МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗАСОБАМИ ТЕХНОЛОГІЙ JAVA SCRIPT»**

Виконав: студент 2 курсу, групи 8.1212-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
В.А. Доценко
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н., Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Доценку Владиславу Андрійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційної системи засобами технологій Java Script

керівник роботи Кривохата Анастасія Григорівна, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 30.11.2023 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка інформаційної системи засобами технологій Java Script.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 03.05.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	17.05.2023	
2.	Збір вихідних даних.	07.06.2023	
3.	Обробка методичних та теоретичних	28.06.2023	
	джерел.		
4.	Розробка першого та другого розділу.	27.08.2023	
5.	Розробка третього розділу.	31.10.2023	
6.	Оформлення та нормоконтроль	27.11.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	14.12.2023	

Студент _____ В.А. Доценко
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Кривохата
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка інформаційної системи засобами технологій Java Script»: 56 с., 38 рис., 5 табл., 11 джерел.

БЕКЕНД, ІНФОРМАЦІЙНА СИСТЕМА, КОРИСТУВАЧ, ПРЕЦЕДЕНТ, ТЕСТУВАННЯ, ФРОНТЕНД, FIGMA, JAVASCRIPT, MONGODB, MYSQL NODE.

Об’єкт дослідження – процес розробки інформаційної системи.

Мета роботи: розробка інформаційної системи засобами технологій Java Script.

Метод дослідження – аналіз, проєктування, вивчення та узагальнення, реалізація.

У магістерській кваліфікаційній роботі досліджуються методи проєктування та реалізації інформаційної системи за допомогою технологій мови програмування JavaScript. У результаті роботи було зроблено аналіз основних вимог до інформаційної системи. За допомогою інструменту Figma було створено шаблон зовнішнього вигляду головної сторінки вебдодатку. Зокрема, були розроблені діаграми, для облегшення процесу проєктування системи, а саме: прецедентів, послідовностей, класів, компонентів та розгортання.

Крім того, була реалізована функціонуюча інформаційна система «Aquapeak» із використанням середовища виконання JavaScript Node.js та бібліотеки React.js, що були використані для ефективної реалізації серверної та клієнтської частини додатку. У якості бази даних, була використана NoSQL база даних MongoDB. Реалізована інформаційна система повністю відповідає усім заданим при проєктуванні вимогам, із привабливим та зрозумілим для користувача інтерфейсом та повним функціоналом додатку для адміністраторів.

SUMMARY

Master’s qualifying paper «Development of the Information System using Java Script Technologies»: 56 pages, 38 figures, 5 tables, 11 references.

BACKEND, INFORMATION SYSTEM, USER, PRECEDENT, FRONTEND, FIGMA, JAVASCRIPT, MONGODB, MYSQL, NODE.

Object of the study – the process of developing of the information system.

Aim of the study: development of the information system using Javascript technologies.

Methods of research – analysis, design, study and generalization, implementation.

The master’s thesis delves into the methods of designing and implementing an information system using JavaScript programming language. The work involved analyzing the basic requirements for the information system and comparing them to other development methods. A template for the appearance of the main page of the web application was created using the Figma tool. Diagrams such as precedents, sequences, classes, components, and deployment were developed to facilitate the system design process.

The «Aquapeak» information system is functioning well and was developed using the Node.js JavaScript runtime and the React.js library. These tools were used to effectively implement the server and client parts of the application. A NoSQL database, MongoDB, was used to manage the database. The implemented information system meets all the specified design requirements and includes an attractive and easy-to-use user interface, as well as full application functionality for administrators.

ЗМІСТ

ВСТУП

На сьогодні потреба в інтернет ресурсах дуже стрімко збільшується. Кожна компанія, навчальний заклад, ресторан, автосалон, магазин хоче мати свій сайт в інтернеті оскільки це забезпечує збільшення кількості відвідувачів та попит на товар або послуги.

Головними критеріями будь-якого сайту є: багатофункціональність, зручність та гарне та приємне оформлення. Ці критерії можна реалізувати використовуючи різні мови програмування при написанні сайту. Найчастіше замовник вказує програмісту певну програмну мову для реалізації та певну базу даних в якій будуть зберігатись дані.

Головною метою роботи є реалізація інформаційної системи засобами JavaScript. Для повноцінної та вірної реалізації інформаційної системи, повинні бути виконані певні дії та умови. Від аналізу та проєктування проєкту, розробки діаграм та схеми бази даних, до процесу розробки та використання бібліотек мови JavaScript для покращення коду та додатку в цілому.

Основні завдання дослідження:

- ознайомлення із іншими прикладами реалізації подібних інформаційних систем;
- огляд інструментів та бібліотек JavaScript, що наддадуть допомогу у розробці інформаційної системи;
- розробити макет вигляду інформаційної системи;
- спроєктувати структуру інформаційної системи за допомогою діаграм та схем;
- розробити інформаційну систему за допомогою спроєктованої структури та інструментів розробки JavaScript;
- провести тестування системи та виправити усі помилки знайдені при тестуванні.

Робота включає у себе три розділи. Перший розділ присвячений огляду

бібліотек та інших інструментів мови програмування JavaScript, що можуть облегшити чи покращити розробку інформаційної системи. Другий розділ містить інформацію про процес розробки системи, діаграми, схеми даних та класів, а також аналіз вимог до системи. У третьому розділі наведено програмну реалізацію системи, продемонстровано використання бібліотек мови програмування JavaScript для створення компонентів сайту. Окрім цього, у третьому розділі проведено тестування додатку та детальний огляд розробленої інформаційної системи.

1 ІНСТРУМЕНТИ РОЗРОБКИ САЙТУ

1.1 ReactJS як інструмент розробки

Питання розробки інформаційних систем та додатків залишається досить актуальним та важливим у сучасному світі з багатьох причин. В наш час всі сучасні компанії та власники бізнесу намагаються розширити свій бізнес за допомогою впровадження інформаційних систем. Оскільки це дуже полегшує та пришвидшує надання інформації про продукцію або послуги та задоволення потреб клієнтів в онлайн-комунікації. Окрім цього, з кожним роком розширюється інтернет аудиторія, а для власників бізнесу це потенційні клієнти. У цей час, онлайн-покупки та інтернет-торгівля мають пік своєї популярності серед людей. Разом із постійним розвитком мобільних технологій, користувачі все більше використовують мобільні телефони для доступу до вебдодатків, тому вебсайти повинні бути адаптивними до різних пристроїв та розмірів екрану, що надає можливість розширити круг потенційних користувачів та клієнтів [11].

JavaScript відіграє ключову роль у розробці вебдодатків і є однією з тих мов програмування, що були створені та підточені для розробки сайтів. Звісно окрім нього є і інші мови, що використовуються у розробці. Наприклад Python чи Java, або PHP, котрий так само як і JavaScript повністю створений задля розробки вебдодатків. Але завдяки фреймворкам та бібліотекам, мова програмування JavaScript постійно вдосконалюється та має велику базу інструментів, щоб надавати можливість розробляти якісну та захищену серверну та клієнтську частини сайту [8].

Для розробки фронтенду сайту було обрано JavaScript бібліотеку для створення користувацьких інтерфейсів React. React.js розроблений компанією Facebook. Цей інструмент добре підходить для використання у проєкту, бо відповідає усім вимогам, що задані на початку розробки, бо є ефективним при розробці масштабованих, швидких та модульних інтерфейсів користувача.

Центральним елементом бібліотеки React є його компонентна архітектура, яка дозволяє розробляти програми шляхом створення безлічі незалежних компонентів, кожен з яких є частиною інтерфейсу користувача. Компоненти React можуть бути використані повторно та легко підтримуватись завдяки своїй ізольованій природі [1].

1.2 Node.js як інструмент розробки

Окрім розробки клієнтської частини сайту дуже великою частиною роботи є розробка серверної частини. Як і для клієнтської частини, основною вимогою до роботи є використання мови JavaScript. При розробці бекенду з використанням цієї мови програмування, частіше за все використовується середовище виконання JavaScript Node.js. Вона орієнтована на серверну розробку, яка дозволяє розробникам створювати масштабовані та ефективні програми. Він заснований на движку JavaScript V8 від Google Chrome і надає можливість виконувати JavaScript на сервері поза браузером. Node.js працює асинхронно і не блокує виконання коду при очікуванні операцій вводу-виводу. Це дозволяє обробляти безліч запитів паралельно, покращуючи продуктивність програми, що є досить важливим пунктом у розробці. Node.js заохочує модульність коду, дозволяючи розробникам створювати безліч маленьких модулів та повторно використовувати їх у своїх додатках [2].

Пакетний менеджер npm забезпечує доступ до бібліотек та інструментів, спрощуючи процес розробки. Він є одним з основних інструментів при роботі із Node.js і надає доступ до багатьох бібліотек, модулів та інструментів для розробки. Він дозволяє інтегрувати бібліотеки та модулі у проєкт. Використання цього інструменту є необхідним для правильного та розширеного використання Node.js [2].

Окрім цього, є бібліотеки та фреймворки, що дозволяють поліпшати роботу з кодом. Прикладом такого є фреймворк Express.js. Express це дуже

популярний вебфреймворком, написаним на JavaScript і працюючим усередині середовища виконання node.js. Ві призначений для розробки вебдодатків та API. Express надає безліч вбудованих функцій для обробки маршрутів, запитів, відповідей, обробки помилок та багато іншого, що робить процес створення вебдодатків у Node.js більш простим та зручним. Окрім цього, сам фреймворк має декілька плагенів для роботи із базою даних, реєстрацією та авторизацією. Наприклад «express-fileupload» він використовується для обробки та загрузки файлів з використанням HTTP-запитів [3].

У подальшому в роботі буде використано модуль Node.js «dotenv», що надає можливість завантажувати змінні оточення з файлу .env у додаток. Зазвичай у Node.js проєктах секретні ключі та всі паролі до бази даних і уся інша конфіденційна інформація зберігаються у змінних оточення для безпеки. Модуль спрощує керування цими змінними оточення, дозволяючи завантажувати їх із файлу .env у процес програми.

Більш детального огляду потребує і бібліотека хешування паролів для Node.js. Бібліотека «bcrypt» надає розробнику методи для хешування паролей користувача, що забезпечує безпеку при роботі та зберіганні паролей у базі даних. У бібліотеці використовується одностороння функція хешування, що створює із заданого паролю хеш, що неможливо розшифрувати [10].

Для того, щоб мати спробу автоматично перезапускати додаток після внесення змін у файлову систему, використовується інструмент «nodemon». Він досить сильно прискорює розробку, та автоматизує процес перезапуску додатку, що дуже економить час для розробника, бо йому не доводиться кожен раз перезапускати програму вручну після кожного внесення змін. Він особливо добре підходить при розробці додатків та скриптів Node.js, API та додатків Express.js [9].

Увесь процес тестування інформаційної системи проходитиме за допомогою бібліотеки Node.js «supertest». Це бібліотека, що надає зручний інтерфейс для написання тестів запитів до серверу або API. Тобто за допомогою бібліотеки, можна проводити як unit-тестування, так і прості тести отримання

відповідей з серверу, щоб переконатись в тому, що сервер коректним чином відповідає на них. Також, ця бібліотека надає можливість емулювати HTTP запити не використовуючи реального серверу.

Окрім цієї бібліотеки, для проведення тестування, використовується фреймворк для тестування JavaScript коду в проєктах на Node.js, включаючи тестування серверної та клієнтської частини [4]. Фреймворк «Jest» надає легкий синтаксис написання тестів, що дуже прискорює та спрощує їх написання. Він включає в себе інструменти для автоматичної заміни модулів, генерації звітів про покриття коду, підтримки асинхронного тестування і багато інших корисних функцій, що не потребують додаткового налаштування. «Jest» має оптимізований та швидкий процес запуску тестів, що робить його ефективним інструментом для контролю якості коду [5].

1.3 Система управління базами даних

Дуже важливим пунктом у розробці інформаційної системи є вибір бази даних як інструменту розробки. MongoDB і MySQL це різні системи управління базами даних, що мають різні підходи до зберігання, організації та обробки даних. Вибір між ними залежить від конкретних вимог проєкту та особливостей програми. Для проєкту, що буде розроблено у кваліфікаційній роботі, було обрано базу даних MongoDB через те, що вона має гнучку схему даних. А також, що важливо на перших порах розробки та тестування, MongoDB дозволяє швидко розпочати роботу з даними без необхідності створення жорсткої схеми. Це може бути корисним у початковій стадії розробки, коли потрібно експериментувати з даними та швидко вносити зміни [7].

MongoDB це високопродуктивна, гнучка та масштабована система управління базами даних, орієнтована на документи, яка дозволяє зберігати та організовувати дані у форматі JSON-подібних документів. Ця база даних входить до класу NoSQL і відрізняється від традиційних реляційних баз даних, таких як

MySQL та інших. Завдяки своїй архітектурі MongoDB забезпечує високу швидкість доступу до даних, що є досить важливим. Вона може ефективно обробляти великі обсяги даних та операції читання, або запису [7].

Також, для MongoDB існує бібліотека для Node.js, яка полегшує роботу з MongoDB через об’єктно-документне відображення. Бібліотека називається «Mongoose», надає зручний спосіб моделювання даних для MongoDB, дозволяючи розробникам створювати схеми даних, виконувати запити та взаємодіяти з базою даних MongoDB, використовуючи JavaScript-об'єкти. Mongoose надає можливість створювати схеми даних, які визначають структуру колекції MongoDB. Це дозволяє задавати типи даних, встановлювати обмеження, індекси та інші правила зберігання даних. Окрім цього, вона дозволяє використовувати хукі та інше програмне забезпечення для, наприклад, збереження файлу [7].

1.4 Опис предметної області

Інформаційна система «Aquapeak» це платформа, що була розроблена та в подальшому буде використовуватись для продажу морської продукції у форматі онлайн магазину. Цей додаток надає можливість користувачу продивлятись товар, розміщений продавцем на сайті, додавати його до свого електронного кошику, для подальшої оплати та залишати відгуки про якість товару.

Основні аспекти предметної області:

- товар: до нього відноситься уся інформація пов’язана із товаром (назва, опис, вигляд товару, ціна товару);
- користувачі: сюди входить уся інформація та дії пов’язані з користувачами інформаційної системи (реєстрація, авторизація, особистий кабінет, відгуки);
- віртуальний кошик та оформлення заказу: можливість додавання товару до віртуального кошику, видалення товару з нього, оформлення заказу та

оплата заказу;

– панель адміністратора: сюди входить увесь функціонал, що має адміністратор сайту (це панель адміністратора для управління товаром, користувачами, заказами, новинами сайту та відгуками).

2 ПРОЄКТУВАННЯ ВЕБДОДАТКУ

2.1 Аналіз вимог

Аналіз вимог це один з головних етапів розробки вебдодатку. Цей етап визначає основні характеристики та функціональні можливості, якій буде мати кінцевий продукт.

У цьому розділі кваліфікаційної роботи представлено розгорнутий аналіз основних вимог до інформаційної системи, яка включає в себе: можливість реєстрації та подальшої авторизації користувачів, забезпечення адаптивності інтерфейсу, створення привабливого візуального оформлення та проведення тестування функціональності.

Можливість реєстрації та авторизації користувача є важливим пунктом для будь-якого вебдодатку, особливо коли додаток є інтернет магазином, чи так чи інакше взаємодіє з оплатою та товаром. Вебдодаток «Aquapeak» повинен мати спосіб реєстрації через заповнення реєстраційної форми, автентифікацію, та зберігання інформації про користувачів у базі даних. Окрім цього, персональна інформація користувача повинна бути захищеною.

Вимога до адаптивності інтерфейсу передбачає, що вебдодаток повинен коректно відображатись на усіх пристроях і екранах, включаючи ноутбуки, планшети та смартфони. Тому потрібно, ретельно розробити дизайн, що буде зручним для користувачів. Впровадити технологічні рішення, щоб забезпечити оптимальну взаємодію та сприйняття контенту незалежно від пристрою, на якому він відображається.

Візуальне оформлення це досить важливий аспект для інформаційної системи. Система повинна мати гарне візуальне оформлення для створення естетично привабливого, зрозумілого та узгодженого зовнішнього вигляду. Для досягнення цього результату, необхідно підібрати кольори, компонування елементів інтерфейсу і загальний стиль дизайну. За цими вимогами було

розроблено шаблон майбутнього вебдодатку «Aquapeak» за допомогою інструменту Figma [6]. Цей макет є досить простим та зручним для орієнтації користувача у додатку. Приклад розробленого макету головної сторінки сайту можна побачити на рисунку 2.1.

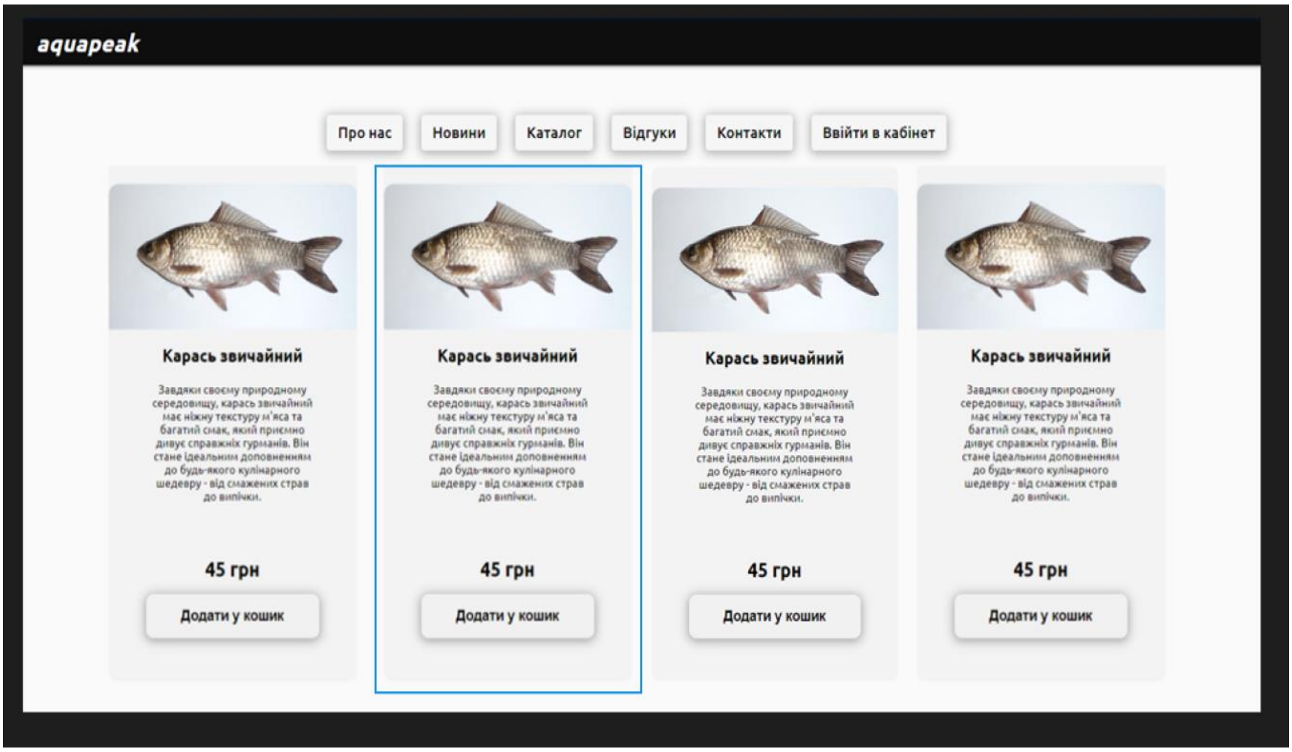

Рисунок 2.1 – Макет головної сторінки сайту «Aquapeak»

На етапі проєктування системи, потрібно передбачити план основних вимог до інформаційної системи. Такі вимоги, як можливість реєстрації, авторизації, адаптивність додатку, тестування додатку на багатьох етапах реалізації, потрібно проєктувати ще до початку реалізації. Це дозволяє розробити ефективний, функціональний та привабливий інформаційний ресурс, що сприяє успішній взаємодії з користувачем. Окрім цього, треба визначити список ролей, що буде створено на сайті, тобто типів користувачів системи.

У проєкті буде створено два типи користувачів, а саме: адміністратор та користувач. Звичайний користувач буде мати стандартний функціонал, тобто зможе продивлятись новини, товар, додавати його у кошик та проводити оформлення заказу, після оплати та отримання якого, зможе залишити відгук.

Адміністратор в свою чергу окрім функціоналу звичайного користувача матиме доступ до редагування, додавання чи видалення даних товару, користувачів чи новин.

2.2 Діаграма прецедентів та діаграма послідовностей

У цій частині роботи представлено детальне концептуальне проєктування для вебсайту «Aquapeak». Концептуальне проєктування це важливий етап у розробці будь-якого вебдодатку. Під час проєктування, проходить створення основних структурних та функціональних аспектів вебдодатку.

Для початку необхідно визначити основні цілі та цільову аудиторію сайту. Основною метою проєкту є створення онлайн-платформи для продажу свіжих морських продуктів із зручним та надійним інтерфейсом користувача, використовуючи інструменти та технології розробки мови програмування JavaScript.

Цільовою аудиторією сайту є любителі морепродуктів, ресторани та всі, хто цінує високоякісні морепродукти. В цілому будь-який користувач мережі Інтернет може зайти до сайту «Aquapeak» та після проходження реєстрації та авторизації зробити замовлення на сайті.

Перед тим, як переходити до розробки інформаційної системи, необхідно проаналізувати та розробити план проєкту. Для зручного аналізу та візуалізації різних ролей у системі, та того, як вони взаємодіють між собою та функціоналом, буде використана діаграма прецедентів (див. рис. 2.2).

Вебдодаток повинен надавати можливість Інтернет користувачу при потраплянні на сайт мати можливість зареєструватися. Авторизований користувач Інтернет магазину повинен мати можливість робити замовлення товарів онлайн та надавати можливість вибору варіантів отримання товарів: доставка, самовивіз. Клієнт магазину за допомогою браузеру отримує доступ до каталогу товарів, новин сайту та статусу свого замовлення.

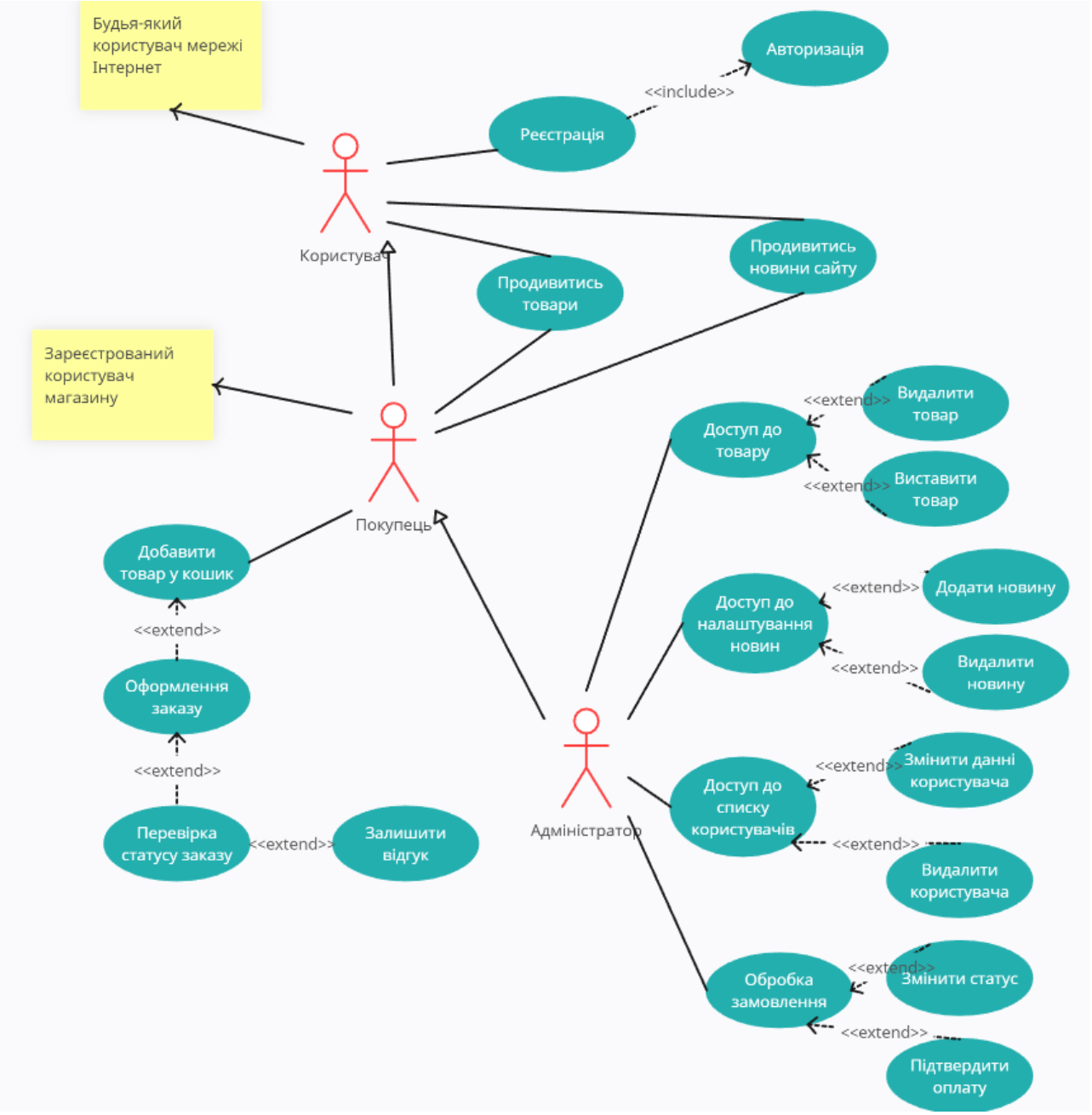

Рисунок 2.2 – Діаграма прецедентів сайту

Коли авторизований клієнт обирає якийсь з товарів, цей товар потрапляє до його віртуального кошику, звідки клієнт по своєму бажанню може добавляти або видаляти товари. Після завершення вибору товарів клієнт має можливість завершити оформлення заказу.

У якості акторів діаграми прецедентів обрані три типи користувача сайту: незареєстрований користувач «Користувач», авторизований користувач «Покупець» та адміністратор сайту «Адміністратор».

Як можна побачити на діаграмі (див. рис. 2.2), адміністратор сайту має найбільш розширені права доступу на сайті, та може робити будь-які дії та навіть ті, що може робити авторизований або незареєстрований користувач. Водночас, звичайні користувачі сайту не мають доступу до функціоналу адміністратору і не можуть: змінювати, додавати, видаляти товар, користувачів чи новини сайту. Тому, між трьома акторами показано відношення узагальнення.

Відношення асоціації на діаграмі показані між актором, що має роль «Адміністратор» та прецедентами «Доступ до товару», «Доступ до налаштування новин», «Доступ до списку користувачів» та «Обробка замовлення», бо тільки в адміністратору сайту є доступ до цього функціоналу.

Окрім діаграми прецедентів, для аналізу послідовностей дій між об’єктами та компонентами у певному часовому порядку під час певного процесу, була створена діаграма послідовностей. Вона допоможе краще зрозуміти та дослідити процес авторизації користувача на сайті (див. рис. 2.3). Завдяки цій діаграмі можна побачити візуально увесь процес авторизації користувача, з моменту натискання кнопки «Авторизуватися» і до моменту підтвердження авторизації та отримання доступу до даних авторизованого користувача.

Після того, як клієнт вводить у форму свою поштову адресу, пароль у поля вводу на фронтенді та натискає кнопку авторизації, усі дані потрапляють до клієнтської частини.

На клієнтській частині є функція, що реагує на натиск кнопки та передає данні далі на сервер, формуючи POST-запит. Сервер Node.js у свою чергу отримує цей запит із усіма даними, що ввів користувач.

Далі у серверній частині проходить перевірка даних, пароль та поштова адреса порівнюються з тими, що існують в базі даних.

Якщо дані співпадають, процес авторизації підтверджується та дані повертаються до клієнтської частини, та створюється сесія, а користувач отримує доступ до свого особистого кабінету.

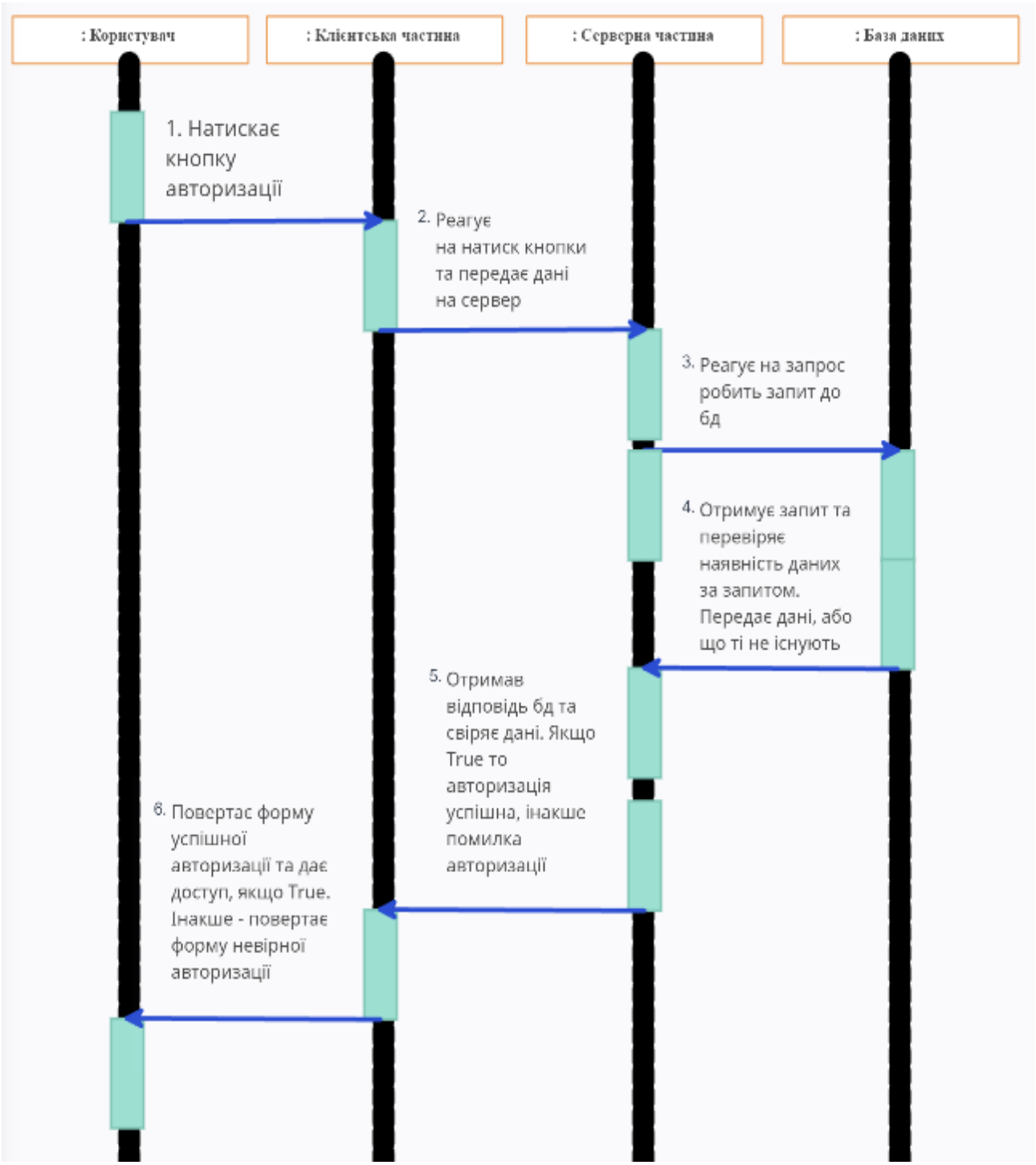

Рисунок 2.3 – Діаграма послідовностей для процесу авторизації

2.3 Діаграма класів та схема бази даних

Для будь-якого вебдодатку, де є потрібність працювати із великою кількістю різних масивів даних, важливою частиною є моделювання та

подальша розробка бази даних. Перед тим як перейти до процесу розробки бази даних, треба її спроєктувати, для того, щоб мати візуальне розуміння її структури.

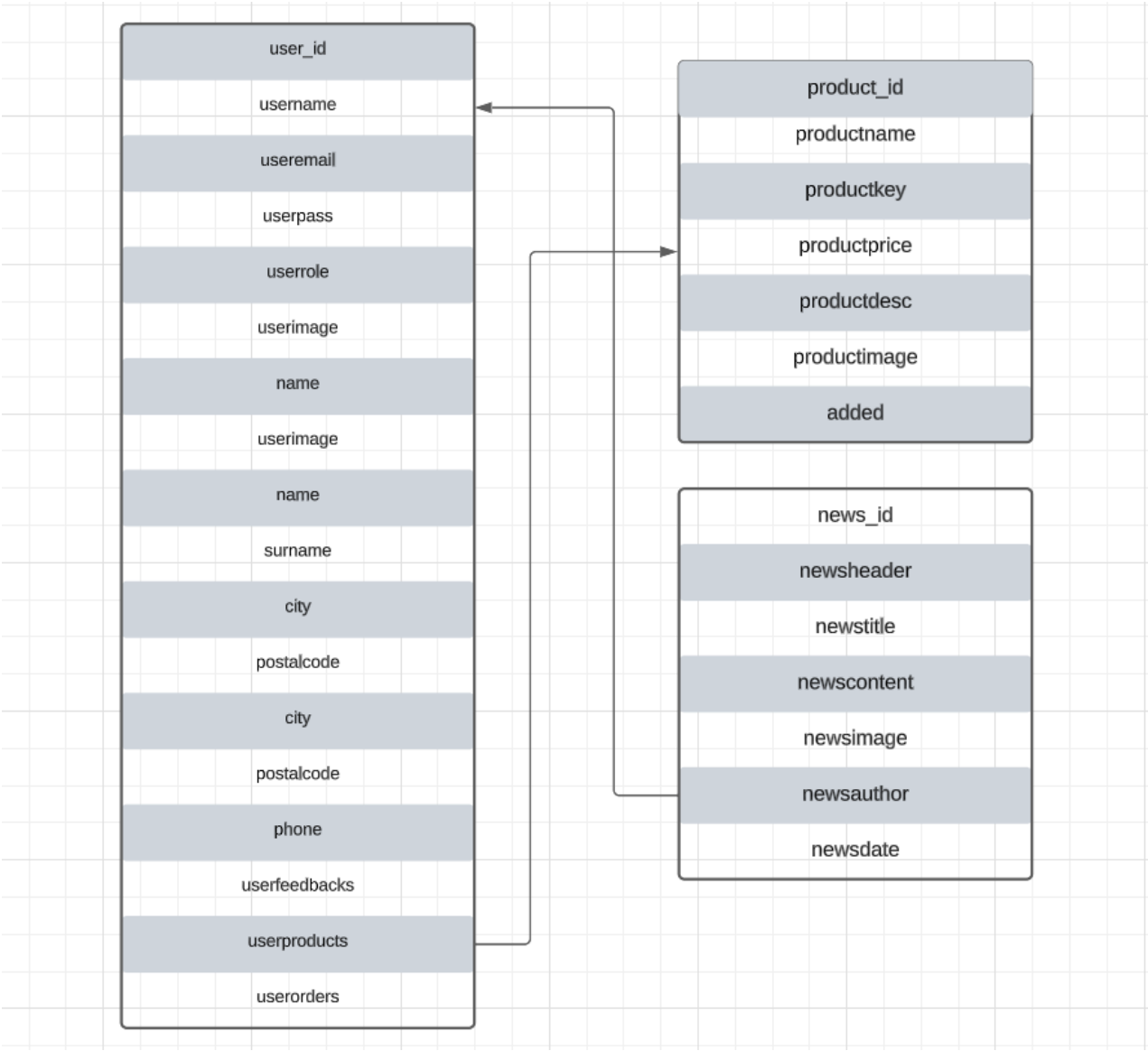

Рисунок 2.4 – Схема структури бази даних сайту «Aquapeak»

Для створення бази даних сайту «Aquapeak» буде використано NoSQL базу даних MongoDB через її зручність, гнучкість та багатофункціональність. Виходячи з розробленої схеми бази даних сайту (див. рис. 2.4), можна виділити три основні таблиці бази даних.

Перша таблиця утримує усі дані пов’язані із користувачами та їх персональними даними. Таблиця користувачів «User», має досить велику

кількість атрибутів, а саме такі поля як: унікальний ідентифікатор «user_id», нікнейм або логін користувача «username», поштова адреса користувача «useremail», пароль користувача у хешованому вигляді «userpass», функціональна роль користувача, тобто адміністратор чи звичайний користувач «userrole», картинка що використовується у особистому профілі користувача «userimage», ім'я користувача «name», прізвище користувача «surname», місто проживання користувача «city», поштовий індекс міста користувача «postalcode», номер телефону користувача «phone», відгук користувача «userfeedbacks», список товарів, що були доданими користувачем до кошику «userproducts» та усі зроблені користувачем на сайті закази «useroders».

Друга таблиця утримує усі дані пов'язані із товаром на сайті. Таблиця товарів «Product», має такі атрибути: унікальний ідентифікатор «product_id», назва продукту «productname», виставлена адміністратором ціна продукту «productprice», детальний опис продукту «productdesc», картинка або фото виставлені для товару «productimage».

Тертя та остання таблиця бази даних сайту буде утримувати інформацію пов'язану з новинами на сайті. Таблиця новин «News» має наступні атрибути: унікальний ідентифікатор «news_id», поле в якому знаходиться коротка назва новини, її швидкий опис «newsheader», поле в якому знаходиться основна назва новини «newstitle», поле в якому знаходиться основний текст новини «newscontent», поле в якому знаходиться картинка або фото що була додана до новини «newsimage», поле в якому знаходиться інформація про автора новини «newsauthor», поле в якому знаходиться інформація про дату випуску новини «newsdate».

Отримав схему бази даних, можна перейти до її створення, але спочатку треба розробити діаграму класів. Це дуже важливий інструмент для розробки, з її допомогою можна візуально побачити структуру класів проєкту та їх взаємодії у системі. У діаграмі відображаються класи, методи цих класів та їх атрибути, а також зв'язок між класами (див. рис. 2.5).

Як можна побачити по діаграмі (див. рис. 2.5), на сайті планується

створення декількох класів, а саме: «Admin», «Product», «Guest», «User», «Cart» та «Payment». Клас «Admin» забезпечує роботу та функціонал для адміністраторів сайту. Цей клас має багато атрибутів, що його описують. Усі атрибути та їх типи даних можна побачити на діаграмі. Окрім атрибутів, клас має методи, або операції класу. Так як клас «Admin» повинен мати найбільший функціонал, він має велику кількість методів. Метод «addProducts» надає адміністратору сайту можливість добавити новий товар до списку товарів, кінцевим результатом виконаного методу, повинен бути створений екземпляр класу «Product» із вказаними адміністратором полями. Метод «deleteProducts» дозволяє при його виконанні знайти певний товар із списку товарів та видалити його.

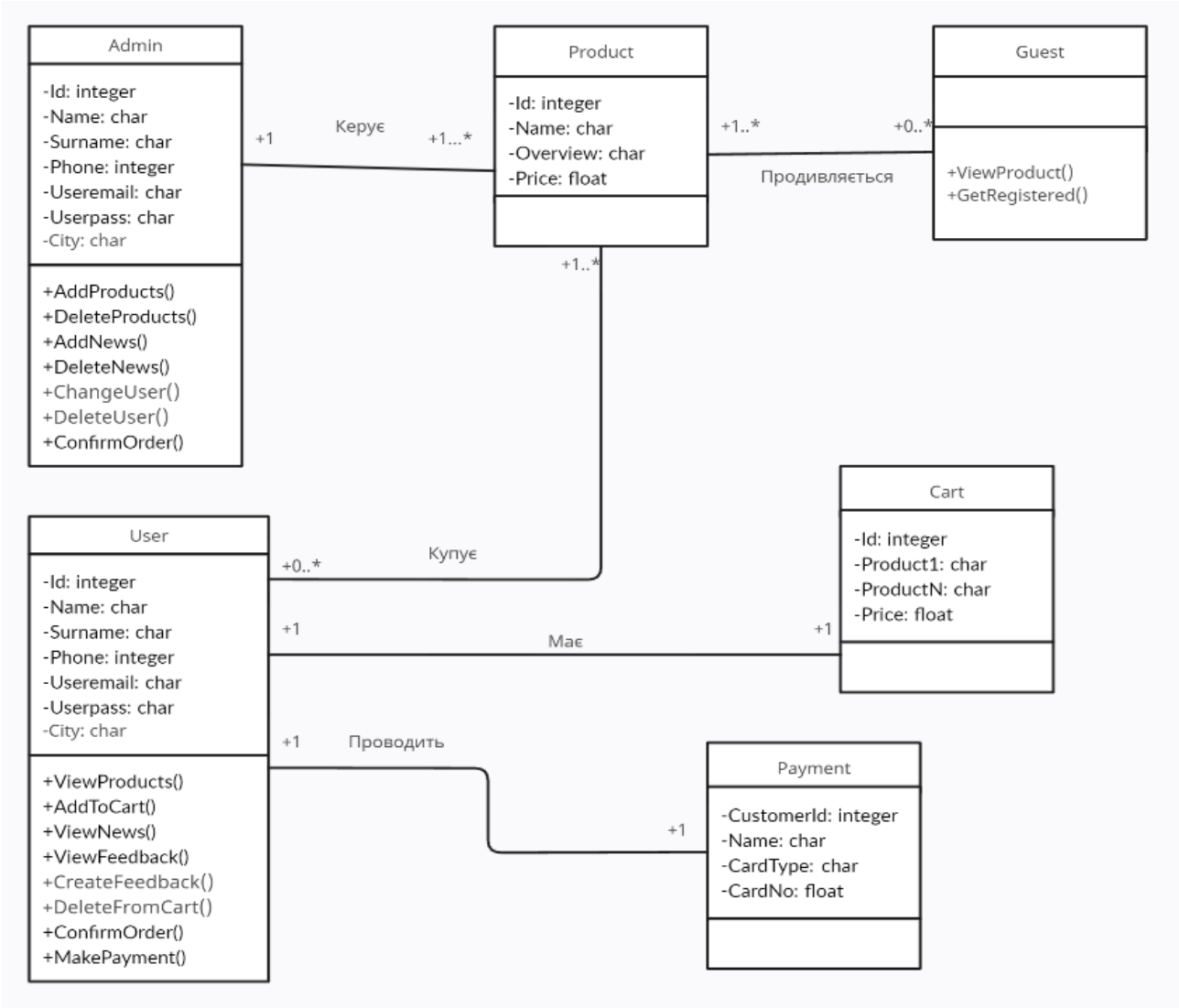

Рисунок 2.5 – Діаграма класів

Клас «addNews» так само як і клас «addProducts» дозволяє додати новину до списку новин. Клас, що надає можливість адміністратору змінити дані користувача сайту називається «changeUser», а для видалення якогось певного користувача використовується метод «deleteUser». Для того, щоб підтвердити оплату та змінити статус заказу на сайті, буде використано метод «confirmOrder».

Клас «User» представляє собою функціонал звичайного авторизованого користувача сайту. Так само, як і попередній клас, він має список атрибутів, що задаються для цього класу, та методи для реалізації самого функціоналу. За допомогою своїх методів класу, буде реалізована можливість користувача оглядати товар методом «viewProducts», добавляти товар до кошику методом «addToCart», продивлятись новини, відгуки, та робити відгуки самому, за допомогою методів «viewNews», «viewFeedback» та «createFeedback». При роботі з кошиком, користувач отримує можливість видалити товар з кошику, за допомогою методу «deleteFromCart», підтвердити оформлення заказу, та провести оплату, за допомогою методів «ConfirmOrder» та «makePayment».

Клас «Guest» має лише стандартний функціонал для незареєстрованого користувача, тобто він може або подивитись товар, або зареєструватись. Класи «User», «Cart» та «Payment» мають в собі атрибути, але не мають ніяких методів, бо використовуються для організації даних.

2.4 Діаграма компонентів та діаграма розгортання

Перед тим як перейти до написання коду сайту, необхідно розробити його архітектуру. Для її візуальної реалізації добре підходить діаграма компонентів. Для того щоб забезпечити роботу та функціональність сайту, використовується велика кількість різних елементів, що взаємодіють між собою. Основні з них це клієнтський додаток та серверний додаток і їх компоненти та фреймворки, база даних, платіжні засоби та інші сервіси. Більш детально їх взаємодія демонструється на діаграмі компонентів (див. рис. 2.6).

Основні компоненти вебсайту «Aquapeak» це: клієнтський React.js додаток та серверний Node.js додаток, база даних сайту та платіжний шлюз. Наданий інтерфейс згідно діаграми представляє спосіб згідно якому, інші компоненти клієнтський та серверний додатки використовують функціонал цього компоненту або обмінюються даними.

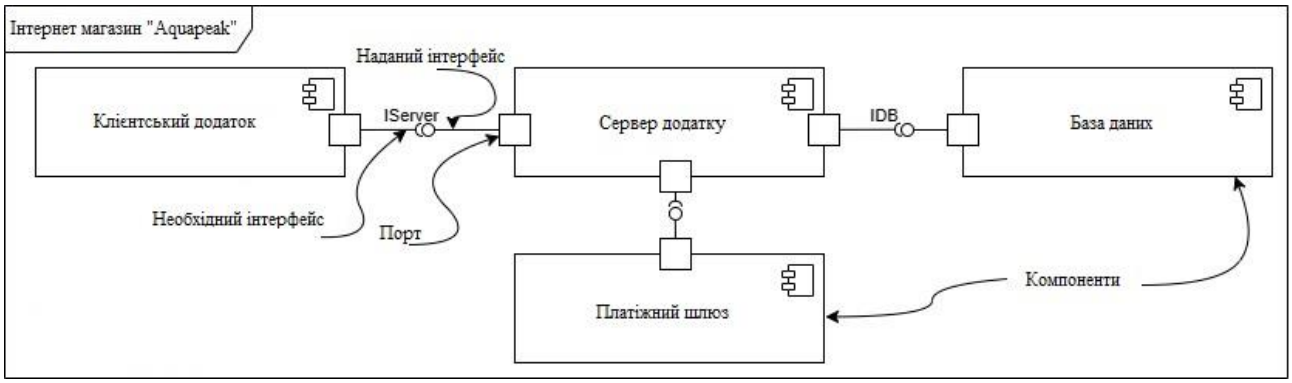

Рисунок 2.6 – Діаграма компонентів

Таким чином обидва компоненти можуть функціонувати та обмінюватись даними для подальшої роботи з ними та обробки запитів. За допомогою порту представляється точка входу або виходу компоненту, з її допомогою компонент може взаємодіяти з іншими компонентами. Через нього компонент отримує можливість за допомогою інтерфейсу визивати різні методи, або передавати повідомлення.

На діаграмі ми можемо візуально побачити взаємодію клієнтської частини сайту із серверною частиною. Далі при необхідності серверна частина сайту робить запит до бази даних, щоб отримати інформацію, що в ній зберігається та використати її надалі у своїх методах. Прикладом для діаграми можна привести процес оформлення на сайті замовлення та подальшої його оплати за допомогою платіжного шлюзу.

Окрім діаграми компонентів, діаграма розгортання добре відображає фізичну архітектуру та відносини між компонентами, що забезпечують повноцінну роботу сайту. Також за допомогою цієї діаграми можна проілюструвати процес того, як елементі розгортаються на апаратних

компонентах (див. рис. 2.7). Як можна побачити на діаграмі, є декілька основних компонентів сайту, що взаємодіють між собою та іншими додатками.

Перший компонент це клієнтський компонент, користувач використовує веббраузер для того, щоб користуватись вебдодатком. За допомогою веббраузеру відображається клієнтський інтерфейс. Код, що виконується на клієнтський частині організовує зовнішній вигляд сайту. Також, надсилає запити на сервер для отримання даних.

Другий компонент це серверний, він зв’язаний із клієнтським інтерфейсом та базою даних. На ньому проходить обробка запитів надісланих до системи від клієнтського інтерфейсу та передача їх до серверного додатку. Окрім цього проходить відправка отриманих даних у відповідь на запити клієнтської частини. Інша частина компоненту це сам додаток, що проводить усю логіку та основний функціонал, а також забезпечує роботу із базою даних.

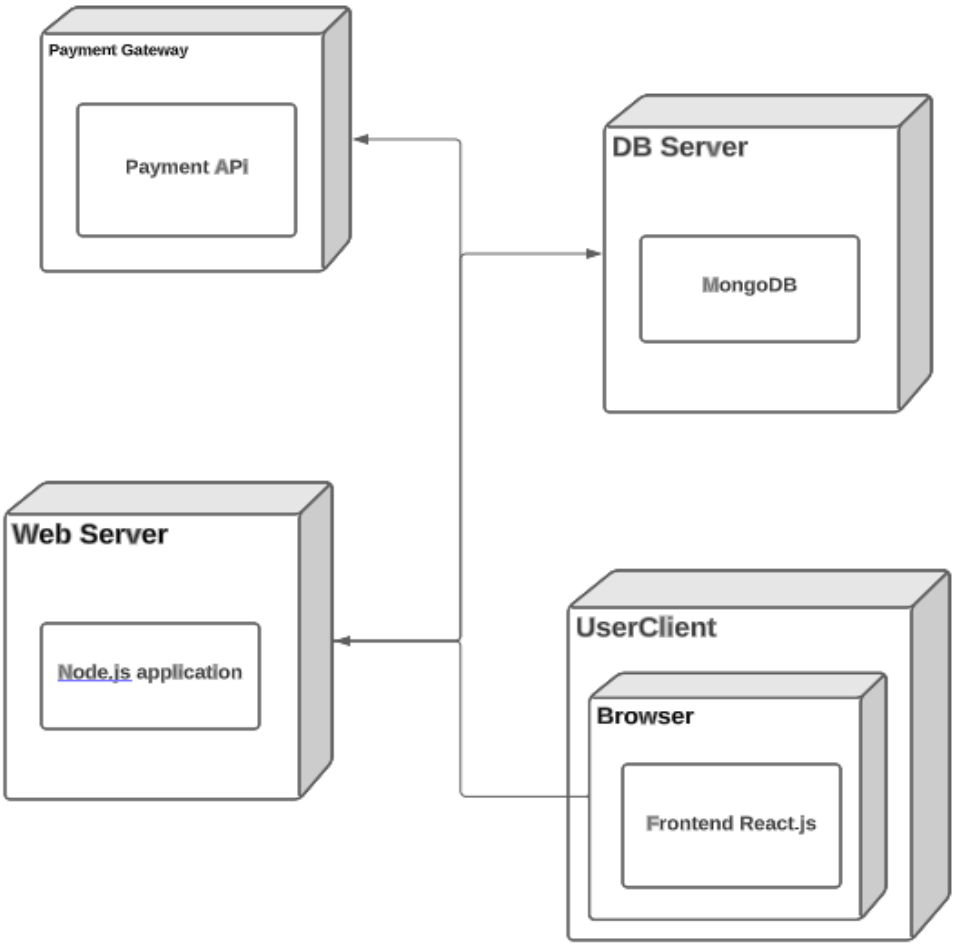

Рисунок 2.7 – Діаграма розгортання

Третій компонент системи це база даних. У цьому компоненті зберігаються дані о замовленнях, товарах та інша інформація, пов’язана із клієнтами. База даних пов’язана із серверною частиною. Таким чином, вебсервер робить запити до бази даних для того щоб отримати дані, змінити чи видалити їх. Останньою з компонентів, що можна побачити на діаграмі є платіжна система. Вона оброблює усі платежі та транзакції зроблені на сайті. Вебсервер надсилає запити на оплату, після чого платіжний шлюз оброблює запит та проводить усі взаємодії із банківськими системами для здійснення транзакції. У свою чергу, мережева інфраструктура забезпечує увесь зв’язок між компонентами.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Реалізація інформаційної системи

Після закінчення підготовки та створення діаграм та схем, згідно планів йде перехід до розробки вебдодатку «Aquapeak». Одними з головних вимог до проєкту були швидкість, масштабованість та комфортний та зрозумілий дизайн сайту. У якості серверного середовища виконання було обрано Node.js через те, що він добре підходить для виконання заданих вимог. Та позволяє створювати швидкі та масштабовані вебдодатки. Для розробки клієнтської частини та створення інтерфейсу вебдодатку, було обрано бібліотеку React.js. Після детального ознайомлення із спроєктованою схемою даних, було обрано документоорієнтовану систему управління базами даних, NoSQL-систему MongoDB.

Серверна частина інформаційної системи створена завдяки Node.js із використанням Express.js. Express.js використовується для роботи та управління запитами, а також для обробки маршрутів. Вебсервер у свою чергу забезпечує співпрацю із клієнткою частиною та базою даних. Надає обробку запитів клієнтської сторони та зворотну передачу даних клієнту.

Клієнтська частина створена за допомогою додатку React надає зручну можливість створювати компоненти, що допомагають реалізовувати повторне використання коду, що надало можливість розробити динамічний інтерфейс користувача із використанням компонентного підходу.

База даних MongoDB надає можливість реалізувати зберігання даних та швидко отримувати доступ з серверної частини сайту по запиту. Створення самої бази даних та таблиць спрощене, та достатньо швидко реалізоване за допомогою зручного візуального інтерфейсу користувача бази даних. З його допомогою було створено три таблиці з даними: «news», «products», «users» (див. рис. 3.1). Окрім створених таблиць, для них були створені відповідні поля згідно схеми бази даних, продемонстрованої на рисунку 2.4.

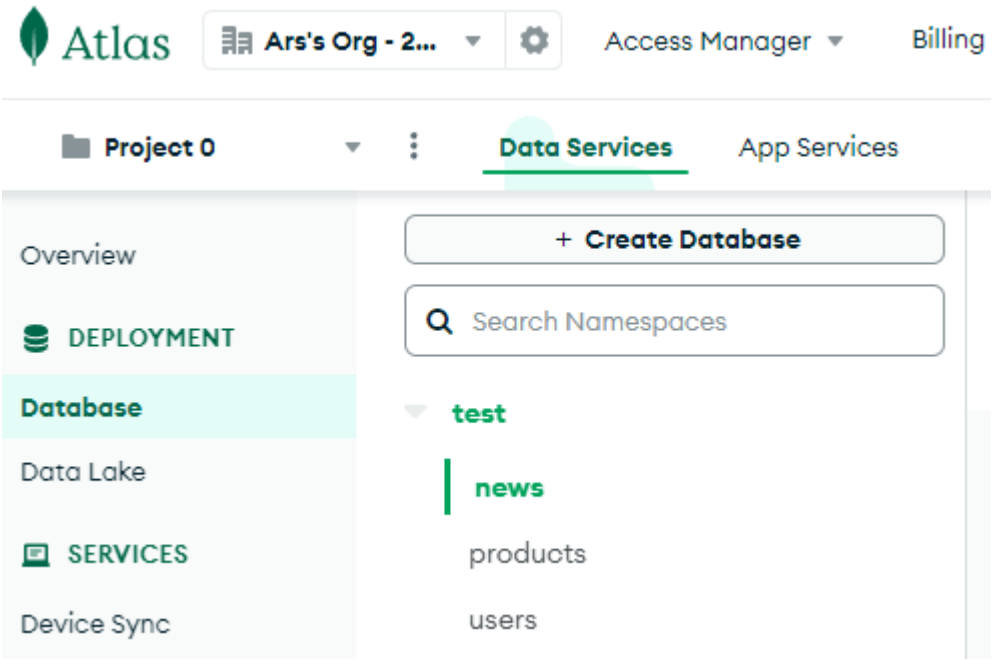

Рисунок 3.1 – Створені таблиці бази даних на MongoDB

Приклад першого тестового заповнення таблиці «Users», поля якої були заповнені згідно даних користувача із роллю адміністратора можна побачити на рисунку 3.2. Усі поля таблиці були внесені згідно схеми бази даних, розробленої у другому розділі. Під час розробки бази даних, та подальшої роботи з нею, були випадки, коли зустрічалась проблема отримання доступу до даних якоїсь з таблиць, або до самої бази даних.

Після ретельного дослідження проблеми, причина такої помилки була знайдена. Причиною була динамічна IP-адреса, що час від часу змінювалась під час розробки, що і видавало у результаті помилку із отриманням доступу до додатку. За замовчуванням база даних у MongoDB має певний список білих IP-адресів які отримують доступ до взаємодії з додатком під час розробки та роботи.

Основний варіант вирішення цієї проблеми це виставлення сайту на хостинг із постійним IP-адресом, це вирішило би проблему на фінальному процесі розробки додатку, щоб користувачі могли стабільно отримувати доступ до контенту сайту.

На момент розробки, було обрано інший варіант вирішення проблеми. Для отримання постійного доступу до сайту під час розробки, було використано

функціонал MongoDB для оновлення списку IP-адрес, що можуть отримати доступ до взаємодії з базою даних та сайту.

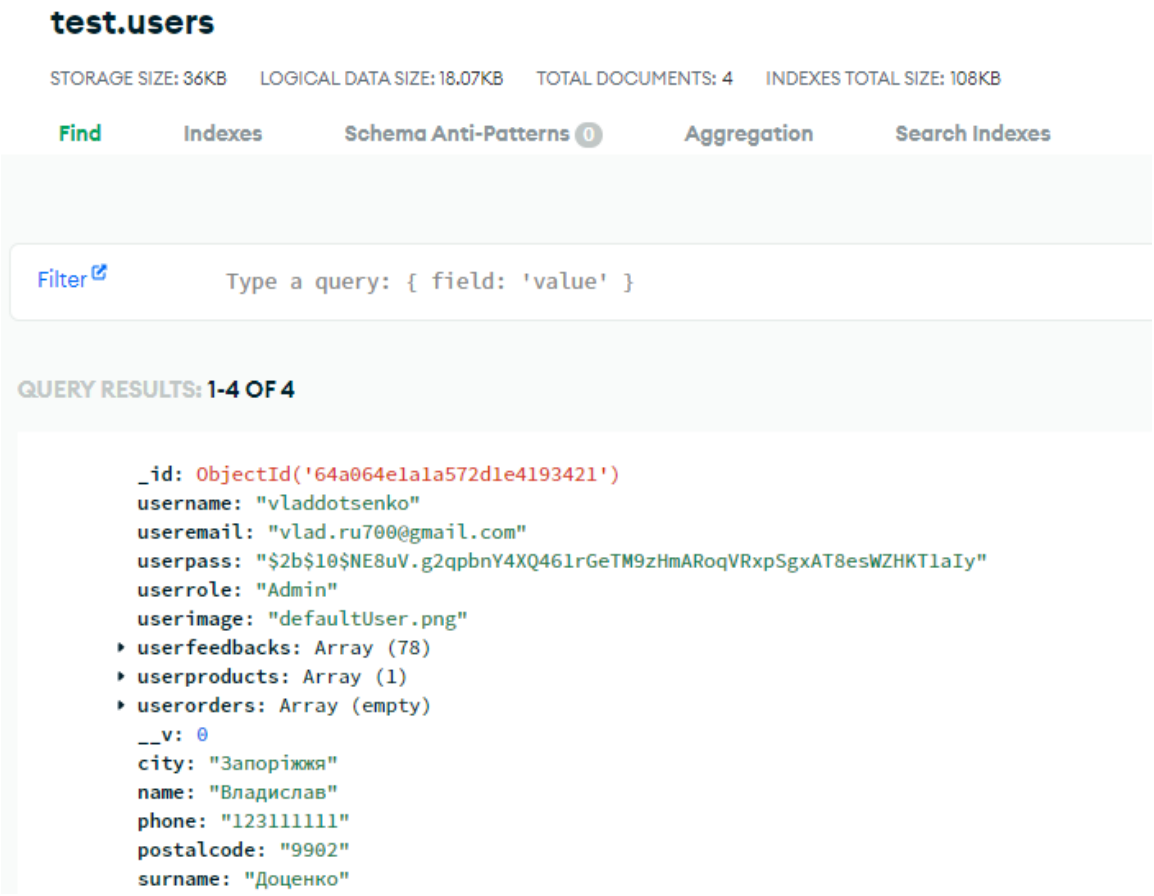

Рисунок 3.2 – Таблиця «Users» зі створеним користувачем ролі «Admin»

Під час розробки інформаційної системи було створено комфортний інтерфейс користувача із роллю адміністратора сайту. Коли не авторизований адміністратор відкриває сайт, він отримує форму сайту для звичайного користувача. Але після того, як він натискає на кнопку авторизації, та авторизується користувачем із роллю «Admin», він одразу отримує форму згідно нової ролі користувача. Надалі, статус користувача перевіряється системою, та якщо при перевірці отримується результат, що він не авторизований, то сайт більше не надає доступ до даних та функціоналу адміністратору сайту. Для того, щоб побачити повний приклад функціоналу взаємодії користувача із своїм профілем сайту, перейдемо до файлу «Profile.jsx». Побачити частину коду клієнтської частини можна на рисунках 3.3 та 3.4.

```
const Profile = () => {

    const [user, setUser] = useState({ userimage: 'default.png', username: '', useremail:
    const [orders, setOrders] = useState([]);
    const [modalActive, setModalActive] = useState(false);

    const handleLoadData = async () => {
        setLoading(true);

        let userdata = JSON.parse(localStorage.getItem('users'));
        setUser(userdata);

        const data = { 'Userdata': userdata };
```

Рисунок 3.3 – Код клієнтської частини профілю користувача

UseEffect() – це хук React.js, що використовується для виконання різних ефектів, що не є пов’язаними на пряму із візуалізацією компонентів, він запускається при прогрузці сторінки, навіть на нуль відсотків. Також, він запускає функцію handleLoadData(), котра виповнює загрузку користувача та усіх його даних.

```
        setOrders(jsonData.orders.userorders);
        setLoading(false);
    }

    const handleLogout = () => {
        localStorage.setItem('login', false);
        localStorage.removeItem('user');
        setStatus(prev => !prev);
    }

    useEffect(() => {
        handleLoadData();
    }, []);
```

Рисунок 3.4 – Функція завантаження користувача handleLoadData()

Далі проходить перевірка «statusLogin == true», якщо результат перевірки позитивний, то запускається функція для загрузки даних. Якщо «statusLogin == false», то користувач перенаправляється на сторінку авторизації, або реєстрації, якщо не був зареєстрованим раніше. У разі позитивного результату після загрузки даних, ми отримуємо роль користувача. Якщо роль користувача

«Admin», то користувач отримує доступ до панелі робітників (див. рис. 3.5).

```
<Button name='Особиста інформація' func={() => { setModalActive(prev => !prev); }} />
{
    user.userrole === 'Admin' ?
    <NavLink to='/adminpanel'>Панель робітника</NavLink> : null
}
<NavLink to="/" onClick={handleLogout}>Вийти з кабінету</NavLink>
```

Рисунок 3.5 – Перевірка ролі користувача «Profile.jsx»

Далі переходимо до файлу «Admin.jsx». Тут можна побачити функцію handleLoadData(), яка загружає дані як і у попередньому випадку. Далі проходить перевірка на statusLogin і подальша перевірка ролі користувача. Після чого отримується доступ до основного функціоналу панелі адміністратору, якщо приходить підтвердження ролі.

```
const Admin = () => {
    const [user, setUser] = useState({ userimage: 'default.png', username: '', useremail:
    const [modalNews, setNews] = useState(false);
    const { statusLogin } = useContext(AuthContext);

    const handleLoadData = async () => {
        setLoading(true);
        let usersdata = JSON.parse(localStorage.getItem('users'));
        setUser(usersdata);
        setLoading(false);
    }

    useEffect(() => {
        handleLoadData();
    }, []);
```

Рисунок 3.6 – Клієнтська частина панелі адміністратору

Коли звичайний користувач натискає кнопку авторизації, виконується функція «handleSubmit», приклад реалізації якої можна побачити у файлі «Signin.jsx». Ця функція використовується у формах React для обробки відправлення форми, що була заповнена користувачем. Далі форма отримується у серверній частині додатку та оброблюється за допомогою функції «app.post()» фреймворку Express.js. Функція оброблює запит на маршрут «/user-login», та

приймає об’єкти «res» запит та «req» відповідь, а «req.body» тримає в собі усі дані, що надсилаються клієнтською частиною, тобто пошта та пароль користувача. Далі пошта передається до методу «User.findOne», це метод, що використовується у базі даних для пошуку певного запису у таблиці. Метод використовується за допомогою бібліотеки «Mongoose» в Node.js. Таким чином проходить пошук пошти вказаної користувачем у базі даних, а за допомогою бібліотеки «bcrypt» методу «bcrypt.compare», проходить пошук та порівняння хешованого паролю, що вводив користувач та хешованого паролю у базі даних. Якщо пароль та пошта співпадають, відправляє дані та відповідь з кодом стану 201, що означає, що авторизація пройдена (див. рис. 3.7).

```
return res.status(201).json({
    user: {
        _id: user._id,
        useremail: user.useremail,
        userrole: user.userrole,
        userimage: user.userimage,
        name: user.name,
        surname: user.surname,
```

Рисунок 3.7 – Відправка даних та відповіді із кодом 201

За подібним принципом, розроблена і реєстрація користувача. Але у запиті передаються дані та записуються до бази даних. Процес створення нового товару виконується за допомогою функції «createProduct» у якій використовується об’єкт FormData. Це об’єкт в JavaScript, який використовується для спрощення процесу відправки даних завдяки методу POST запиту. Завдяки методу «formData.append()» об’єкту FormData, створюється набор ключ-значення, а після відправляється за допомогою AJAX на сервер, де оброблюється за допомогою функції «app.post()». Де «req.body» знов отримує на зберігання дані, що надсилаються клієнтською частиною. Для зберігання картинки використовується метод бібліотеки «express-fileupload». Метод «file.mv()» використовується для переміщення картинки у тимчасовому елементі, щоб передати його до бази даних при створенні елементу. Спочатку отримані дані

перевіряються, чи існує подібний товар у базі даних за допомогою методу «Product.findOne». Якщо подібний товар не існує, створюється новий товар із вказаними даними. Він зберігається у базу даних завдяки методу «newProduct.save()», після чого повертається код 201, що свідчить про вдалу операцію (див. рис. 3.8.).

```
    return res.status(400).json({ errors: { message: `Продукт з такою назвою існує!` } });
}

const newProduct = new Product({
    productname: productName,
    productimage: file.name,
    productdesc: productDesc,
    productprice: productPrice,
});
await newProduct.save();
return res.status(201).json({ product: newProduct });
```

Рисунок 3.8 – Створення нового продукту на стороні серверу

Для того, щоб сервер запускався та функціонував була створена функція «start». При визові цієї функції, за допомогою методу «mongoose.connect», із бібліотеки «Mongoose», проходить підключення та встановлення з’єднання з базою даних MongoDB за її адресою. Після чого проводиться перевірка поточного оточення виконання додатку, щоб переконатися , що вона проходить не у тестовому оточенні. Далі за допомогою методу «app.listen()», проходить запуск вебсерверу та прослуховування порту. Таким чином сервер становиться доступним для подальшої роботи (див. рис. 3.9).

```
const start = async () => {
    try {
        await mongoose.connect('mongodb+srv://vladdotsenko:Moqparoloch

        if (process.env.NODE_ENV !== "test") {
            app.listen(PORT, () => console.log(`started: ${PORT}`));
        }
```

Рисунок 3.9 – Код, що відповідає за запуск серверу

3.2 Тестування інформаційної системи

Закінчивши реалізацію коду, продемонстрованого у підрозділі 3.1, треба провести тестування отриманого продукту. Існує дуже велика кількість варіантів для проведення тестування продукту. Приклади мануального тестування можна буде побачити в підрозділі 3.3 кваліфікаційної роботи. У цьому розділі будуть описані приклади автоматизованого тестування. Спочатку, треба визначити набір тест-кейсів. Тест-кейси важливі для контролювання і перевірки того, як додаток буде поводити себе при позитивних та негативних кейсах. Перше, для чого будуть створені тест-кейси – це реєстрація користувача на сайті.

Для того, щоб отримати доступ до оформлення замовлення, користувачу по-перше треба бути зареєстрованим у системі та бути авторизованим. Для отримання позитивного сценарію цього тест кейсу, після проведення тесту, у базі даних повинен бути створений новий користувач із цими даними. Тому, зробимо перший тест-кейс, що буде зображено на таблицях 3.1 та 3.2.

Таблиця 3.1 – Тест-кейс «Реєстрація нового користувача, позитивний сценарій»

Номер дії	**Дія**	**Очікуваний результат**
1	Зайти до сайту «Aquapeak»	Відображається головна сторінка сайту «Aquapeak»
2	Натиснути на кнопку «Ввійти в кабінет»	Користувач перенаправляється на сторінку авторизації «Signin»
3	Натиснути на кнопку «Стати рибкою»	Користувач перенаправляється на сторінку реєстрації «Signup»
4	Заповнити поле «username» даними не існуючого користувача на сайті	Дані введені в поле «username» та відображаються коректним чином

Продовження таблиці 3.1

Номер дії	**Дія**	**Очікуваний результат**
5	Заповнити поле «useremail» даними не існуючого користувача на сайті	Дані введені в поле «useremail» та відображаються коректним чином
6	Заповнити поле з паролем «userpass» валідними даними	Дані введені в поле «userpass» та відображаються коректним чином
7	Заповнити поле з повторним паролем «confpass» валідними даними	Дані введені в поле «confpass» та відображаються коректним чином
8	Натиснути кнопку «Створити акаунт»	Користувача переправлено на сторінку «Signin». Дані нового користувача внесені в базу даних

Також треба створити тест кейс для перевірки негативного сценарію реєстрації користувача. У цьому тест-кейсі буде перевірка на існування цього користувача у системі та базі даних. Якщо такий користувач існує, система повинна повідомити про це та запропонувати пройти авторизацію користувача, згідно таблиці 3.2.

Таблиця 3.2 – Тест-кейс «Реєстрація нового користувача, негативний сценарій, користувач з цими даними вже існує»

Номер дії	**Дія**	**Очікуваний результат**
1	Зайти до сайту «Aquapeak»	Відображається головна сторінка сайту «Aquapeak»

Продовження таблиці 3.2

Номер дії	**Дія**	**Очікуваний результат**
2	Натиснути на кнопку «Ввійти в кабінет»	Користувач перенаправляється на сторінку авторизації «Signin»
3	Натиснути на кнопку «Стати рибкою»	Користувач перенаправляється на сторінку реєстрації «Signup»
4	Заповнити поле «username» даними існуючого користувача на сайті	Дані введені в поле «username» та відображаються коректним чином
5	Заповнити поле «useremail» даними існуючого користувача на сайті	Дані введені в поле «useremail» та відображаються коректним чином
6	Заповнити поле з паролем «userpass» валідними даними	Дані введені в поле «userpass» та відображаються коректним чином
7	Заповнити поле з повторним паролем «confpass» валідними даними	Дані введені в поле «confpass» та відображаються коректним чином
8	Натиснути кнопку «Створити акаунт»	Користувач отримує помилку «Користувач з цими даними вже існує».

Для проведення тестування буде використана бібліотека JavaScript «Supertest». З її допомогою ми отримаємо доволі зручні засоби для проведення тестування у середовищі виконання JavaScript Node.js.

У прикладі коду нижче, приведено приклад створеного тесту за допомогою бібліотеки «supertest». У цьому тестовому прикладі ми симулюємо процес реєстрації користувача згідно текст-кейсу таблиці 3.1 та передаємо функції дані не існуючого користувача для проведення реєстрації. Далі ми перевіряємо статус

код, що поверне функція який повинен дорівнювати 201. Що означає, що код було успішно виконано та новий користувач був зареєстрований у системі та надалі існуватиме у базі даних. Для тест кейсу на рисунку 3.11, також створено кодову реалізацію тесту за допомогою бібліотеки «supertest» (див. рис. 3.11). У цьому тестовому прикладі симулюється процес реєстрації користувача згідно тест-кейсу 3.2 та передаємо функції дані вже зареєстрованого користувача для спроби проведення реєстрації.

```
const request = require('supertest');
const app = require('../index.js');

describe('User-register Endpoints', () => {
    it('User Register Test [passed]', async () => {
        const res = await request(app)
          .post('/user-register')
          .send({
            Username: 'testname1',
            Useremail: 'testemail1@gmail.com',
            Userpassword: 'TestPass123',
            Userrole: 'User'
          });
        expect(res.statusCode).toEqual(201);
        expect(res.body).toHaveProperty('user');
    });
```

Рисунок 3.10 – Фрагмент коду тесту «Реєстрація нового користувача, позитивний сценарій»

Далі ми перевіряємо статус код, що поверне функція який повинен дорівнювати 400. Що означає, що код було успішно виконано, але новий користувач не був створений у системі так як він вже був зареєстрований раніше та існує у базі даних. Результат виконання обох тестів можемо побачити далі на рисунку (див. рис. 3.13).

Обидва тести були виконані згідно тестових кейсів та очікуваних результатів. Окрім перевірок реєстрації, важливо звернути увагу на авторизацію користувача. Перед розробкою тестів авторизації користувача, треба розробити тест-кейси. Для авторизації так само як і для реєстрації буде створено

позитивний та негативний сценарії. Перший тест-кейс створено для позитивного сценарію авторизації користувача, побачити його можна на таблиці 3.3.

```
it('User Register Test [failed][users is already been created]', async () => {
    const res = await request(app)
      .post('/user-register')
      .send({
        Username: 'testname',
        Useremail: 'testemail@gmail.com',
        Userpassword: 'TestPass123',
        Userrole: 'User'
      });
    expect(res.statusCode).toEqual(400);
    expect(res.body).toHaveProperty('error');
  });
```

Рисунок 3.11 – Фрагмент коду тесту «Реєстрація нового користувача, негативний сценарій, користувач з цими даними вже існує»

Таблиця 3.3 – Тест-кейс «Успішна авторизація користувача»

Номер дії	**Дія**	**Очікуваний результат**
1	Зайти до сайту «Aquapeak»	Відображається головна сторінка сайту «Aquapeak»
2	Натиснути на кнопку «Ввійти в кабінет»	Користувач перенаправляється на сторінку авторизації «Signin»
3	Заповнити поле «useremail» валідними даними існуючого користувача на сайті	Дані введені в поле «useremail» та відображаються коректним чином
4	Заповнити поле «userpass» валідними даними існуючого користувача на сайті	Дані введені в поле «userpass» та відображаються коректним чином
5	Натиснути кнопку «Поринути у море»	Користувач перенаправляється на сторінку особистого кабінету

Наступний тест-кейс створений для перевірки негативного сценарію реєстрації користувача. Згідно тест-кейсу при спробі авторизації, вводиться некоректна та неіснуюча поштова адреса, система повинна повідомити про це та адекватно відреагувати на помилку. Приклад тест-кейсу можна побачити у таблиці 3.4.

Таблиця 3.4 – Тест-кейс «Авторизація користувача, негативний сценарій, невірний email»

Номер дії	**Дія**	**Очікуваний результат**
1	Зайти до сайту «Aquapeak»	Відображається головна сторінка сайту «Aquapeak»
2	Натиснути на кнопку «Ввійти в кабінет»	Користувач перенаправляється на сторінку авторизації «Signin»
3	Заповнити поле «useremail» даними не існуючого користувача на сайті	Дані введені в поле «useremail» та відображаються коректним чином
4	Заповнити поле «userpass» валідними даними	Дані введені в поле «userpass» та відображаються коректним чином
5	Натиснути кнопку «Поринути у море»	Користувач отримує помилку «Невірно вказана поштова адреса»

Та останній тест-кейс створений для перевірки негативного сценарію реєстрації користувача. Але згідно тест-кейсу при спробі авторизації, вводиться коректна поштова адреса та некоректний пароль, система повинна повідомити про це та адекватно відреагувати на помилку пов’язану з невірним паролем. Приклад тест-кейсу можна побачити на. таблиці 3.5.

Таблиця 3.5 – Тест-кейс «Авторизація користувача, негативний сценарій, невірний пароль»

Номер дії	**Дія**	**Очікуваний результат**
1	Зайти до сайту «Aquapeak»	Відображається головна сторінка сайту «Aquapeak»
2	Натиснути на кнопку «Ввійти в кабінет»	Користувач перенаправляється на сторінку авторизації «Signin»
3	Заповнити поле «useremail» даними існуючого користувача на сайті	Дані введені в поле «useremail» та відображаються коректним чином
4	Заповнити поле «userpass» не валідними даними	Дані введені в поле «userpass» та відображаються коректним чином
5	Натиснути кнопку «Поринути у море»	Користувач отримує помилку «Невірно вказаний пароль»

Для цих тестових кейсів створено кодову реалізацію за допомогою бібліотеки «supertest» (див. рис. 3.12). Перший тест симулює авторизацію користувача. В функцію передається пошта та пароль користувача, шо вже зареєстрований та існує в базі даних. Після чого, на сайті проходить перевірка даних та авторизація, функція повертає позитивний результат.

Другий та третій тести так само симулюють авторизацію користувача, але дані, що вводяться є некоректними. В другому варіанті вводиться неправильний пароль користувача який існує. На сайті проходить перевірка всіх даних, після звірки, функція повертає негативний результат, бо пароль користувача є невірним. Так само і в третьому тесті, електрона пошта користувача не є валідною, тому перевірка даних повертає негативний результат, та перевірка негативного сценарію закінчується успішно.

```
const request = require('supertest');
const app = require('../index.js');

describe('User-login Endpoints', () => {
    it('User Login Test [passed]', async () => {
        const res = await request(app)
          .post('/user-login')
          .send({
            Useremail: '113index@gmail.com',
            Userpassword: '12345678',
          });
        expect(res.statusCode).toEqual(201);
        expect(res.body).toHaveProperty('user');
    });

    it('User Login Test [failed][password is not correctly]', async () => {
        const res = await request(app)
          .post('/user-login')
          .send({
            Useremail: 'testemail@gmail.com',
            Userpassword: 'TestPass124433',
          });
        expect(res.statusCode).toEqual(400);
        expect(res.body).toHaveProperty('error');
    });

    it('User Login Test [failed][email is not correctly]', async () => {
        const res = await request(app)
          .post('/user-login')
          .send({
            Useremail: 'testema1il@gmail.com',
            Userpassword: 'TestPass123',
          });
        expect(res.statusCode).toEqual(400);
        expect(res.body).toHaveProperty('error');
    });
});
```

Рисунок 3.12 – Фрагмент коду тесту «Авторизація нового користувача»

Для того щоб запустити процес перевірки, виконується команда у терміналі «yarn test». Після виконання команди, запускається два тести, в першому тесті три тесових випадки, у третьому тесті два тестових випадки (див. рис. 3.13). Як можна побачити, усі тести були виконані та пройшли.

```
yarn run v1.22.19
warning ..\..\..\package.json: No license field
$ nodemon index.js
[nodemon] 2.0.15
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Server started on port: 5000
 PASS  __tests__/user_login.test.js (7.152 s)
  User-login Endpoints
    √ User Login Test [passed] (3111 ms)
    √ User Login Test [failed][password is not correctly] (665 ms)
    √ User Login Test [failed][email is not correctly] (139 ms)

 PASS  __tests__/user_register.test.js
  User-register Endpoints
    √ User Register Test [passed] (3451 ms)
    √ User Register Test [failed][users is already been created] (145 ms)
```

Рисунок 3.13 – Результат виконаних тестів

3.3 Огляд готового додатку

Увесь вебдодаток онлайн магазину «Aquapeak» було розподіллено на дві основні частини: клієнтська частина та адміністративна частина (див. рис. 3.14).

Рисунок 3.14 – Розподілені частини сайту «Aquapeak»

Перша частина сайту була розроблена для користувачів онлайн магазину тобто є клієнтською частиною. У цій частині є декілька кнопок для переходів на інші сторінки, а саме: «Про нас», «Новини», «Каталог товарів», «Відгуки», «Контакти» та «Ввійти в кабінет», якщо користувач не був авторизованим на

сайті раніше. Якщо користувач вже проходив реєстрацію, то користувач побачить кнопку «Особистий кабінет».

При переході користувача на сайт «Aquapeak», для нього відкривається сторінка сайту «Каталог», що є головною сторінкою. На цій сторінці користувач може отримати будь-яку інформацію про товар, що було виставлено на продаж адміністрацією, або власником сайту (див. рис. 3.15).

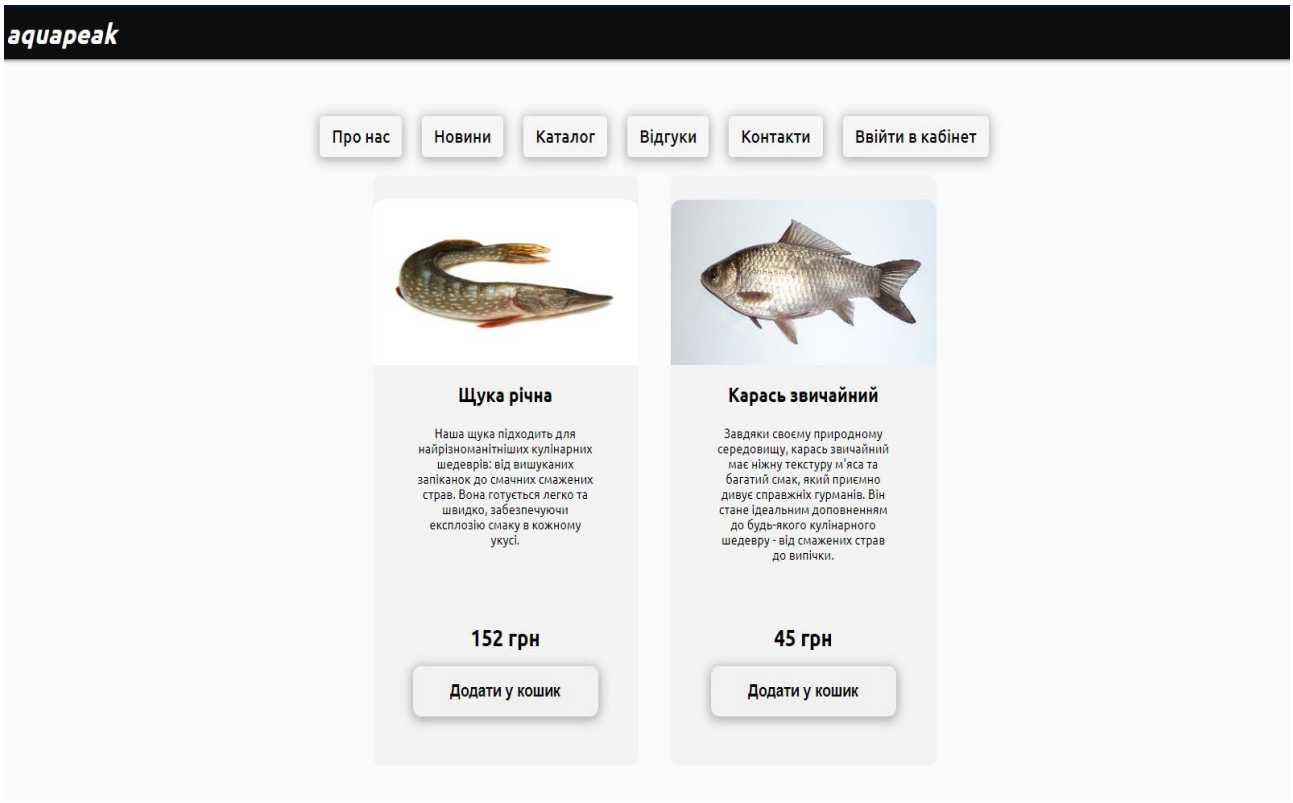

Рисунок 3.15 – Головна сторінка сайту «Aquapeak»

При виборі першої кнопки «Про нас», користувача буде автоматично переправлено на розділ сайту, де він може отримати більше інформації про компанію. Там він зможе прочитати декілька статей «Наша команда», «Про роботу» та «Інформація о товарі», в яких буде детально представлена інформація про партнерів сайту та звідки отримується товар. З цієї сторінки користувач має доступ до інших сторінок (див. рис. 3.16).

При обранні кнопки «Новини», користувач перенаправляється на нову сторінку, де йому представляється інформація та усі новини, що були

опубліковані адміністратором або власником сайту та внесені в базу даних.

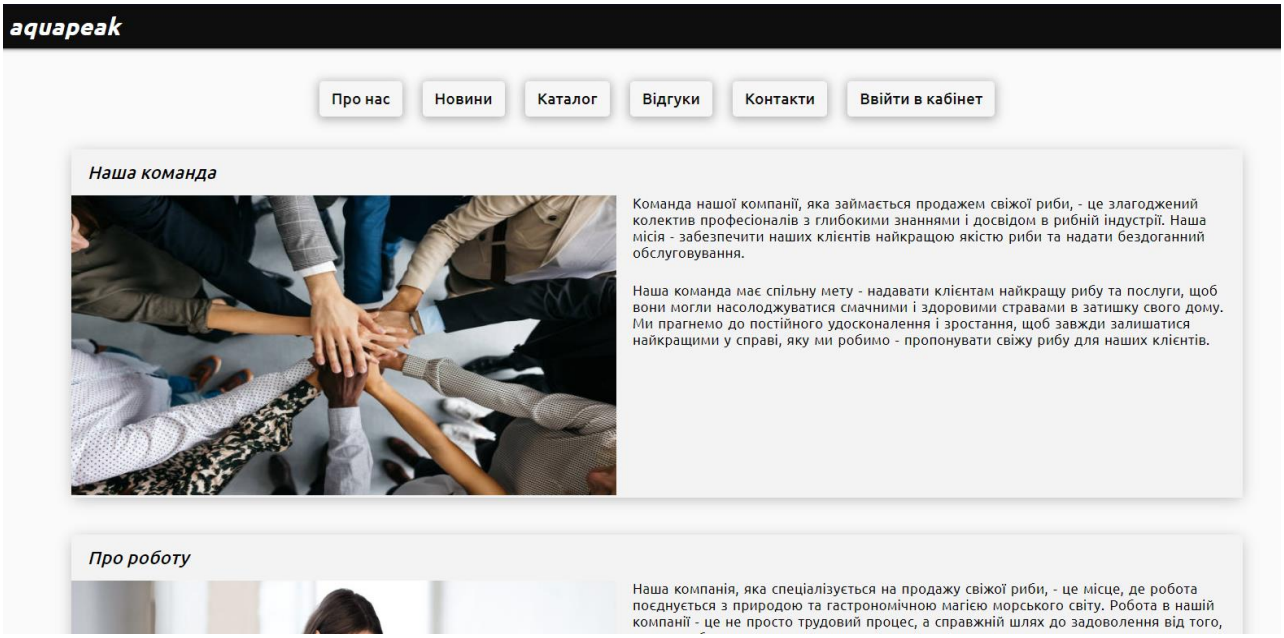

Рисунок 3.16 – Вигляд сторінки «Про нас» сайту «Aquapeak»

На цій сторінці, користувач може переглянути усі новини, продивитися викладені фото та картинки. Також, зможе побачити дату виставленої новини та автора, що її виставляв (див. рис. 3.17).

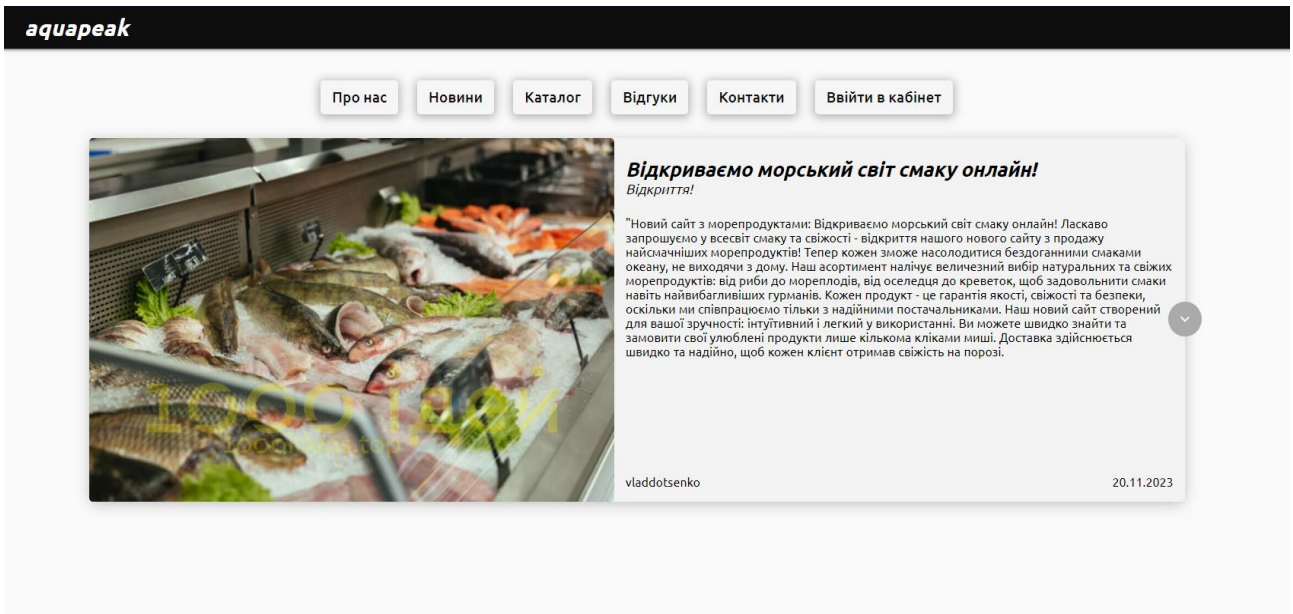

Рисунок 3.17 – Вигляд сторінки «Новини» сайту «Aquapeak»

При переході на іншу сторінку після натискання кнопки «Відгуки», користувач потрапляє на сторінку, де може побачити цілу сторінку відгуків від інших користувачів, що вже придбали товар на сайті раніше, та встигли залишити свої відгуки, щодо надійності та якості придбаного товару. Кожен відгук має свій номер замовлення, оцінку та коментар від користувача, що робив замовлення. Також, у коментарі зазначається ім’я та прізвище користувача, що залишив відгук, а також дата та час створення цього відгуку (див. рис. 3.18).

Після натискання кнопки «Контакти», користувач перенаправляється на сторінку, де може отримати усю інформацію стосовно онлайн магазину, електронної скриньки та телефону підтримки, що зазначені на сайті. Також, користувач отримує доступ до посилань на інші соціальні мережі, такі як: Facebook, Telegram, Whats Up, Youtube, Instagram та Discord.

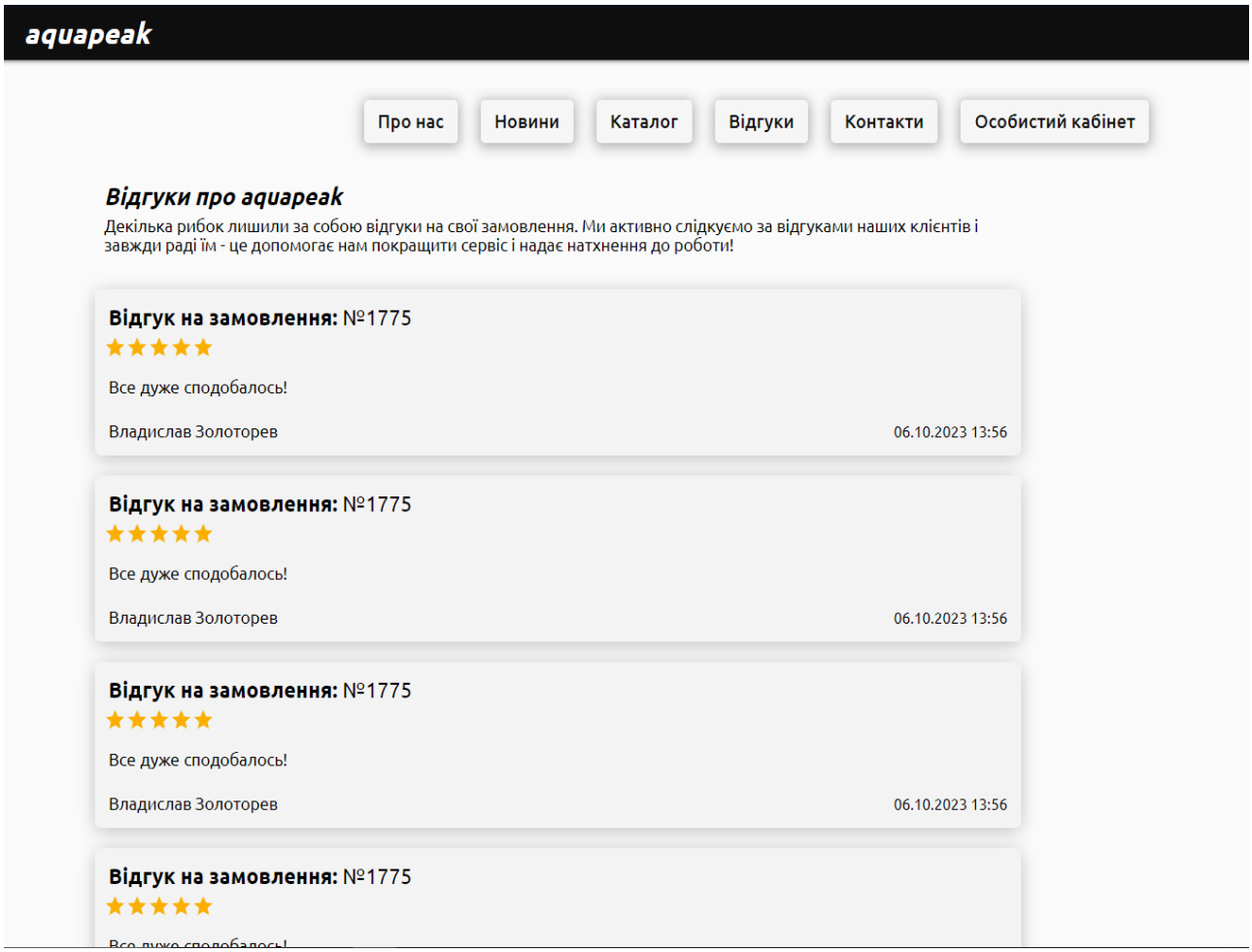

Рисунок 3.18 – Вигляд сторінки «Відгуки» сайту «Aquapeak»

Нижче, користувач може отримати інформацію стосовно адреси нашого магазину, та гугл карту для отримання інформації щодо маршруту та візуального орієнтиру на місцевості (див. рис. 3.19).

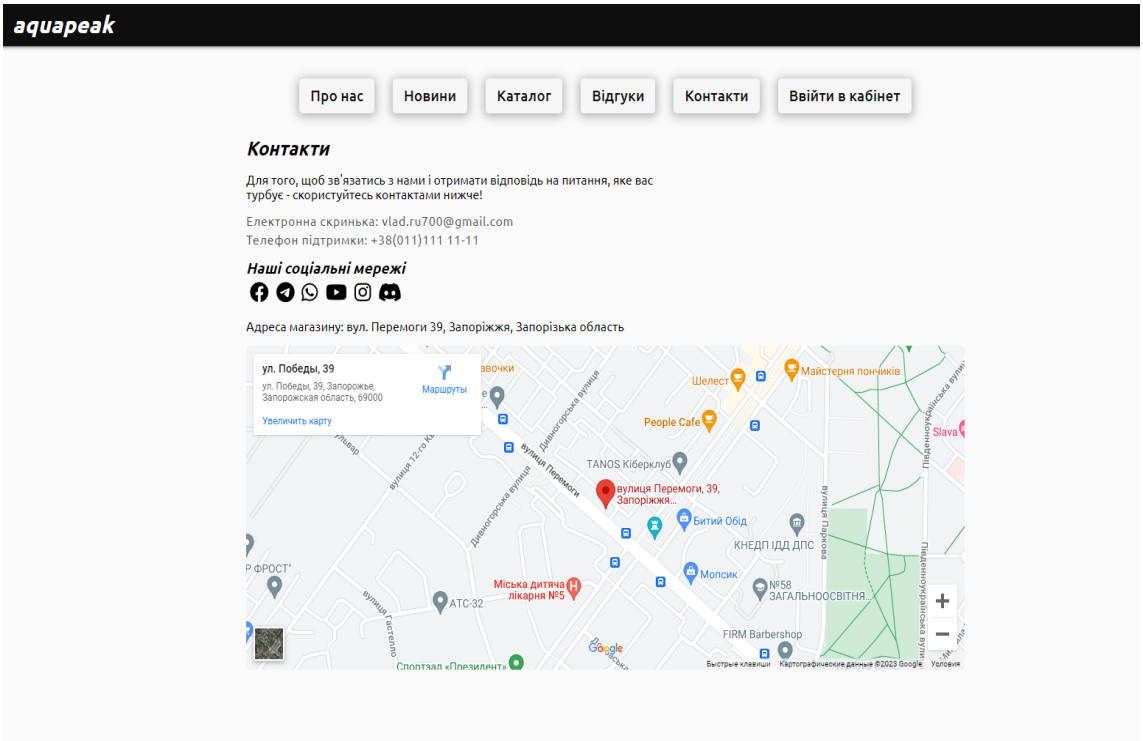

Рисунок 3.19 – Вигляд сторінки «Контакти» сайту «Aquapeak»

Остання кнопка доступна для користувача це «Особистий кабінет», або «Ввійти в кабінет», якщо користувач авторизований. Незареєстрований користувач перенаправляється на форму авторизації, де він повинен ввести свій логін та пароль, щоб авторизуватись (див. рис. 3.20).

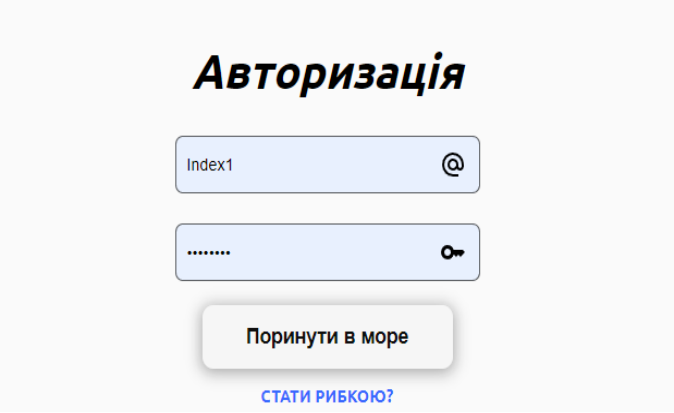

Рисунок 3.20 – Вигляд сторінки авторизації сайту «Aquapeak»

Якщо у користувача ще немає свого кабінету і він не був зареєстрованим, він може обрати кнопку «стати рибкою», після чого для нього буде доступна форма для реєстрації на сайті (див. рис. 3.21).

Користувач повинен придумати та ввести свій логін, свою поштову адресу та свій пароль. Коли всі поля будуть заповнені, користувач натискає кнопку «Створити акаунт» після чого його перенаправляє на сторінку з товаром.

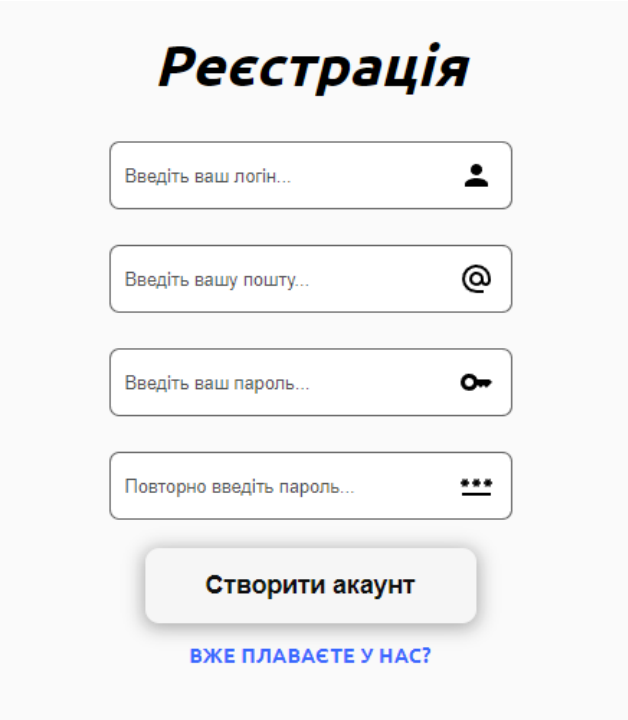

Рисунок 3.21 – Вигляд сторінки реєстрації сайту «Aquapeak»

Якщо користувач зареєстрований та авторизований, після натискання на кнопку «Особистий кабінет», користувача переправить на сторінку його особистого кабінету, де користувач може побачити совою особисту інформацію та має чотири кнопки, а саме: «Головна», «Кошик», «Особиста інформація» та «Вийти з кабінету» (див. рис. 3.22).

За допомогою кнопки «Головна», користувач може повернутись на сторінку з товаром, запропонованим на сайті. Натиснувши кнопку «Кошик», користувач потрапляє на сторінку із списком усього товару, що він додав до свого кошику раніше (див. рис. 3.23).

Рисунок 3.22 – Вигляд сторінки «Особистий кабінет» сайту «Aquapeak»

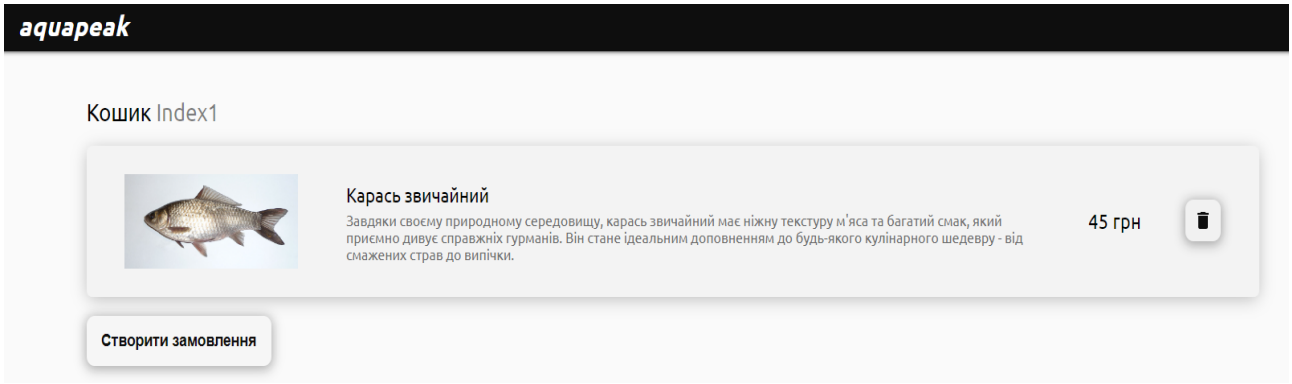

Рисунок 3.23 – Вигляд сторінки «Кошик» сайту «Aquapeak»

Далі в своєму кошику, користувач може обрати, що йому робити із своїм товаром далі. Він може видалити його з кошику, натиснувши на кнопку видалення, чи може перейти до створення покупки завдяки кнопки «Створити замовлення».

Після натиску кнопки переходу до замовлення, користувач перенаправляється на форму «Оформлення замовлення» (див. рис. 3.24).

В цьому меню, користувач повинен обрати, який спосіб отримання замовлення для нього є більш зручним: «Замовити доставку», «Забрати в магазині» чи «Забрати на пошті» (див. рис. 3.25).

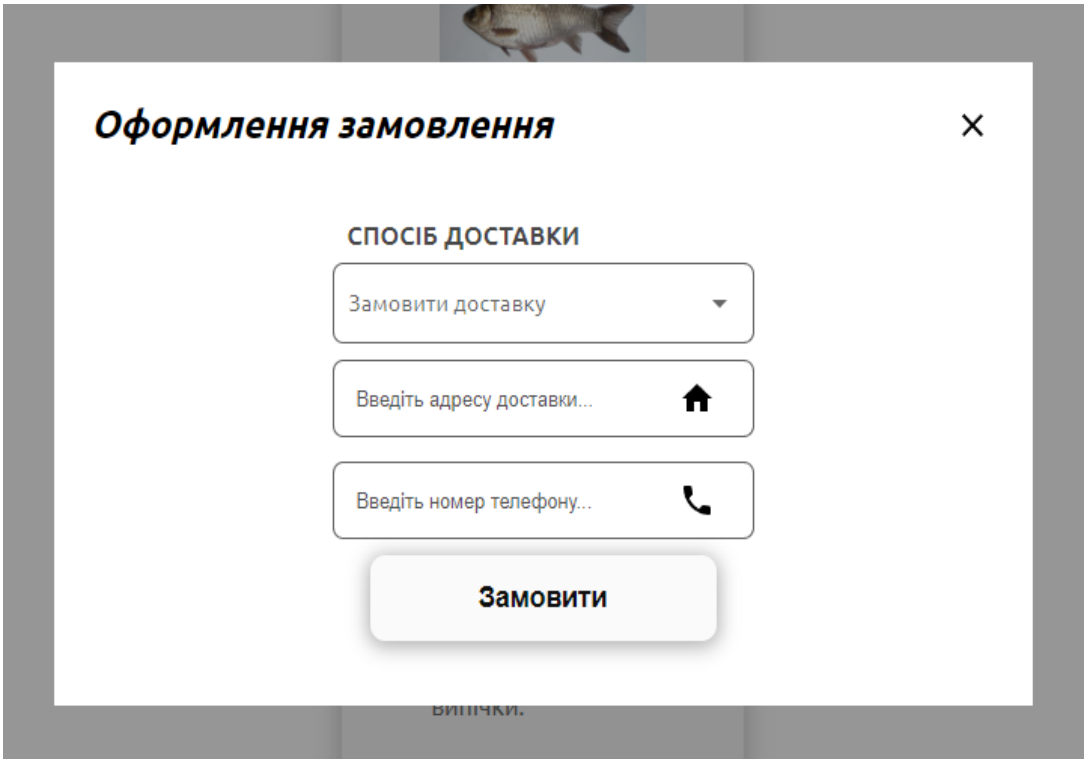

Рисунок 3.24 – Форма «Оформлення замовлення» сайту «Aquapeak»

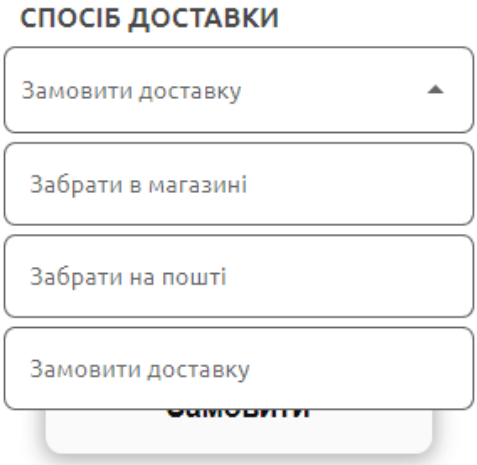

Рисунок 3.25 – Можливість обрати спосіб отримання товару

Після обрання способу отримання товару, користувач заповнює форму згідно обраного варіанту отримання товару. Після заповнення форми, користувач натискає кнопку «Замовити», після чого перенаправляється до свого кабінету, де може перевіряти статус свого замовлення, та може заробити оплату на сайті за допомогою картки (див. рис. 3.26).

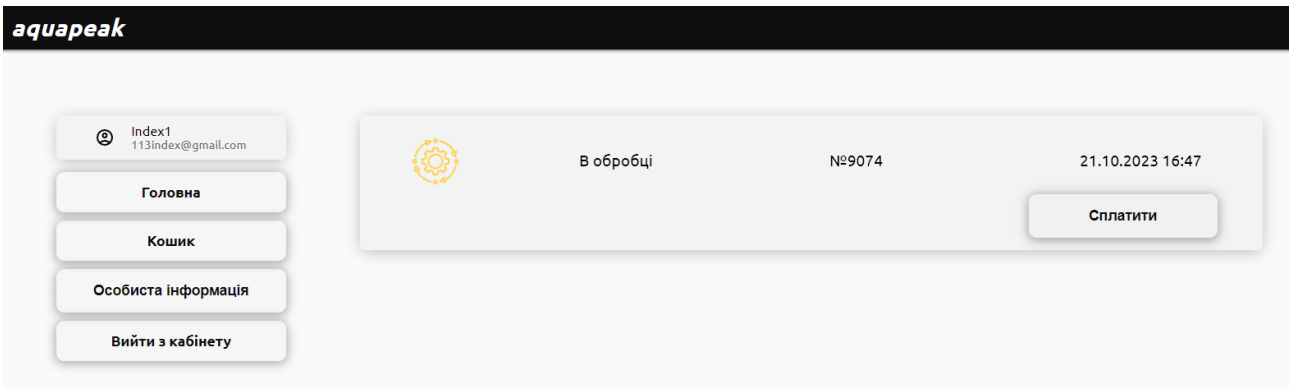

Рисунок 3.26 – Перевірка статусу товару

Адміністраторська частина сайту передбачає панель адміністратору у особистому кабінеті користувача, що авторизований як адміністратор сайту. Завдяки кнопці «Панель робітника», адміністратор сайту отримує доступ до функціоналу взаємодії з товаром, новинами на сайті, має доступ до повного списку користувачів та замовлень (див. рис. 3.27).

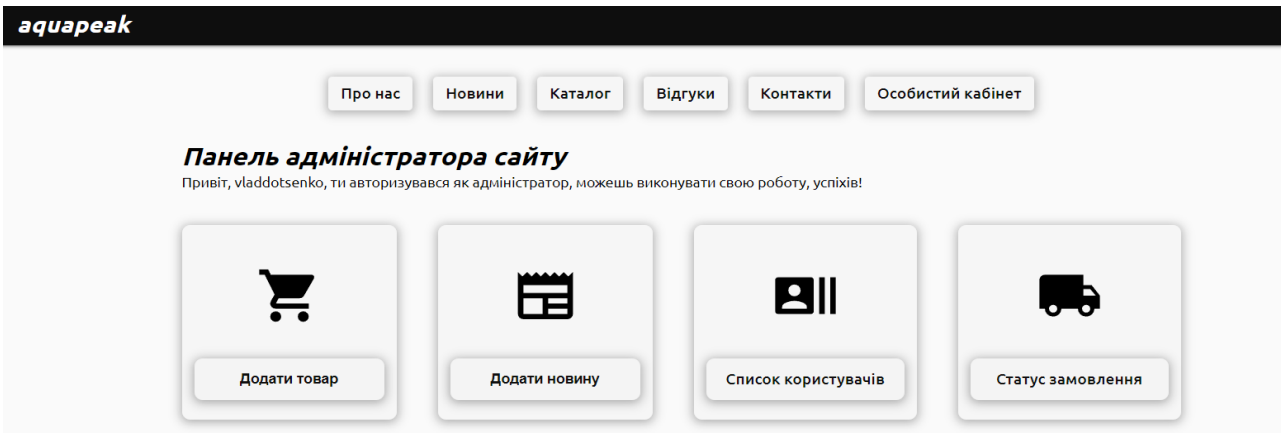

Рисунок 3.27 – Панель адміністратору сайту «Aquapeak»

Адміністратор може взаємодіяти із товаром за допомогою кнопки «Додати товар». Після натиску цієї кнопки, відкривається форма, в яку користувач адміністратор повинен ввести інформацію стосовно товару та заповнити усі поля (див. рис. 3.28). Після введення усіх пунктів форми та натиску на кнопку «Створити», товар потрапляє у базу даних, та становиться доступним у каталозі сайту. Також доданий товар може бути видалений адміністратором у каталозі.

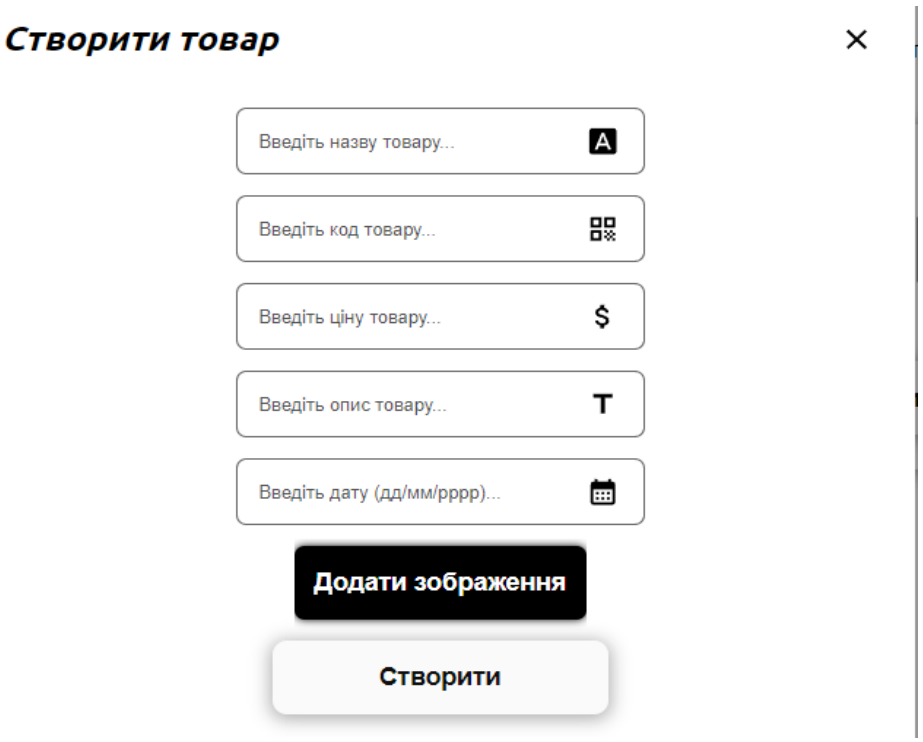

Рисунок 3.28 – Панель додавання нового товару

Так само, користувач адміністратор сайту, може взаємодіяти з новинами сайту. Адміністратор може додати нову новину натиснувши кнопку «Додати новину», після чого відкривається форма створення новин (див. рис. 3.29).

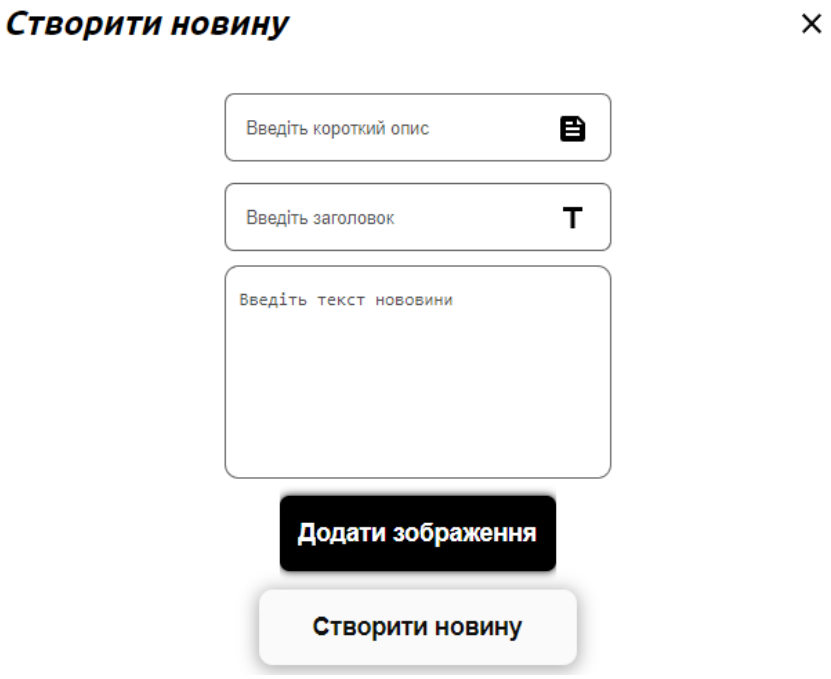

Рисунок 3.29 – Панель додавання нової новини

Адміністратор повинен заповнити усі поля форми, після чого підтвердити створення форми.

Дані з форми передаються до бази даних, а звідти до списку новин в розділі «Новини» на сайті.

Також, адміністратор може отримати повний список усіх користувачів, що були зареєстровані на сайті.

Разом з цим, адміністратор може видалити користувача, для цього йому потрібно натиснути на кнопку видалення під відповідним користувачем.

Окрім видалення, адміністратор може змінити деякі дані користувача, такі як: поштова адреса, пароль або надати користувачу функціонал адміністратору (див. рис. 3.30).

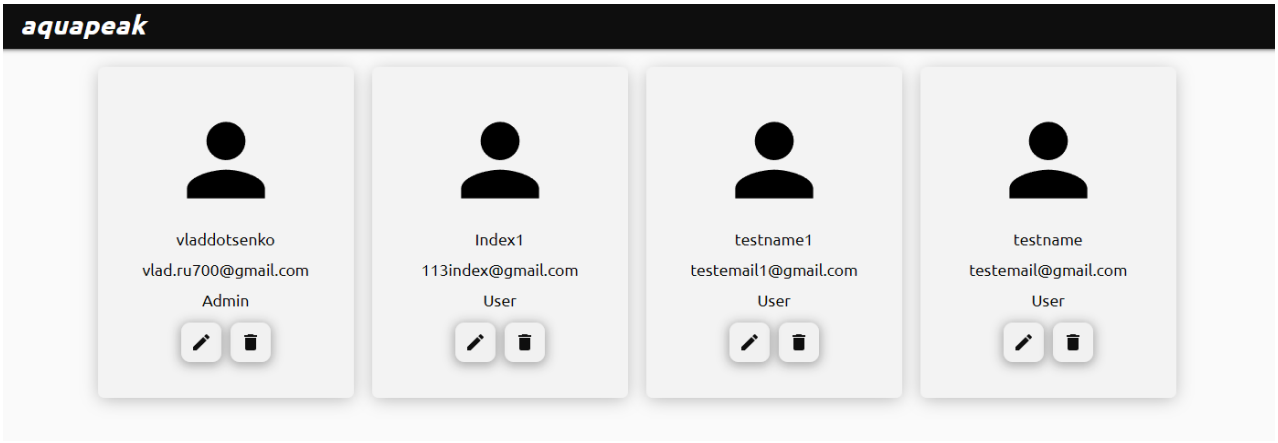

Рисунок 3.30 – Панель списку користувачів

Остання з кнопок доступних користувачу із функціоналом адміністратору сайту це «Статус замовлення».

Переходячи за цим посиланням, адміністратор отримує сторінку із списком замовлень, що були зроблені, їх номерами замовлень, датами замовлень, інформацією що надав користувач та статусами цих замовлень (див. рис. 3.31).

На цій сторінці, адміністратор може перевірити чи була зроблена оплата за замовленням та якщо потрібно, змінити статус замовлення, якщо той було виконано, відправлено, чи якщо заказ очікує клієнта у магазині.

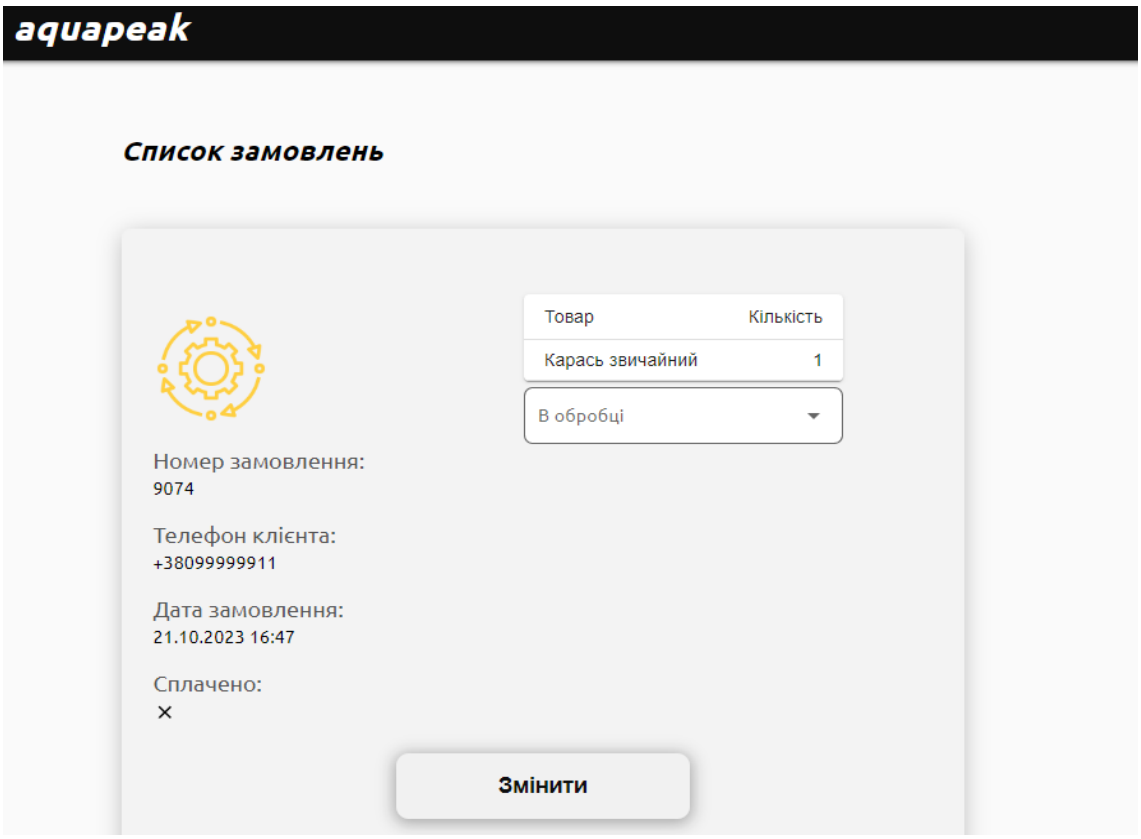

Рисунок 3.31 – Панель обробки замовлень

ВИСНОВКИ

За результатом проведеної роботи отриману успішну реалізацію інформаційної системи «Aquapeak» за допомогою мови програмування JavaScript та її бібліотек, а також бази даних MongoDB. На етапі проєктування, був створений макет зовнішнього вигляду головної сторінки сайту за допомогою багатофункціонального вебінструменту для дизайну інтерфейсів Figma. Окрім цього, була спроєктована структура інформаційної системи завдяки діаграмі класів, схемі бази даних та діаграмі компонентів.

Для реалізації додатку були використані дві різні технології, а саме середовище виконання JavaScript Node.js та JavaScript бібліотека React.js для розробки інтерфейсів користувача. Завдяки Node.js, було розроблено багатофункціональний бекенд інформаційної системи з реєстрацією, авторизацією та адміністраторською частиною. У розробленому додатку, користувач має можливість взаємодіяти із запропонованим йому товаром, та має функціональний онлайн кошик. Після отримання товару, користувачу надається можливість залишити відгук на сайті. Користувач із роллю адміністратору сайту, має розширений функціонал, включно з додаванням або видаленням товарів та користувачів.

За допомогою бібліотеки React.js було створено динамічний та зручний інтерфейс користувача вебдодатку, що відповідає вимогам поставленим на етапі проєктування інформаційної системи та згідно створеному шаблону вебдодатку. Інтерфейс користувача є ефективним, зручним та привабливим. Сайт має просту та зрозумілу для користувача навігацію, що дозволяє швидко орієнтуватися.

На фінальному етапі кваліфікаційної роботи було проведено тестування інформаційної системи. За результатами тестування були виправлені помилки пов’язані із доступом до бази даних. Фінальні спроби тестування продемонстрували, що вебдодаток працює відповідно очікуваних результатів та має функціонал, що був запланований на етапі розробки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Sarrion E. Master React in 5 Days: Become a React Expert in Under a Week. Apress, 2023. 283 p.
2. Buckler C. Node.js: Novice to Ninja. SitePoint Pty. Ltd, 2022. 341 p.
3. Lim G. Beginning Node.js, Express & MongoDB Development. Pakt, 2019. 131 p.
4. Supertest. Supertest module. URL: https://www.npmjs.com/package/supertest (дата звернення: 27.09.2023).
5. Supertest. Supertest module. URL: https://testing-library.com/docs/react-testing-library/intro (дата звернення: 28.09.2023).
6. Schwarz D. The Designer's Guide to Figma: Master Prototyping, Collaboration, Handoff, and Workflow. SitePoint Pty Ltd, 2023. 224 p.
7. Gehlot A., Beri R. Full Stack Development with MongoDB. BPB Publications, 2022. 260 p.
8. Springer S. React: The Comprehensive Guide. Rheinwerk Computing, 2023. 273 p.
9. Osmani A. Learning javascript Design Patterns: A javascript and React Developer's Guide. O'Reilly Media, Inc, 2023. 346 p.
10. Hibbard J., Wanyoike M. Node.js: Tools & Skills. SitePoint, 2022. 143 p.
11. Кудін О. В. Кривохата А. Г. Методи класифікації акустичних даних. *Актуальні проблеми математики та інформатики* : збірка тез та доповідей дев'ятої Всеукраїнської, шістнадцятої регіональної наукової конференції молодих дослідників. (Запоріжжя, 26-27 квітня 2018). Запоріжжя, 2018. С. 37–38.