

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

Інтернет-магазин будівельних матеріалів

Виконав студент групи КН-19-2
Сторожук Сергій Олександрович

Керівник роботи:
ПЕТРОСЯН Руслан Валерікович

Рецензент:
ТЕРЕЩУК Степан Олександрович

Житомир – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **СТОРОЖУК Сергій Олександрович**

Керівник роботи: **ПЕТРОСЯН Руслан Валерікович**

Тема роботи: **«Інтернет-магазин будівельних матеріалів»,**

затверджена наказом закладу вищої освіти від **«05» січня 2023 р., №10/с**

Термін здачі закінченої роботи: **«10» червня 2023 р.**

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Процес організації онлайн маркетингу для людей які цікавляться будівництвом та алгоритми та методи розробки інтернет-магазину

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 лютого 2023 – 24 квітня 2023	Виконано
2	Проектування структури системи	24 квітня 2023 – 12 травня 2023	Виконано
3	Написання програмного коду	12 травня 2023 – 21 травня 2023	Виконано
4	Тестування системи	21 травня 2023	Виконано
5	Оформлення пояснювальної записки	21 травня 2023 – 23 травня 2023	Виконано
6	Попередній захист	1 червня 2023	Виконано

Здобувач вищої освіти **Сергій СТОРОЖУК**
Керівник **Руслан ПЕТРОСЯН**

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з проекту “Інтернет-магазин будівельних матеріалів” та пояснювальної записки. Пояснювальна записка до випускної роботи містить 57 сторінок , 31 рисунка , 32 лістинга та 3 таблиць.

Метою роботи є розробка інтернет-магазину, на якому користувач не виходячи з дома зможе ознайомитись з необхідним для нього продуктом, представленим в інтернет-магазині. Користувач зможе дізнатися вартість продукту, його характеристики, побачити як виглядає продукт зовні

В роботі визначено основні завдання на розробку інтернет-магазину. Було проведено проектування інтернет-магазину, проаналізовані аналоги розробленої системи, розроблено React застосунок, був описаний порядок роботи з інтернет-магазином.

КЛЮЧОВІ СЛОВА: ПРОДУКТ, ІНТЕРНЕТ МАГАЗИН, БАЗА ДАНИХ

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		3

ABSTRACT

The graduation thesis of the bachelor consists of the project "Internet store of building materials" and an explanatory note. The explanatory note to the final thesis contains 57 pages, 31 figures, 32 listings and 3 tables.

The goal of the work is the development of an online store where the user can get acquainted with the necessary product presented in the online store without leaving home. The user will be able to find out the price of the product, its characteristics, see how the product looks from the outside

The work defines the main tasks for the development of an online store. The design of the online store was carried out, analogs of the developed system were analyzed, the React application was developed, and the procedure for working with the online store was described.

KEYWORDS: PRODUCT, INTERNET STORES, DATABASE

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	6
ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ	8
1.1 Постановка задачі	8
1.2 Аналіз існуючих програмних продуктів за тематикою дипломної роботи 9	
1.3 Обґрунтування вибору інструментальних засобів розробки	11
РОЗДІЛ 2 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ	22
2.1 Проектування та розробка програмного продукту	22
2.2 Структура бази даних, взаємодія між собою компонентів системи	24
2.3 Реалізація функціоналу серверної частини інтернет-магазину	30
РОЗДІЛ 3 ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ, ТЕСТУВАННЯ	42
3.1 Інтерфейс сайту	42
3.2 Розробка архітектури сайту	44
3.3 Тестування роботи системи	47
ВИСНОВКИ.....	50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	52
ДОДАТОК.....	53

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface, використовується для взаємодії між фронтендом і бекендом, дозволяє отримувати та відправляти дані.

JWT – JSON Web Token, зберігає інформацію про сесію та дозволи, використовується для аутентифікації.

БД – база даних.

NPM – (Node Package Manager) - це менеджер пакетів для JavaScript, який використовується разом з Node.js., також дозволяє розробникам легко встановлювати, оновлювати та керувати залежностями проекту.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми: Впровадження електронної комерції та розширення сфери інтернет-торгівлі стали невід'ємною частиною сучасного бізнесу. В даний час, коли велика кількість споживачів шукає зручні та доступні способи придбання товарів, інтернет-магазини здобувають все більшу популярність і стають основою ефективного продажу продукції та послуг.

У сфері будівельних інструментів і матеріалів також спостерігається зростання попиту на електронну торгівлю. Клієнти, будівельні підприємства та приватні особи шукають способи зручного та швидкого замовлення необхідних інструментів та матеріалів без витрат на дорогу до звичайного магазину.

Ця кваліфікаційна робота присвячена дослідженню та розробці інтернет-магазину будівельних інструментів, який відповідає потребам сучасного ринку та сприяє зручному та ефективному процесу замовлення та постачання продукції.

Мета: Метою дослідження є надати користувачам сайт на якому не виходячи з дома зможе ознайомитись з необхідним для нього продуктом, дізнатись ціну, побачити всі характеристики даного продукту.

Об'єкт дослідження: процес організації онлайн маркетингу для людей які цікавляться будівництвом.

Предмет дослідження: алгоритми та методи розробки інтернет-магазину.

Практичне значення полягає в тому, що результат роботи можуть бути використані для того щоб користувач не виходячи з дому зміг ознайомитись з товаром та його характеристиками.

Структура та обсяг роботи. Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел із 12 найменувань (1 стор.), 2 додатків(4 стор.) та 32 таблиць лістингу програм. Повний обсяг роботи становить 57 сторінок, з них 52 сторінок основного тексту

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

1.1 Постановка задачі

Основна задача – розробити сайт, який буде включати в себе ефективний продаж будівельних матеріалів через Інтернет, забезпечуючи зручний і надійний процес покупки для клієнтів. Основні завдання які потрібно виконувати включають наступне:

1. Великий асортимент: Забезпечте широкий вибір будівельних матеріалів, включаючи різноманітність марок, типів і розмірів. Клієнти повинні мати можливість знайти потрібні їм матеріали на вашому сайті без необхідності шукати в інших місцях.
2. Зручний пошук і навігація: Забезпечте простий і ефективний пошуковий і навігаційний інтерфейс на веб-сайті, щоб клієнти могли швидко знайти потрібні їм товари.
3. Детальна інформація про товари: Забезпечте докладний опис кожного товару, включаючи характеристики, фотографії, інструкції та можливість перегляду відгуків клієнтів. Це допоможе клієнтам прийняти рішення про покупку і впевнитися в якості продукту.
4. Безпека та конфіденційність: Забезпечте безпечний протокол передачі даних, щоб гарантувати конфіденційність особистої і платіжної інформації клієнтів. Використовуйте сертифіковані платіжні системи та застосовуйте заходи для запобігання шахрайству та кібератакам.
5. Зручність доставки: Надайте різні варіанти доставки, такі як доставка на дім або самовивіз з пункту видачі. Забезпечте розумні терміни доставки і розрахунок вартості доставки на основі ваги, розміру та місця доставки.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		8

1.2 Аналіз існуючих програмних продуктів за тематикою дипломної роботи

Для пошуку та аналізу аналогічних інтернет-магазинів, спочатку потрібно виділити основні критерії за якими будуть визначені відповідні інтернет-магазини для дослідження.

Критерії для пошуку існуючих інтернет-магазинів:

1. Асортимент товарів: На сайті інтернет магазину представлена велика кількість продукції

2. Якість продукції: Присутність коментарів до продукту на сайтах, рейтинг товарів та іншу доступну інформацію.

3. Ціни та акції: Порівняння ціни на товари у різних магазинах. Деякі магазини можуть пропонувати акції, знижки або безкоштовну доставку.

4. Користувацький інтерфейс та зручність використання: Зручний інтерфейс легкий пошук товарів та зручна система оформлення замовлення.

1. Sem – це інтернет-магазин з продажу жалюзів. Не зважаючи на вміст, візьмемося основний функціонал

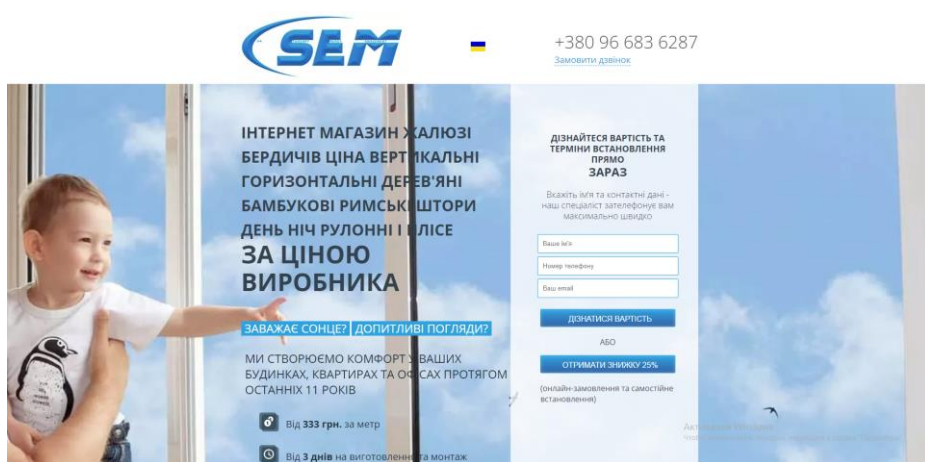


Рисунок 1.1 - Головна сторінка інтернет магазину Sem

Переваги

- Присутня авторизація на сайті

Недоліки:

- не зрозумілий інтерфейс

- відсутня інформація про продукт

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				9
Змн.	Арк.	№ докум.	Підпис	Дата		

- ціна вказана лише за метр
- відсутній показ асортименту
- відсутні коментарі.

2. АЛЛО – інтернет магазин мобільних телефонів, ноутбуків, телевізорів, фотоапаратів, відеокамер, побутової техніки.

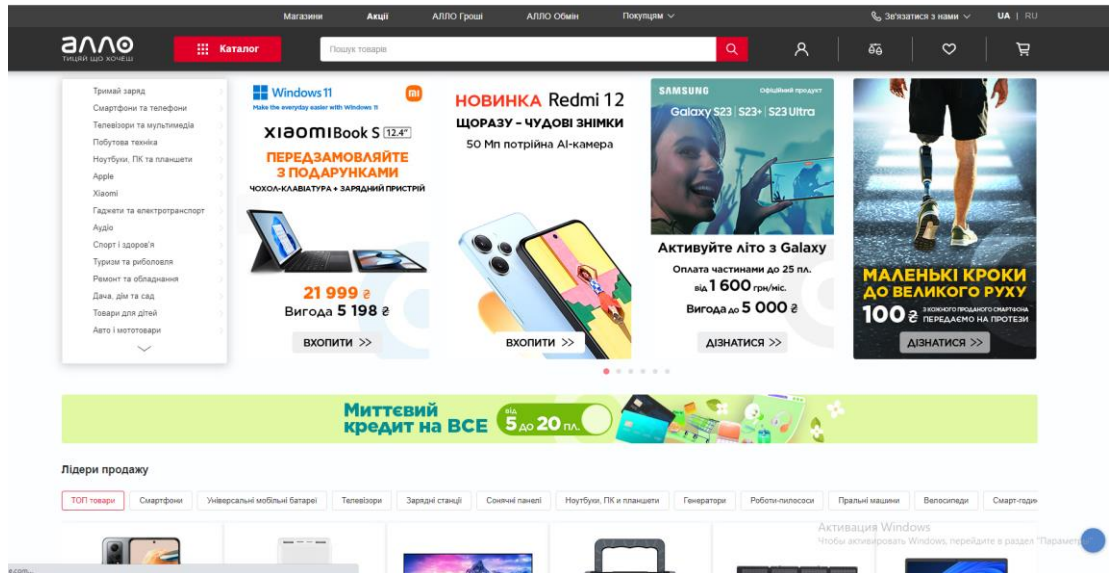


Рисунок 1.2 - Основна сторінка інтернет-магазину АЛЛО

Переваги:

- Легкий, приємний, зручний інтерфейс
- Присутній пошук
- Великий асортимент
- Присутні коментарі та оцінка продукту
- Присутня авторизація на сайті
- Зручна система оплати
- Присутні знижки та акції

Недоліки:

- Немає весь функціонал присутній

Отже, як можна побачити, майже всі інтернет-магазин мають схожий функціонал та загалом мають все, що потрібно, все що відповідає тематиці. Тож визначились критерії, за якими здійснювався пошук та аналіз аналогів, як основу для розробки власного інтернет-магазину.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Тож основні риси проекту це: зручний, легкий та приємний інтерфейс, присутні зображення продукту, є розділи на категорії продукту, присутній пошук, авторизація, можливість оцінювати продукт та залишати коментарі, зручна навігація по інтернет-магазині.

1.3 Обґрунтування вибору інструментальних засобів розробки

Веб-технології існують вже досить давно, тому ні для кого не буде дивним, що вони постійно розвиваються, постійні оновлення, покращуються бібліотеки та фреймворки. Наразі існує безліч підходів до розробки веб-сайтів і розробники використовують різні типи архітектури залежно від потреб проекту. Ось декілька з них:

1.Монолітна архітектура (Monolithic Architecture): Вся логіка додатка розгортається як один цілий блок. Усі компоненти, такі як сервер, база даних, фронтенд і т. д., працюють разом і знаходяться в одному програмному модулі або пакеті. Цей підхід простий у використанні, але може бути складним для масштабування та розширення.

2.Клієнт-серверна архітектура (Client-Server Architecture): Цей підхід передбачає розподіл функціональності між клієнтською та серверною сторонами. Клієнтська сторона відповідає за взаємодію з користувачем і відображення інтерфейсу, тоді як серверна сторона виконує бізнес-логіку, зберігає та обробляє дані. Цей підхід дозволяє кращу масштабованість та можливість розподілу завдань між клієнтом і сервером.

3.Мікросервісна архітектура (Microservices Architecture): Веб-додаток розбивається на невеликі, автономні та взаємодіючі сервіси, кожен з яких відповідає за конкретну функціональність. Кожен сервіс може бути розгорнутий, масштабований та оновлюваний незалежно від інших. Вони комунікують між собою за допомогою API або інших механізмів обміну повідомленнями. Цей підхід дозволяє гнучкість, швидкодію та легкість розробки та масштабування окремих компонентів системи.

4.Серверна архітектура (Serverless Architecture): В цьому підході розробник не має прямого контролю над серверами. Інфраструктура піклується

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		11

про автоматичне масштабування, керування ресурсами і виконання коду. Розробник зосереджується на написанні функціонального коду, який виконується в середовищі, наданому платформою. Цей підхід спрощує розгортання та зменшує затрати на обслуговування інфраструктури.

Кожен підхід має свої переваги та недоліки, і вибір залежить від потреб вашого проекту, його масштабу, складності та інших факторів.

Це лише кілька з багатьох типів архітектур, які використовуються в сучасній веб-розробці. Вибір підходу залежить від конкретних потреб проекту, його масштабу та цілей.

У випадку розробки кваліфікаційної роботи, дивлячись на вимоги до інтернет-магазину, його функціоналу та можливостей найбільш підходить саме клієнт-серверна архітектура.

Слід обрати архітектурний підхід до розробки інтернет-магазину. Дійсно, існує багато різних архітектурних підходів та протоколів для розробки інтернет-магазинів та веб-сервісів. Найбільш поширені це REST API, GraphQL, SOAP, Odata та тд.

Зупинюсь на перших двох.

Так, дійсно, існує багато різних архітектурних підходів та протоколів для розробки інтернет-магазинів та веб-сервісів. Наведу деякі з них:

1. REST API (Representational State Transfer Application Programming Interface): Це стандартний архітектурний стиль для побудови веб-сервісів. REST API використовує HTTP-протокол для взаємодії між клієнтом та сервером. Він базується на роботі з ресурсами та використовує HTTP-методи, такі як GET, POST, PUT і DELETE, для операцій з цими ресурсами. REST API є популярним і простим у використанні підходом.

2. GraphQL: Це запитова мова та середовище виконання, яке дозволяє клієнтам запитувати саме ті дані, які їм потрібні, і отримувати їх у компактному форматі. GraphQL дозволяє клієнтам визначати структуру запиту та отримувати зворотну відповідь з точними даними, що були запитані. Цей підхід дає більшу гнучкість у виборі даних та зменшує надлишкові передачі даних.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Зважаючи на потреби і вимоги інтернет магазину та орієнтуючись на дослідження і аналіз, оберемо REST API архітектурним підходом до розробки ігрового порталу.

Тепер потрібно обрати технології розробки. Виберемо базу даних, їх існую величезна кількість, основна їх різниця використання SQL чи ні, та гнучкість схем та таблиць. Я мене впав вибір на PostgreSQL.

PostgreSQL є відмінним вибором для створення бази даних для інтернет-магазину. Ось деяка інформація та переваги PostgreSQL для цього конкретного використання:

1.Надійність та стабільність: PostgreSQL відомий своєю стабільністю та надійністю. Для інтернет-магазину, де доступність та безпека даних є критичними, PostgreSQL надає механізми відновлення після збоїв, журналування та резервне копіювання, що забезпечує збереження даних та високу доступність.

2.Розширені можливості: PostgreSQL має багатий функціонал, який допомагає створити потужну базу даних для інтернет-магазину. Він підтримує складні типи даних, геопросторові запити, повнотекстовий пошук, транзакційну безпеку та інші функції, які дозволяють зберігати та керувати даними про товари, замовлення, клієнтів, оплати, доставку та інші аспекти інтернет-магазину.

3.Швидкодія: PostgreSQL забезпечує високу продуктивність та швидкодію завдяки своєму оптимізатору запитів та ефективному використанню ресурсів. Це важливо для інтернет-магазину, де потрібно швидко обробляти запити на пошук товарів, оновлення інформації про товари та виконання транзакцій.

4.Розширюваність: PostgreSQL дозволяє розширити функціональність бази даних за допомогою власних розширень та додатків. Це означає, що ви можете налаштувати базу даних під свої потреби та використовувати розширення, які специфічні для вашого інтернет-магазину.

5.Спільнота та підтримка: PostgreSQL має велику та активну спільноту користувачів та розробників, яка надає підтримку, документацію та ресурси для вирішення питань та проблем. Ви можете отримати допомогу, якщо зіткнетесь з

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		13

якими-небудь труднощами під час розробки та підтримки свого інтернет-магазину.

Керування базою даних буде здійснюватись через pgAdmin – безкоштовний інструмент адміністрування та розробки для PostgreSQL, який надає графічний інтерфейс для виконання різноманітних завдань з керування базою даних.

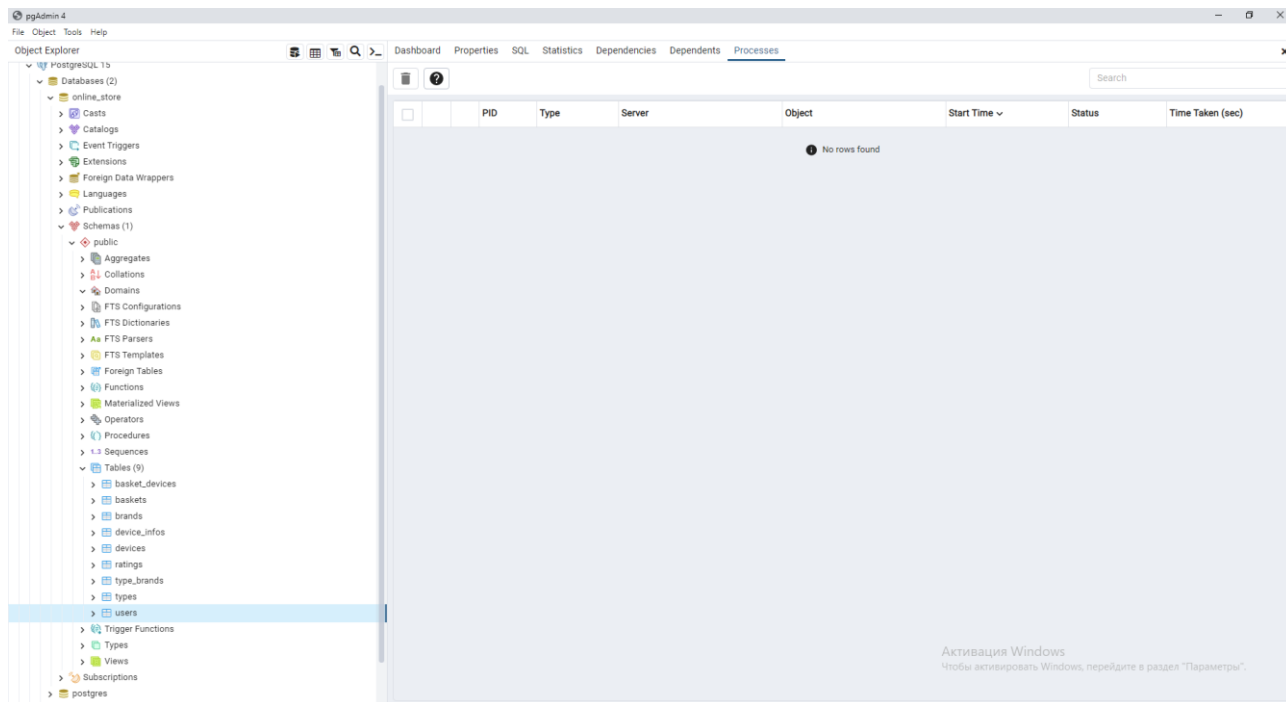


Рисунок 1.3 - PostgreSQL pgAdmin4

Розробка серверної частина – ключовий момент. Оскільки було обрано архітектурний підхід REST, то такі технології як Node.js та PHP – можуть бути використанні для розробки серверної частини. Обидві мають відповідні інструменти та фреймворки, які підтримують реалізацію REST API.

Node.js 18.16.1 – крос-платформне середовище, що дозволяє виконувати JavaScript на стороні сервера. Одна з головних переваг Node.js полягає в тому, що воно незалежне від платформи, що дозволяє розробникам використовувати одну мову програмування (JavaScript) для створення як клієнтських, так і серверних додатків. Декілька ключових особливостей про Node.js:

1. Підтримка різних операційних систем: Node.js підтримує роботу на різних операційних системах, що дозволяє розробникам використовувати його на своїх платформах розробки

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				14
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Крос-платформеність модулів та пакетів: Node.js використовує платформу npm(Node Package Manager), яка є найбільшою колекцією відкритих пакетів та модулів для JavaScript.

3. Широке застосування: Node.js використовується для розробки різноманітних додатків, включаючи веб-сервери, мережеві додатки, інструменти командного рядка, розширення браузера та багато іншого.

PHP 8.2 - це скриптова мова програмування, яка використовується для розробки веб-додатків та серверних додатків. Вона широко використовується для створення динамічних веб-сторінок, роботи з базами даних і взаємодії з іншими компонентами веб-сервера. Однак, важливо відзначити, що швидкість виконання PHP може бути трохи повільнішою порівняно з деякими іншими технологіями, такими як Node.js, особливо при великому обсязі обчислень або роботі з великими навантаженнями. Проте, з використанням правильних оптимізацій та кешування можна досягти високої продуктивності з PHP.

Таблиця 1.1 – Таблиця порівняння між Node.js та PHP

Характеристика	Node.js	PHP
Тип	Серверна технологія	Серверна технологія
Синтаксис	JavaScript	PHP
Швидкодія	Висока	Середня
Використання	В основному для мережевих застосунків та серверів	Широке використання для веб-розробки
Асинхронність	Вбудована підтримка асинхронного програмування	Синхронне програмування(з підтримкою асинхронності у PHP 8+)

Продовження таблиці 1.1

Популярність	Значно зростає та активно використовується	Довгий час був дуже популярним, але швидше зростання замінили інші мови
Веб-фреймворки	Express.js, Noa.js, Meteor, etc.	Larevel, Symfony, CodeIgniter, etc.
Розширюваність	Добре підтримується сторонніми пакетами та модулями	Широкий вибір розширень, але менше в порівнянні з Node.js
Виконання на сервері	Використовує V8 JavaScript Engine	Використовується веб-сервером, таким як Apache або Nginx

Ця таблиця порівняння демонструє основні різниці та подібності між Node.js та PHP. Мій вибір пав на Node.js тому що на мій погляд Node.js зараз має певну кількість переваг, наприклад: швидкодія, асинхронність, єдина мова.

Node.js має різноманітні фреймворки, які допомагають розробникам побудувати веб-додатки швидко та ефективно. Ось кілька популярних фреймворків для Node.js:

1. Express.js 4.x API: Express.js є одним з найпопулярніших та найпростіших у використанні фреймворків Node.js. Він надає мінімальний набір функцій для побудови веб-додатків, таких як маршрутизація, обробка шаблонів та обробка запитів і відповідей. Він також підтримує розширення за допомогою додаткових пакетів.

2. Koa.js 2.13.0: Koa.js є легковаговим фреймворком, який побудований на основі синтаксису генераторів ES6. Він надає потужні можливості для обробки запитів та відповідей, а також дозволяє використовувати проміси для управління асинхронним кодом.

3. Nest.js v9: Nest.js є прогресивним фреймворком для побудови ефективних та масштабованих серверних застосунків. Він використовує TypeScript і пропонує структуру, що базується на модулях, декораторах та ін'єкції залежностей. Nest.js також має вбудовану підтримку для створення API за допомогою GraphQL.

4. Нарі.js 21.3.2: Нарі.js цінується за свою простоту використання та гнучкість. Він надає розширені можливості для обробки запитів, маршрутизації, валідації та автентифікації. Нарі.js також має велику спільноту та плагіни, які полегшують розробку веб-додатків.

5. Meteor.js 3.0: Meteor.js є повноцінним фреймворком, який включає не тільки серверну, але й клієнтську частину. Він надає інтегроване середовище розробки, а також підтримує реал-тайм оновлення даних та автоматичну синхронізацію між сервером і клієнтом.

Зробимо порівняння за допомогою таблицею між двома фреймворками - Express.js та Koa.js

Таблиця 1.2 – Таблиця порівняння фреймворків Node.js

Аспекти	Express.js	Koa.js
Синтаксис	Стандарт JavaScript та колбеки	Генератори ES6 та проміси
Розмір	Великий	Менший
Асинхронність	Обмежена	Повна підтримка асинхронності
Управління помилками	Власне управління помилками	Підтримка промісів для управління помилками
Middleware	Middleware базується на колбеках	Middleware використовує генератори
Розширення	Велика кількість пакетів розширень	Повільніший темп розширення

Продовження таблиці 1.2

Стан розробки	Завершений, стабільний	Активна розробка, менша стабільність
Спільнота	Велика спільнота розробників	Зростаюча спільнота розробників
Використання	Розповсюджений для веб-додатків	Переважно використовується в нових проектах або більш специфічних вимог

За підсумками таблиці можна зробити вивід що для даної роботи підійде більше Express.js тому що на даний час він більш стабільний та розповсюджений у використанні для веб-додатків.

Фронтенд.

У сучасній веб-розробці існує три ключові технології, які найчастіше застосовуються. Це Vue, React та Angular. Незалежно від усіх різниць, вони всі мають спільну мету – створення веб-інтерфейсів, що включає в себе надання зручних інструментів та підходів для цієї задачі. Вибір залежить від унікальних особливостей проекту та від самої команди розробників. Отже, я віддаю перевагу React і вибираю його для своїх потреб.

React 18.2 – це JavaScript бібліотека з відкритим вихідним кодом, яка призначена для розробки користувацьких інтерфейсів. React використовує віртуальний DOM для оптимізованого оновлення відображення. Він порівнює віртуальне дерево DOM з попереднім станом і застосовує лише необхідні зміни, що сприяє ефективності процесу оновлення. Одна з особливостей React – це модель компонентів, яка дозволяє розбити елементи на незалежні компоненти, що можуть бути повторно використані в різних місцях. Це спрощує процес розробки і, загалом, є зручним підходом.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Ось деякі переваги використання React у розробці веб-додатків:

- 1.Компонентний підхід: React побудований навколо компонентного підходу, що дозволяє розділити користувацький інтерфейс на невеликі та багаторазові компоненти. Це робить код більш організованим, легко зчитуваним та підтримуваним.
- 2.Висока продуктивність: Завдяки використанню алгоритмів порівняння віртуального DOM, React оптимізує процес оновлення інтерфейсу. Це дозволяє зменшити кількість маніпуляцій з реальним DOM, що призводить до поліпшення продуктивності та реагує на дії користувача швидше.
- 3.Однонаправлений потік даних: React підтримує однонаправлений потік даних, що сприяє простоті управління станом додатку. Це дозволяє зручно керувати та оновлювати дані в додатку.
- 4.Розширюваність: React має широкий вибір сторонніх бібліотек і компонентів, що дозволяє швидко розширювати можливості вашого додатку. Крім того, React легко інтегрується з іншими бібліотеками або фреймворками.
- 5.Активна спільнота: React має велику та активну спільноту розробників. Це означає, що ви можете швидко знайти допомогу, поради та рішення для вашої розробки.

Також React має декілька переваг при використанні з Node.js:

- 1.Спільна мова програмування: React використовує JavaScript як мову програмування, що є стандартною мовою для Node.js. Це означає, що ви можете використовувати одну мову для розробки і на стороні клієнта (React) і на стороні сервера (Node.js), що спрощує процес розробки та спілкування між фронтендом і бекендом.
- 2.Єдина модель компонентів: React пропонує модель компонентів, яка дозволяє розбити веб-додаток на незалежні частини, які легко управляти та перевикористовувати. Це також працює на користь використанню React з Node.js, оскільки ви можете використовувати ту саму модель компонентів і на серверному боці, що дозволяє перевикористовувати код та логіку між клієнтом і сервером.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		19

3.Швидкість та ефективність: React має вбудовану оптимізацію для віртуального дерева DOM, яке дозволяє ефективно оновлювати лише необхідні елементи і компоненти. Це особливо корисно в поєднанні з Node.js, оскільки ви можете виконувати обробку та рендеринг на сервері, а потім відправляти готовий HTML клієнту, що поліпшує загальну продуктивність та швидкість відгуку.

4.Розширення функціональності: Node.js дозволяє використовувати різноманітні модулі та бібліотеки з екосистеми JavaScript. React використовується з Node.js, ви можете легко інтегрувати різноманітні пакети, наприклад, для роботи з базами даних, мережевими запитами, роутингом і т.д., що дозволяє розширити функціональність вашого додатку та працювати з різними серверними сервісами.

5.Масштабованість: Node.js добре підходить для побудови масштабованих додатків завдяки своїй асинхронній та подієвій моделі. Разом з React, ви можете побудувати масштабовані додатки, які легко масштабувати на багатоядерних серверах та вирішувати багатофункціональні завдання з високою продуктивністю.

Узагалі, поєднання React та Node.js створює потужну комбінацію для розробки сучасних веб-додатків, які мають високу продуктивність, швидкість та розширюваність.

Після огляду технологій розробки складається стек PERN. Стек PERN (PostgreSQL, Express, React, Node.js) – це популярний стек технологій для розробки повнофункціональних веб-додатків.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				20
Змн.	Арк.	№ докум.	Підпис	Дата		

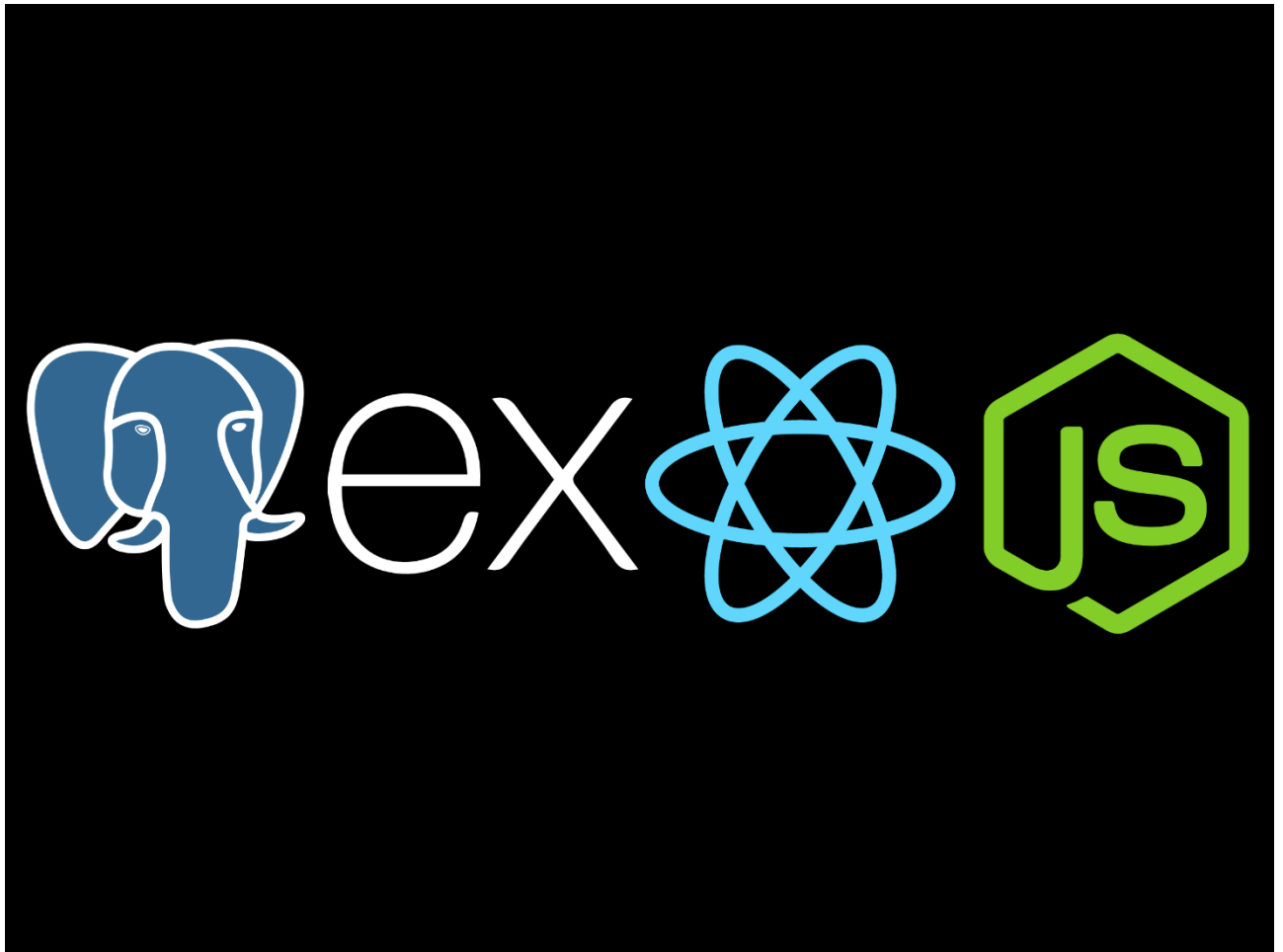


Рисунок 1.4 - Стек PERN

Висновки до першого розділу

У ході виконання першого розділу була сформульовано задачі на виконання кваліфікаційної роботи, визначені основні вимоги щодо розробки проекту.

В результаті аналізів аналогів інтернет-магазинів по поставленим критеріям. Було визначено та пояснено вибір основних цілей розробки, визначено необхідного мінімального функціоналу та визначено вимог до побудови користувацького інтерфейсу.

Отримавши вимоги та бачення інтернет-магазинів, для розробки проекту була вибрана архітектура, яка є потрібною та оптимальною. Визначено та пояснено вибір стеку технологій для реалізації проекту – PostgreSQL, Express.js, React, Node.js

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

2.1 Проектування та розробка програмного продукту

У даному розділі буде описано проектування та розробка програмного продукту, відповідно до поставлених задач та вимог з першого розділу дипломної роботи.

Як було визначено раніше, один з етапів розробки це проектування. Надзвичайно важливий етап, адже він дає чіткий напрямок для подальших дій. На етапі проектування ми визначаємо можливі варіанти використання системи, її алгоритми роботи та загалом майбутню модель, це дозволить більш точно і коректно будувати систему інтернет-магазину. Вже на даному етапі ми можемо побачити як має все працювати задля уникнення проблем у подальшій розробці.

Почнемо з діаграми варіантів використання (Use Case Diagram) – UML діаграма, яка зображає вимоги до розроблюваної системи, ідентифікує зв'язки та відношення між акторами і прецедентами в системі.

На даній діаграмі виділено 3 актори – користувач, зареєстрований користувач, адміністратор. Різниця між акторами – це їх ролі та доступ до певних дій на сайті, наприклад звичайний користувач немає можливості користуватись «Додати тип товару».

Тобто таким чином, ми бачимо усі можливі варіанти використання системи для відповідних акторів.

Можливості акторів:

- Користувач: може переглядати асортимент продукту представлений на сайті, може вибирати типи та бренди продуктів, може зареєструватись;
- Зареєстрований користувач: може авторизуватись, вийти зі свого акаунту та добавляти товар до кошика;
- Адміністратор: такі ж можливості як зареєстрований користувач + має доступ до додавання брендів та типів та додавання нового продукту на сайт;

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		22

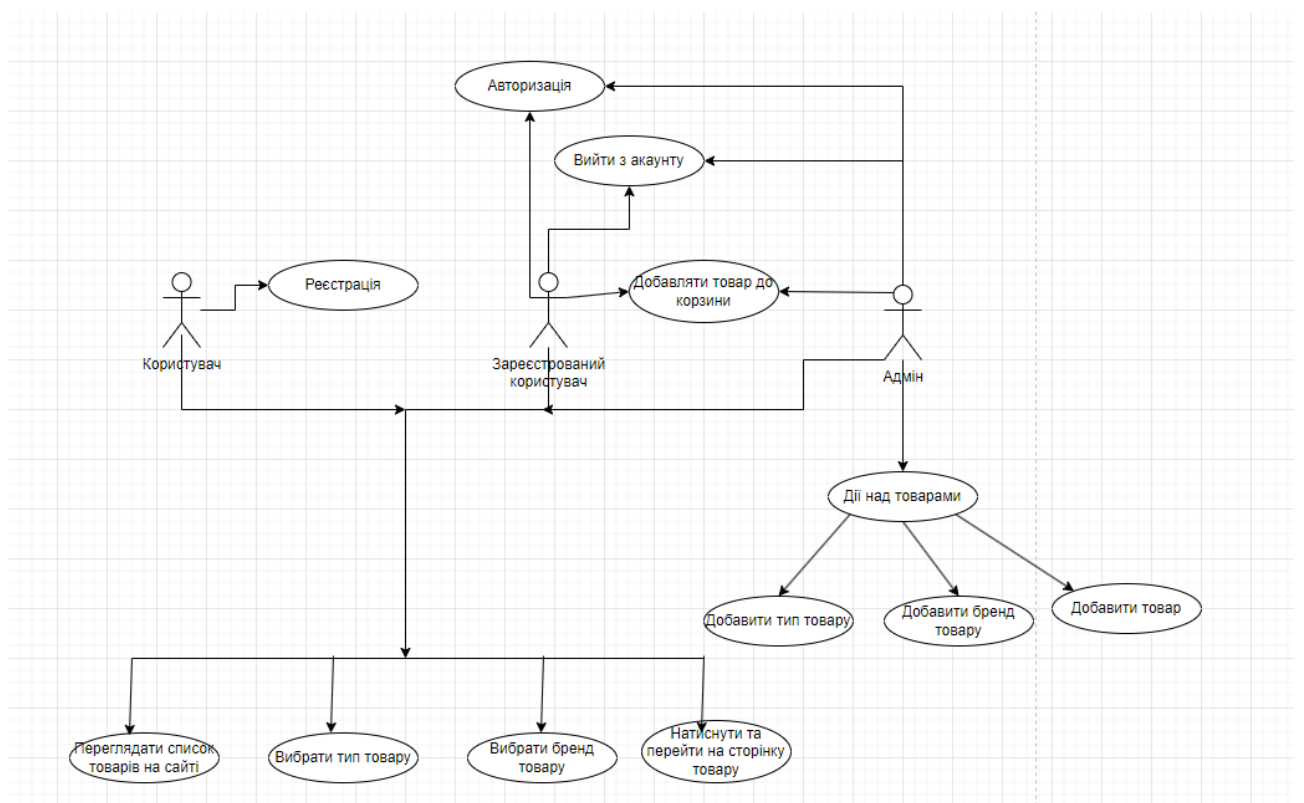


Рисунок 2.1 - Діаграма варіантів використання

Нижче буде представлений 1 з прикладів взаємодії сайту з користувачем.

Прикладом буде авторизація/реєстрація користувача.

Початкова точка – відкриття сторінки після якої в рамках даної діаграми користувач переходить на сторінку авторизації. Далі він має такі дії як можливість входу у акаунт або реєстрація. У випадку авторизації, користувач має ввести потрібні дані про свій акаунт, після яких йде перевірка чи існує такий користувач перевірка чи правильно введено пароль та пошту. Якщо дані були введені правильно, акаунт існує – користувач авторизується, відбудеться перехід у кінцеву точку на основну сторінку сайту. У разі реєстрації, алгоритм схожий, у разі введення правильних даних та перевірки чи існує вже такий акаунт, користувач може успішно зареєструватись та перейти у кінцевий вузол, який зображений на даній діаграмі. Якщо буде допущено десь помилку користувач зможе знову спробувати ввести свої данні.

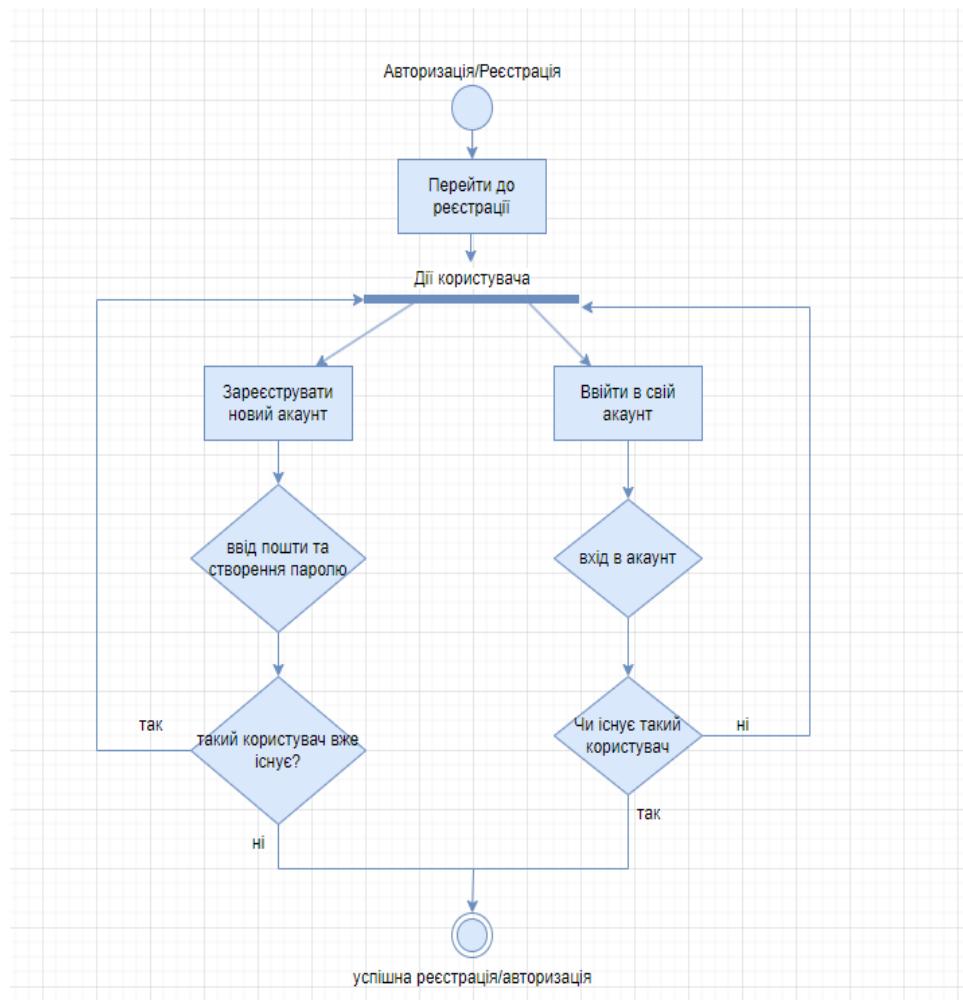


Рисунок 2.2 - Діаграма діяльності, авторизація
2.2 Структура бази даних, взаємодія між собою компонентів системи

Для бази даних я використовував PostgreSQL. Побудуємо діаграму схеми бази даних. Почнемо с опису сутності користувача. У кожної сутності обов’язково має бути унікальний ідентифікатор який однозначно визначає сутність це – id. Також у користувача буде email, пароль а також у користувача буде роль.

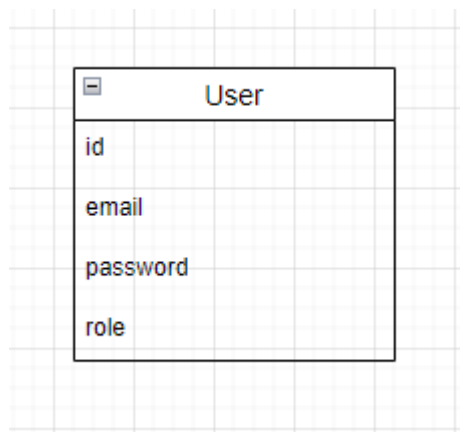


Рисунок 2.3 - Таблиця користувача

Наступна сутність це кошик користувача, тут буде всього 2 поля, а саме унікальний ідентифікатор та зовнішній ключ на користувача для того щоб розуміти кому цей кошик належить.

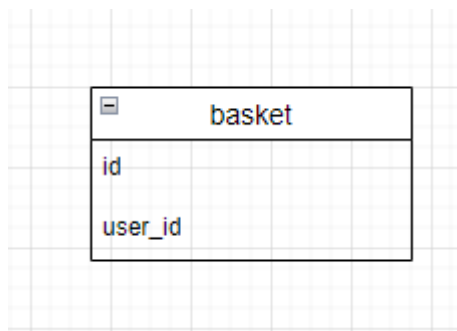


Рисунок 2.4 - Таблиця кошика

Зв'язок між цими таблицями буде один до одного, тобто 1 користувачу може належати тільки 1 корзина, в той час як 1 корзина може належати 1 користувачеві

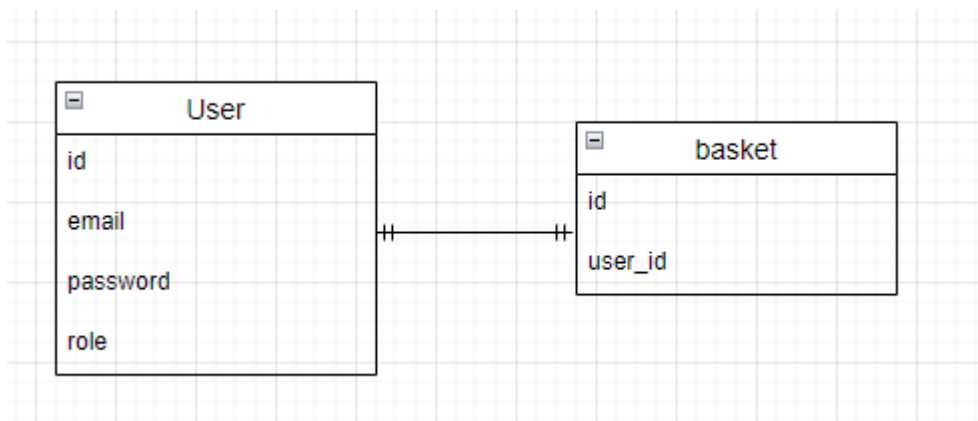


Рисунок 2.5 - Зв'язок між таблицями user та basket

Наступна сутність, це сутність окремого конкретного пристрою. В ньому будуть наступні поля: id, назва пристрою, його ціна. Також буде зображення, та в цій сутності ще буде 2 посилання, 2 зовнішніх ключа, один на тип пристрою, інший на бренд

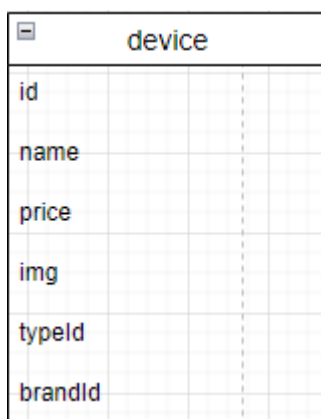


Рисунок 2.6 - Таблиця device

Типи та бренди будуть зберігатись також в окремих двох таблицях. Одна буде називатись відповідно type, інша буде мати назву brand.

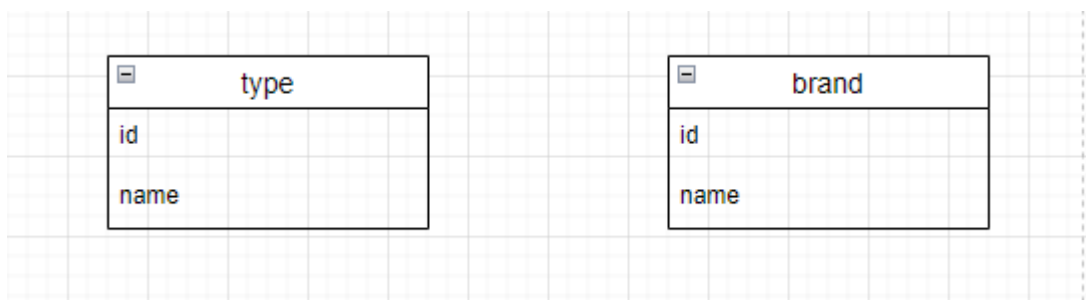


Рисунок 2.7 - Таблиці type та brand

Зв'язок між таблиця type та device, та зв'язок між brand та device буде наступною, одному типу або бренду може належати декілька пристроїв, отже зв'язок один до багатьох

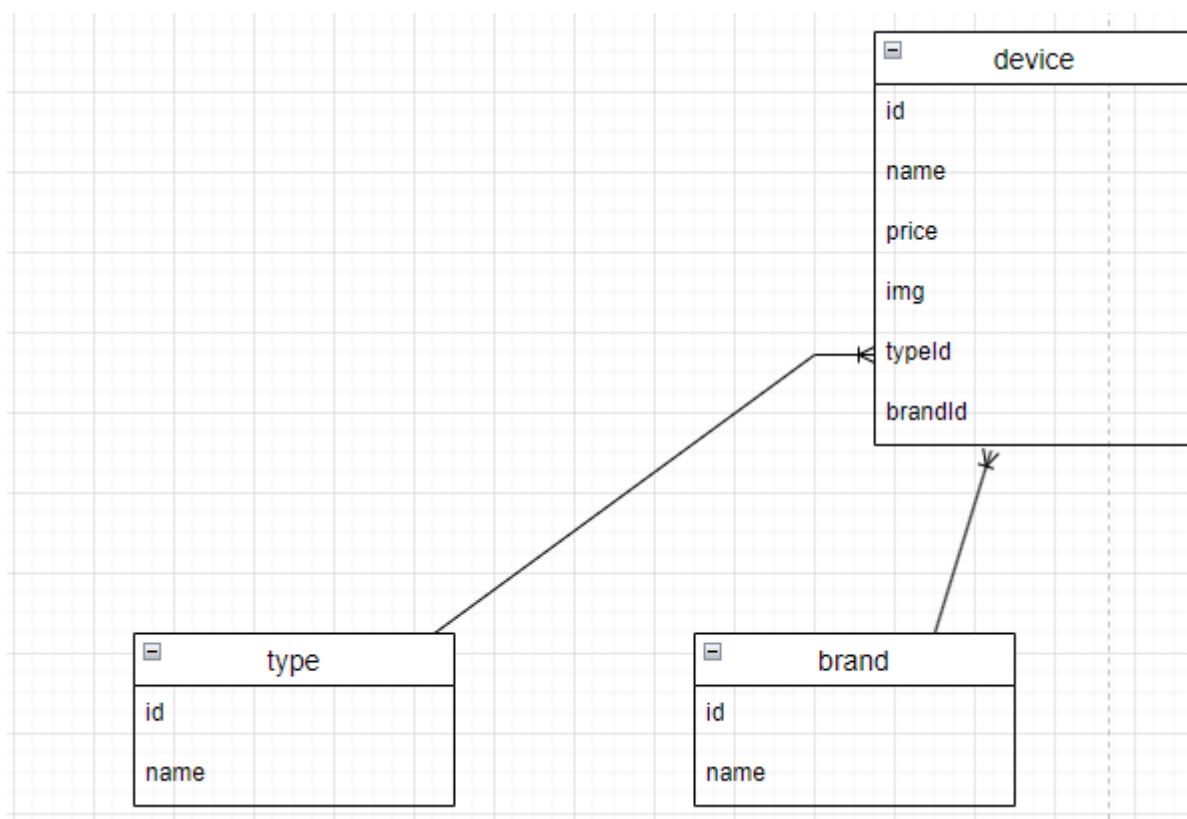


Рисунок 2.8 - Зв'язки між таблицями device, type та brand

Також таблиці type та brand будуть зв'язані між собою оскільки не кожному типу належить бренд. Зв'язок між цими сутностями буде багато до багатьох, оскільки один бренд може належати до декількох типів і тей же час один тип може належати кільком брендам

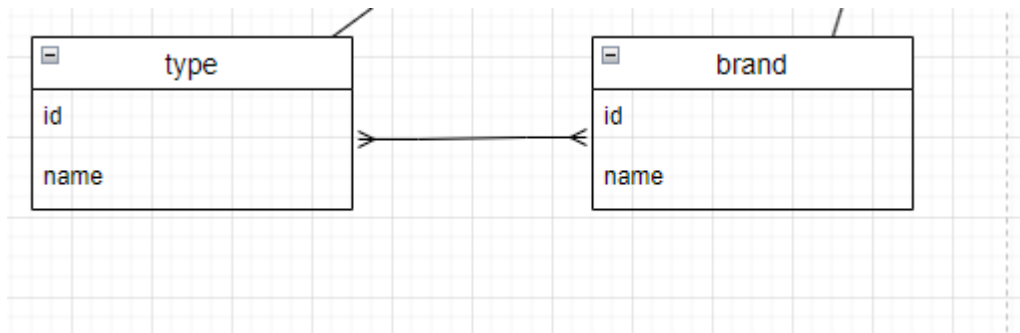


Рисунок 2.9 - Зв'язки між таблицями type та brand

Також для кожного пристрою буде якась індивідуальна інформація. Наприклад у шуруповерта можна описати якісь його характеристики які має тільки він. Тому створимо окремо таблицю, яка буде зберігати інформацію про конкретний прилад. В ній буде id посилання на прилад, також у якихось полях ці характеристики необхідно буде зберігати тому створимо title там буде назва характеристики та description там вже буде зберігатись вже сам опис

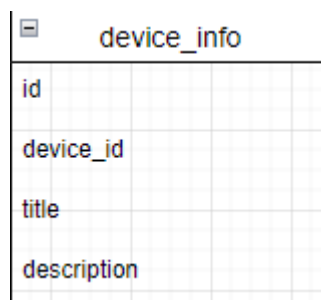


Рисунок 2.10 - Таблиця про інформацію для приладу

Зв'язок з таблицею device буде один до багатьох. Один прилад може мати багато характеристик.

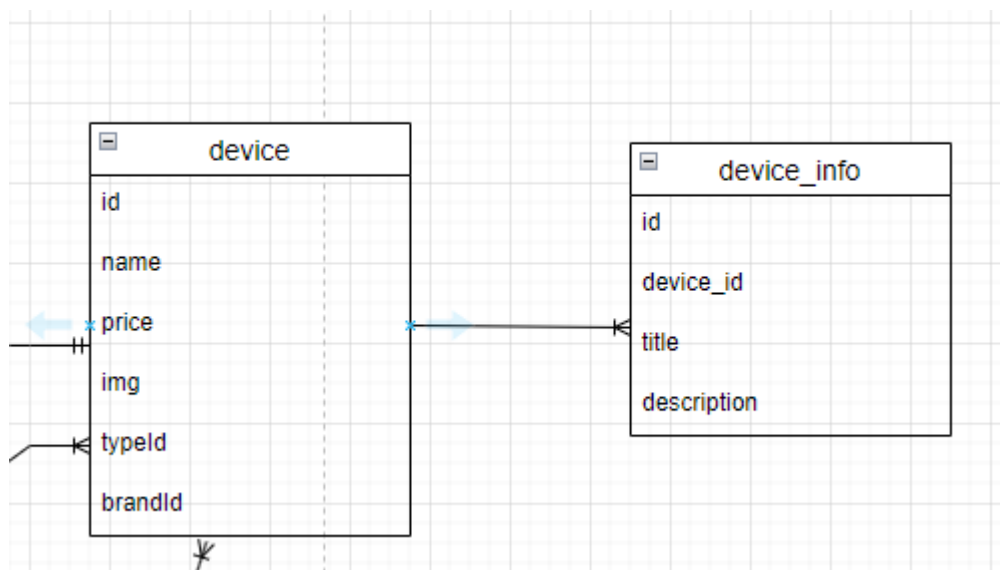


Рисунок 2.11 - Зв'язок між таблицями device та device_info

Також створимо ще одну таблицю, назовемо її basket_device це таблиця буде деякою сполучною ланкою між кошиком та пристроєм, вона міститиме посилання на корзину і відповідно на пристрій.

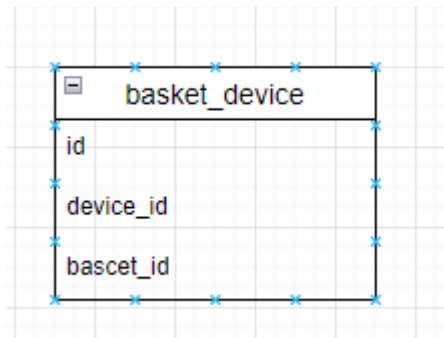


Рисунок 2.12 - Таблиця basket_device

Зв'язки будуть з таблицями basket, там буде зв'язок один до багатьох оскільки один кошик може містити велику кількість пристроїв, в той же час прилад який знаходиться в кошику однозначно посилається на реальний прилад який є у нас в магазині тому тут зв'язок буде один до одного

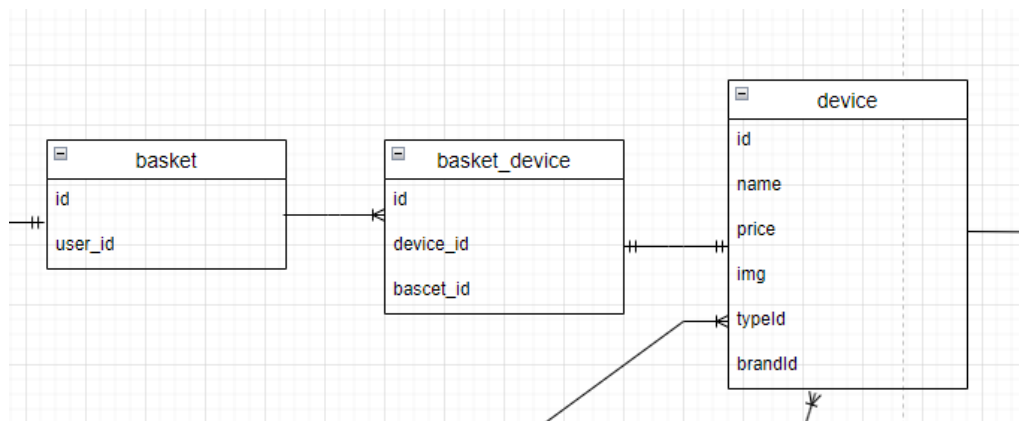


Рисунок 2.13 - Зв'язок між таблицями basket, basket_device та device
В результаті маємо готову діаграму бази даних.

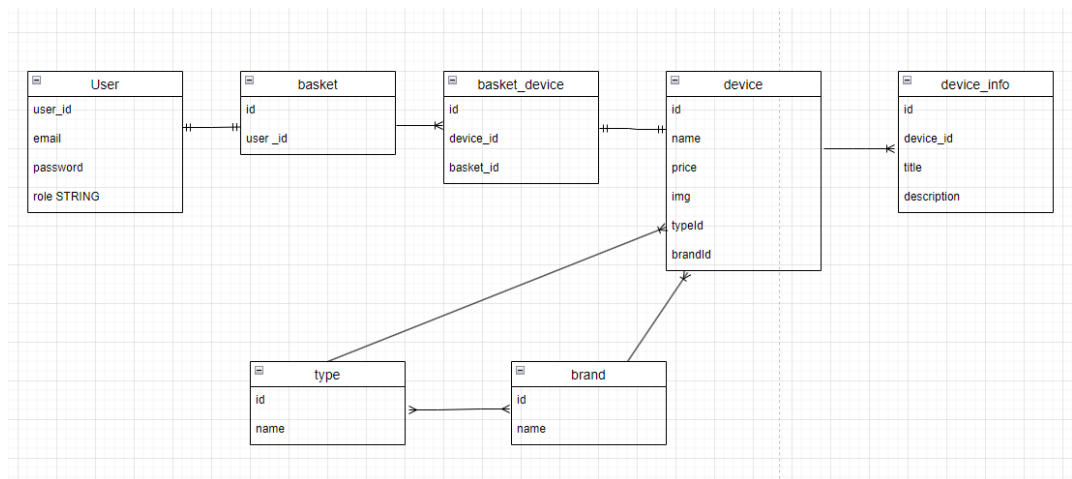


Рисунок 2.14 - Діаграма бази даних

Для проекту даної бд буде достатньо, якщо потрібно буде розширити функціоналу на сайті, ввести нові колекції не є проблемою. Розглянемо більш детальну наявну діаграму:

Колекція User – містить дані про користувачів. Містить такі поля:

- user_id: ідентифікатор користувача, тип цього поля INTEGER, тобто числовий
- email: електронна пошта, потрібна для авторизації, має бути унікальною, тип STRING;
- password: пароль до акаунту користувача, потрібен для авторизації, тип STRING;
- role: роль користувача на сайті, тип STRING

Колекція basket – корзина для користувача. Містить такі поля:

- user_id: ідентифікатор користувача, тип цього поля INTEGER
- id ідентифікатор корзини, тип INTEGER

Колекція basket_device – корзина для користувача. Містить такі поля:

- user_id: ідентифікатор корзини продукту, тип цього поля INTEGER
- device_id ідентифікатор дивайсу, тип INTEGER
- basket_id ідентифікатор корзини, тип INTEGER

Колекція device – містить дані про продукт. Містить такі поля:

- id: ідентифікатор продукту, тип цього поля INTEGER
- name: назва продукту тип STRING;
- price: ціна продукту, тип цього поля INTEGER

-img: зображення , тип STRING

-typeId: ідентифікатор типу продукту, тип INTEGER

-brandId: ідентифікатор бренду продукту, тип INTEGER

Колекція device_info – містить детальну інформацію про продукт.

Містить такі поля:

- id: ідентифікатор корзини продукту, тип цього поля INTEGER

- device_id ідентифікатор дивайсу, тип INTEGER

- title назва характеристики, тип STRING

-description опис характеристики продукту, тип STRING

Колекція brand – зберігаються данні про бренди. Містить такі поля:

- id: ідентифікатор корзини продукту, тип цього поля INTEGER

- name назва бренду, тип STRING

Колекція type – зберігаються данні про типи. Містить такі поля:

- id: ідентифікатор корзини продукту, тип цього поля INTEGER

- name назва типу, тип STRING

2.3 Реалізація функціоналу серверної частини інтернет-магазину

Почнемо з побудови серверної частини. Після ініціалізації проекту, встановлення початкових залежностей можемо почати з кореневого файлу index.js. Конфігурація зберігаються у файлі package.json:

Лістинг 2.1 - Конфігурація сервера, package.json

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.0",
    "cors": "^2.8.5",
    "dotenv": "^16.1.4",
    "express": "^4.18.2",
    "express-fileupload": "^1.4.0",
```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    "jsonwebtoken": "^9.0.0",
    "pg": "^8.11.0",
    "pg-hstore": "^2.3.4",
    "sequelize": "^6.32.0",
    "uuid": "^9.0.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  }
}

```

Тут вкажемо назву проекту, версію, опис(опціонально), вказуємо головний, кореневий файл – це index.js, вказуємо скрипти, які можемо виконувати з терміналу ,було вказано скрипт “dev” для запуску сервера, nodemon – дозволяє запустити сервер і автоматично зберігати та застосовувати зміни, тим самим не потрібно кожного разу після змін власноруч запускати сервер. Також файл містить назву ліцензії і залежності, які необхідні для роботи додатку, їх буде розглянуто пізніше, яка і де використовується.

В головному файлі index.js створюємо сервер за допомогою express.js, конфігуруємо його та підключаємо базу даних, підключаємо роути.

Даний код створює екземпляр додатку Express, що дозволить далі налаштувати роути, middleware тощо. Також вказуємо, що додаємо middleware яка дозволяє розпарсити вхідні дані у форматі JSON та далі використовуємо middleware «cors», яка вирішує проблему політики міжсайтових запитів, таким чином дозволяємо використовувати додаток всім джерелам, не потрібно тепер це прописувати вручну:

Лістинг 2.2 - Створення сервера

```

const app = express()
app.use(cors())
app.use(express.json())

```

Далі розглянемо підключання бази даних до сервера. Для цього створимо окремий файл db.js.

Лістинг 2.3. - Підключення бази даних

```

const {Sequelize} = require('sequelize')

module.exports = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASSWORD,
  {

```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		31

```

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

```

Також хочеться нагадати що всі налаштування до бази даних винесенні в змінні оточення. Там були створенні наступні змінні:

- назва бази даних
- назва головного користувача який створюється при встановленні сервера
- пароль який був вказаний при встановленні системи управління базами даних
- також в режимі розробки хост вказуємо локальним (localhost)
- а також був встановлений порт по замовчуванням (5432)

Лістинг 2.4 - Налаштування до бази даних

```

DB_NAME=online_store
DB_USER=postgres
DB_PASSWORD=ewqhvxc
DB_HOST=localhost
DB_PORT=5432

```

Після створення файлу db.js повертаємося в index.js, і тепер сюди імпортуємо об'єкт який був зроблений в файлі db.js та який звідти ж експортували

Лістинг 2.5 - Імпортуємо базу даних до сервера

```
const sequelize = require('./db')
```

Далі визвемо функцію для підключення до бази даних. Для цього буде створено асинхронну функцію (оскільки всі операції з базою даних є асинхронними) “start”. Далі все обернемо в блок “try,catch” що відловлювати відразу можливі помилки та щоб програма не падала та визвемо функцію “listen”.

Також сюди визиваємо функцію “authenticate()” за допомогою неї буде створюватись підключення до бази даних. Наступним етапом визиваємо функцію у об'єкта sequelize “sync” ця функція буде зрівнювати стан бази даних з схемою даних

Лістинг 2.6 - Функція для підключення до бази даних

```

const start = async () => {
  try {
    await sequelize.authenticate()
    await sequelize.sync() // функція яка зрівнює стан бази
    даних із схемою даних
    app.listen(PORT, () => console.log(`Server started on port

```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		32

```

    ${PORT}`))
    }catch {
        console.log{
    }
}

```

А також визиваємо функцію “start”, щоб сервер в нас запустився.

Тепер приступимо до реалізації того як дані будуть вже зберігатись в базі даних. Для цього створимо папку models в ній створимо файл models.js саме тут ми моделі даних як раз і будемо описувати. Сюди імпортуємо об’єкт sequelize який створювали в файлі db.js та з самого пакету sequelize нам потрібно імпортувати клас DataTypes за допомогою якого описуються типи полів (STRING, INTEGER). В приклад я приведу декілька описаних моделей

Лістинг 2.7 - Опис моделі користувача

```

const User = sequelize.define('user', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true}, // INTEGER - тип числовий
  email: {type: DataTypes.STRING, unique: true},
  password: {type: DataTypes.STRING},
  role: {type: DataTypes.STRING, defaultValue: »USER»}
})

```

Лістинг 2.8 - Опис моделі предмету

```

const Device = sequelize.define('device', {
  id: {type: DataTypes.INTEGER, primaryKey: true, 33 отуди3333ment:
true},
  name: {type: DataTypes.STRING, unique: true, allowNull: false},
  price: {type: DataTypes.INTEGER, allowNull: false},
  rating: {type: DataTypes.INTEGER, defaultValue: 0},
  img: {type: DataTypes.STRING, allowNull: false},
})

```

Тепер перейдемо до каркасу програми. Почнемо з маршрутів по яким будуть відпрацьовувати ті чи інші методи. Створимо папку Routes для типів, брендів, дивайсів, користувачів також у файлі index.js ці всі маршрути будуть об’єднуватись, тобто основний роутер програми. В ньому отримуємо роутер з express, створюємо об’єкт цього роутеру, і за підсумком цей роутер експортуємо. Так як файли typeRouter, brandRouter, deviceRouter, userRouter будуть під основним роутером, тобто якоюсь його частиною, тому в основному роутері це вказуємо. Викликаємо функцію use першим параметром вказуємо URL по якому роутер буде відпрацьовувати, а іншим потрібно передати сам роутер. Також

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		33

імпортуємо всі створенні роутери та наступним етапом зіставляємо маршрути, які вказали відповідним роутерам

Лістинг 2.9 - Головний роутер

```
const Router= require('express')
const router = new Router()
const deviceRouter = require('./deviceRouter')
const userRouter = require('./userRouter')
const brandRouter = require('./brandRouter')
const typeRouter = require('./typeRouter')
router.use('/user', userRouter)
router.use('/type', typeRouter)
router.use('/brand', brandRouter)
router.use('/device', deviceRouter)
module.exports = router
```

Також розглянемо інші роутери наприкладі brandRouter. По роботі з брендами будемо використовувати методи post та get. Метод post будемо використовувати для того щоб створювати бренд, а метод get буде використовуватись для того, щоб усі бренди отримувати. Імпортуємо контролер та в кожен post та get методи другим параметром передаємо відповідну функцію.

Лістинг 2.10 - Роутер brandRouter

```
Const Router= require('express')

const router = new Router()

const brandController = require('../controllers/brandController')

router.post('/',brandController.create)

router.get('/',brandController.getAll)

module.exports = router
```

Для брендів та типів маршрути будуть абсолютно однакові, а для користувача вже буде небагато відрізнятися, тут буде 2 post метода реєстрація та авторизація, і також буде один метод get за допомогою якого будемо перевіряти авторизований користувач 34 оту, це буде відбуватись по jwt token. Також сюди імпортуємо контролер та в кожен post та get методи другим параметром передаємо відповідну функцію.

Лістинг 2.11 - Роутер userRouter

```
Const Router = require('express')
```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				34
Змн.	Арк.	№ докум.	Підпис	Дата		

```
const router = new Router()
const userController = require('../controllers/userController')
const authMiddleware = require('../middleware/authMiddleware')
router.post('/registration',userController.registration)
router.post('/login',userController.login)
router.get('/auth',authMiddleware,userController.check)
module.exports = router
```

Для того щоб сервер зрозумів що об'єднали всі 4 роутера в 1, повертаємося до кореневого index.js та імпортуємо туди основний роутер.

Лістинг 2.12 - Підключення головного роутра до серверу

```
const router = require('../routes/index')
```

Далі по аналогії викликаємо функцію use першим параметром вказуємо url по якому роутер повинен оброблятися в нашому випадку це('/api'), другим параметром передаємо сам роутер

Лістинг 2.13 - Виклик функції

```
app.use('/api',router)
```

Зараз функція написана безпосередньо в роутері це є поганою практикою, оскільки роутер стає занадто товстим і логіку по-доброму потрібно відокремлювати для цього створимо папку controllers та в ньому створимо для кожного роутеру відповідний контролер. На прикладі контролера користувача тут створимо однойменний клас та створимо відповідні функції, а саме registration login та функцію яка перевірятиме авторизований користувач назовемо її check.

Авторизація буде відбуватись по jwt token або ж токен доступу-це простий рядок розділена крапками на 3 частини, здебільшого нас цікавить частина payload туди будемо ховати дані про користувача це його пошта, id, та його роль. Після того як користувач зареєструвався, ввів пошту та пароль для нього генерується подібний токен, всі його дані вшиваються всередину, і важливо розуміти що ці дані ніяк не шифруються, тому на клієнті можемо спокійно їх розшифровувати, але перевірити на те валідний токен можна тільки за допомогою секретного ключа, який оголошений на сервері та ніхто знати не повинен.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Підсумовуючи користувач вводить пошту та пароль насамперед перевіряємо чи існує користувач з такою поштою в системі якщо так то відразу порівнюємо пароль який знаходиться в базі даних з паролем який написав користувач, якщо ці паролі співпадають генеруємо jwt token та відправляємо його на клієнт. Після чого на клієнті цей токен зберігається, потім цей токен прикріплюється до запитів, де необхідна авторизація, наприклад в корзину не зайдемо якщо не авторизовані, авторизуємося сервер генерує токен повертає його назад на клієнт, потім натискаємо кнопку увійти до корзини і в нас відправляється запит на отримання товарів до цього запиту хедери в http заголовки додаємо authorization та туди вміщуємо той самий токен котрий який дістали, сервер валідує цей токен за секретним ключем, якщо він валідний то користувач потрапляє на сторінку кошика та отримує всі товари які в нього там перебувають.

Почнемо з реєстрації на сам перед отримуємо з тіла запиту пошти та пароллю, а також роль

Лістинг 2.14 - Тіло запиту

```
const {email, password, role} = req.body
```

Зробимо невелику перевірку, якщо в тілі запиту пошти або пароллю порожні то будемо повертати помилку на клієнт, напишемо що “Некоректний email чи password”

Лістинг 2.15 - Перевірка

```
if (!email || !password) {
    return next(ApiError.badRequest('Некоректний email чи password'))
}
```

Наступний етап це перевірити чи існує користувач з таким email в системі, тому вказуємо умови до нього передаємо email.

Лістинг 2.16 - Перевірка на існування користувача

```
Const candidate = await User.findOne({where: {email}})
```

Далі задаємо умови, якщо нам повернувся користувач та він не пустий то тоді знову повертаємо помилку в якій вказуємо що користувач з таким email вже існує

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Лістинг 2.17 - Повертання помилки

```
if (candidate) {  
    return next(ApiError.badRequest('Користувач з таким email вже існує'))  
}
```

Якщо ця умова не виконалась користувача не знайшли тоді можемо зафіксувати пароль та створити вже нового користувача функція хешування пароля асинхронна викликаємо її, першим параметром надаємо безпосередньо сам пароль користувача який отримали у тілі запиту, а другим параметром передаємо скільки разів хешуватимемо

Лістинг 2.18 - Хешування паролю

```
const hashPassword = await bcrypt.hash(password, 5)
```

Наступним етапом за допомогою функції create користувача створюємо, передаємо туди email роль та захеширований пароль.

Лістинг 2.19 - Створення користувача

```
Const user = await User.create({email, role, password: hashPassword})
```

Відразу для користувача створюємо кошик, викликаємо функцію create моделі basket та все що необхідно передати це id користувача яке можемо отримати вже з створеного об'єкту самого користувача

Лістинг 2.20 - Створення корзини

```
const basket = await Basket.create({userId: user.id})
```

Тепер треба згенерувати цей jsonwebtoken. Викликаємо функцію generateJwt та в неї потрібно передати деякі дані. Передаємо id користувача пошту та його роль яку отримаємо вже з самого користувача.

Лістинг 2.21 - Використання функції generateJwt

```
Const token = generateJwt(user.id, user.email, user.role)
```

Оскільки генерувати токен нам доведеться і в функції логіну створимо для цього окрему функцію назвемо її generateJwt

Лістинг 2.22 - Створення функції generateJwt

```
const generateJwt = (id, email, role) =>{  
    return jwt.sign(  
        {id,email,role},  
        process.env.SECRET_KEY,  
        {expiresIn: '24h'}  
    )  
}
```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				37
Змн.	Арк.	№ докум.	Підпис	Дата		

Отож відкриємо Postman та затестуємо, прописуємо url “http://localhost:5000/api/user/registration”, в тілі запиту вже буде рядок в форматі json, і потім вказуємо пошту на пароль.

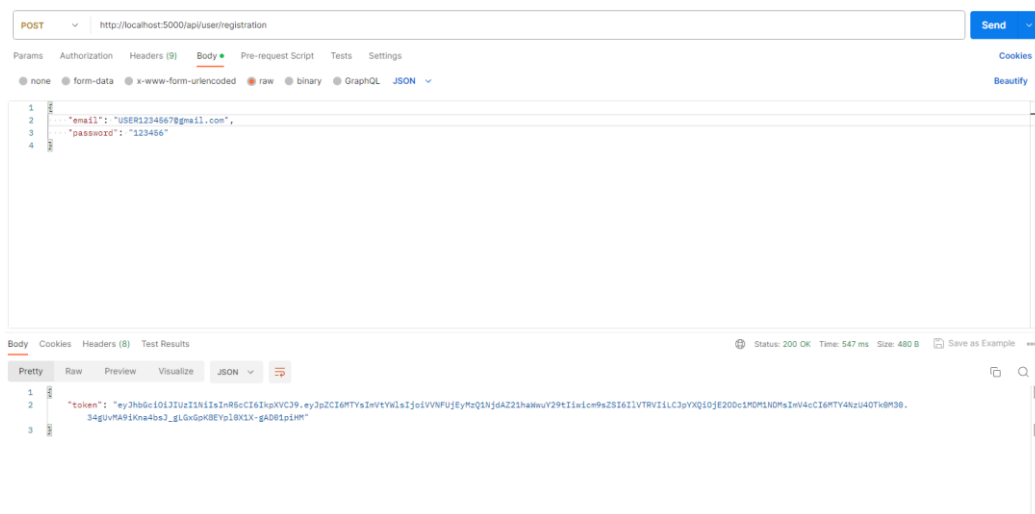


Рисунок 2.4 - Тестування реєстрації

Реалізуємо функцію логіну, тут також з тіла запиту отримуємо email
пароль, відразу робимо деструктуризацію

Лістинг 2.23 - Тіло запиту

```
const {email,password} = req.body
```

Тепер нам треба переконатись що користувач з такою поштою в базі даних у нас вже існує, тому за умовою шукаємо цього користувача, але тут навідміну від реєстрації вже перевіряємо, що якщо користувача не знайдено то тоді повертаємо помилку, та напишемо що “Користувач з таким іменем не знайдено”.

Лістинг 2.24 - Пошук користувача в базі даних, та повернення помилки

```
Const user = await User.findOne({where:{email}})
```

```
if(!user){
    return next(ApiError.internal('Користувач не знайдений'))
}
```

Якщо ці умови пройшли, тоді нам потрібно переконатись в тому що пароль який написав користувач в формі збігається з тим що знаходиться в базі даних, але в базі даних в нас лежить захешований пароль, тому за допомогою bcrypt функції compareSync вона є синхронною, порівнюємо ці паролі. Першим

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				38
Змн.	Арк.	№ докум.	Підпис	Дата		

параметром передаємо пароль який написав користувач інший отримуємо з бази даних, якщо паролі не співпадають також повертаємо помилку

```
Лістинг 2.25 - Перевірка валідності паролю, та повернення помилки
let comparePassword = bcrypt.compareSync(password,user.password)
    if(!comparePassword){
        return next(ApiError.internal('Вказаний невірний пароль'))
    }
```

Потім також генеруємо токен за допомогою функції generateJwt та передаємо туди всі необхідні параметри

Лістинг 2.26 - Генерація токenu

```
const token = generateJwt(user.id,user.email,user.role)
```

Та на клієнт повертаємо поаналогії з реєстрацією сам токен

Лістинг 2.27 - Повернення токenu

```
return res.json({token})
```

Залишається тільки функція check, але перед цим реалізую middleware. Саме в ньому декодуватимемо токен та перевірятимемо його на валідність, якщо токен не валідний будемо повертати відразу помилку про те що користувач не авторизований. Тому відразу експортуємо функцію middleware. Відразу зробимо перевірку, якщо метод у нас дорівнює “OPTIONS” тоді пропускаємо. Якщо все таки десь виникла в нас помилка повертаємо status code 401, в повідомленні напишемо що користувач не авторизований

Лістинг 2.28 - Функція middleware

```
module.exports = function (req,res,next){
    if(req.method ==="OPTIONS"){
        next()
    }
    res.status(401).json({message:"Не авторизований"})
}
```

Тепер перейдемо до логіки, перше, що нам необхідно зробити, це з хедерів вичепити сам токен, так як в хедер зазвичай поміщають спочатку тип токена, а потім вже сам токен, тому нам потрібно по пробілу два ціх слова відділити і по першому індексу отримати безпосередньо сам токен, якщо цього токenu немає, якщо він не існує знову ж таки повертаємо помилку

Лістинг 2.29 - Отримання токenu, та повернення помилки

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		39

```

        const token = req.headers.authorization.split(' ')[1]// Bearer(тип
        токenu) fasfasdfdsafdsa(сам токен)
        if(!token){
            return res.status(401).json({message:"Не авторизований"})
        }
    }

```

Йдемо далі, якщо токен є, цю умову пропустили нам потрібно його декодувати імпортуємо модуль `jsonwebtoken`, та визиваємо функцію `verify`, яка і буде перевіряти токен на валідність. Сюди необхідно першим параметром передати сам токен, а другим секретний ключ

Лістинг 2.30 - Декодування токenu

```

const jwt = require('jsonwebtoken')
const decoded = jwt.verify(token,process.env.SECRET_KEY)

```

Наступним етапом до `req` наприклад в полі `user` додамо дані, які витягли з цього токenu та у всіх функціях цей користувач буде доступний. Потім викликаємо функцію `next`

Лістинг 2.31 - Передача даних з токenu та виклик функцію `next`

```

req.user = decoded
next()

```

Далі переходимо в `userRouter` сюди імпортуємо цей `middleware`, та передаємо його другим параметром в `get` запит, який буде перевіряти користувача на авторизованість

Лістинг 2.32 - Імпорт `middleware`

```

const authMiddleware = require('../middleware/authMiddleware')

```

Лістинг 2.33 - Перевірка користувача на авторизованість

```

router.get('/auth',authMiddleware,userController.check)

```

Тепер перейдемо до реалізації, відкриваємо `UserController`, вся функція `check` буде зводитись до того щоб згенерувати новий токен і відправити його назад на клієнт, передаємо всі необхідні параметри функцію `generatejwt`, та повертаємо згенерований токен на клієнт

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		40

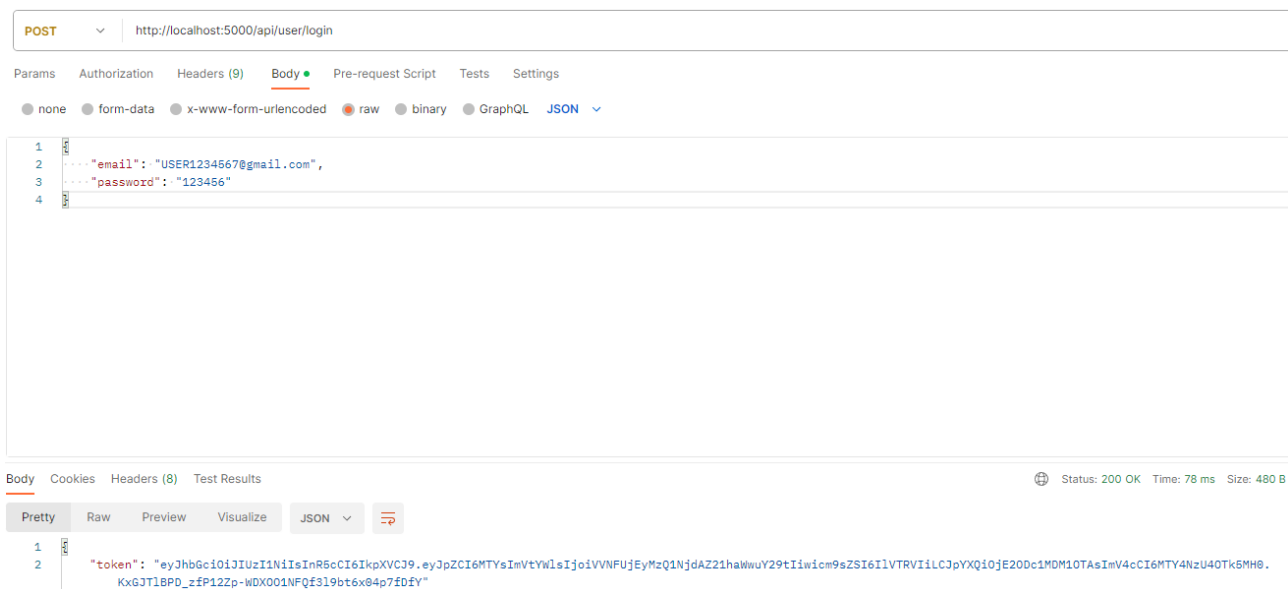


Рисунок 2.5 Перевірка логіну

Висновки до другого розділу

Під час виконання другого розділу було пройдено чимало моментів розробки інтернет-магазину, пророблено етапи проектування та створення серверної частини веб-додатку.

По початковому баченні сайту, оснований на вимогах та критеріях з першого розділу кваліфікаційної роботи, були побудовані діаграми діаграма варіантів використання, яка показує можливі дії користувачів, діаграма діяльності, яка зображує алгоритми роботи активностей в системі.

Була побудована діаграми бази даних, визначені залежності між колекціями, типи полів.

Далі була проведено масштабна робота по побудові повноцінного сервера інтернет-магазину на основі проектування, за допомогою засобів розробки Node.js в поєднання з Express.js. У висновку отримали пророблену бекенд частину, яка готова до взаємодії з фронтендом.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		41

РОЗДІЛ 3 ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ, ТЕСТУВАННЯ

3.1 Інтерфейс сайту

Було розроблено 4 сторінки сайту: сторінка адмін-панелі, сторінка авторизації та реєстрації, основна сторінка та сторінка з інформацією та описом продукту.

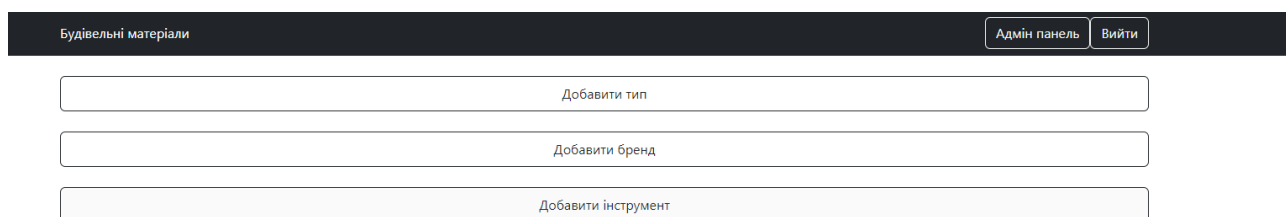


Рисунок 3.1 – Адмін-панель сайту

Сторінка де представлено вікно реєстрації та авторизація. Користувач може створити власний акаунт на сайті.

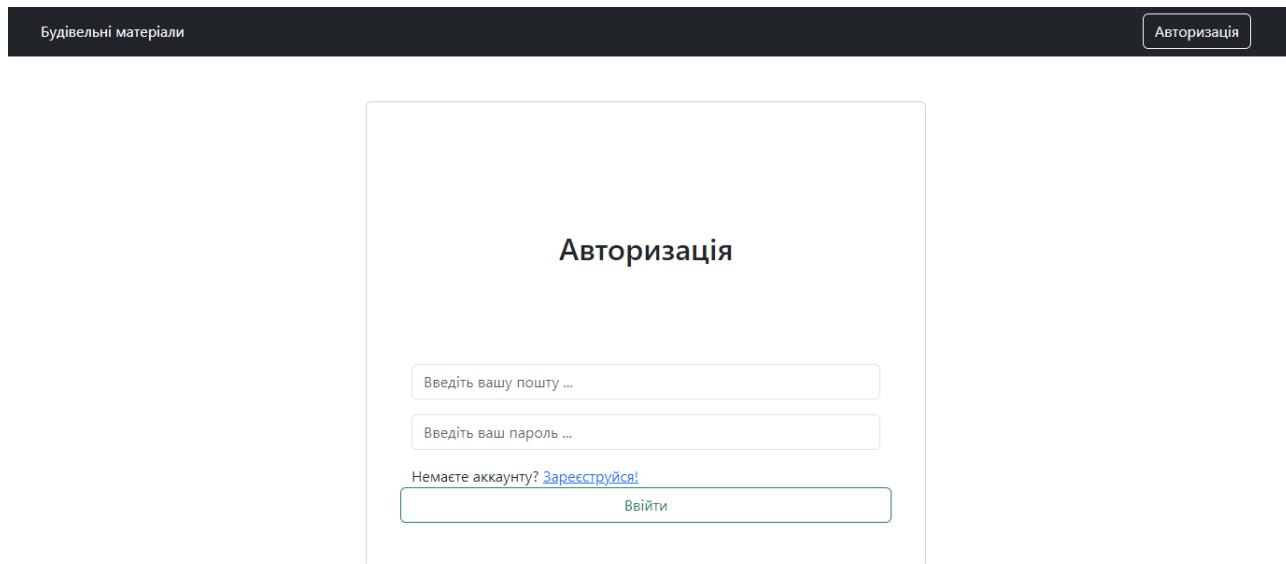


Рисунок 3.2 - Сторінка авторизації на сайті

Регістрація

Введіть вашу пошту ...

Введіть ваш пароль ...

Маєте акаунту? [Ввійдіть!](#)

Регістрація

Рис 3.3 - Сторінка реєстрації на сайті

шурупверт

Перфоратори

Циркулярні пилки

Шліфувальні та полірувальні
машини (болгарки)

Електролобзики

Дрилі та міксери

Електрорубанки

Фрезери

Metabo

Makita

Intertool Storm

CAT

Bosch

DWT

DeWalt

Dnipro-M

Einhell

GMT

Зеніт

9999 грн
DX1219000 грн
DX1310000 грн
CAT DX1124299 грн
DeWalt DCD991T24200 грн
Bosch Professional GBH 22022000 грн
Акумуляторний безщітковий
перфоратор CAT DX219500 грн
Bosch Professional GBH 2-28 F6400 грн
Перфоратор Makita HR2470

Активация Windows

Чтобы активировать Windows, перейдите в г

Рисунок 3.4 - Основна сторінка сайту

Сторожук С.О

Петросян Р.В

Змн.

Арк.

№ докум.

Підпис

Дата

КН.КР.Б – 122 – 23 – ПЗ

Арк.

43

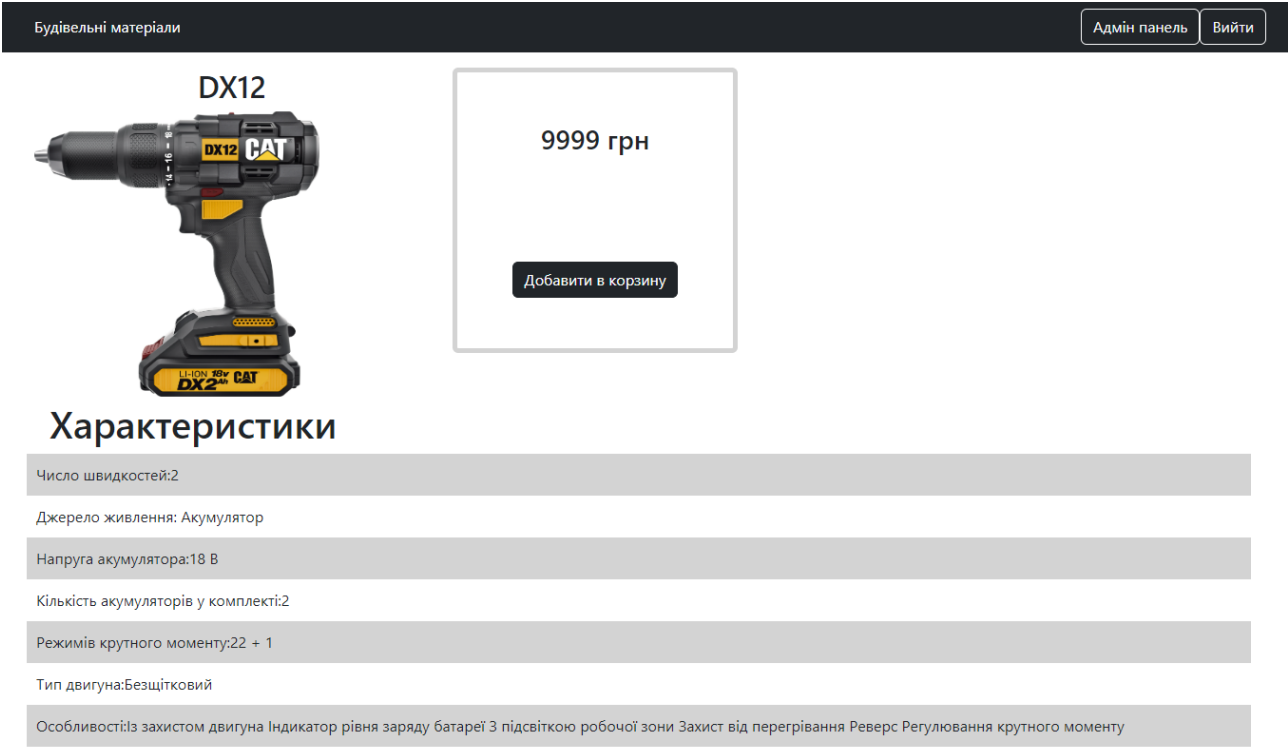


Рисунок 3.5 -Сторінка продукту

3.2 Розробка архітектури сайту



Рисунок 3.6 - Схема навігації по сайту

За допомогою навігації користувач може перейти на головну сторінку або на сторінки реєстрації та авторизації, також користувач з правами адміну має змогу заходити на адмін-панель

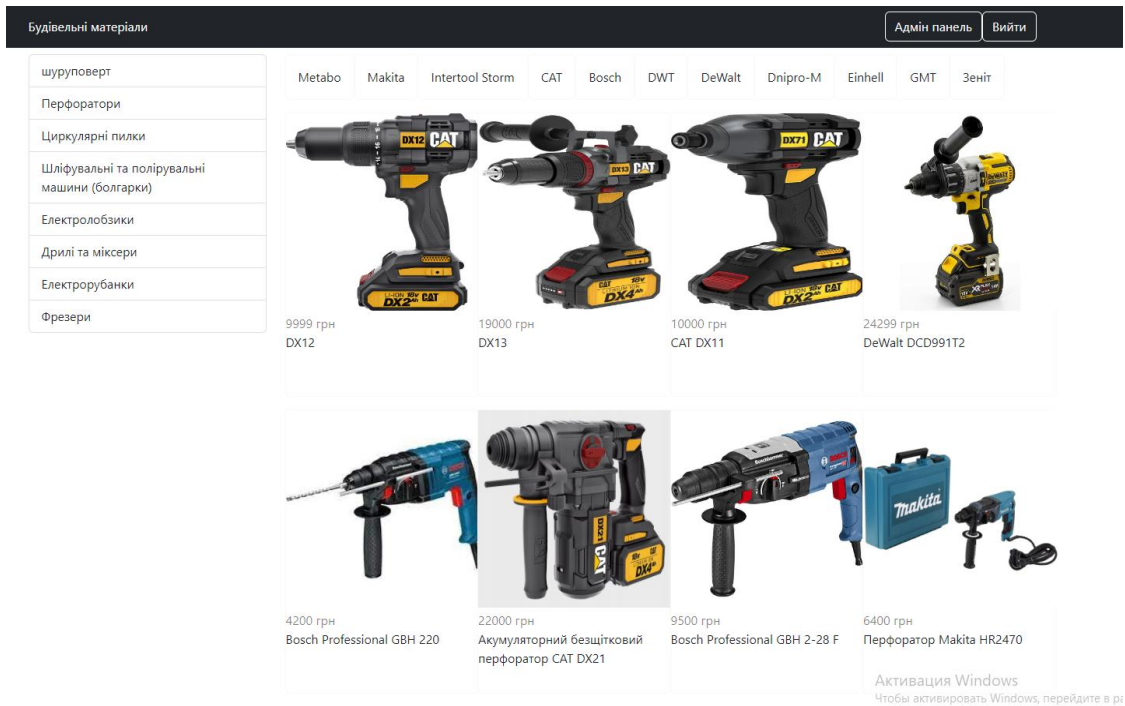


Рисунок 3.7 - Основна частина сайту

Було зроблено основні частини сайту де знаходиться інформація про продукт який представлений в інтернет-магазині(Рис.2.3.), можна побачити візуально продукт, побачити його характеристики також можна побачити ціну, клієнт зможе самостійно вибрати потрібний йому бренд та тип продукту який йому потрібний

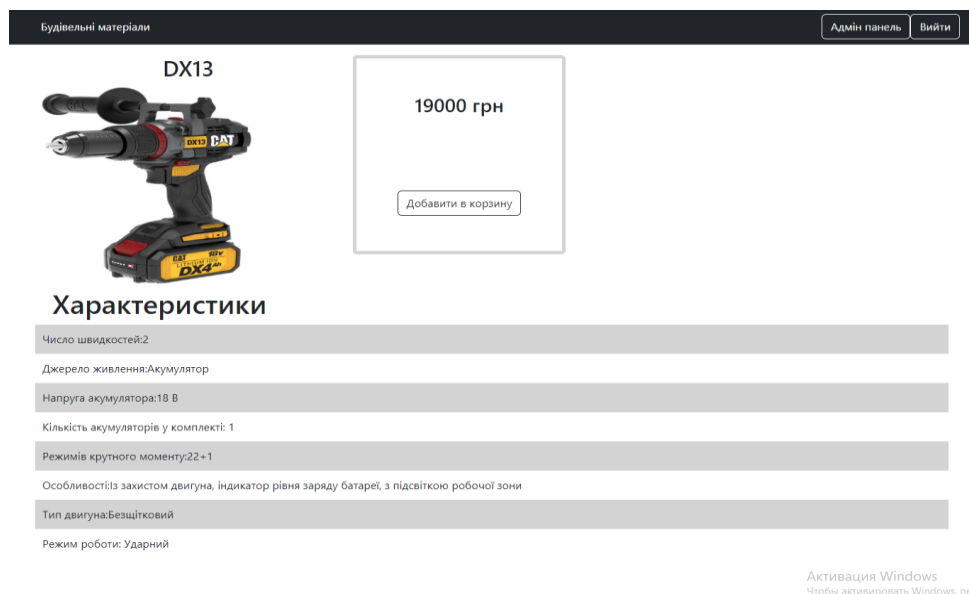


Рисунок 3.8 - Інформація про продукт

Було розроблено адмін-панель. Адмін-панель – це сторінка де користувач маючи роль адміну зможе добавляти якийсь продукт до інтернет-магазину або ж має можливість добавляти тип продукту та бренд.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Будівельні матеріали

Адмін панель Вийти

Добавити тип

Добавити бренд

Добавити інструмент

Рисунок 3.9 - Сторінка адмін-панелі

Також на сайті було розроблено сторінку де реалізована реєстрація та авторизація на сайті. Сторінка реєстрації

Будівельні матеріали

Авторизація

Введіть вашу пошту ...

Введіть ваш пароль ...

Немаєте акаунту? [Зареєструйся!](#)

Ввійти

Рисунок 3.10 - Сторінка авторизації

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Регістрація

Введіть вашу пошту ...

Введіть ваш пароль ...

Маєте акаунту? [Вийдіть!](#)

Регістрація

Рисунок 3.11 - Сторінка реєстрації

3.3 Тестування роботи системи

Для того щоб бути впевненим в правильності роботи інтернет магазину зробимо тестування функціоналу представленого на сайті.

Тестування програми - це процес перевірки та оцінки програмного забезпечення з метою виявлення помилок, дефектів або неправильностей у функціонуванні програми. Його основна мета полягає у забезпеченні якості продукту та підтвердженні його відповідності вимогам та очікуванням користувачів.

Існує багато різних методів тестування програм, а основні з них включають:

1. Юніт-тестування (Unit Testing): Цей метод використовується для тестування найменших можливих одиниць програми, таких як функції або методи. Він дозволяє перевірити, чи працюють ці одиниці правильно окремо від решти програми.

2. Інтеграційне тестування (Integration Testing): Під час інтеграційного тестування перевіряється взаємодія між різними компонентами або модулями програми. Цей метод дозволяє виявити проблеми, що виникають при злитті компонентів разом.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		47

3. Системне тестування (System Testing): У системному тестуванні перевіряється поведінка всієї системи або програми в цілому. Він включає тестування функціональності, продуктивності, надійності та інших аспектів програми.

4. Автоматизоване тестування (Automated Testing): Автоматизоване тестування використовується для автоматизації виконання тестів, замість їх ручного виконання. Це дозволяє прискорити процес тестування та забезпечити більшу стабільність результатів.

Метою тестування програми є забезпечення якості програмного забезпечення та виявлення помилок або дефектів, які можуть вплинути на його функціонування. Відповідне тестування допомагає забезпечити надійність, функціональність, продуктивність та зручність використання програми, а також зменшити ризик виникнення проблем після випуску на ринок.

Приклад тестування авторизації/реєстрації на сайті.

Таблиця 3.1 - Тестування авторизації/реєстрації

Procedure	Expected Result	Status	Comment/Actual
Натиснути кнопку “Авторизація”	+	Пройшов	Відбувається перехід на сторінку авторизації
Натиснути кнопку “Зареєструватися”	+	Пройшов	Відбувся перехід на сторінку реєстрації
Натиснути кнопку “Реєстрація” не ввівши ні пошти ні паролю	+	Пройшов	Нічого не вводити. З’явилась помилка що введені невірний пароль та пошта
Спробувати зареєструватись ввівши пароль 12345 без пошти	+	Пройшов	

Продовження таблиці 3.1

Настиснути на кнопку “Реєстрація” ввівши всі данні вірно	+	Пройшов	Реєстрація успішна перехід на основну сторінку
З реєстрації настиснути на кнопку ”Ввійдіть”	+	Пройшов	Відбудеться перехід на сторінку авторизації
Спробувати авторизуватись, ввівши тільки пароль	+	Пройшов	Виб’є помилку такого користувача не знайдено
Спробувати ввести пошту admim@gmail.com	+	Пройшов	З’явиться помилка “Вказаний невірний пароль”
Спробувати авторизуватись, ввівши всі дані існуючого користувача вірно	+	Пройшов	Авторизація успішна. Було здійснено перехід на головну сторінку

Висновки до третього розділу

Під час виконання третього розділу було зроблено фронтенд частину інтернет-магазину, згідно всіх вимог та критеріїв. Це було зроблено за допомогою бібліотеки React.

Також було продемонстровано порядок роботи з інтернет-магазином, були оглянуті його основні можливості. Було продемонстровано тестування одну з багатьох функціональностей представлених на сайті.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		49

ВИСНОВКИ

У ході виконання першого розділу була сформульовано задачі на виконання кваліфікаційної роботи, визначені основні вимоги щодо розробки проекту.

В результаті аналізів аналогів інтернет-магазинів по поставленим критеріям. Було визначено та пояснено вибір основних цілей розробки, визначено необхідного мінімального функціоналу та визначено вимог до побудови користувацького інтерфейсу.

Отримавши вимоги та бачення інтернет-магазинів, для розробки проекту була вибрана архітектура, яка є потрібною та оптимальною. Визначено та пояснено вибір стеку технологій для реалізації проекту – PostgreSQL, Express.js, React, Node.js

Під час виконання другого розділу було пройдено чимало моментів розробки інтернет-магазину, пророблено етапи проектування та створення серверної частини веб-додатку.

По початковому баченні сайту, оснований на вимогах та критеріях з першого розділу кваліфікаційної роботи, були побудовані діаграми діаграма варіантів використання, яка показує можливі дії користувачів, діаграма діяльності, яка зображує алгоритми роботи активностей в системі.

Була побудована діаграми бази даних, визначені залежності між колекціями, типи полів.

Далі була проведене масштабна робота по побудові повноцінного сервера інтернет-магазину на основі проектування, за допомогою засобів розробки Node.js в поєднання з Express.js. У висновку отримали пророблену бекенд частину, яка готова до взаємодії з фронтендом.

Під час виконання третього розділу було зроблено фронтенд частину інтернет-магазину, згідно всіх вимог та критеріїв. Це було зроблено за допомогою бібліотеки React.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		50

Також було продемонстровано порядок роботи з інтернет-магазиномі, були оглянуті його основні можливості. Було продемонстрованно тестування одну з багатьох функціональностей представлених на сайті.

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		51

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Node.js. *Node.js*. URL: <https://nodejs.org/uk> (дата звернення: 15.05.2023).
2. PHP: Hypertext Preprocessor. *PHP: Hypertext Preprocessor*. URL: <https://www.php.net/> (дата звернення: 15.05.2023).
3. Express - Node.js web application framework. *Express - Node.js web application framework*. URL: <https://expressjs.com/> (дата звернення: 16.05.2023).
4. Koa - next generation web framework for node.js. *Koa - next generation web framework for node.js*. URL: <https://koajs.com/> (дата звернення: 16.05.2023).
5. React. *React*. URL: <https://react.dev/> дата звернення: 18.06.2023).
6. Інтернет магазин жалюзі Бердичівціна. URL: <https://sem.rv.ua/internet-magazin-zhalyuzi-berdichiv.html> (дата звернення: 25.06.2023).
7. АЛЛО - національний маркетплейс із найширшим асортиментом. *АЛЛО - національний маркетплейс із найширшим асортиментом*. URL: <https://allo.ua/> (дата звернення: 25.06.2023).
8. JWT.IO. *JSON Web Tokens - jwt.io*. URL: <https://jwt.io/> (дата звернення: 19.06.2023).
9. Home v6.14.0. *Home v6.14.0 | React Router*. URL: <https://reactrouter.com/en/main> (дата звернення: 18.06.2023).
10. npm. *npm*. URL: <https://www.npmjs.com/> (дата звернення: 25.06.2023).
11. React Bootstrap | React Bootstrap. *React Bootstrap | React Bootstrap*. URL: <https://react-bootstrap.netlify.app/> (дата звернення: 25.06.2023).
12. React integration · MobX ua. *MobX*. URL: <https://mobx.js.org/react-integration.html> (дата звернення: 25.06.2023).

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				52
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТОК

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Лістинг `deviceController.js`, контролера приладу на сервері, містить в собі створення приладу, реалізація функції для отримання всіх приладів та функція отримання одного конкретного дивайсу.

```
const uuid = require('uuid')
const path = require('path');
const {Device, DeviceInfo} = require("../models/models");
const ApiError = require('../error/ApiError')
class DeviceController{
  async create(req,res,next){
    try {
      let {name,price, brandId,typeId,info} = req.body
      const {img} = req.files
      let fileName = uuid.v4() + ".jpg"
      img.mv(path.resolve(__dirname, '..', 'static',
fileName))
      const device = await
Device.create({name,price,brandId,typeId,img:fileName})

      if(info){
        info = JSON.parse(info)
        info.forEach(i=>
          DeviceInfo.create({
            title: i.title,
            description: i.description,
            deviceId: device.id
          })
        )
      }

      return res.json(device)
    }catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }

  async getAll(req,res){
    let {brandId ,typeId,limit,page } = req.query
    page = page || 1
    limit = limit || 9
    let offset = page* limit - limit
    let devices;
    if(!brandId && !typeId){
      devices = await Device.findAndCountAll({limit,
offset})
    }
    if(brandId && !typeId){
```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        devices = await
Device.findAndCountAll({where:{brandId}, limit,offset,})

    }
    if(!brandId && typeId){
        devices = await
Device.findAndCountAll({where:{typeId}, limit,offset,})

    }
    if(brandId && typeId){
        devices = await Device.findAndCountAll({where:{typeId,
brandId}, limit,offset,})

    }
    return res.json(devices)

}

async getOne(req,res){
    const {id} = req.params
    const device = await Device.findOne(
        {
            where:{id},
            include:[{model:DeviceInfo,as:'info'}]
        },
    )
    return res.json(device)
}

}

module.exports = new DeviceController()

```

Реалізація бази даних в проєкті та реалізація схеми як дані будуть зберігатись в базі даних.(models.js)

```

const sequelize = require ('../db')
const {DataTypes} = require('sequelize')

const User = sequelize.define('user',{
    id:{type:DataTypes.INTEGER, primaryKey:true,autoIncrement:
true},// INTEGER - тип числовий
    email:{type:DataTypes.STRING, unique: true},
    password:{type:DataTypes.STRING},
    role:{type:DataTypes.STRING,defaultValue:"USER"}
})

const Basket = sequelize.define('basket',{
    id:{type:DataTypes.INTEGER, primaryKey:true,autoIncrement:
true},
})

const BasketDevice = sequelize.define('basket_device',{
    id:{type:DataTypes.INTEGER, primaryKey:true,autoIncrement:

```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

true},
}))

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},
  rating:{type:DataTypes.INTEGER, defaultValue: 0},
  img:{type:DataTypes.STRING, allowNull:false},
}))

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

const Brand = sequelize.define('brand',{
  id:{type:DataTypes.INTEGER, primaryKey:true,autoIncrement:
true},
  name:{type:DataTypes.STRING,unique:true, allowNull: false},
}))

const Rating = sequelize.define('rating',{
  id:{type:DataTypes.INTEGER, primaryKey:true,autoIncrement:
true},
  rate:{type:DataTypes.INTEGER, allowNull: false},
}))

const DeviceInfo = sequelize.define('device_info',{
  id:{type:DataTypes.INTEGER, primaryKey:true,autoIncrement:
true},
  title:{type:DataTypes.STRING, allowNull: false},
  description:{type:DataTypes.STRING, allowNull: false},
}))

const TypeBrand = sequelize.define('type_brand',{
  id:{type:DataTypes.INTEGER, primaryKey:true,autoIncrement:
true},
}))

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

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

Basket.hasMany(BasketDevice)
BasketDevice.belongsTo(Basket)

Type.hasMany(Device)
Device.belongsTo(Type)

```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		56

```

Brand.hasMany(Device)
Device.belongsTo(Brand)

Device.hasMany(Rating)
Rating.belongsTo(Device)

Device.hasMany(BasketDevice)
BasketDevice.belongsTo(Device)

Device.hasMany(DeviceInfo, {as: 'info'});
DeviceInfo.belongsTo(Device)

Type.belongsToMany(Brand, {through: TypeBrand})
Brand.belongsToMany(Type, {through: TypeBrand})

module.exports = {
  User,
  Basket,
  BasketDevice,
  Device,
  Type,
  Brand,
  TypeBrand,
  DeviceInfo,
}

```

		Сторожук С.О			КН.КР.Б – 122 – 23 – ПЗ	Арк.
		Петросян Р.В				
Змн.	Арк.	№ докум.	Підпис	Дата		57