Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБ-МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ DEFILE

DEVELOPMENT OF A WEB STORE DEFILE FOR THE SALE OF MUSICAL INSTRUMENTS

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗ-41
Попко Віталій Сергійович
_____
(підпис)

Керівник
д.т.н., професор
Гордєєв Олександр Олександрович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«2» 02 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Попко Віталій Сергійович

1. Тема кваліфікаційної роботи *Розробка веб-магазину з продажу музичних інструментів Defile*
Керівник роботи: *д.т.н. Професор Гордєєв О.О.*
затверджені наказом закладу вищої освіти від «*23*» *грудня* 2022 р. №*961*01-02*

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи
«08» червня 2022 р.

3. Вихідні дані до роботи: *Вимоги до розробки програмного забезпечення. Мова програмування серверної частини – JavaScript з використанням Node. Технології програмування клієнтської частини – HTML, CSS, JavaScript, React.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз існуючих методів та засобів для розробки програмного забезпечення, та їх вибір, аналіз та опис функціонального наповнення для програмного забезпечення, детальний опис розробки та тестування програмного забезпечення, опис роботи програми.

5. Перелік графічного матеріалу:
11 рис. 2 діаграми, 1 схема

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		Завдання видав	Завдання прийняв
Аналіз предметної області	*Гордєєв О.О.*		
Специфікація вимог до розробленої системи	*Гордєєв О.О.*		
Розробка об'єкта проектування	*Гордєєв О.О.*		
Нормоконтроль	*Гордєєв О.О.*		
Гарант ОП	*Ліщина Н.М.*		
Показник запозичень тексту	3,07%		
Академічна доброчесність	*Яцук А.А.*		

7. Дата видачі завдання « 2 » лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*05.02.2023 р.*	вик
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2023 р.*	вик
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*10.03.2023 р.*	вик.
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2023 р.*	вик.
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2023 р.*	вик.
6	*Тестування та налагодження об'єкта проектування*	*01.06.2023 р.*	вик.
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*08.06.2023 р.*	вик.

Здобувач вищої освіти _____ (Попко В.С.)
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (Гордєєв О.О.)
(підпис) (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти: *Попко Віталій Сергійович*
(ПІБ)

Тема кваліфікаційної роботи:
Розробка веб-магазину з продажу музичних інструментів Defile / Development of a web store Defile for the sale of musical instruments

(повна назва згідно наказу)

Коротка характеристика кваліфікаційної роботи та прийнятих рішень:
Дана робота складається з таких розділів як: Аналіз предметної області і постановка завдань на кваліфікаційну роботу, специфікація вимог до програмного забезпечення та розробка веб-додатку.

Самостійні розробки і пропозиції автора:
Дипломант розробив сайт у вигляді веб-магазину з використанням PERN технологій. Під час виконання кваліфікаційної роботи, здобувач вищої освіти повністю описав принцип розробки бази даних, серверної та клієнтської частини у вигляді веб-сторінок.

Практичне значення роботи:
Дипломантом розроблено проект, який може бути використаний у вигляді фундаменту як для удосконалення розроблюваного сайту, так і для інших подібних проектів.

Недоліки:
Не виявлено

Загальний висновок:
Вважаю, що дана робота задовольняє всім вимогам, які описані в методичних вказівках. Рівень виконання роботи виді є високим. Усі поставленні завдання дипломант реалізував на високому рівні. Робота рекомендується до захисту та заслуговує на оцінку «відмінно».

Рецензент _____ (*к.т.н., доцент, Поліщук М.М.*)
(підпис) (науковий ступінь, вчене звання, посада, ПІБ)

«02» 06 2023 р.

АНОТАЦІЯ

Попко В.С. Розробка веб-магазину з продажу музичних інструментів Defile. Рукопис.

Кваліфікаційна робота бакалавра. Кваліфікаційна робота за спеціальністю 121 Інженерія програмного забезпечення. Луцьк 2023. 52 с.

Кваліфікаційна робота присвячена розробці та реалізації інтернет магазину з продажу музичних інструментів з використанням React та Node.

Робота складається з трьох розділів, в яких описано проблематику веб-додатків та розв’язання завдань, пов’язаних з нею. Описано процес розробки інтернет магазину, здійснено дослідження фреймворків React та Node для мови програмування JavaScript, розкрито суть їхнього використання. Показано принцип роботи суб’єкту управління базами даних PostgreSQL.

Ключові слова**:** інтернет-магазин, база даних, HTML, CSS, JavaScript, React, Node, PostgreSQL.

ABSTRACT

Popko V.S. Development of a web store Defile for the sale of musical instruments. – Manuscript.

Bachelor thesis. Bachelor thesis in the specialty 121 Software engineering. Lutsk 2023. 52p.

The thesis is devoted to the development and realization web-shop for the sale of musical instruments with using React and Node.

The work consists of three, in which the problems of web apps and tasks have been analyzed. The process of development of web shop has been descripted, research of React and Node frameworks for programming language JavaScript have been carried out, the sense of these using has been determined. The principle of work with database management system has been shown.

Keywords: web-shop, data base, HTML, CSS, JavaScript, React, Node, PostgreSQL.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ 10
1.1 Аналіз сучасного стану проблеми 10
1.2 Постановка завдання на кваліфікаційну роботу бакалавра 18
Висновки до розділу 1 19
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ 21
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення 21
2.2 Вибір засобів, методів та алгоритмів вирішення поставленого завдання 25
Висновки до розділу 2 28
РОЗДІЛ 3 РОЗРОБКА ВЕБ ДОДАТКУ 30
3.1 Практична реалізація об’єкта проектування 30
3.2 Тестування та налагодження інформаційно-комп’ютерної системи 31
3.3 Розробка бази даних та оформлення серверної частини додатку 34
3.4 Захист інформаційно-комп’ютерної системи 37
3.5 Розробка клієнтської частини додатку 39
Висновки до розділу 3 47
ВИСНОВКИ 49
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 50
ДОДАТКИ 51

ВСТУП

Веб технології змінили життя людей по всьому світу. Прикладів таких змін дуже багато. Наприклад, замість довгих черг за оплатою різних рахунків тепер можна сплатити всі послуги в глобальній мережі Інтернет; замість очікування улюбленого серіалу по телевізору тепер можна переглянути той самий серіал вже на своєму комп'ютері чи телефоні коли завгодно; замість досить тривалого очікування відповіді від товариша, який живе на іншій півкулі, тепер можна моментально отримати відповідь від нього, використовуючи різні соціальні мережі.

Як можна замітити, досить незручні та складні речі, які виконувались раніше, тепер здаються дрібницею з появою веб технологій. Без появи нових технологій в розробці веб-додатків ми би до сих пір витрачали багато часу в незрозумілих на теперішній час речах.

Саме напрям веб-технологій є однією з найбільш затребуваною в світі. Багато початківців-розробників залишаються працювати саме за цим напрямком. Розробка веб-додатків є досить легкою темою для вивчення, у порівнянні з іншими галузями розробки, що саме і приваблює новачків у вивченні даної галузі.

Темою наведеної кваліфікаційної роботи є розробка веб-магазину з продажу музичних інструментів.

Актуальність теми полягає в тому, що завдяки швидкому розвитку інформаційних технологій, звичайний користувач зможе замовити необхідний товар за допомогою декількох натиснень, а адміністратор – легко керувати всім асортиментом даного магазину. Без актуальних веб технологій весь цей процес здавався б в рази складнішим, оскільки замість оформлення всієї документації, що необхідна в декілька натиснень, люди б мали багато проблем з оформленням. Та й користувачеві, замість того, щоб йти в магазин, та перевіряти, чи є даних товар хорошим, можна просто переглянути відгуки від інших користувачів.

Метою даної роботи є створення веб-додатку, який буде створений за PERN технологією (PostgreSQL, Express, React, Node).

Завданням даної роботи є:

– опис схожих технологій, які використовуються для розробки подібного програмного забезпечення;

– реалізація можливості розглядати та замовляти необхідний товар;

– реалізація реєстрації та авторизації користувачів;

– реалізація можливості адміністраторам добавляти необхідний товар та їх типи у базу даних через клієнтську частину додатку.

Об’єктом дослідження даної роботи є інтернет-магазин для огляду та оформлення музичних інструментів.

Предметом дослідження є методи та технології розробки сайтів за допомогою PERN.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Веб розробка є на даний момент однією з найбільш розвинених та потребуючих гілок в IT індустрії. Зараз практично кожна престижна компанія чи установа містить свій веб-сайт.

Веб-сторінка є практично єдиним способом взаємодії між компанією та звичайним рядовим користувачем без прямого контакту. Саме цей факт робить розробку веб-додатків одним з найбільш популярних в IT.

Сама структура створення та підтримки веб-додатків ділиться на 2 гілки.

Front-End – створення інтерфейсу для користувача у веб-сторінці чи програмі. Це дозволяє взаємодіяти між клієнтом (користувачем) та сервером [1].

Back-End – розробка логіки веб-сторінки чи програми. Найпопулярніший приклад роботи Back-End – авторизація та реєстрація користувачів. Однак він не єдиний.

За допомогою Front-End користувач може взаємодіяти вже з самим Back-End, коли Back-End виконує практично весь основний функціонал, будь то обчислення, чи пошук відеоролика у соціальних мережах. Без Back-end побудувати повністю функціональний сайт неможливо.

Розглянемо спочатку першу гілку веб-додатків. Кожен сучасний веб-сайт містить в собі три складові, з яких вони будуть написані: HTML, CSS та JavaScript, які показано на рисунку 1.1

Рисунок 1.1 – Мови, які використовуються у технологіях Front-end

HTML (Hyper Text Markup Language) – мова розмітки документів, за допомогою якої створюється сама структура сайту (текст, заголовки, абзаци, списки, тощо) [1]. На даний момент використовується найбільше саме 5 версія HTML, вона є найбільш оптимізованою та має набагато більше функцій, ніж його попередні версії.

Веб-сторінка лише з використанням HTML буде виглядати досить сиро та непривабливо. Тому тут потрібно ще використовувати CSS.

CSS (Cascading Style Sheets) – мова для опису та стилізації документів. За допомогою CSS можна буде наприклад: змінити шрифт, колір тексту, розмір шрифту, задній фон і т.д.

Використання лише HTML та CSS не є достатнім, ще потрібно надати необхідний функціонал самій сторінці. Для цього використовують JavaScript.

JavaScript – об’єктно-орієнтовна, динамічна мова програмування. Найчастіше ця мова використовується для створення функціоналу у веб-сайтах, що дає можливість клієнту взаємодіяти з серверною частиною сайту. Також далі у списку буде наведено весь можливий функціонал використання цієї мови у самому веб-додатку:

– зміна інтерфейсу без використання різних IDE за виконання деяких умов. Така функція використовується для розробки інтерфейсу саме для мобільних додатків, так як функціоналу мобільної платформи (смартфон, планшет) зазвичай недостатньо для повної роботи з ним саме на цій платформі,

ніж на персональному комп’ютері. Тому тут використовується більш спрощена версія інтерфейсу, ніж та, яка присутня на ПК;

– взаємодія з сервером. За допомогою JavaScript можна «підштовхнути» серверну частину до роботи (наприклад, надіслати запит) безпосередньо у самому клієнті;

– надання веб-сторінці «життя». Додавання різних анімацій, використання спливаючих вікон або видання інформації про успіх або помилку у самому сайті не буде лишнім.

Також для полегшення написання коду в JavaScript використовують бібліотеку jQuery. Хоча повноцінною бібліотекою, як наприклад ті, які будуть описані коротко далі, її досить складно назвати, оскільки вона не може повністю покрити весь код, а лише використати окремі функції, які є відповідно у цій бібліотеці.

Зазвичай використання цих трьох мов буде достатньо для створення веб-сайту. Ще також є різні бібліотеки, за допомогою яких розробка веб-додатку може стати більш простим, а сам сайт вийде більш функціональним. Розглянемо коротко декілька таких бібліотек:

– React.js – відкрита JavaScript бібліотека, творцями якої є Meta (в минулому Facebook), для створення інтерфейсу для користувачів у простий та швидкий спосіб [2]. Ця бібліотека підходить для тих, хто тільки наважився вивчити якусь з подібних бібліотек. Ця бібліотека фокусується здебільшого лише на створення інтерфейсу для користувачів. React має величезну популярність серед розробників, тому існує безліч статей, з якими можна швидко виконати будь-яку проблему. Найчастіше використовується з’єднання React з іншою JavaScript бібліотекою для Back-end Node.js;

– Vue.js – бібліотека JavaScript, яка використовується здебільшого для написання невеликих та нескладних сайтів. Але не зважаючи на те, що ця бібліотека має досить обмежений функціонал, вона все таки має досить хорошу документацію низький рівень входу та сам невеликий розмір бібліотеки [3];

– Angular – написана на TypeScript веб-бібліотека з відкритим кодом, творці якого являються Angular Team, під керівництвом Google. Ця бібліотека має в 7-8 разів більший розмір, ніж React, чи Vue.js, але платою за це є досить великий функціонал. На даному сервісі написані такі проекти, як Google (Gmail, Google Play, YouTube та немало інших проектів від Google), Upwork, Lego, Adidas. На рисунку 1.2 показано графік завантажень даної бібліотеки, що робить її однією з найпопулярніших бібліотек на сьогодні [3-5].

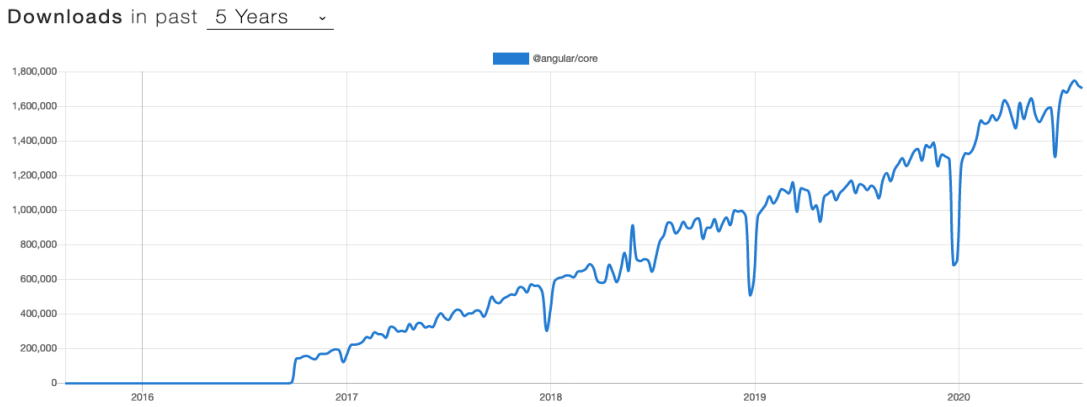

Рисунок 1.2 – Графік завантажень бібліотеки Angular за останні 5 років

Використання бібліотек, про які було написано вище, значно можуть полегшити написання веб-додатку, а тому зараз використання голого JavaScript (навіть з використанням jQuery) не є на даний момент сильно доцільним.

Тепер варто описати принцип роботи Back-end. Найчастіше при роботі з Back-End ми стикаємось з базами даних. Для роботи з базами даних прийнято використовувати різні суб'єкти управління базами даних (СУБД). Далі буде наведено список декількох з них:

– Microsoft SQL Server – СУБД, яка була розроблена компанією Microsoft. Вона оптимальна в роботі з Windows, хоча має сумісність з Linux. Дана СУБД містить досить чималий функціонал, а тому місця займає немало. Дана система управління використовується здебільшого для керування великими базами даних, яким функціоналу від решти СУБД буде недостатньо;

– MySQL є однією з найбільш популярних СУБД. MySQL – вільна СУБД з відкритим кодом. В першу чергу вона використовується для розробки та створення динамічних веб-сторінок, оскільки вона підтримується немалою кількістю мов програмування;

– PostgreSQL – вільна об’єктно-реляційна СУБД. Особливістю даної системи керування базами даних є те, що вона не контролюється жодною компанією. Ця СУБД також підтримує немало мов програмування, як і попередня MySQL. Дана СУБД є досить простою у використанні, тому є також досить популярною серед розробників-початківців;

– MongoDB – документо-орієнтована система управління базами даних, з відкритим кодом. Особливістю даної СУБД є те, що вона у деяких випадках не потребує використання мови SQL. Ця особливість робить дану СУБД однією з найшвидших та масштабованих систем.

Для керування базами даних, які вже були створені та налаштовані в СУБД, необхідно далі налаштувати взаємодію між сервером та клієнтом. Це робиться за допомогою різних мов програмування. Обмежень в самих мовах немає, можна писати код на будь-якій мові, але при умові, якщо сама система керування базами даних підтримує дану мову. Тому далі буде розглянуто найпопулярніші мови програмування, які використовують для взаємодії між клієнтською та серверною частиною:

– C# – об’єктно-орієнтовна мова програмування. Мова була створена компанією Microsoft, тому здебільшого дану мову використовують лише на пристроях, які використовують операційну систему Windows. Дана мова є досить швидкою, тому використання даної мови при керуванні базами даних є доцільним;

– Java – об’єктно-орієнтовна мова програмування, яка була створена компанією Sun Microsystems, яку потім придбали Oracle. Дана мова підтримує понад 3 мільярди пристроїв, що робить дану мову досить затребуваною. Синтаксис даної мови досить схожий на синтаксис мов C та C++, тому тим, хто використовував ці мови в минулому не буде ніяких проблем у вивченні Java.

Дана хоча і поступає попередній мові у швидкості, але тим не менш її все ще вистачає для зручного керування базами даних;

– Python – високорівнева, інтерпретована, об’єктно-орієнтовна мова програмування. Цю мову можна назвати універсальною, оскільки за її допомогою можна зробити чимало інших програм, окрім серверної частини веб-сторінки (наприклад, боти, які використовуються в месенджері Telegram, програми з використанням штучного інтелекту та чимало інших прикладів). Дивлячись на те, що ця мова є інтерпретованою, що означає, що дана мова не є досить швидкою, її все ще може вистачати для невеликих баз даних;

– PHP не потребує ніякого компілятору, що дозволяє працювати на будь-якому пристрої. Також дана мова містить відкритий код та знаходиться у вільному доступі. Дана мова програмування є золотою серединою для кожного веб-додатку, як для незначного, де використовується невелика база даних, так і значного, де цей додаток контролює весь процес роботи компанії [4];

– C++ – об'єктно-орієнтовна, компільована мова програмування. Сюди можна ще віднести таку мову, як C. Зі слова компільована (мова, яка відразу переводить весь код програми в машинний код) можна зрозуміти що дані мови підходять для використання її саме в базі даних. Але на перший погляд, C++ та C не є досить простими мовами програмування. Зазвичай, їх основне призначення є саме створення складних програм, наприклад: драйвера для комп’ютерного обладнання, комп’ютерні ігри та рушії до них, операційні системи і так далі. Тому використовувати саме ці мови для написання запитів не є сильно доречним. Краще використовувати простіші варіанти;

– Node.js – бібліотека JavaScript, яка була придумана саме для роботи з Back-end. Сам JavaScript використовується для обробки веб-сторінок, коли Node.js був придуманий для обробки запитів у сервер. Для керування модулями необхідний встановлений пакетний менеджер npm.

Після написання коду для запитів в базу даних, необхідно кожну з функцій протестувати. Це можна зробити за допомогою платформ для тестування API. Вони допомагають без перешкод надіслати або отримати

запити для внесення змін у базу даних. Для тестування написаних потім API для запитів існують наступні платформи:

– Postman – платформа для тестування API з відкритим доступом. Дана платформа містить всі необхідні для тестування функції, має 2 версії (веб версія та сам додаток). Очевидно, додаток буде мати більш широкий функціонал, ніж веб-версія, тому перше підходить для більш складних систем, а друга – для простіших;

– Insomnia – платформа для тестування API. Дану платформу використовують такі компанії, як: Netflix, Facebook, Tesla, MasterCard, WeWork та багато інших компаній[6].

Для комфортного створення веб-продукту і взагалі будь-якого програмного забезпечення та подальшої його підтримки необхідно чітко визначити, яку саме архітектуру необхідно використовувати. Надалі буде наведено один з прикладів архітектури для веб-додатків.

Перше, що необхідно визначити, це на якій мові треба створювати клієнтську та серверну частини програмного продукту. Тому спочатку у папці з проектом потрібно створити дві папки, одна для клієнтської частини (Client), друга – для серверної (Server). Тепер вже безпосередньо в цих папках необхідно встановити необхідні основні пакети, які будуть використовуватись протягом розробки. У клієнтській частині це буде насамперед бібліотека, приклади яких були описані вище. У серверній можна просто встановити JSON файл, де можна буде потім в процесі встановити необхідні пакети.

Після встановлення необхідних пакетів в обох папках, можна перейти вже безпосередньо до розробки. Першим, як правило, розробляється серверна частина, оскільки доцільніше краще її спочатку випробовувати саме на платформах для тестування API. Це робиться для того, щоб потім не розбиратись зі всіма помилками у самому клієнті, коли не сильно буде зрозуміло, у чому саме помилка. Структура серверної частини (на прикладі буде розглянуто саме мій проект) буде показано в таблиці 1.1.

Таблиця 1.1 – Перелік складових частин архітектури серверної частини веб-додатка

№ п\п	Назва папки або файлу	Призначення
1	2	3
1	Package.json, package-lock.json	Ці два файли допоможуть зрозуміти, які саме пакети були встановлені. За допомогою цих файлів можна вказується, де починається запуск.
2	node_modules	Папка node_modules містить всі встановлені пакети
3	index.js	Основний файл, з якого розпочинається запуск серверу.
4	db.js	Цей файл допомагає підключити сервер до СУБД.

Продовження таблиці 1.1

1	2	3
5	.env	Конкретно у даному випадку тут написані основна інформація для підключення серверу до СУБД.
6	controllers	Папка з основним функціоналом сайту (Створення запитів).
7	error	Папка з функціями для відображення помилок сервера.
8	middleware	Папка з проміжним програмним забезпеченням (функції, які мають доступ до об’єкту запиту, відповіді та до наступних функцій проміжної обробки в циклі «запит-відповідь» додатку.
9	models	Папка, яка містить файл для створення таблиці в СУБД, та його з’єднання для роботи з базою даних.
10	routes	Папка, яка містить файли з кодом, де описані конкретні маршрути для навігації сервера.
11	static	Папка, де будуть розміщуватись всі зображення, які коли небудь були задіяні при створенні товару

Джерело: складено автором

Клієнтська частина буде створена за допомогою команди у командному рядку в самому середовищі розробки npm create-react-app (судячи з команди можна зрозуміти, що клієнт буде написаний за допомогою бібліотеки React). Після закінчення встановлення бібліотеки в проект, будуть додані вже необхідні для роботи пакети (разом з файлами Package.json та package-lock.json, які були розглянуті в таблиці 1.1), вказівка у форматі md, де буде вказано те, що клієнт працює саме на React, невелику документацію щодо самого React та дві папки: public та src. Перша папка містить в собі HTML файл, за допомогою якого буде виконуватись саме запуск клієнта (досить часто вона залишається без змін). Друга папка виконує весь функціонал клієнта. Архітектура тут може

використовуватись довільна, але все ж таки треба розділити всі компоненти на частини. Наприклад, компоненти, такі як: навігаційна панель, модальні вікна і т.д. варто вказати в одній папці, а самі сторінки, які будуть відрізнятись як зовнішнім виглядом, так і у функціоналі можна вказати вже в іншій папці.

Лише за правильним слідуванням інструкцій у побудові архітектури додатку, можна створити зручний як у розробці, так і у користуванні додаток.

Для того, щоб при можливості, якщо якимось чином, проект був зламаний, а знайти помилку є непосильною задачею, існує такий сервіс, як GitHub. Хоча він потрібен не лише для цього, щоб відкочуватись до попередньої версії. Цей сервіс має дуже великий функціонал, як перегляд всього коду, який написаний в проекті, можливість іншим користувачам переглядати та редагувати код для покращення функціоналу або додаванням нових функцій в проект. Коротше кажучи, цей сервіс є дуже зручним середовищем для обміну коду.

Оскільки темою цієї роботи є створення веб-магазину з продажу музичних інструментів, варто розглянути декілька аналогів:

– Rozetka (rozetka.com.ua) – український інтернет магазин, який надає послуги з продажу різних товарів, такі як наприклад: комп’ютери, побутова техніка, різні інструменти, товари для дому та багато іншого. Як можна зрозуміти, тут представлений товар у вигляді музичних інструментів. Моя робота буде все ж таки більше стосуватись саме однієї категорії, а не багатьох, тому можна сказати, що ця платформа буде лише частково схожа на розроблений мною продукт;

– Guitarhouse (guitarhouse.com.ua) – український інтернет магазин з продажу гітар та їх обладнання. Цей сайт вже більше буде схожим на мій проект, оскільки тут представлені виключно музичні інструменти.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Протягом виконання даної роботи повинні бути виконані наступні пункти:

– верстка та написання коду для навігації та користуванням веб-додатку. У даному пункті потрібно зробити веб-додаток максимально зручним у користуванні та легким у вивченні;

– створення бази даних зі всіма товарами та користувачами. Тут необхідно створити базу даних, яка буде містити всю необхідну інформацію про товар та користувачів;

– можливість авторизації та реєстрації користувачів. У даному пункті розглядається зручне керування своїм обліковим записом (Авторизація та реєстрація в окремому вікні);

– контроль ролей користувачів. Цей пункт дозволить видавати роль адміністратора звичайним користувачам. Це дасть завади звичайним користувачам добавляти чи видаляти товар або так само вести контроль над всіма користувачами;

– додавання та видалення товару та їх типів у базу даних. Тут береться до уваги створення зручного інтерфейсу у веб-додатку для додавання товару. При створенні товару потрібно буде вказати назву, тип товару, ціну, обрати зображення з товаром, та, за потреби, додати характеристики товару;

– можливість замовлення товару. Принцип замовлення товару робиться через кошик. Якщо користувачеві даний продукт сподобався, йому необхідно додати цей товар у кошик, де йому потрібно буде остаточно оформити замовлення.

Саме за допомогою цих завдань можна побудувати веб магазин з необхідним набором функцій

Висновки до розділу 1

У даному розділі було представлено принцип розробки веб-додатків. Розкрито відмінність між Front-end та Back-end та описано їх принцип роботи,

які мови та бібліотеки тут можуть використовуватись. Також наведений приклад реалізації архітектури веб-додатка та розглянуто аналоги сайтів, які мають схожу з моєю роботою тематику з платформою GitHub, та його призначенням. Постановлено завдання на кваліфікаційну роботу.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Для початку треба скласти UML діаграму, яким чином буде проект виконуватись. Для цього можна використати діаграму варіантів використання, де детально буде розглянуто функціонал проекту для окремого типу користувачів. Сама діаграма представлена на рисунку 2.1.

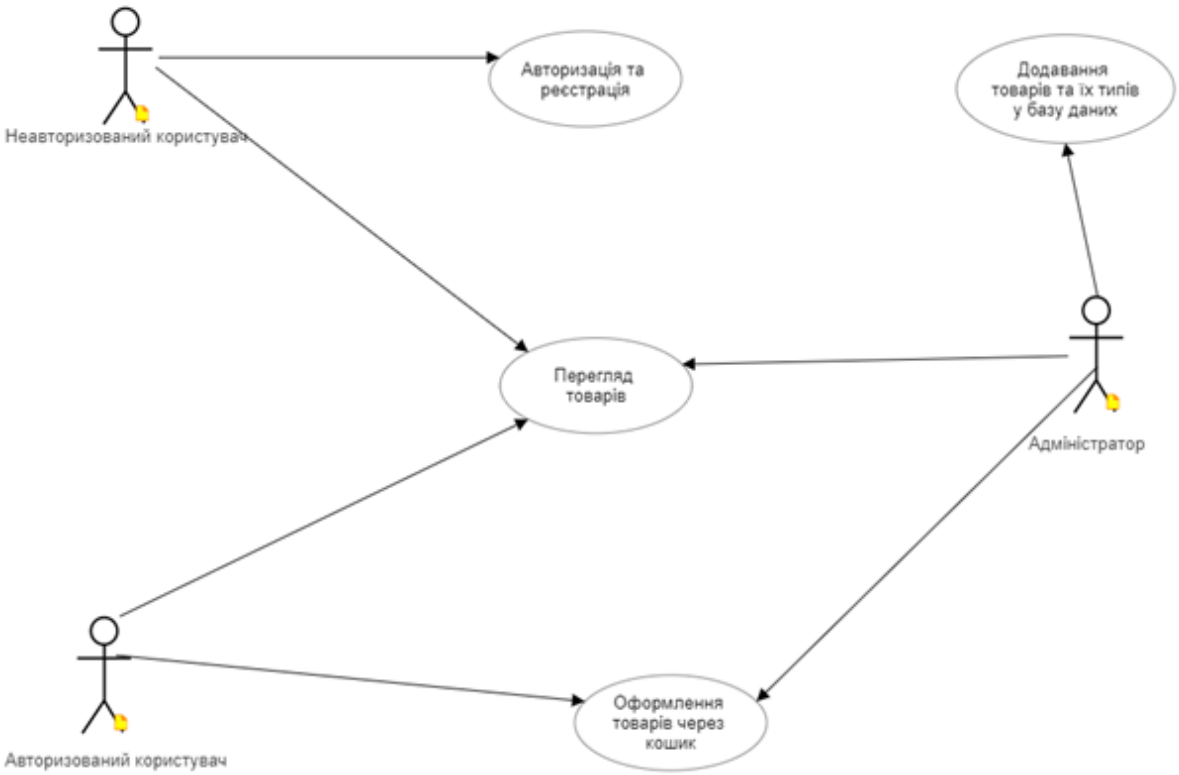

Рисунок 2.1 – Діаграма варіантів використання

Перед розробкою потрібно зрозуміти з яких складових буде розроблятися додаток. Як вже було згадано в попередньому розділі, проект буде складатись з двох частин: серверної та клієнтської частин. Буде логічніше розглянути спочатку серверну частину, оскільки саме на ній ми будемо створювати клієнтську частину (а саме дизайн сайту, його функціонал і т.д.).

Перш за все розглянемо питання проектування бази даних. На рисунку 2.2 представлена ER-діаграма.

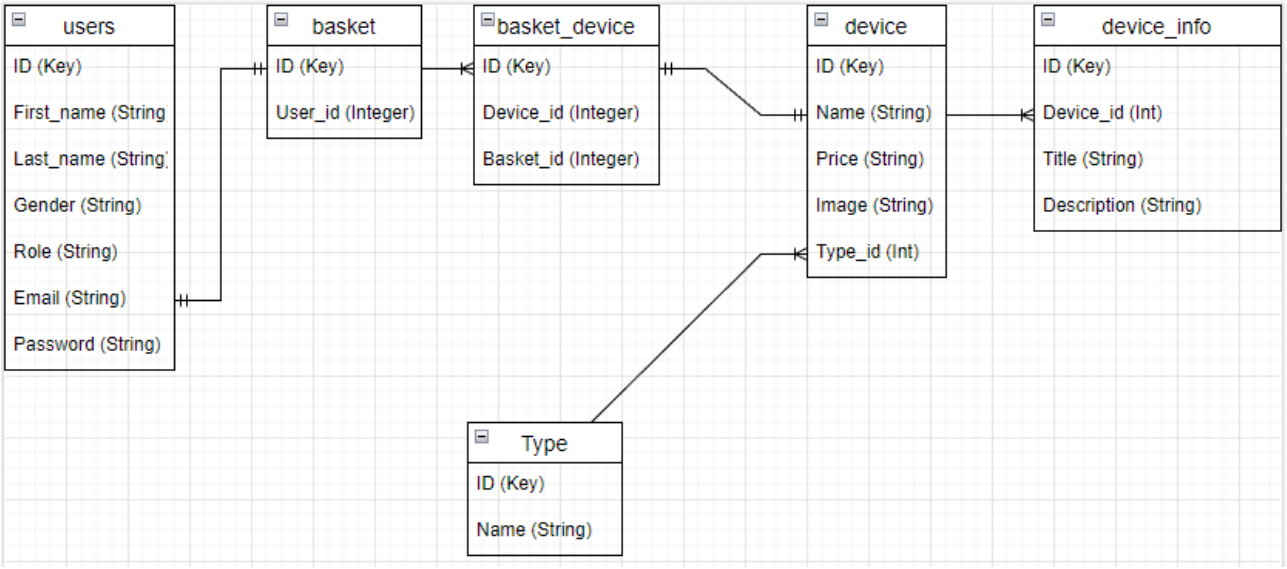

Рисунок 2.2 – ER діаграма

Розглянемо дану діаграму детальніше. У комірці Users буде показана вся детальна інформація про користувача (ім’я, прізвище, стать), також його логін та пароль. Всі змінні у даній таблиці, окрім ID (він буде виконувати роль ключа, як і в подальших таблицях) мають значення string.

У Basket показано, до якого користувача ця корзина була під’єднана. Треба звернути увагу до того, що Email та ID корзини з’єднані зв’язком 1 до 1, тобто від одного користувача, до однієї корзини. Це означає, що лише одна корзина може належати лише одному користувачу, і навпаки, один користувач може мати лише одну корзину.

У Basket_device показується зв’язок між корзиною користувача та товарами. Можна побачити, що тут також прокладений зв’язок між сусідніми колонками а саме Basket (відношення 1 до багатьох, оскільки одна корзина може містити в собі безліч товарів) та Device (відношення 1 до 1, тому що тут корзина може посилатись лише на 1 товар).

У комірці Device показана вся інформація про товар (назва товару, його ціна, зображення та до якого типу він відноситься). Назва товару, тип та

зображення будуть мати тип string (у випадку з зображенням буде використовуватись метод з вираховуванням саме шляху во самого файлу з зображенням), а ціна – integer.

У комірці Type вказано, які типи товарів є в базі даних. Тут вказано зв'язок з товарами, як 1 до багатьох, оскільки чимало товарів можуть мати один і той самий тип. Тип змінної Name буде string.

У комірці Device_info вказано всі характеристики окремого товару. Ця комірка взаємодіє з товарами як безліч до одного, що дасть змогу детально та зрозуміло описати окремий товар. Тут дані будуть розглядатись у вигляді масиву, де всі змінні будуть мати тип string.

З базою даних та її компонентами розібрались, тепер можна перейти до створення шаблону інтерфейсу. Він розробляється за допомогою програмного забезпечення Figma.

Додаток буде мати 6 сторінок. Розглянемо детально кожну з них:

– приклад з готовим інтерфейсом головної сторінки в програмі Figma буде показано на рисунку 2.3. Тут показується список зі всіма типами зліва та всі товари у вигляді таблиці справа від списку;

– сторінка з авторизацією та реєстрацією буде містити колонки для вводу інформації, яка необхідна для користування (у випадку з реєстрацією треба буде вказувати персональну інформацію, наприклад ім'я, прізвище, стать і т.д., а у випадку з авторизацією просто пошту та пароль);

– сторінка з окремим товаром буде містити зображення справа та детальною інформацією про обраний користувачем товар зліва (назва, ціна та характеристики);

– сторінка зі змістом корзини буде представляти собою список з обраними для замовлення товарами, їх ціною за кожен товар та загальна вартість за всі товари;

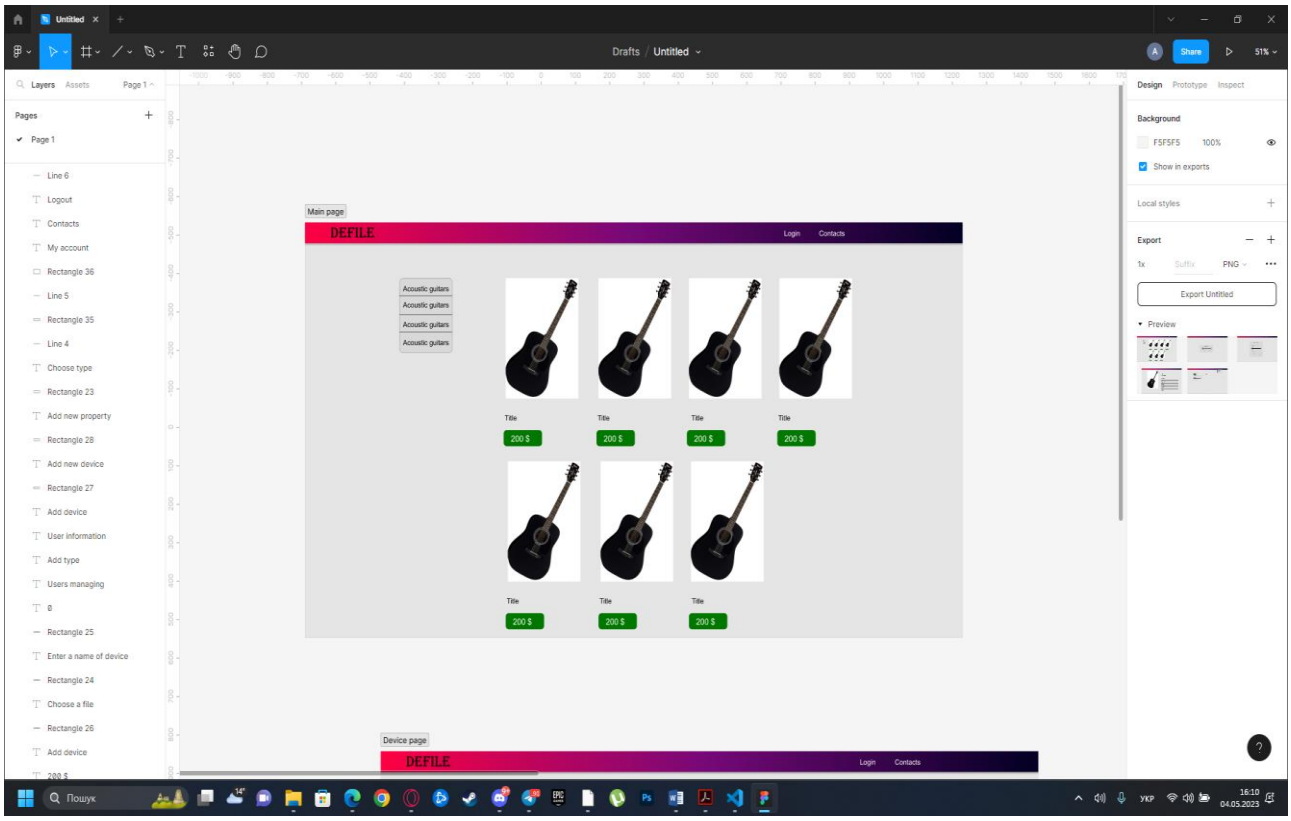

Рисунок 2.3 – Програма Figma з готовим дизайном головної сторінки

– сторінка з особистим кабінетом буде мати декілька контейнерів.. В них буде показано інтерфейс з додаванням типу та товару. У випадку з першим контейнером зліва буде кнопка, яка при натисканні буде показувати список зі всіма типами, 2 поля, де треба вказати назву товару та контейнер для вибору зображення, яке необхідно завантажити на сервер саме для цього товару. Справа буде показана кнопка, при натисканні якої створюється колонка для введення характеристики товару з можливістю видалення.

Структуру розроблюваного сайту буде зображено на рисунку 2.4. Ті комірки, які залиті чорним кольором будуть доступні для всіх, а ті, які не обведені – білим.

Саме такий інтерфейс буде найбільш зручним у користуванні та найбільш дружнім як з користувачем, так і для адміністратора, який буде слідкувати за товарним наповненням.

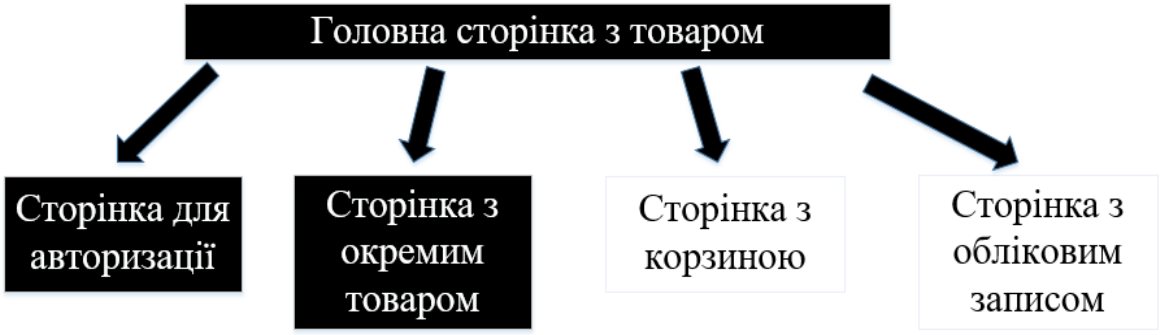

Рисунок 2.4 – Структура сайту

2.2 Вибір засобів, методів та алгоритмів вирішення поставленого завдання

Перш за все, треба розібрати саме серверну частину проекту. Серверна частина буде написана на мові JavaScript з використанням платформи Node.js, так як ця платформа містить дуже велику кількість різних пакетів, бібліотек чи плагінів, які в рази спрощують написання коду. Варто для початку розглянути, які саме пакети будуть встановлені:

– Bcrypt – шифрує паролі в самому токені, що унеможливлює витік всієї інформації користувача;

– Cors – пакет, який дозволяє взаємодіяти клієнтській частині з серверною;

– Dotenv – дозволяє налаштовувати окремі змінні середовища;

– Express – пакет, який дозволяє розробляти веб-додатки та API до них з мінімальною кількістю коду;

– Express-fileuload – доповнення до Express, яке дозволяє вже завантажувати файли безпосередньо у саму базу даних не використовуючи суб’єкт керування базами даних;

– Jsonwebtoken – модуль для роботи з JWT токенами;

– Pg – модуль для роботи з суб’єктом управління базами даних PostreSQL;

– Pg-hstore – доповнення до попереднього модуля;

– Sequelize – ORM (Object-Relational Mapping або об’єктно-реляційне відображення), технологія програмування, яка дозволяє зв'язати об'єктно-орієнтовану модель даних з реляційною базою даних. Це означає, що замість того, щоб вручну писати SQL-запити для отримання даних з бази даних, можна працювати з об'єктами в програмі та додавати, змінювати та видаляти дані, а ORM-бібліотека буде перетворювати ці операції на відповідні SQL-запити та виконувати їх на базі даних. Це значно спрощує розробку програм та зменшує кількість коду, що потрібен для взаємодії з базою даних;

– Uuid – пакет для ідентифікації об’єкту. Це потрібно для того, щоб при створенні якогось об’єкту в таблиці бази даних появлявся окремий ідентифікатор, і за його допомогою вже можна було б знайти цей самий об’єкт в додатку;

– Nodemon – пакет для розробника, який спрощує розробку додатку на платформі Node.js. Наприклад, при перезавантаженні окремого елемента коду, він відразу перезавантажує цей фрагмент, що економить час.

Було прийнято рішення запити вже у самому додатку, використовуючи Express та Sequelize. Тому нам необхідно використовувати програму, яка буде тестувати API, які вийшли під час написання коду, на різні помилки. Для цього можна використовувати програму Postman. Він досить простий у використанні, має історію всіх запитів. Також є окреме вікно для тестування запитів (будь це просто отримання даних, їх видалення, чи додавання) та вікно з результатом, де буде показувати статус запиту, в чому саме помилка (якщо сервер повернув запит з помилкою) і саме результат запиту (може бути у декількох виглядах, наприклад у вигляді json, xml, чи html формату, та у вигляді саме сторінки у браузері). Цей сервіс дасть нам змогу протестувати запити, не використовуючи клієнтську частину проекту.

Система авторизації та реєстрації користувачів буде відбуватись через систему JWT (JSON Web Token). Це один з способів представлення даних для передачі між двома чи більше сторонами у вигляді json-об'єкту[7]. Приклад одного такого токена показано в лістингу 2.1

Лістинг 2.1 – Приклад JSON Web Token

```
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJvcmFvcm
FvcmFAb3JhLmNvbSIsInJvbGUiOiJBRE1JTiIsImlhdCI6MTY4MzIxNTk2NSwiZXhwIjoxNjgzMzAyMzY1f
Q.zA80IAcNJefDjSKo8eHhnfi9hm1CFUkn5BrHfbxtzrk"
```

Кінець лістингу 2.1

Сам токен складається з трьох частин:

– Header – заголовок;

– Payload – основна інформація, яка міститься в токені, яку треба розшифрувати;

– Signature – підпис (можливі випадки, коли вони можуть бути відсутні);

Кожна частина токену розділяється крапками. За допомогою цього легко розділяються всі елементи токену.

За допомогою сайту https://jwt.io можна розшифрувати саме цей токен. Варто лише вставити той токен у велике вікно, щоб переглянути, чи добре токен зашифрував інформацію[7]. Приклад вдалого розшифрування токену на вказаному сайті показано на рисунку 2.5.

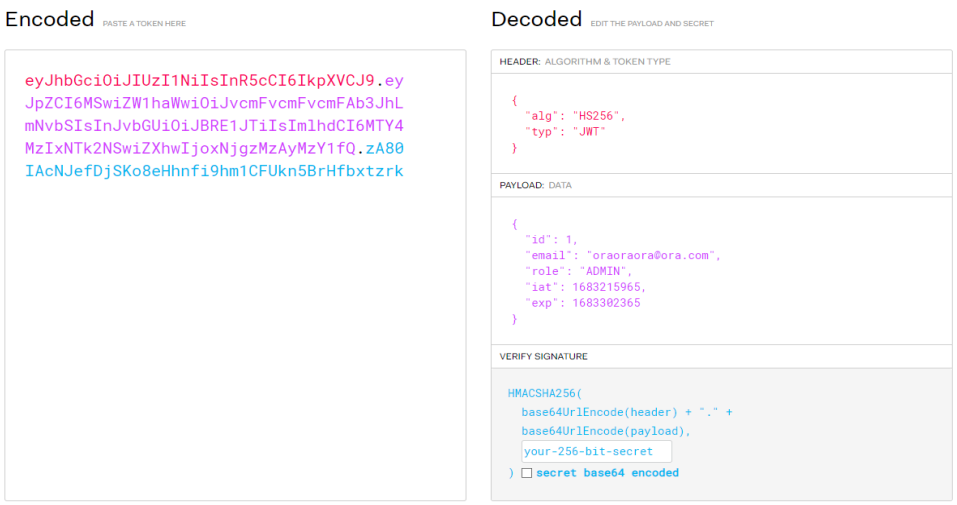

Рисунок 2.5 – Процес розшифрування JWT

Використання саме JWT дозволяє кожен раз при авторизації змінювати токен, і відсилати на сервер, що робить крадіжку облікового запису дуже складною.

Також, як можна замітити, токен не показує саме пароль (це треба буде налагодити вже саме на сервері, що потім буде розглянуто більш детально під час розробки додатку), що також ускладнює крадіжку облікового запису.

Технології, які будуть використовуватись у серверній частині проекту розглянуто, ще необхідно розглянути клієнтську частину. Основна частина контенту, а саме верстка інтерфейсу у нас буде виконуватись через бібліотеку React, так як вона, як і Node.js має чималу кількість пакетів, які суттєво можуть прискорити та спростити розробку проекту:

– Axios – плагін, який змушує клієнт працювати з серверною частиною, тобто, виконувати всі запити, які були вказані у коді клієнту;

– Jwt-decode – розшифровує JWT токен, який був викликаний сервером безпосередньо у самому клієнті, що дасть змогу керувати цим токеном безпосередньо в браузері;

– Mobx – бібліотека, яка дозволяє легко керувати змістом додатку, оскільки для його користування не треба велика кількість коду;

– Mobx-react-lite – зв’язує весь функціонал Mobx з React;

– React-bootstrap – пакет, який має елементи для стилізації сторінок, що значно спрощує та скорочує розробку клієнтської частини проекту;

– React-dom, React-router та React-router-dom – ці плагіни дозволяють з’єднувати сторінки у проекті, тим самим створюючи навігацію;

– React-icons – великий збірник SVG зображень, які можна легко стилізувати під окремі об’єкти.

Висновки до розділу 2

У даному розділі буди представлена база даних, яка буде найбільш приближеною до такої, яка має бути в звичайному Інтернет-магазині, та яким чином вона повинна бути приєднана до клієнтської частини проекту, а саме наведено таблиці та їхні зв’язки між ними, обрано СУБД, на якій буде само працювати база даних. Розглянуто спосіб розробки авторизації та реєстрації користувачів через використання технології JWT, та детально описано принцип його роботи. Було обрано такі платформи для розробки додатку, а саме Node.js для серверної частини та React для клієнтської та бібліотеки до них (з детальним описом про кожну з них), за допомогою яких треба буде розробляти проект.

РОЗДІЛ 3
РОЗРОБКА ВЕБ ДОДАТКУ

3.1 Практична реалізація об’єкта проектування

Для розробки проектів такого типу потрібно розділити серверну (її зазвичай прийнято розробляти першою, тому зараз буде розглянуто саме її), та клієнтську. Для початку розробки, необхідно встановити платформу Node.js. Після цього, у терміналі необхідно прописати шлях до папки з сервером, після чого прописати команду npm init -y. Ця команда встановить JSON файл, який буде мати практично всю інформацію про проект (у нашому випадку, ще такий буде у клієнтській частині, але там буде створюватись автоматично разом зі створенням React додатку).

Далі необхідно встановити необхідні пакети, які були вказані в розділі 2. Вони встановлюються за командою npm install (тут необхідно далі вказати назву пакету). Після успішного встановлення хоча б одного пакету, можна побачити, що появилась папка Node_modules. Там будуть подалі знаходитись всі необхідні файли для роботи.

Після встановлення всі нам необхідних пакетів, переходимо до кодування. Щоб почати працювати з пакетами, які були встановлені, треба їх імпортувати в js файл (таким чином можна імпортувати не лише пакети, але й інші файли, які повинні взаємодіяти з файлом, в який треба імпортувати код. Приклад імпортування декількох пакетів показано в лістингу 3.1.

Лістинг 3.1 – Приклад імпортування в Node.js

```
const cors = require('cors')
const fileUpload = require('express-fileupload')
const router = require('./routes/index')
```

Кінець лістингу 3.1

В даному прикладі, імпортування виконувалося як публікація змінної з використанням функції require(). В дужках було прописано або шлях до

необхідного об’єкту (якщо це звичайний код у файлі), або назва пакету, з яким треба зв’язатись.

Для створення клієнтської частини додатку необхідно у терміналі вказати місцезнаходження папки, де повинна знаходитись сама клієнтська частина. Після вказання місцезнаходження треба вписати команду npx create-react-app (назва проекту). Під час виконання цієї команди будуть встановлюватись всі необхідно мінімальні пакети, які треба для роботи з React, і самі його компоненти (разом з файлом package.json, де знаходиться вся інформація про встановлені пакети). Також створяться дві папки src та public. Для розробки самого додатку треба використовувати саме src, де будуть міститись всі необхідні для розробки файли.

Пакети тут встановлюються абсолютно ідентично, як і у випадку з серверною частиною. Тільки замість кореневої папки серверу треба встановлювати їх у кореневу папку клієнта.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Щоб отримати інформацію про помилки, які будуть зв’язані з сервером, треба окремо створити код для їх відображення. Для цього треба створити клас, який буде відображати всі помилки, які буде повертати програма. У самому класі треба оголосити конструктор, у якому будуть вказані змінні зі статусом помилки та їх повідомленням. Далі опишемо 3 помилки, які можуть потенційно трапитись на сервері, а саме:

– сторінку з запитом вказано не правильно (404);

– сервер прийняв запит, але відмовився його виконувати (403);

– сервер не може обробити запит безпосередньо через клієнт (500).

Виведення повідомлення з помилкою буде виконуватись через метод, у якому після імені вказується повідомлення з конструктора. У самому методі необхідно вказати повернення класу, де вказується у дужках статус помилки, та його повідомленням.

Для подальшої роботи методу треба буде його ще запустити через функцію у проміжному програмному забезпеченні. Воно якраз буде показувати та повертати у консоль помилку зі статусом та його повідомленням у вигляді JSON формату. Також у випадку, якщо таку помилку у класі зі списком помилок виявлено не було, просто повертається помилка зі статусом 500, де у повідомленні вказується, що помилка не була передбачена.

Далі зрозуміло, у чому може бути помилка, обернувши все у блок try/catch, де у try буде виконуватись сам запит на сервер, а у catch відповідно виклик повідомлення про помилку. Далі можна тестувати всі запити за допомогою програми Postman. Оскільки браузер по замовчуванню може виконувати лише запити отримання інформації (GET). Для інших запитів треба вже створювати код для виконання тих самих запитів на сервер. Postman дозволяє виконувати всі запити на сервер без використання стороннього коду.

Сам Postman має досить простий інтерфейс, у якому досить просто освоїтись, де він буде вказаний у рисунку 3.1.

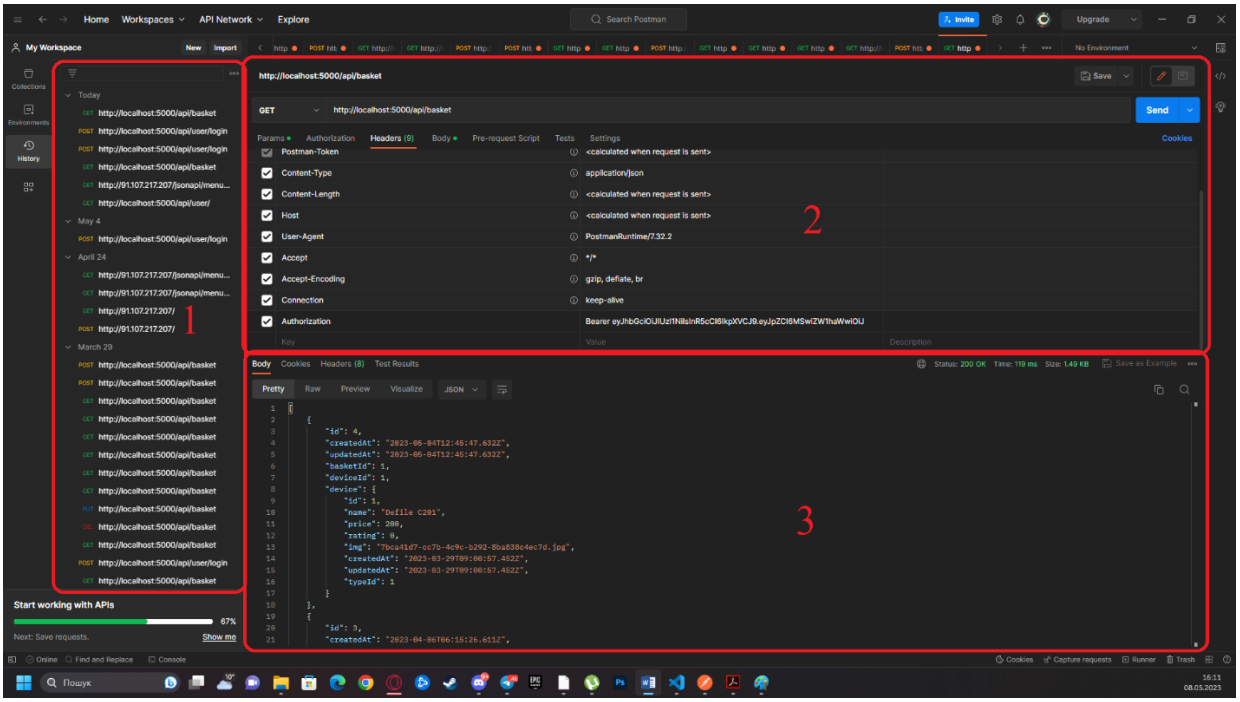

Рисунок 3.1– Інтерфейс програми Postman

Далі буде розглянуто кожен елемент, які показані на рисунку 3.1:

– елемент 1 – основний функціонал програми Postman. (Наприклад, можна створити групу з запитами для тестування або перегляди саму історію запитів);

– елемент 2 – основна інформація про запит. Над цим елементом показано вкладки, це ті запити, які на даний момент обробляються. У самому елементі є сам тип запиту (GET, POST, DELETE і так далі), біля нього справа розташовано строку, де саме прослуховується сервер (у нашому випадку це локальний сервер). Під цими елементами розташоване тіло запиту. Тобто тут можна задати певні змінні, які можна надіслати на сервер, перевірити, чи надсилається запит на сервер при певній умові (наприклад, чи є користувач авторизованим на даний момент на сервері, і якщо так);

– елемент 3 – вікно з відповіддю від сервера. Зверху справа показується статус запиту, за скільки часу він виконав запит та скільки той запит важить. Над JSON кодом знаходяться кнопки, які можуть показати у довільному форматі як може відобразитись відповідь (Також замість JSON формату може показатись вигляд як HTML, XML, або просто текст). Сам текст є результатом запиту. У даному прикладі запит було виконано успішно.

Для налагодження клієнтської частини нічого вже придумувати не треба, оскільки при встановленні React додатку в пакетах добавлені функції відображення помилок відносно клієнтської частини (У випадку, якщо з’явилась якась помилка, React, замість відображення сторінки, покаже помилку та її можливе джерело). А для верстки та швидкої стилізації самої сторінки можна використовувати програмне забезпечення розробника, яке доступне практично у кожному браузері. Для стилізації просто вибираємо необхідний нам елемент, та у вікні справа просто або змінюємо всі значення, якщо вони є, або добавляємо. У випадку, якщо все влаштовує, можна просто це все добавляти вже сам клас у CSS файл, де він буде прив’язаний до необхідного файлу.

Також, за допомогою цього можна в консолі побачити всі значення, які ми задавали для консолі або всі помилки, які нам, скоріше за все, необхідно буде вирішити.

3.3 Розробка бази даних та оформлення серверної частини додатку

Розробляти базу даних будемо на основі PostgreSQL. Створити базу даних можна напряму в панелі адміністратора самого СУБД. Інтерфейс самого СУБД буде показано на рисунку 3.2, за допомогою якого буде легше розібратись в навігації

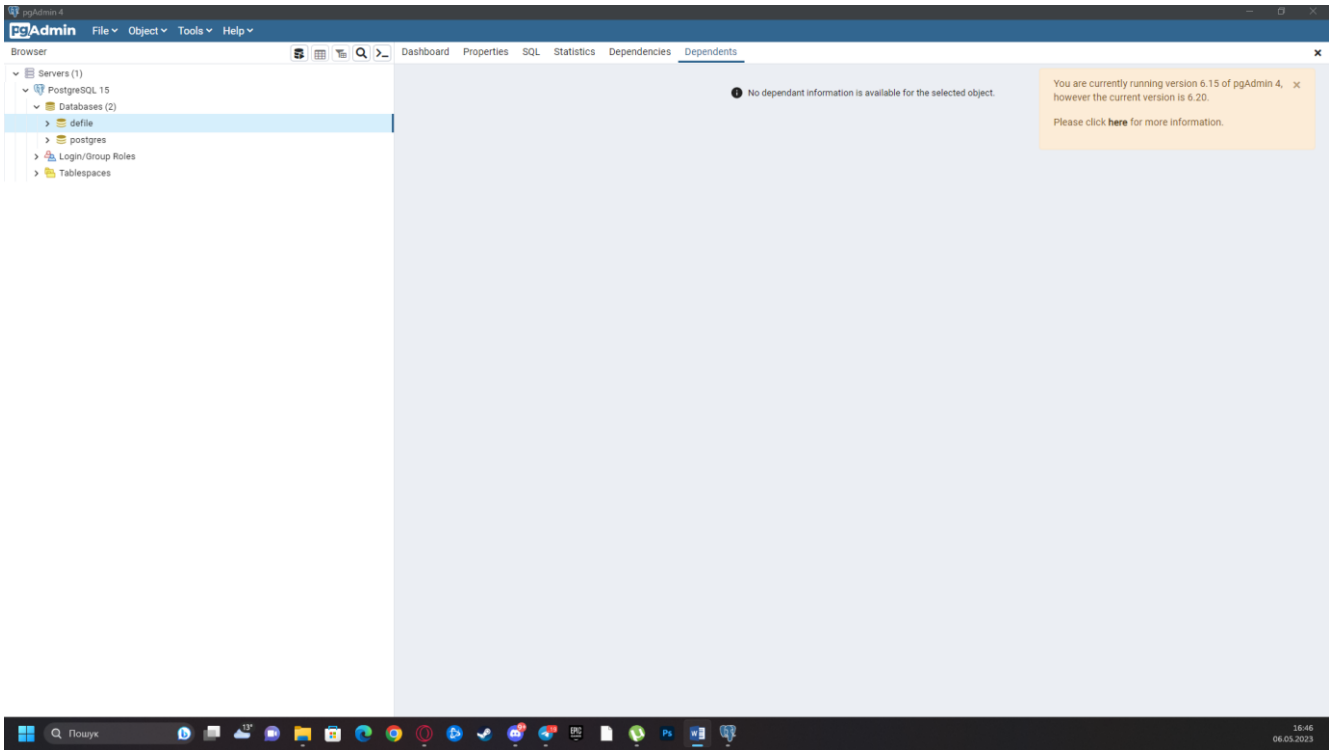

Рисунок 3.2 – Інтерфейс панелі адміністратора PostgreSQL

Створити базу даних модна натиснувши правою кнопкою миші на Databases, де треба навести потім на Create та натиснувши Database. Далі появиться вікно для створення бази даних. Можна замітити, що тут є багато вкладок, які можуть детальніше налаштувати базу даних. Детальне налаштування не знадобиться, тому просто вписуємо назву бази даних.

Далі необхідно вказати, з якої бази даних сервер буде приймати інформацію. Це можна зробити за допомогою пакету Sequelize та змінного середовища dotenv. Створюємо у кореневій папці файл .env, у якому ми вписуємо всі змінні, які ми вказували під час встановлення PostgreSQL (назва користувача, пароль, хост (де тримається сервер), порт СУБД), порт на якому той сервер слухається, назву бази даних та секретний ключ. Змінну з портом просто вписуємо як PORT, також так само записуємо секретний ключ (SECRET_KEY), а перед всіма змінними з СУБД треба вписати префікс DB, щоб воно виглядало, наприклад, як DB_USER.

Але цього ще мало для роботи з базою даних. Треба ще підключити це через пакет Sequelize. Для цього створюється JS файл, де імпортується Sequelize, та за його допомогою добавляються всі змінні, наприклад як process.env.DB_USER. Окремо хост і порт оголошується таким чином, щоб сервер зрозумів, що він бере інформацію саме через PostgreSQL, і далі підключається вже до локального хоста з відповідним портом, до якого приєднаний СУБД. Приклад цього можна побачити в лістингу 3.2.

Лістинг 3.2 – Підключення СУБД до сервера

```
{
    dialect: 'postgres',
    host: process.env.DB_HOST,
    port: process.env.DB_PORT
}
```

Кінець лістингу 3.2

Далі треба підключити цей об’єкт у кореневий файл сервера index.js, імпортуючи його. Далі оголосити його в асинхронній функції, через authenticate (за його допомогою буде підключатись база даних) та sync (буде порівнювати стан бази даних зі схемою даних) де вже надалі буде прослуховується сервер.

Так як створювана система не потребує створення таблиць напряму у СУБД, треба їх створювати через JS файл, тим самим налагодивши всі зв’язки,

згідно рисунку 2.1. В лістингу 3.3 буде показано приклад коду, за допомогою якого буде створюватись модель, та в якому буде розглянуто кожен елемент коду більш детально.

Спочатку у sequelize оголошується функцію define, яка саме створює модель. Першим параметром у прикладі вказано ‘device’, це буде назвою моделі. У фігурних дужках всередині функції описано всі змінні, які будуть використовуватись у таблиці.

Лістинг 3.3 – Створення моделі в базі даних зі змінними

```
const Device = sequelize.define('device', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    name: {type: DataTypes.STRING, unique: true, allowNull: false},
    price: {type: DataTypes.INTEGER, allowNull: false},
    img: {type: DataTypes.STRING, allowNull: false},
})
```

Кінець лістингу 3.3

Після назви змінної у нас ідуть всі налаштування для кожної змінної. Type у нас відповідає за тип змінної. Для оголошення самого типу треба ще імпортувати DataTypes через sequelize.

За основний ключ у даній таблиці відповідає параметр primaryKey. Без нього таблиця у базі даних, як мінімум, не буде працювати так як треба, оскільки побудувати зв'язки з такою таблицею буде неможливо. Також для ключа, щоб автоматизувати створення даних у таблиці, треба добавити параметр autoIncrement, та зробити його true. Це дасть змогу автоматично створити один рядок у таблиці.

Далі будуть розглянуті параметри, які стосуються виключно таблиці device. Кожна змінна у name має бути унікальна, оскільки при реєстрації умовного товару з абсолютно ідентичним іменем, такий товар не повинен створюватись. Параметр allowNull дозволяє вказувати пусте значення в параметрі. Так як у даному випадку у кожній змінні має бути хоча б одне

значення (будь це назва товару чи його ціна) то у нас у кожній змінні даний параметр дорівнює false.

Далі потрібно встановити зв’язки між цими таблицями, як було продемонстровано на рисунку 2.1. Розглянемо в лістингу 3.4 приклад зв’язку між таблицями. Там показано два зразка того, як можна утворити зв’язок між елементами. У першому випадку у нас тут оголошується зв’язок один до одного, то у другому один до багатьох.

Лістинг 3.4 – Утворення зв’язку між таблицями

```
User.hasOne(Basket)
Basket.belongsTo(User)

User.hasMany(Rating)
Rating.belongsTo(User)
```

Кінець лістингу 3.4

Тепер треба всі ці моделі експортувати в СУБД для подальшої роботи через module.exports, де всередині прописуємо всі моделі, які були спочатку створені у цьому js файлі.

Щоб перевірити, чи створились таблиці вдало, треба розглянути у терміналі, чи не появилась чимала кількість запитів, а у панелі адміністратора PostgreSQL повинні появитись таблиці, назви яких повинні співпадати з назвою моделі.

3.4 Захист інформаційно-комп’ютерної системи

Як вже відомо, у даному проекті буде використовуватись JWT технологія. Для захисту від можливих витоків зайвої інформації під час реєстрації або авторизації, треба зробити так, щоб сервер зміг розв’язати шифр JSON Web Token, при тому, щоб ніде не пройшов витік секретної інформації, який буде знати лише користувач облікового запису і був записаний лише в базі даних з користувачами.

Для генерації самого токена, треба імпортувати пакети jsonwebtoken та bcrypt у файл, де знаходиться код для авторизації та реєстрації користувача. Ще перед створенням запитів, треба створити код, який саме буде генерувати такий токен. Для цього, у тілі зі змінними вказуються всі змінні з бази даних, які використовуються для генерації. Всі ці змінні далі вписуються у вигляді об'єкту в саму функцію використавши функцію jwt.sign, після змінних беремо з файлу .env змінну SECRET_KEY, і далі так само вписуємо опції. Тут буде одна опція, це обмежена кількість життя самого токена (expiresIn). Ця опція буде рівна добі, тобто, через добу цей токен самостійно знищується.

Далі буде розглянуто приклад саме авторизації користувача з використанням JWT токена. Приклад запиту авторизації показано в лістингу 1.

Це буде асинхронний метод, де в якості змінних виступають отримання змінних від користувача (через функцію req), повернення відповіді (у нашому випадку це буде якраз JWT токен), та next, який у випадку чого буде сповіщати про помилку від користувача.

Одразу після написання коду, програма надсилає запит до сервера, чи є дана пошта зареєстрована у базі даних. У випадку, якщо ні, то програма скаже, що дана пошта не є на даний момент базі даних, і запропонує користувачеві зареєструвати її.

Якщо ж дана пошта була зафіксована в базі даних, то треба буде впевнитись, що пароль, який ввів користувач, співпадає з тим, що лежить в базі даних. Оскільки пароль в базі даних буде зашифрований, необхідно його розшифрувати за допомогою функції compareSync, яка стає доступною за допомогою бібліотеки bcrypt. У випадку, якщо пароль введено невірно, сервер про це повідомить користувача, і запит не виконається, залишивши за собою помилку. Якщо ж все вірно, то буде надсилатись JWT токен на сервер, і користувач зможе зайти на свій обліковий запис.

У випадку з реєстрацією користувача, все практично робиться так само. Тільки вже помилки від користувача будуть трохи інші, наприклад інформація введена неправильно, або обліковий запис з такою поштою вже існує. Як вже

стало відомо з авторизації, пароль у базі даних лежить вже шифрований. Він шифрується за допомогою функції з пакету bcrypt.hash(), де у дужках вказуються такі змінні, як пароль, які введе користувач, та кількість спроб хешування. Велику цифру ставити не варто, оскільки розшифрувати пароль назад буде досить складно. У нашому випадку їхня кількість буде дорівнювати п’яти.

Тепер, якщо всі попередні дії виконуються належним чином, буде створюватись персональний кошик для користувача, і виконається запит на створення облікового запису користувача. Авжеж, після цього створиться такий же JWT токен.

3.5 Розробка клієнтської частини додатку

Після встановлення пустого додатку React, та завантаження необхідних пакетів для подальшої роботи та полегшення розробки, треба ще його підготувати. Для цього необхідно у файлі Index.js створити класи, з яких ми будемо брати інформацію з клієнту, а потім, під час розробки, приєднати до сервера, і вже звідси збирати всю інформацію. Це робиться за допомогою компонента createContext, щоб потім можна було брати інформацію вже використовуючи useContext.

Після цього необхідно створити глобальне сховище, де буде саме зберігатись вся інформація. Для цього треба використати компонент makeAutoObservable з пакету mobx. Оголошується для кожного сховища клас, у якому міститься конструктор зі всіма необхідними змінними та запитами до сервера. У випадку з користувачами там у конструкторі змінна, яка повинна перевіряти, чи є цей користувач зареєстрований та його дані. Клас з товарами містить інформацію про типи, товари, корзини. Також тут буде створена можливість пагінації товарів, щоб не було показано лише великого списку з товарами.

Оскільки у веб-додатку буде не одна сторінка, потрібно зробити навігацію по сайту. Це можна зробити за допомогою пакету React-router-dom.

Для початку у файлі App.js імпортується ця бібліотеку, а саме функція BrowserRouter, та покривається всією функцією всі значення, які будуть повертатись. У середині вставляється компонент, який буде відповідати за логіку навігації, який з буде розглянуто потім детально.

Далі необхідно створити шляхи для навігації. Кожен шлях у нас буде мати назву, та компонент (у нашому випадку це вже буде сторінка), до якого користувач може приєднатись. Приклад прокладання шляху для авторизованих користувачів буде показано у лістингу 3.5, у випадку з неавторизованими користувачами, опис дуже схожий, тільки треба оголосити шляхи у іншому масиві.

Лістинг 3.5 – Шляхи для навігації по веб-додатку

```
export const authRoutes = [
    {
        path:ADMIN_ROUTE,
        Component: Admin
    },
    {
        path:BASKET_ROUTE,
        Component: Basket
    }
]
```

Кінець лістингу 3.5

Для реалізації логіки навігації треба імпортувати компоненти Switch, Route, Redirect з пакету React-router-dom. Прописуємо у return компонент Switch, далі будемо перевіряти, чи є на даний момент користувач авторизований. Код з реалізацією логіки буде показано у лістингу 2 (Додаток А).

Для початку, беруться дані про користувача через useContext, який вже налаштований у файлі Index.js. Далі у поверненні будуть оголошуватись всі можливі маршрути навігації для користувача. У випадку, якщо користувач не

авторизований, то йому доступні не всі можливості та сторінки у додатку, якщо ж авторизований, то навпаки, йому доступні всі сторінки. Також, якщо ж у пошуковому рядку після адреси ввести якісь інші символи, що не вказані у шляхах, то користувача буде повертати на початкову сторінку. В принципі такої логіки для навігації по інтернет магазину буде достатньо.

Тепер можна розглянути оформлення та стилізацію самого додатку. Його треба зробити ідентичним, яким було зроблено раніше у програмі Figma.

Щоб полегшити роботу зі стилізацією об'єктів, треба використати таку бібліотеку, як react-bootstrap. Ця бібліотека має дуже велику базу з різними елементами.

Для створення та подальшої стилізації навігаційної панелі буде використовуватись такий елемент, як Navbar. Оголошуватись цей елемент буде у файлі App.js. Але спершу, треба оформити цю саму навігаційну панель, приклад якої буде показано у лістингу 3 (Додаток А).

Можна взяти вже готовий пакет з сайту самого пакету (саме там можна знайти більшість всіх готових елементів). Копіюємо вміст скопійованого з сайту матеріалу у файл, який буде імпортуватись у return.

Весь елемент обводиться у функцію observer, що дозволить більш раціонально використовувати навігацію по додатку. Саме посилання по іншим сторінками можна робити за допомогою метода useNavigate, що можна імпортувати з пакету react-router-dom. Але, щоб скористатись цією функцією, треба оголосити її через якусь змінну, у нашому випадку це буде navigate. Тепер для оголошення функції по натисканню на елемент, треба у якомусь елементі оголосити опцію onClick, де вже далі вписується ця функція.

Для зміни елементів (які будуть змінюватись залежно від того, чи є користувач авторизований), треба виконати перевірку. Для цього у return, після оголошення логотипа, потрібно розділити 2 фрагменти коду. Спочатку задається умова, чи буде вона виконуватись. У випадку, якщо вона буде виконуватись, буде відображатись іконка з кошиком (можна взяти дану іконку з пакету, попередньо імпортувавши react-icons), де можна переглянути всі

замовлені товари, та кнопка з спливаючим вікном (nav-dropdown), де можна буде або зайти в особистий кабінет, або вийти з облікового запису. Якщо ж не буде відображатись, то там буде показуватись лише кнопка для входу облікового запису.

Для оформлення головної сторінки будуть вже використовуватись окремі елементи, які будуть оформлюватись самостійно, але все ще за допомогою react-bootstrap. Для контейнеру з типами використовується ListGroup, де будуть братись з бази даних всі типи через функцію map. Справа від контейнеру з типами вже буде відображатись список зі всіма товарами (тут так само інформація береться через функцію map), правда сам контент знаходиться виключно у елементі Row (рядок). Інтерфейс готової головної сторінки показано на рисунку 3.3.

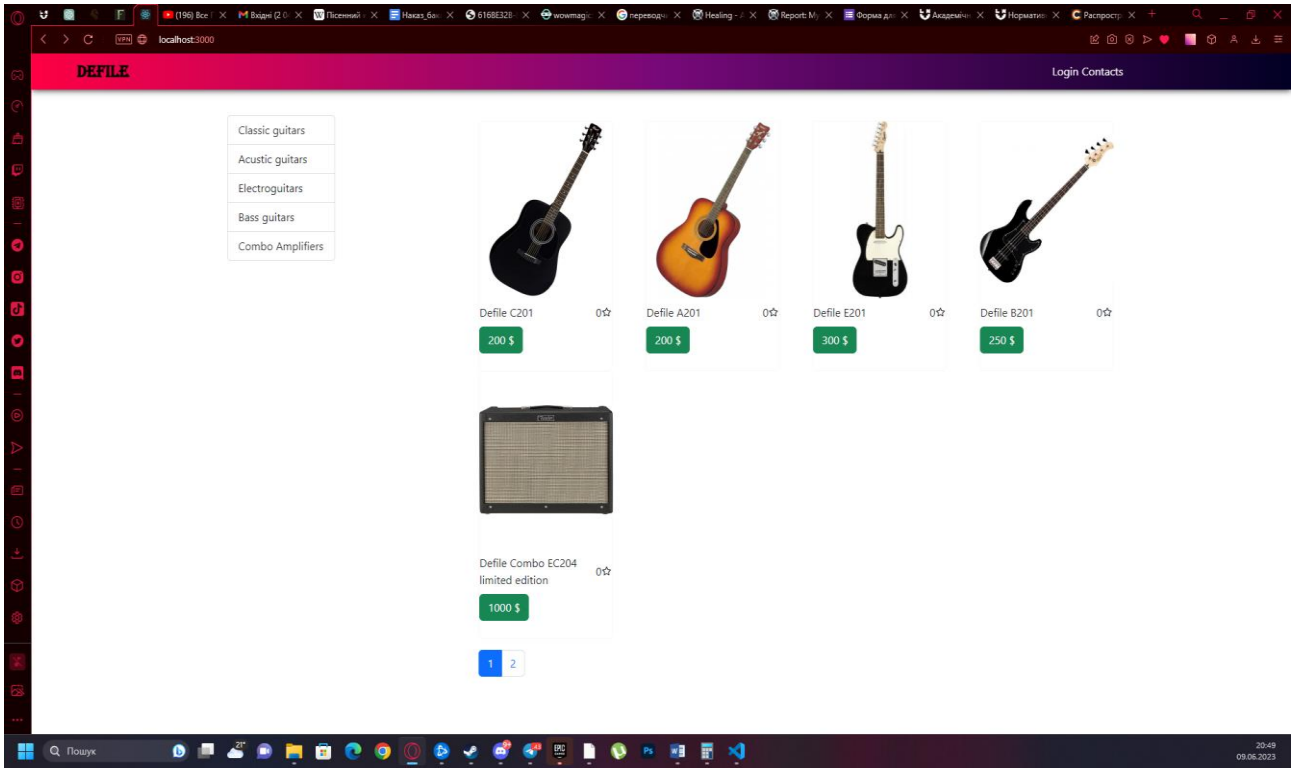

Рисунок 3.3 – Готовий інтерфейс головної сторінки

Також для цього елементу буде організована пагінація. Тобто можна буде гортати сторінки, де на кожній сторінці буде окрема кількість товарів, а не просто весь список зі всіма товарами, оскільки у такому випадку список може

бути безкінечний, а вигляд воно буде мати ніякий. Приклад пагінації буде показано в лістингу 3.6.

Для сторінки з конкретним товаром будуть використовуватись такі елементи з react-bootstrap, як: Image (зображення самого товару, яке буде братись з бази даних), Card (Коротка інформація, яка описана в базі даних (назва та ціна з кнопкою «купити»). Ще варто зазначити, що під Card буде ще список всіх характеристик даного товару. Також для розділення кожного з елементів варто використати такі елементи, як Row та Col.

Лістинг 3.6 – Пагінація елементів списку з товарами

```
const Pages = observer(() => {
    const {device} = useContext(Context)
    const pageCount = Math.ceil(device.totalCount / device.limit)
    const pages = []

    for (let i = 0; i < pageCount; i++) {
      pages.push(i + 1)
    }

  return (
    <Pagination className='mt-3'>
      {pages.map(page =>
          <Pagination.Item
            key={page}
            active={device.page === page}
            onClick={() => device.setPage(page)}
          >
            {page}
          </Pagination.Item>
      )}
    </Pagination>
  )
})
```

Кінець лістингу 3.6

Для сторінки авторизації буде використовуватись контейнер Card. Тут буде все по простому, вікно з полями для вводу електронної пошти та паролю, надпис з посиланням на реєстрацію облікового запису та кнопка «увійти». З реєстрацією все те саме, тільки вікон буде трохи більше.

Основним елементом в сторінці з обліковим записом буде використовуватись такий елемент з react-bootstrap як Tabs, та його дочірній елемент Tab. Цей елемент чимось нагадує вікно браузера, де зверху знаходяться вкладки, по яким можна легко переміщатись, а нижче нього – весь контент. Всього мають знаходитись 2 вкладки:

– з додаванням типу товару (просто поле, де треба вписати новий тип для товарів);

– для додавання товару (тут весь контент поділиться на 2 частини, зліва треба буде обрати тип товару, його назву, ціну та зображення з кнопкою добавити, а справа вікно, де можна добавити окремо характеристики для даного товару);

Приклад готового інтерфейсу особистого кабінету показано на рисунку 3.4.

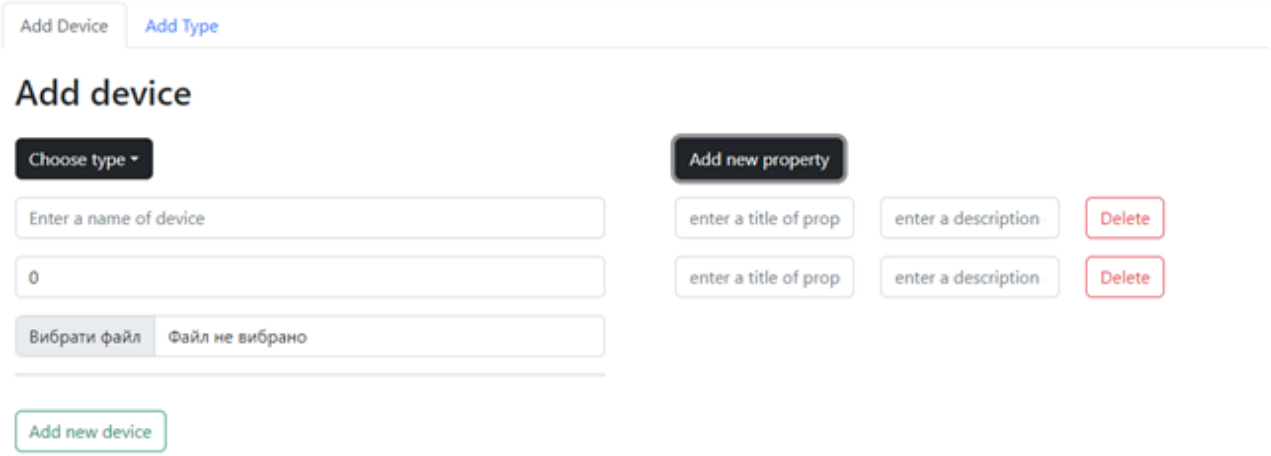

Рисунок 3.4 – Фрагмент інтерфейсу з додаванням товару у магазин

Сторінка з корзиною буде мати дуже простий інтерфейс, буде всього показуватись вся вартість за всі товари, та список з обраними користувачем товарами.

Після того, як всі сторінки вже були створені, згідно макетів у програмі Figma, то треба тепер зв’язати клієнт відповідно з сервером, у якому знаходиться вся база даних. Для цього знадобиться пакет axios. Але спершу

треба дати клієнтській частині проекту зрозуміти, де саме знаходиться сервер. В цьому допоможе глобальне сховище .env, де треба лише оголосити змінну REACT_APP_API_URL, де у результаті вказується шлях до сервера.

Потім створюємо файл, де будуть налаштовані шляхи до сервера. Всього їх буде 2, у одному випадку будуть виконуватись запити, коли користувач не авторизований, у іншому – навпаки, коли авторизований. Усередині всіх методів просто вставляємо змінну baseURL, де вказується змінна з глобального середовища.

Для того, щоб ці дві шляхи відрізнялись можливостями. Треба добавити можливість, що система буде розуміти, що користувач авторизований, та дасть йому більший функціонал. Це можна зробити за допомогою Interceptor. Для його роботи треба ще зробити функцію, яка буде зчитувати JWT токен після авторизації користувача. Далі треба підключити сам шлях до тієї функції через interceptors.request. Залишається лише експортувати з цього файлу самі шляхи.

Далі треба зробити самі запити, використовуючи ці шляхи. Так як API вже готові та протестовані за допомогою Postman, то тут залишається лише прописати шляхи для кожного конкретного запиту в окремому файлі.

Для API користувачів треба добавити ще localstorage, для того, щоб розмістити JWT токен на клієнт для його розшифрування. Він буде розшифровуватись за допомогою бібліотеки jwt-decode.

Тепер залишається лише добавити можливість виконання запитів в клієнтську частику додатку при виконанні певних дій. Розглянемо спочатку відображення всіх елементів на головній сторінці. У випадку з типами, оголошується device через useContext, який оголошувався у файлі index.js. Для початку треба помістити всю функцію в observer. Усередині ListGroup оголошуються type через функцію map. Для того, щоб показувався лише окремий тип, треба оголосити наступні опції у ListGroup:

– active треба поставити значення type.id===set.selected.type.id;

– на опцію onClick оголосити функцію setSelectedType.

Список з товарами буде оголошуватись також використовуючи функцію map, попередньо оголосивши device з використанням useContext, та помістивши всю функцію в observer.

У сторінці з окремим товаром для відображення інформації використовується useState, де усередині береться у вигляді масиву. Також треба вирахувати конкретний ідентифікаційний код конкретного товару. Він буде вираховуватись за допомогою useEffect, де береться запит на отримання окремого товару, і потім дані надсилаються на клієнт.

Вся інформація оголошується таким чином, назва у масиві useState та сама змінна у базі даних. У випадку з інформацією оголошується та змінна, яка вказана всередині значення самого useState, та елемент масиву

Для авторизації та реєстрації користувачів у клієнтській частині буде виконуватись функції. У випадку з авторизацією оголошується блок try/catch. У try оголошується змінна, яка буде збирати дані з сервера. Сервер ставить авторизованим користувача, і переносить клієнт на головну сторінку. Якщо ж користувач ввів дані неправильно, то вилізе повідомлення, у якому буде сказано, що користувач ввів щось неправильно. Реєстрація виконується аналогічно, лише варто вказати те, що вводяться більше даних у сервер.

Так як сторінка з обліковим записом містить в собі декілька вкладок, розглянемо декілька з них. Розглянеться спочатку додавання типу. Оголошується useState, де буде братись інформація з серверу. Використовуючи змінні у useState, створюється функція, де виконуватиметься запит, беручи інформацію з поля у клієнті. Запит буде виконуватись при натисканні кнопки знизу, де у ній вписуємо опцію onClick, де буде вписуємо значення цієї функції.

У випадку з товарами ситуація досить складніша. Тут так само до кожної змінної в базі даних оголошується змінна через useState. Далі оголошується функція, де будуть братись значення з форм у вигляді FormData. Детально код буде показано у лістингу 4 (Додаток А). До кожної змінної оголошується метод append, за допомогою якого буде виконуватись прийняття інформації у FormData. Потім ця FormData буде приймати всі значення у базу даних.

Також треба приділити увагу окремо на зображення, тип та масив з інформацією. У випадку з файлом оголошується функція, де буде братись зображення через useState. Також у формі, де буде братись зображення оголошується опція type, який буде рівний file та onChange з функцією, яка обговорювалась раніше.

Для типу буде братись значення з таблицями type, де береться звідси ідентифікатор. Масив з інформацією буде оброблятись через JSON.stringify.

Після оброблення інформації, буде виконуватись функція createDevice, де всередині вписується сама FormData.

У випадку з корзиною все дуже просто. Все що треба, це оголосити useContext через index.html, виконати всі запити через useEffect та оголосити всі значення через функцію map. Для того, щоб просумувати всі значення з ціною, так само всі змінні з ціною в одному кошику через map, де потім вписуємо змінну, яка обчислила все в сумі.

Після кожної зміни, або як проект вже буде готовий, можна вже завантажити код у платформу GitHub. Це робиться на випадок того, коли вже у проект були внесені вже такі зміни, що вони поверненню до попередньої версії вже підлягати не можуть. Для цього треба створити репозиторій з відповідними ім’ям. Далі описувати, як завантажувати код на дану платформу не потрібно, так як там показано покроково, що треба далі робити. Після його завантаження показано всі файли, які використовуються в проекті, де можна відкотити попередню версію.

Висновки до розділу 3

У цьому розділі було описано процес розробки веб-застосунку. Показано, яким чином встановлюються всі необхідні компоненти та пакети, які необхідні для подальшої розробки додатку та його подальшої роботи.

Для серверної частини було зроблено базу даних через СУБД PostgreSQL, згідно діаграми, яка була показна на рисунку 2.1. Також було оформлено

зв’язки між ними. Для оформлення запитів використовувались вручну написані API, які були протестовані на платформі Postman.

Показано принцип роботи JWT токена. Розказано, які елементи містить сам JWT токен. Було показано, як можна зашифрувати пароль в базі даних, і під час виконання запитів – його розшифрування для авторизації користувача.

Було розібрано принцип роботи клієнтської частини додатку. Обговорено пакет react-bootstrap, який полегшує верстку та стилізацію веб-застосунку. Було розроблено маршрути для навігації по додатку. Показано, яким чином підключається клієнтська частина додатку до серверу. Реалізовано виконання запитів до сервера вже виключно у самому додатку, згідно в якій сторінці перебуває користувач.

Також було розказано про платформу GitHub, та для чого вона потрібна.

ВИСНОВКИ

В даній роботі було розроблено прототип веб-магазину з продажу музичних інструментів. Результат кваліфікаційної роботи бакалавра повністю підтверджує виконання поставленого завдання. Розроблений додаток з використанням PERN технологій повністю відповідає вимогам, які були описані в даній роботі, а саме: розробка бази даних, інтерфейсу, тестування та налагодження системи.

Було описано технологію PERN та схожих аналогів. Виконано порівняння схожих технологій та обґрунтовано, чого було обрано саме те програмне забезпечення, на якому розроблявся даний продукт.

Реалізовано можливість розглядати товари на головній сторінці. Користувач може оглянути окремий товар, та, за бажанням, замовити.

Окрему увагу було приділено авторизації та реєстрації користувачів. Задля захисту від несанкціонованих дій від шахраїв, була використана технологія JWT токену, що шифрує всю інформацію.

Також у даній роботі реалізовано конкретно адміністраторам у клієнтській частині доповнювати товарний ряд та їх типи. Була розроблена функція, що звичайний користувач не зможе добавляти товари, на відміну від адміністратора.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що обрати Frontend чи Backend? Яка між ними різниця? Пояснення на прикладах. Logos IT Acadamy. URL: https://frontend.lviv.ua/chym-vidriznyayetsya-frontend-ta-backend-rozrobka-shho-obraty/ (дата звернення 10.02.2023).

2. Hari N. Just React!: Learn React the React Way / Narayn Hari. Melbourne, VIC, Australia, 2022. 369 с.

3. Огляд фреймворків JavaScript. Що, для чого і коли використовувати. Dou. URL: https://dou.ua/forums/topic/34739/ (дата звернення 12.02.2023).

4. Backend Languages: The First Step for Becoming a Backend Web Developer. Scaler. URL: https://www.scaler.com/topics/software-engineering/backend-languages/ (дата звернення 10.02.2023).

5. Angular (фреймворк) Wikipedia URL: https://uk.wikipedia.org/wiki/Angular_(фреймворк). (дата звернення 12.02.2023).

6. Ahwan A. A. Top 5 Open Source Postman Alternative - Abdul Aziz Ahwan. Abdul Aziz Ahwan. URL: https://www.abdulazizahwan.com/2022/04/top-5-open-source-postman-alternative.html (дата звернення 15.02.2023).

7. Security of JSON Web Tokens (JWT). URL: https://cyberpolygon.com/materials/security-of-json-web-tokens-jwt/ (дата звернення 16.03.2023).

ДОДАТКИ

Додаток А

Лістинг 1 – Авторизація користувачів за допомогою JWT токена

```
    async login(req, res, next) {
        const {email, password} = req.body
        const user = await User.findOne({where:{email}})
        if (!user) {
            return next(ApiError.internal('User with that email is not existed'))
        }
        let comparePassword = bcrypt.compareSync(password, user.password)
        if(!comparePassword){
            return next(ApiError.internal('Password is wrong'))
        }
        const token = generateJwt(user.id, user.email, user.role)
        return res.json({token})
    }
```

Кінець лістингу 1

Лістинг 2 – Реалізація логіки навігації

```
const AppRouter = () => {
  const {user} = useContext(Context)

  console.log(user)
  return (
    <Routes>
        {user.isAuth && authRoutes.map(({path, Component}) =>
            <Route key={path} path={path} element={<Component/>} exact/>
        )}
        {publicRoutes.map(({path, Component}) =>
            <Route key={path} path={path} element={<Component/>} exact/>
        )}
        <Route path='*' element={<Navigate to='/'/>}/>
    </Routes>
  )
}
```

Кінець лістингу 2

Лістинг 3 – Приклад навігації навігаційної панелі в React

```
const NavBar = observer (() => {
  const navigate = useNavigate()
  const {user} = useContext(Context)
  let [basketOpen, setBasketOpen] =useState(false);
```

```
  const logOut = () => {
    user.setUser({})
    user.setIsAuth(false)
  }

    return (
    <Navbar bg="light" expand="lg" className='background' >
    <Container fluid>
      <NavLink to={SHOP_ROUTE} className="logo">Defile</NavLink>
      <Navbar.Toggle aria-controls="navbarScroll" />
      <Navbar.Collapse id="navbarScroll">
        {user.isAuth ?
            <Nav className="ml-auto">
                <SlBasket onClick={() => setBasketOpen(basketOpen =!basketOpen)}
className={`shop-basket-button ${basketOpen && 'active'}`}/>
                {basketOpen && (
                  <Card className='show-basket' >
                    <Button variant='success' onClick={() =>
navigate(BASKET_ROUTE)}>Create an order</Button>
                  </Card>
                )}

                <NavDropdown title="Account name" id="navbarScrollingDropdown">
                    <NavDropdown.Item onClick={() => navigate(ADMIN_ROUTE)}>My
account</NavDropdown.Item>
                    <NavDropdown.Item href="#action4">
                        Contacts
                    </NavDropdown.Item>
                    <NavDropdown.Divider />
                    <NavDropdown.Item onClick={() => logOut()}>
                        Logout
                    </NavDropdown.Item>
                </NavDropdown>
            </Nav>:
            <Nav className="ml-auto">
                <NavLink onClick={() => navigate(LOGIN_ROUTE)}
style={{color:"white", textDecoration:'none'} }>Login</NavLink>
                <NavLink style={{color:"white", marginLeft:"5%" ,
textDecoration:'none'}}>Contacts</NavLink>
            </Nav>
        }
      </Navbar.Collapse>
    </Container>
  </Navbar>
  );
})

export default NavBar;
```

Кінець лістингу 3

Лістинг 4 – Функція з використанням FormData

```
const addDevice=()=>{
    const formData = new FormData()
    formData.append('name', name)
    formData.append('price', `${price}`)
    formData.append('img', file)
    formData.append('typeId', device.selectedType.id)
    formData.append('info', JSON.stringify(info))
    createDevice(formData)
}
```

Кінець лістингу 4