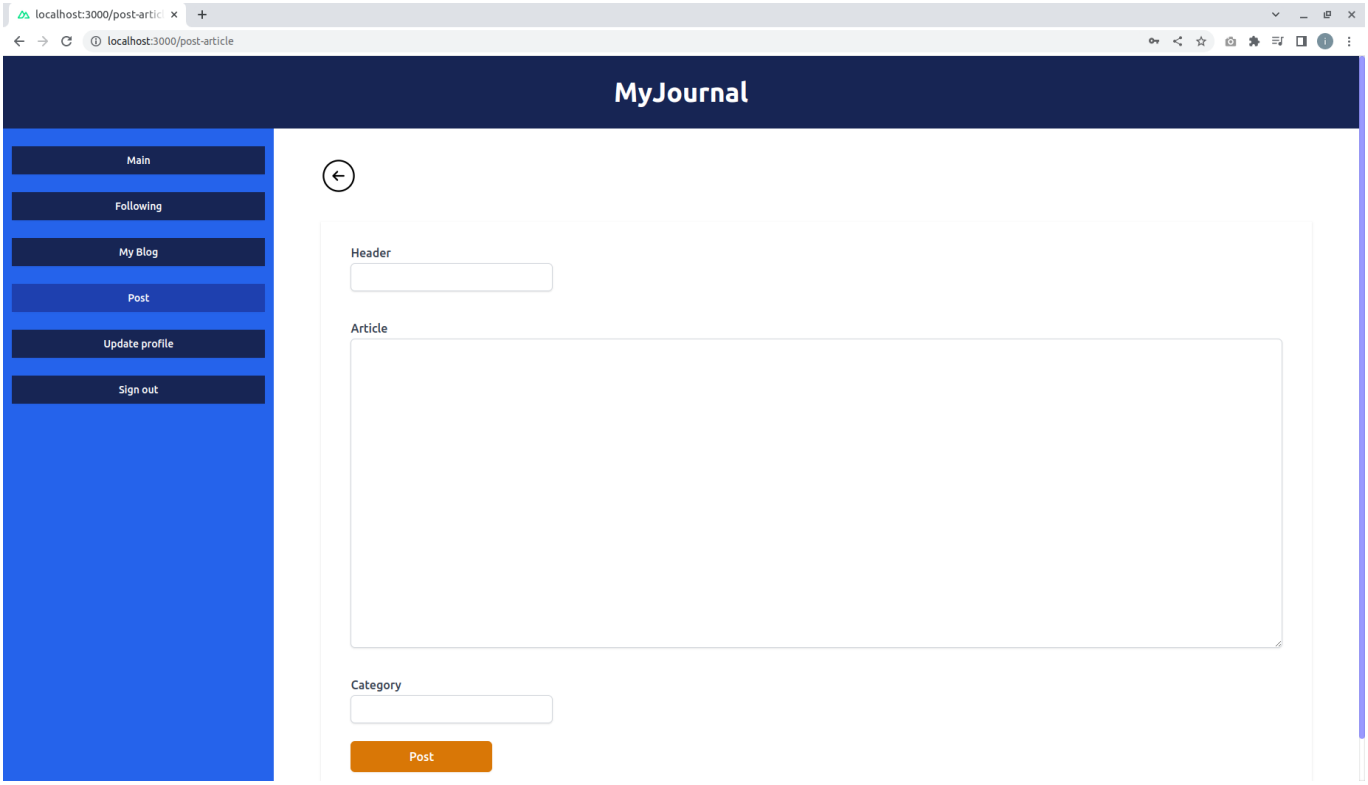

Рисунок 3.6.6 — сторінка створення статті

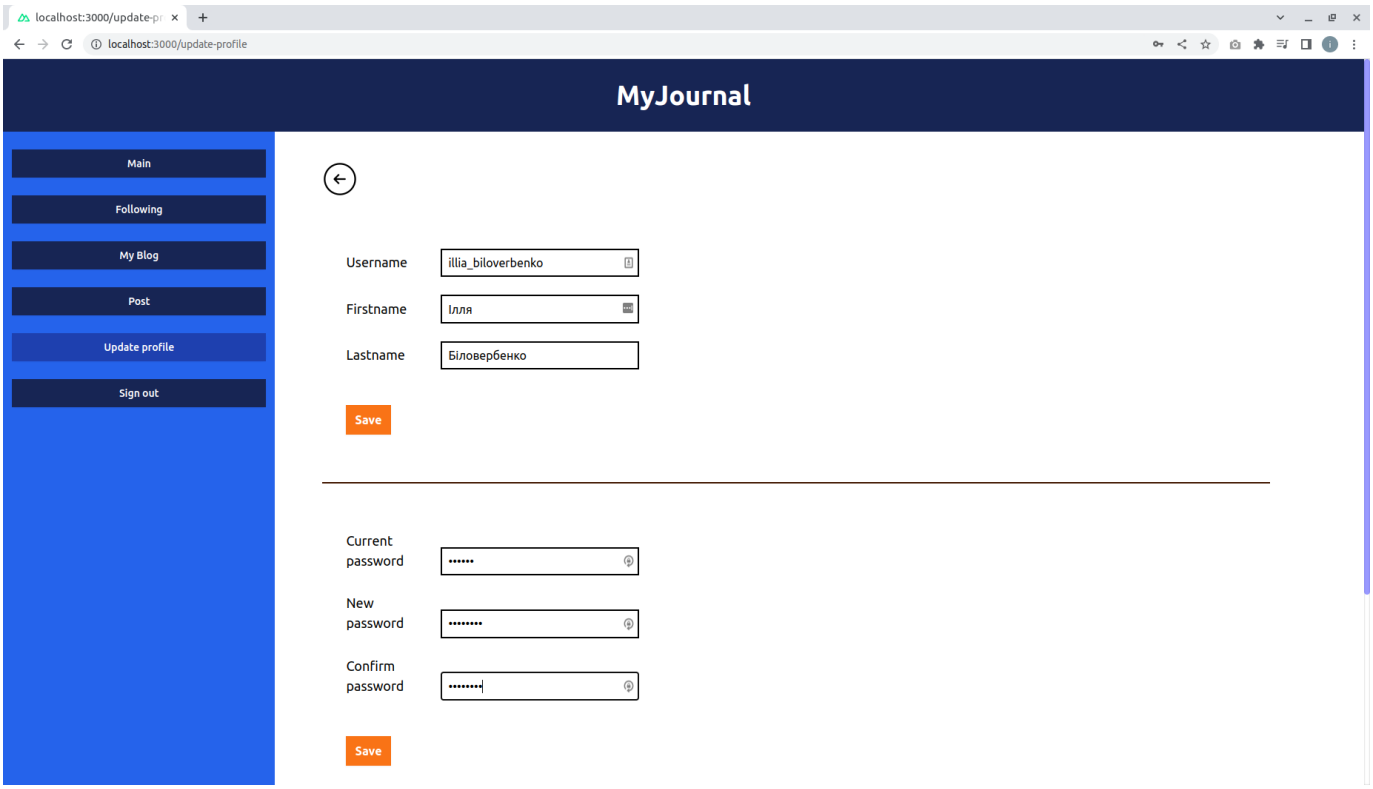

Рисунок 3.6.7 — сторінка зміни даних користувача

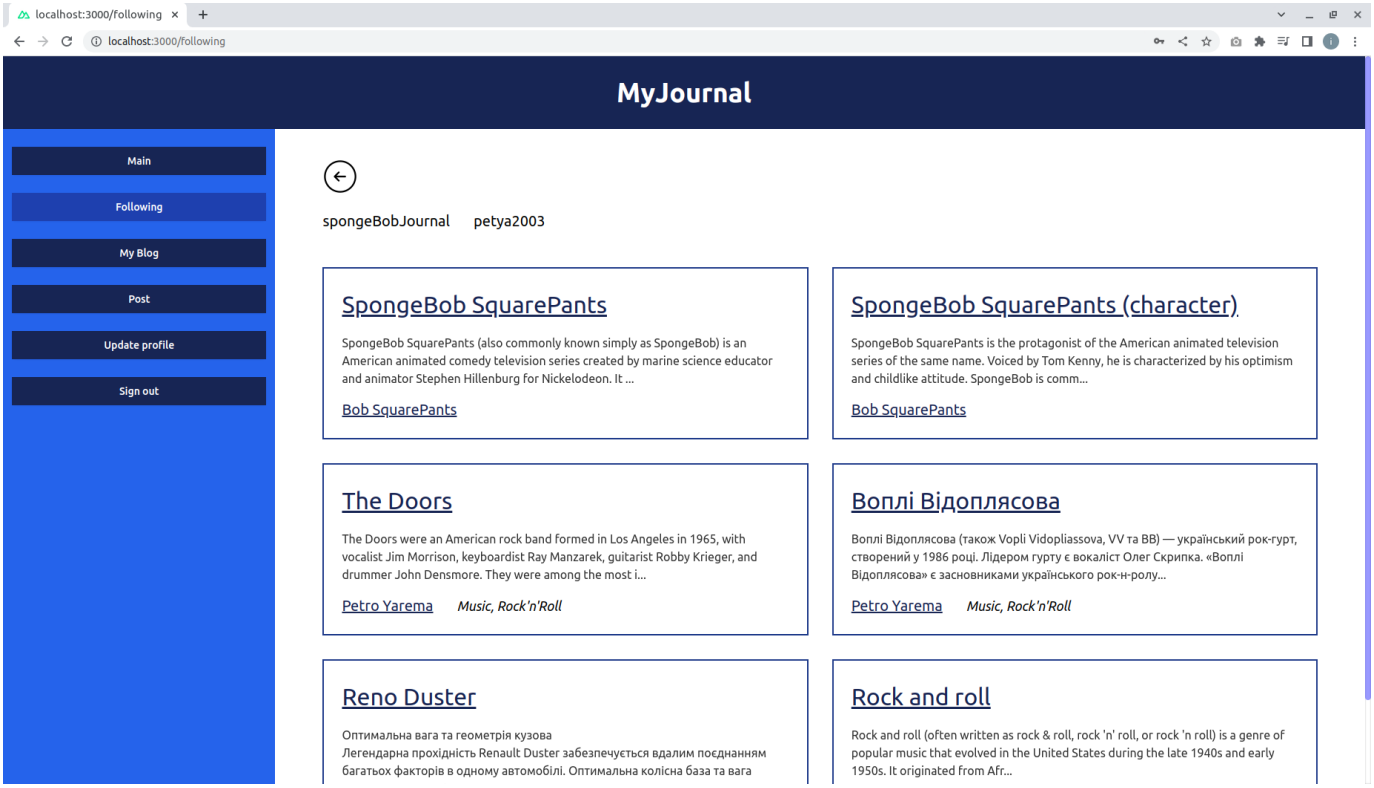

Рисунок 3.6.8 — сторінка відслідковуваних користувачів (зверху - їх горизонтальний список, знизу - їхні статті)

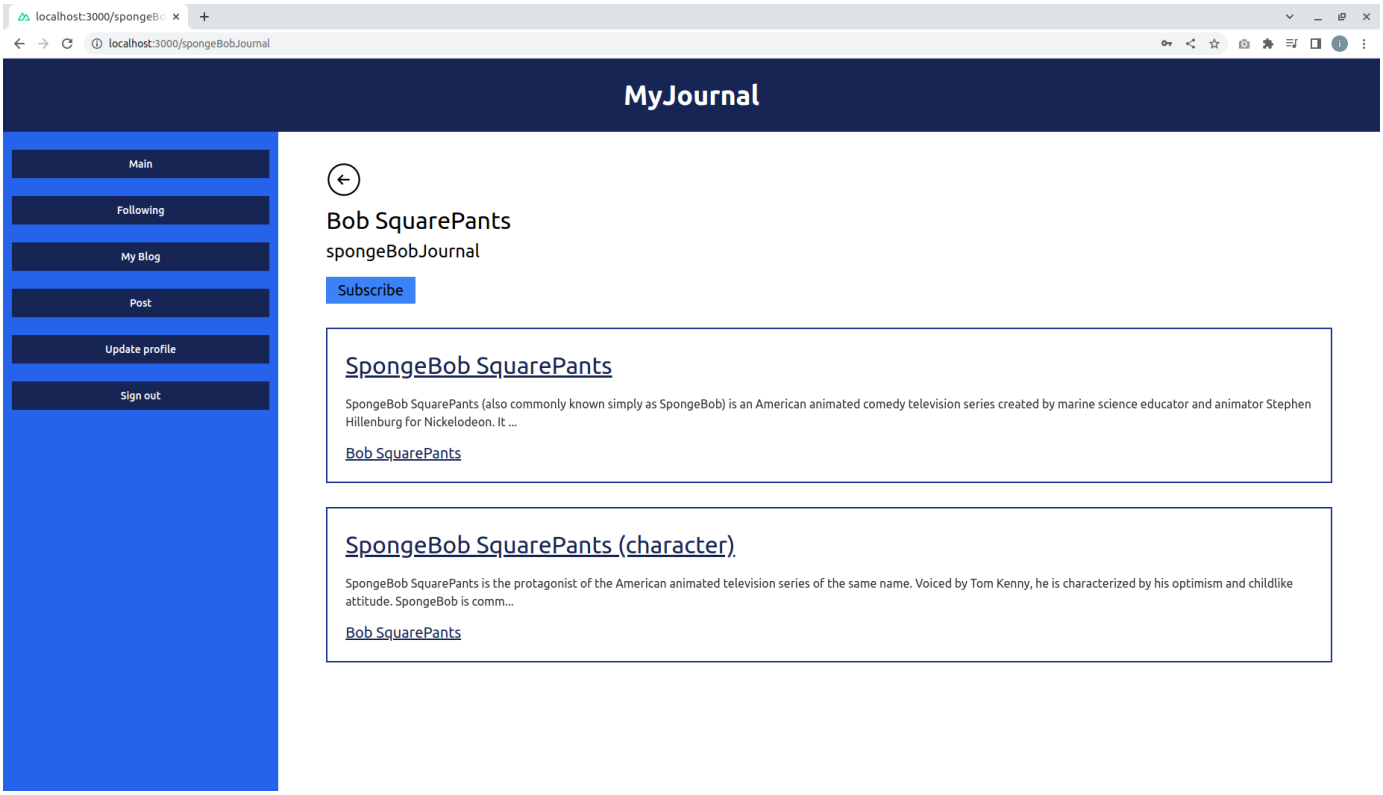

Рисунок 3.6.9 — сторінка профілю конкретного користувача

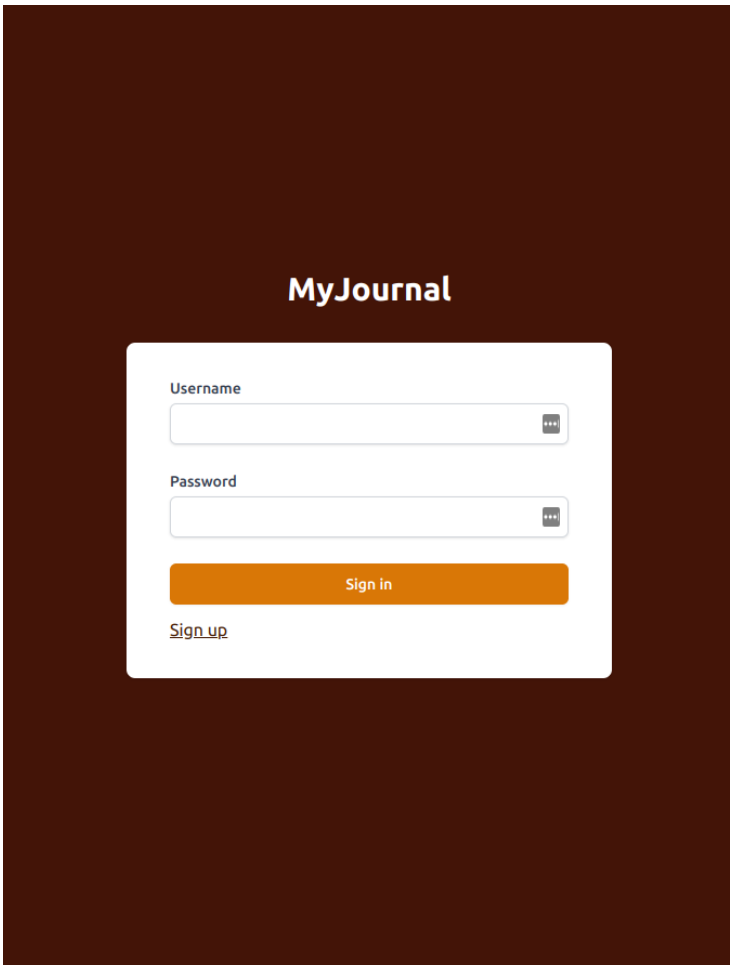

Рисунок 3.6.10 — сторінка автентифікації користувача на девайсі з шириною екрану 650 пікселів

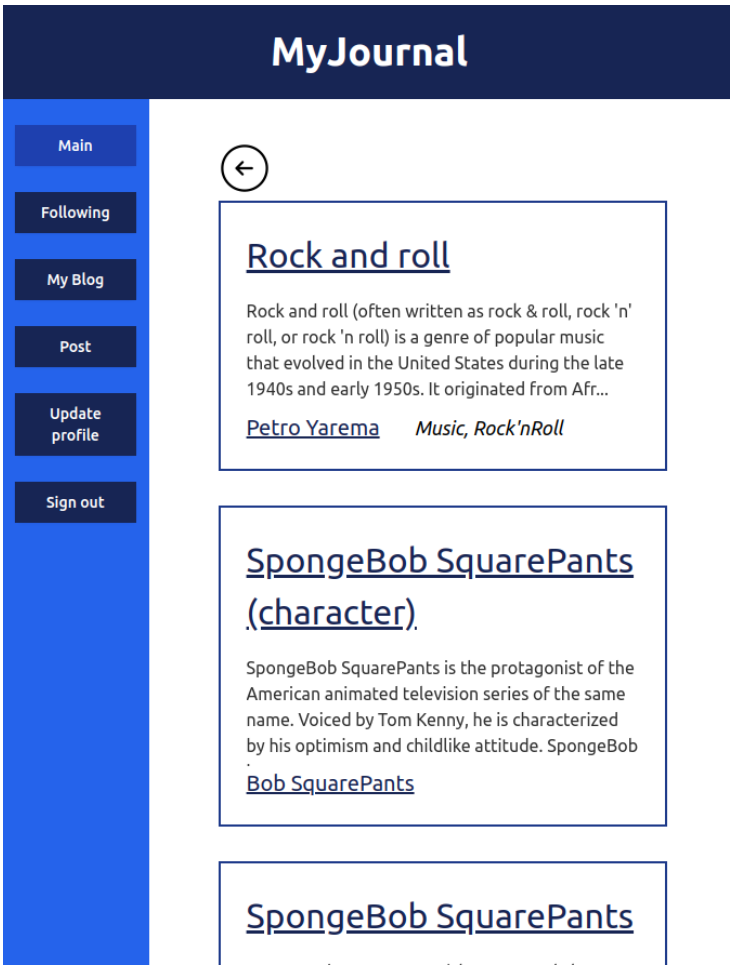

Рисунок 3.6.11 — головна сторінка на девайсі з шириною екрану 650 пікселів

Висновки

У цій курсовій роботі було детально розглянуто процес створення веб-застосунків з використанням Vue (Nuxt 3) та Node.js. Зокрема, розглянуто основні елементи створення веб-застосунків, такі як структурування проекту, роботу з директоріями та роутингом як у серверній, так і клієнтській частині застосунку.

Також було досліджено роботу з базою даних MongoDB та ORM Mongoose. Розглянуто структуру сутностей і взаємозв'язки між ними; CRUD-операції, які можна виконувати з об’єктами у розглянутій базі даних.

Розглянуто процес автентифікації та авторизації користувачів з використанням Passport.js та JSON Web Tokens. Описано та реалізовано процес створення, валідації та зберігання токенів JWT, а також розроблено проміжне програмне забезпечення для перевірки прав доступу користувачів.

Також розглянуто процес роботи з HTTP-запитами з використанням Axios та роботу з фронтендом за допомогою фреймворків Vue.js і Nuxt.js.

Досліджено структуру та функціональні можливості компонентів, проміжного програмного забезпечення та layout-ів у Nuxt.js; доцільність використання фреймворків Vue та Nuxt 3 для розробки веб-застосунків.

Серед переваг розглянутих фреймворків можна виділити:

- Простоту - Vue та Nuxt 3 мають легкий для вивчення синтаксис, що дозволяє створювати додатки, не маючи глибоких знань чи попереднього досвіду в цій галузі.
- Компонентний підхід - Vue має компонентний підхід до розробки, який дозволяє розділити код на окремі компоненти. Таким чином досягається краща організація, масштабованість та підтримка коду.
- Швидкість, адже Vue та Nuxt 3 — це сучасні фреймворки, робота над

вдосконаленням яких все ще триває; вони мають оптимізований код та генерують статичні файли для забезпечення швидкості завантаження файлів та в цілому роботи веб-застосунку.

Цілей щодо функціональності застосунку досягнуто.

Загалом, ця курсова робота дозволила отримати багато знань та практичних навичок у створенні веб-застосунків, дослідити основні елементи розробки, такі як структурування проекту, роботу з базою даних та автентифікацію користувачів, та зрозуміти те, як вони взаємодіють між собою для створення повноцінного веб-застосунку.

Список використаної літератури:

1. Документація Vue https://vuejs.org/guide/introduction.html
2. Документація Nuxt https://nuxtjs.org/docs/get-started/installation
3. Документація NodeJS https://nodejs.org/docs/latest-v20.x/api/
4. Документація Express https://expressjs.com/en/4x/api.html
5. Документація Tailwind https://tailwindcss.com/docs/installation
6. Документація Passport.js https://www.passportjs.org/docs/
7. Документація Mongoose https://mongoosejs.com/docs/guide.html
8. Документація Postman ручне тестування https://learning.postman.com/docs/sending-requests/requests/
9. Документація Postman автоматизоване тестування https://learning.postman.com/docs/writing-scripts/intro-to-scripts/
10. Стаття про автентифікацію в NodeJS з використанням Passport.js та MongoDB https://www.makeuseof.com/user-authentication-in-nodejs/
11. Форум stackoverflow https://stackoverflow.com/