

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр

(назва освітнього ступеня)

на тему: Розробка інтернет-магазину автозапчастин
«Auto Tech» з використанням технологій Node.JS та React

Виконав(ла): студент(ка) 4 курсу, групи СПс-42
спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

	(підпис)	Бобик А.О. (прізвище та ініціали)
Керівник	(підпис)	Цуприк Г.Б. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	(прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-сайту інтернет магазину автозапчастин "Auto Tech" на React.js// Дипломний проект // Бобик Анатолій // Тернопільський національний технічний університет імені Івана Пулюя», група СПс-42// Тернопіль, 2024 // с. – 116 , рис. – 61, табл. – 4, кресл. – 4, додат. – 4 .

Ключові слова: ВЕБ САЙТ, REACT, REDUX, JS, NODE, ІНТЕРНЕТ МАГАЗИН.

Мета роботи - це розробка веб-сайту інтернет магазину автозапчастин «Auto Tech» на React.js та nodeJS.

Пояснювальна записка складається з п'яти розділів.

В першому розділі виконано аналітичний огляд існуючих рішень та підібрано фреймворк. Описано призначення даної розробки, вимоги до програмного, апаратного забезпечення та ведення документації та технічне завдання.

В другому розділі дається загальний опис задачі та її специфічні особливості. Описано методи побудови сайту з використанням вибраних мов програмування, баз даних. Описано процес та логіку здійснення розробки проекту.

В третьому розділі описано процедури по встановленню програмного забезпечення та його налаштуванню. Також описано створення та використання тестів і тестових програм та інструкцію з експлуатації веб-додатку.

В четвертому розділі розраховано вартість розроблюваного веб-сайту, а в п'ятому описано техніку безпеки та екологічні вимоги під час виконання даного проекту.

ABSTRACT

Development of the website of the auto parts online store "Auto Tech" on React.js// Diploma project // Bobyk Anatoliy // Ternopil National Technical University named after Ivan Pulyuy", group SPs-42// Ternopil, 2024 // p. – 87, fig. - 61, tab. - 4, armchair. - 4, add. - 4.

Keywords: WEB SITE, REACT, REDUX, JS, NODE, INTERNET SHOP.

The purpose of the work is the development of the website of the auto parts online store "Auto Tech" using React.js and nodeJS.

The explanatory note consists of five sections.

In the first section, an analytical review of existing solutions was performed and a framework was selected. The purpose of this development, the requirements for software, hardware and documentation, and the technical task are described.

The second section gives a general description of the problem and its specific features. Methods of building a site using selected programming languages and databases are described. The process and logic of project development are described.

The third section describes the procedures for installing the software and configuring it. It also describes the creation and use of tests and test programs and the instructions for using the web application.

The fourth section calculates the cost of the developed website, and the fifth section describes the safety and environmental requirements during the implementation of this project.

ЗМІСТ

АНОТАЦІЯ	4
ВСТУП.....	8
1 ЗАГАЛЬНИЙ РОЗДІЛ.....	9
1.1 Аналітичний огляд існуючих рішень.....	9
1.2 Технічне завдання	Помилка! Закладку не визначено. 7
1.2.1 Найменування та область застосування.....	Помилка! Закладку не визначено. 7
1.2.2 Призначення розробки.....	Помилка! Закладку не визначено. 7
1.2.3 Вимоги до функціоналу веб-сайту	Помилка! Закладку не визначено. 7
1.2.4 Вимоги до програмної документації.....	Помилка! Закладку не визначено. 8
1.2.5 Стадії та етапи розробки	179
1.2.6 Порядок тестування та прийому.....	20
2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ	21
2.1 Розробка структури сайту і веб-сторінок.....	Помилка! Закладку не визначено. 2
2.2 Створення та верстка сторінок сайту.....	27
2.3 Розробка структури бази даних сайту.....	42
2.4 Програмування сайту.....	46
2.4.1 Написання клієнтської частини.....	46
2.4.2 Написання admin-частини.....	48
2.5 Тестування веб-сайту.....	50
3 РОБОТА З САЙТОМ В ІНТЕРНЕТІ	57
3.1 Інструкція з розміщення сайту в Інтернеті.....	57
3.2 Інструкція з обслуговування та наповнення сайту.....	60
3.3 Інструкція з популяризації та підтримки сайту	63
4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ	66
4.1 Основні причини електротравматизму. Електробезпека користувача ВДТ.....	67
4.2 Протипожежне водопостачання	69

ВИСНОВКИ	72
ПЕРЕЛІК ПОСИЛАНЬ	73
ДОДАТКИ	73
ДОДАТОК А – Лістинг файлу «App.jsx».....	74
ДОДАТОК Б – Лістинг файлу «Product.jsx».....	77
ДОДАТОК В – Лістинг файлу «product.js»	80
ДОДАТОК Г – Лістинг файлу «NewProduct.jsx»	83
ДОДАТОК Д – Диск із кваліфікаційною роботою бакалавра	87

ВСТУП

У наш існує чимало нових та сучасних технологій, сфера бізнесу, звичайно також не є винятком. Використання цих технологій в бізнесі значно покращує його роботу, досить часто робить його надзвичайно зручним, а часом просто необхідним.

Якщо подумати, то ще з кінця минулого сторіччя у всій світовій економіці завдяки швидкому розвитку саме інформаційних технологій відбулися значні зміни. Конкуренція завжди зростала і продовжує зростати, а це, в свою чергу, змушує всі виробничі підприємства використовувати і вдосконалювати існуючі інформаційні системи та обов'язково створювати нові. Якщо у підприємства раптом відсутня діяльність якогось характеру, то у всесвітній мережі це буде розцінюватися як недолік. Брак часу у споживачів змушує все більше придбань робити через Інтернет, а це, в свою чергу, зумовлює ще більший розвиток електронної торгівлі завдяки появі нових її різновидів.

Дивлячись на це все, постає завдання покращення ефективності використання персонального часу покупця задля його зручності та швидкого пошуку необхідного товару чи отримання інформації про нього. Особливість сервісу полягає в тому, що він надає достатньо можливостей, які необхідні для заощадження часу та підвищення рівня обслуговування покупців. І в такому разі замовнику не потрібно виходити з дому, їхати в той чи інший магазин, не знаючи точно чи є там необхідний товар, а просто зайти в Інтернет, вибрати магазин, подивитися чи є товар в наявності і якщо так, то замовити його.

Саме тому появилась необхідність створювати інтернет-магазини практично для будь-якого бізнесу і кожен власник старається придбати зручний, швидкий, красивий, привабливий сервіс.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

У наш час інтернет–магазини стали максимально розповсюджені, вони розроблені для ведення зручного та швидкого бізнесу. Їх особливістю є те, що людина має змогу влаштовувати купівлю чи продаж будь-якого товару не виходячи з дому, оформлювати замовлення, обирати саме те, що їй потрібно за максимально швидкий час. Дуже багатьом людям лінє йти в магазини, перевіряти чи є там необхідні їм товари. І навіть самі прогресивні люди не мають бажання затрачати лишній час на пошук необхідних їм товарів, а вважають за краще просто зайти на веб-сайт і одразу замовити те що треба всього в два кліки. Тому, як стало зрозуміло, весь прогрес йде в мережу-інтернет і не користуватися такою чудовою можливістю буде просто не розумно. Із-за всіх цих вищеперелічених причин було прийнято досить серйозне рішення – створити даний інтернет-магазин для нашої сфери бізнесу.

Наша ж сфера бізнесу представляє магазин автозапчастин. Звичайно моя розробка інтернет-магазину автозапчастин «Auto Tech» буде далеко не першим існуючим веб-сайтом саме автозапчастин, тому перед його створенням буде досить логічно спершу подивитися на вже давно створенні та існуючі різні магазини автозапчастин в мережі-інтернет, щоб мати якесь уявлення як він приблизно повинен виглядати та виділити певні моменти, які обов'язково або повинні бути, або має бути їх відсутність. Такий підхід тільки покращить розроблюваний інтернет-магазин та, можливо, зробить його цікавішим.

Отож, після абсолютно легкого пошуку вже існуючих інтернет-магазинів автозапчастин було знайдено чимало їх варіантів. Давайте розглянемо декілька обраних варіантів, щоб провести невеличкий розбір.

Перший магазин, який було обрано можна знайти за наступним посиланням:

https://filterplus.com.ua/ua/?gclid=CjwKCAjwqauVBhBGEiwAXOepkdlf7pryHy2L_ZnkWtqv7tAfW3N9bvWefeafwwQK8tVH8oPdmuHa6hoCCZwQAvD_BwE.

Назва даного інтернет-магазину – «FILTER PLUS». Спершу розглянемо його дизайн, який є достатньо не цікавим, банальним та не привабливим, що однозначно являється великим мінусом для цього сайту. На рисунку 1.1 можна побачити дизайн магазину автозапчастин «FILTER PLUS».

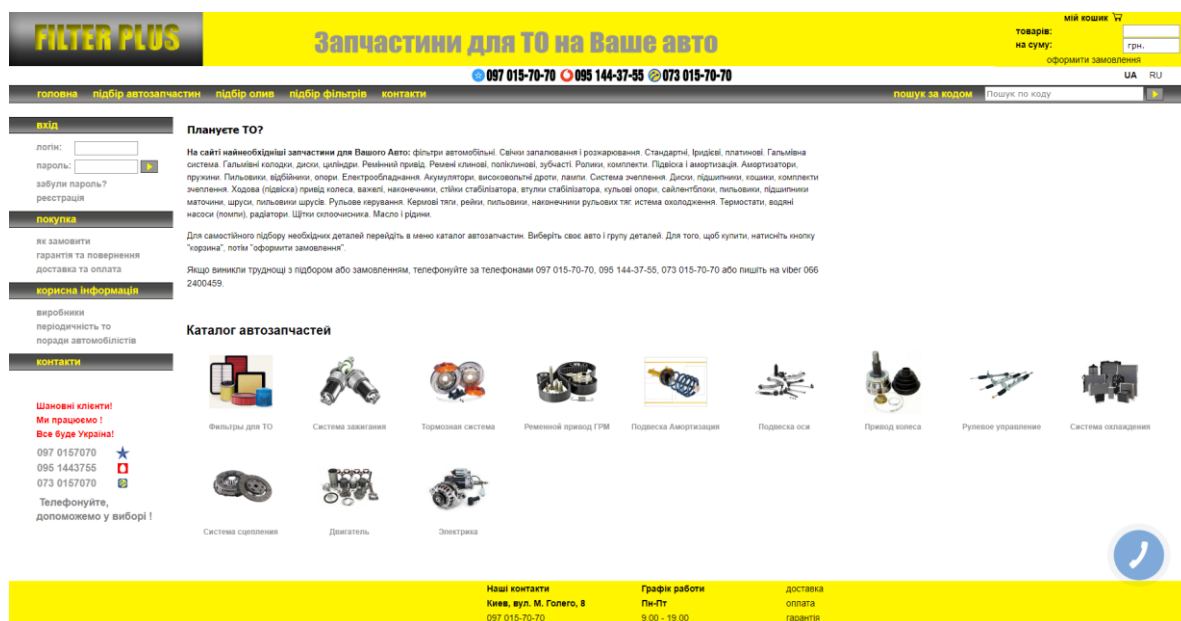


Рисунок 1.1 – Дизайн інтернет-магазину «FILTER PLUS»

Кольори дизайну підібрані не погано, але саме розташування товарів, форм реєстрації чи логіну, корзини та її опису (тобто кількості товарів та суму, якої вони варті), а також інших допоміжних функціоналів не дуже хороше. Оскільки це тільки перший автозапчастин, який ми розглянули, то ще повного уявлення як може виглядати кращий подібний веб-сайт у нас немає. Але цей однозначно виглядає не дуже привабливим. Починаючи від тієї самої невеликої форми реєстрації або логіну, яка чомусь знаходиться зліва, а це є досить не звичним розташуванням, завершуючи корзиною, яка чомусь показує кількість товарів і їх сумарну вартість не заходячи в середину неї.

Додатковим недоліком цього сайту є те, що у ньому немає повноцінного перекладу всього тексту, що знаходиться на ньому. Про що ж тут йдеться?

На перший погляд, здається, що весь текст йде однією мовою – українською, але якщо придивитися, то стає видно, що сам каталог автозапчастин написаний на російській мові, а також назви всіх наявних типів товарів також написано на російській, не кажучи, що контактна інформацію взагалі написана змішано, тобто початок на українській, а кінець – російській. Якщо зайти в опис будь-якого типу автозапчастин, то і там можна помітити змішування українського і російського тексту, приклад цього можна побачити на рисунку 1.2 наведеного нижче.

Система зажигания

Оберіть марку автомобіля, потім модель

марка:

модель:

Система зажигания: свечи зажигания, свечи накаливания, катушки зажигания, реле свечей накала.

Интернет-магазин Filterplus предлагает Вашему вниманию свечи зажигания, свечи накаливания, провода зажигания, катушки зажигания, реле свечей накала ведущих мировых производителей. Если Вы знаете код запчасти и не нуждаетесь в подборе, для просмотра цен и оформления заказа введите код в поле "поиск по коду" (справа в верхнем меню) и нажмите кнопку поиска. Для самостоятельного подбора и оформления заказа перейдите в каталог автозапчастей FilterPlus. Если возникли трудности с подбором или заказом, звоните мы обязательно вам поможем. Ниже приведены самые популярные бренды производителей системы зажигания.



NGK (Германия)

Denso (Япония)

Bosch (Германия)



Beru (Германия)

Champion (США)

Рисунок 1.2 – Опис автозапчастини у інтернет-магазині «FILTER PLUS»

Проте, як і, мабуть, в будь-якого веб-сайту у цього є свої переваги або «плюси». До них можна віднести зручне сортування товару, тобто ви просто вибираєте необхідну вам автозапчастину і вказуєте фільтри на марку вашого автомобіля, і звичайно його модель; в свою чергу магазин одразу вам показує весь наявний у ньому товар, що максимально спрощує пошуки потрібних речей. Також що стосується пошуку товарів, то можна замітити поле, де можна просто ввести (звісно якщо ви знаєте) код товару і одразу його знайти.

Наступний інтернет-магазин автозапчастин, який буде розглянуто, у вас буде можливість знайти за наступним достатньо довгим посиланням:

https://dok.ua/?utm_source=google&utm_medium=cpc&utm_campaign=10549030453&utm_content=&utm_term=%2Вінтернет%20%2Вмагазин%20%2Вавтозапчастин&utm_id=448580544938&gclid=CjwKCAjwqauVBhBGEiwAXOepkfpAB-LrerEyalz9HaYrPMI8ybHIIvITLVxSE1uIXU_peP0B-C52YxoC37AQAvD_BwE.

Назва цього інтернет-магазину – «dok.ua». Давайте спершу розглянемо його дизайн, але одразу хочеться сказати, що він достатньо прийнятний, цікавий і сучасний. Дизайн цього магазину ми можемо побачити на рисунку 1.3, який знаходиться нижче.

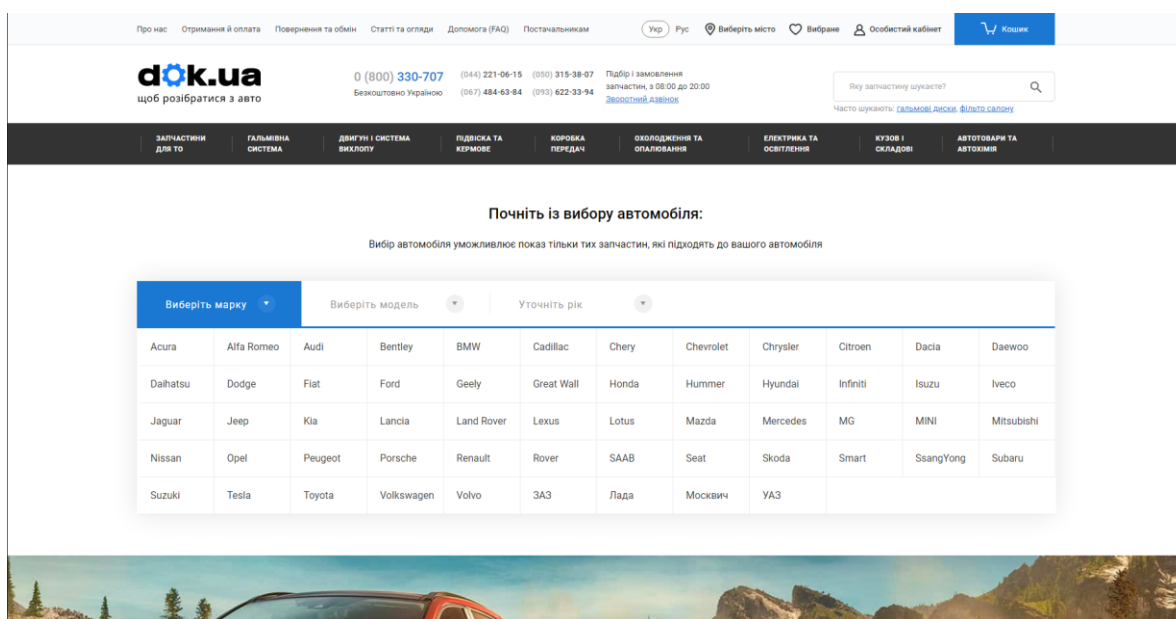


Рисунок 1.3 – Дизайн інтернет-магазину автозапчастин «dok.ua»

Це тільки одна з частин всього великого веб-сайту, він досить довго гортається вниз, тому повністю його показати аж ніяк не вийде, але вже навіть по ній можна сміливо відмітити чимало переваг або «плюсів» даного сайту. Отож, одразу дуже зручно зроблено, що всі допоміжні функції знаходяться з самого верху веб-сайту, тому якщо виникло якесь запитання чи щось потрібно, то користувачу не потрібно гортати весь цей великий сайт в самий низ.

Реєстрація користувача та логін у його особистий кабінет знаходить справу, що досить звично для абсолютно будь-якого сайту і це хороше рішення, на відмінну від попереднього сайту, який ми розглядали. Кошик тут також

знаходиться справа, він не показує одразу які товари ви хочете придбати, їх кількість і загальну суму, а у нього спершу потрібно зайти, і вже тоді буде отримано цю всю інформацію, що також властиве всім інтернет-магазинам, крім того, що був на попередньому розгляді. Окремо хочеться відмітити, що в кошику даного інтернет-магазину також вказується як правильно оформити покупку у ньому.

Переклад між мовами повністю працює на абсолютно весь наявний текст на даному сайті, немає жодних недоліків щодо цього.

Далі ми можемо бачити категорії товарів, яких є досить багато, вікна цих категорії є модальними, що також додає цікавості цьому сайту. Якщо вибрати якусь із категорій, а в ній – автозапчастину, то система видасть список всіх наявних в магазині запчастин за типом, а також одразу запропонує їх фільтрацію, якщо ви цього не зробили одразу. Фільтрація виглядає абсолютно так само як ми можемо бачити на рисунку 1.3. Тобто система нам пропонує вибрати спочатку марку свого автомобіля, тоді його модель і на сам кінець рік випуску цього автомобіля. Хочеться також відмітити, що без вказання одної характеристики у користувача немає можливості вказати іншу, тобто якщо користувач не вибере марку свого автомобіля, він не зможе вказати його модель, а такий підхід в свою чергу застерігає користувачів від помилок. В результаті всіх цих поділів на категорії і фільтрів необхідний товар буде знайдено за максимально швидкий час.

Якщо не затримуватися на цьому і йти далі, то ми можемо бачити красиві слайдові переходи, на кожному з яких пише якась перевага цього магазину над його конкурентами або додаткові його можливості про які не було сказано раніше і це досить актуально та сучасно, такий спосіб на даний час вважається один з найкращих «short презентацій магазину». Представлення одного з таких слайдів зображено на рисунку 1.4.

Також цей магазин представляє спеціальні товари, наприклад новинки або товари по знижці, приклад такого зображено на рисунку 1.5.

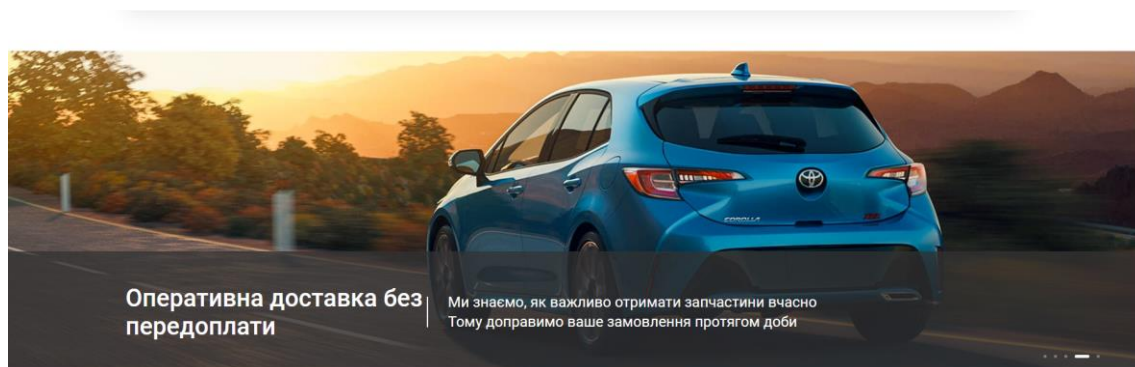


Рисунок 1.4 – Приклад слайдової частини інтернет-магазину «dok.ua»

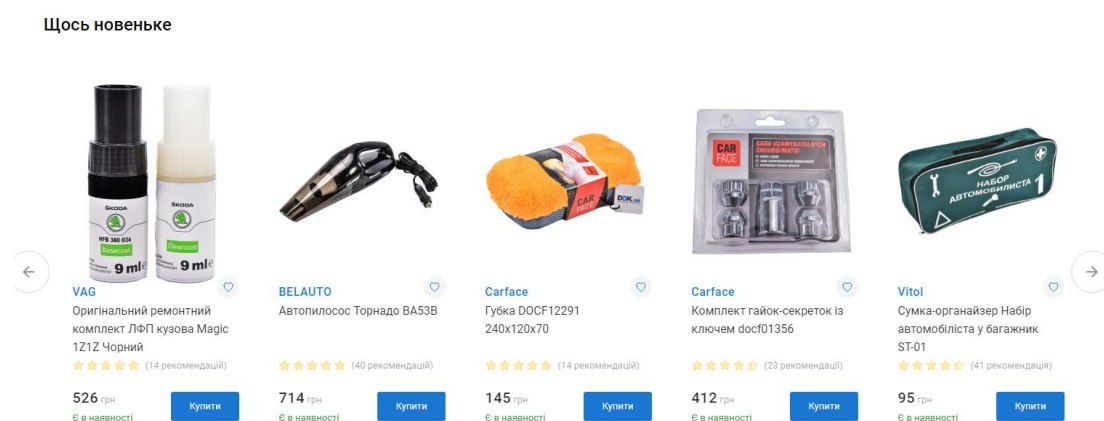


Рисунок 1.5 – Новинки інтернет-магазину «dok.ua»

Все вищеперелічене робить цей магазин максимально комфортним, цікавим, привабливим, уніфікованим та просто привертає увагу користувачів. Він може бути справжнім прикладом для написання власного інтернет-магазину. У ньому практично немає ніяких недоліків, винятком буде тільки те, що після вибору категорії та фільтрації товару важко повернутися на домашню сторінку, тому що пропадає логотип-посилання на неї, але це не є дуже вагомий недолік. Ну і можна ще сказати, що в деяких випадках сайт є досить громіздким і містить надто багато інформації, яку треба прогортати, до якої можна просто не дійти. Але це вже буде досить суб'єктивна думка окремих користувачів.

Отже, ми розглянули два приклади вже давно існуючих інтернет-магазинів автозапчастин, виділили їхні переваги та недоліки. Тепер ми маємо достатнє уявлення про вигляд магазинів автозапчастин в інтернеті, враховуючи їх

переваги та недоліки будемо старатися придумати та написати власний цікавий та хороший дизайн, а також функціональність для нашого сайту.

1.2 Технічне завдання

1.2.1 Найменування та область застосування

Найменування програми – інтернет-магазин автозапчастин «Auto Tech».

Область застосування програми – застосовується у сфері бізнесу, зокрема як описано вище для власників магазинів автозапчастин для ведення зручного і сучасного бізнесу, продажу своїх товарів в інтернеті, а також зручного придбання цих товарів покупцем.

1.2.2 Призначення розробки

Використовується покупцями для зручного придбання товарів онлайн, тобто за допомогою інтернету. Даний інтернет-магазин буде необхідним для покупців, у яких поламався автомобіль і їм необхідно придбати певну автозапчастину для його ремонту. Замість того, щоб тратити велику кількість часу на пошуки необхідних їм автозапчастин проходячи по різних магазинах, вони можуть зайти на сайт магазину автозапчастин і досить швидко перевірити чи є там необхідний їм товар. Також цілком природньо, що дана розробка досить вагомо буде покращувати роботу власнику магазину, оскільки він може вести власний бізнес онлайн, а не перебувати весь час на робочому місці, щоб мати змогу щось продати.

1.2.3 Вимоги до функціоналу веб-сайту

Інтернет-магазин автозапчастин «Auto Tech» повинен забезпечити можливість перегляду наявних автозапчастин у ньому, їх придбання, оптимізацію роботи з базою даних в якій будуть зберігатися певні дані, зокрема про товар, який замовили покупці у даному магазині, отримання даної інформації власником.

Час виконання заданої задачі миттєвий. Вона повинна виконуватись одразу, після натискання на будь-яку з клавіш, якщо умови будуть сумісні. Як тільки будуть введені дані про товар, і буде натиснено на клавішу «замовити», замовлення повинно одразу бути зареєстрованим, і в свою чергу програма це відображає у списку. Або при видаленні покупця зі списку, після натискання клавіші видалення, замовника повинно одразу вилучати зі списку.

Надійне функціонування програми обов'язково повинне бути забезпечене наявністю блоку перевірки правильності введених даних. Це збереже від помилок, що часто виникають при виконанні звичайної рутинної роботи користувачем. Також необхідно забезпечити достатню стійкість програми до некваліфікованих дій користувача, що зазвичай досягається супроводом будь-яких операцій в програмі достатньо чіткими і зрозумілими повідомленнями, адже це досить часто помітна проблема користувача.

Розроблений програмний продукт не потребуватиме будь-яких видів обслуговування, для його використання достатньо буде навиків користувача ПК.

1.2.4 Вимоги до програмної документації

По закінченню даної розробки програмного забезпечення потрібно

підготувати наступну документацію:

- інструкція для користування адміністратору;
- загальні відомості про можливості програми;
- інструкція з експлуатації.

1.2.5 Стадії та етапи розробки

Розробка інтернет-магазину автозапчастин «Auto Tech» буде мати такі стадії:

- аналіз;
- проектування;
- еволюція.

Метою аналізу завжди є максимально повний опис поставленої нам задачі. На цій стадії як правило аналізується предметна область задачі, а також виконується об'єктна декомпозиція майбутньої програми та визначаються особливості поведінки об'єктів. Вже за результатами аналізу буде розроблятися структурна схема програми та описуватись об'єкти і події над ними.

Стадія проектування поділяється на логічне та фізичне. Логічне проектування полягає у розробці структури класів, в ньому визначаються поля, які зберігають складові об'єктів, алгоритми методів, що описують поведінку об'єктів.

Фізичне проектування зазвичай містить об'єднання опису класів у модулі, а також вибір схеми їх підключення, визначення способів взаємодії з обладнанням, розуміння взаємодії з операційною системою та іншим програмним забезпеченням і обов'язково забезпечення синхронізації процесів для систем рівнобіжної обробки.

На етапі проектування зазвичай використовуються наступні методи

розробки класів як спадкування, композиція, наповнення. Результатом даного етапу проектування завжди є створення ієрархії класів та їх опис.

Етап послідовної реалізації та підключення класів до проекту – це еволюція системи. Цей процес, як правило, починається із створення основного інтерфейсу бібліотеки класів, далі описуються тип поля класу, розробляються алгоритми методів, а також уточнюються вимоги до програми і вносяться невеликі зміни. А вже одержаний прототип продукту тестується й налагоджується.

1.2.6 Порядок тестування та прийому

Прийом даного розробленого програмного забезпечення обов'язково повинен відбуватися на об'єкті Замовника в терміни, які зазначені в індивідуальному завданні. Для прийому роботи Виконавець повинен представити:

- його діючу програму, яка цілком відповідає даному технічному завданню;
- вихідний програмний код, який записаний разом із програмою на оптичний носій інформації;
- прийом даного програмного забезпечення повинен відбуватися перед комісією з кількох чоловік (один з яких – Замовник) у такій послідовності:
 - доповідь Виконавця про виконану роботу;
 - демонстрація Виконавцем роботи програми;
 - контрольні випробовування роботи програми;
 - відповіді на запитання і зауваження комісії.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

2.1 Розробка структури сайту і веб-сторінок

Загалом створення будь-якого веб-сайту починається зі створення самої інформаційної моделі розроблюваного сайту. Насамперед, давайте розберемося що ж таке структура сайту та для чого вона взагалі потрібна [1].

Структура сайту - це зазвичай досить грамотно побудована навігація, тобто розподіл елементів оформлення зручним стилем, органічний зв'язок безпосередньо між внутрішніми сторінками і самими розділами. Створення структури сайту часто вирішує чималу кількість важливих завдань, а також демонструє рівень професіоналізму та кваліфікації організації.

Підходячи до цього питання в першу чергу було вирішено вирішено, яку саме інформацію будемо розміщати на сторінках інтернет-магазину. Після довгих та детальних аналізів різних веб-сайтів, було прийнято рішення, що при вході в інтернет-магазин одразу буде можливість подивитися на товари, які як правило, а це підтверджує статистика, цікавлять відвідувачів найбільше.

Якщо ж з того досить нестандартного, проте хорошого переліку їх нічого не зацікавить, то вони, звичайно будуть мати змогу спуститися нижче. Далі їх чекає абсолютно стандартний та мабуть найчастіше практикований підхід практично будь-яким веб-сайтом, тобто всім нам відомі категорії товарів. Категорії собою представляють поділ всіх існуючих товарів даного інтернет-магазину на групи, яким вони відповідають. Тут йдеться про неймовірну зручність класифікації товарів, адже якщо б її не існувало, то товар просто знаходився б на сайті в перемішку, що робить його перегляд достатньо довгим та нудним.

Коли інтернет-магазини писалися вперше і входили в експлуатацію, то всім відомо, що бувало чимало різних випадків, коли відвідувачі таких сайтів просто не витримували над надзвичайно довгим пошуком необхідних їм товарів

і залишали їх, не додивившись всіх наявних товарів до кінця списку, а в гіршому випадку – навіть не дійшовши до середини; досить часто потрібні товари були в подібних магазинах, але їх пошук міг забирати стільки часу, що було легше вийти з дому, потративши трішки свого часу, але без всяких зусиль купити потрібний товар в звичайному магазині.

Впротивагу цьому і виникла ідея створення тих самих категорій. Вони поділяють весь наявний товар на загальні групи, які будуть конкретизувати пошук товару. І якщо умовно в магазині загалом можна переглянути 500 товарів, з яких тільки кожних 50 буде подібними за якоюсь приналежністю і вони будуть розміщені у випадковому порядку серед тих 500-ста товарів, то це погодьтєся буде досить довго, адже прийде́тсья їх всіх переглядати. Якщо ж в цьому інтернет-магазині буде існувати такий поділ на певні групки за якоюсь приналежністю, то відвідувачу потрібно буде переглянути лише одну конкретну групу товарів, тобто всього 50 товарів і вже це буде неймовірно швидко.

Тому було прийнято рішення на обов'язкове їх введення. За допомогою них відвідувач нашого інтернет-магазину зможе максимально швидко перевірити наявний товар, який його цікавить і якщо він його буде задовольняти, то відповідно замовити його.

Далі, що є цілком природньо, в кожній категорії буде міститися якась кількість товарів. Ці товари не повинні потрапляти до певної категорій просто так, чи у випадковому порядку, без всякої спорідненої властивості чи приналежності. Це означає, що всі товари, знаходячись в конкретній категорії будуть мати щось спільне між собою, іншими словами те, що буде їх поєднувати у ній.

Відштовхуючись від цієї думки, можна зробити надзвичайно велику кількість різних категорій, а також їхніх підкатегорій, тому що шукати щось спільне між товарами і робити їх поділ можна майже вічно, бо завжди знайдеться якась характеристика, за якою все ж таки можна буде відрізнити один товар від іншого. Такий підхід, може, бути хорошим для інтернет-магазинів з дуже

великою кількістю товарів, але точно не для нашого скоромного магазинчику.

Тому у моєму інтернет-магазині автозапчастин «Auto Tech» буде всього декілька різних категорій за якими буде можливість класифікувати весь наш наявний товар.

Що ж, далі нам потрібно побачити як будуть розміщені та зв'язані всі категорії та всі товари у нашому інтернет-магазині. Для цього я зобразив структуру сайту і ми можемо побачити її частину на рисунку 2.1, яка знаходиться нижче.

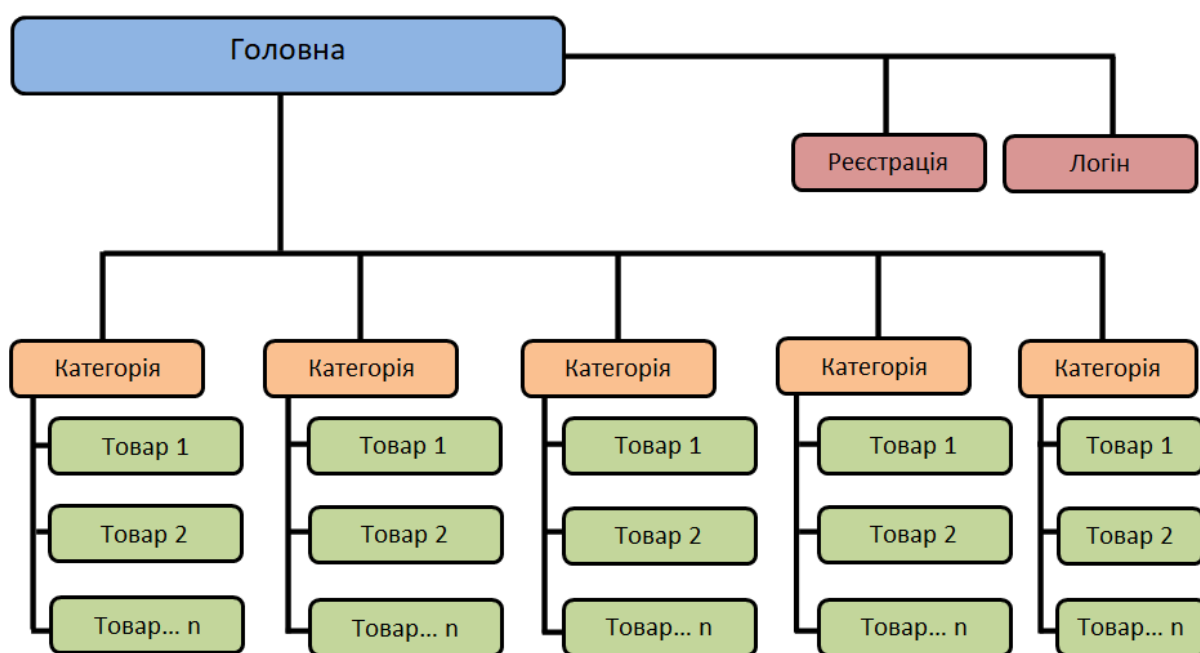


Рисунок 2.1 – Структура інтернет-магазину автозапчастин «Auto Tech»

Тепер, після того як ми побачили розміщення і зв'язки всіх структурних частин нашого інтернет-магазину автозапчастин, можна зробити цілий ряд наступних висновків.

Стає зрозуміло, що в ньому також є форми для реєстрації користувачів або їх авторизації, якщо користувачі були попередньо / раніше зареєстровані. Якщо ж ні, то вони роблять абсолютно стандартну операцію – реєструються на даному сайті для подальшого його використання, а точніше, після реєстрації вони будуть

мати змогу щось купувати у цьому інтернет-магазині.

Як і було сказано вище, як тільки користувач потрапляє на сайт цього інтернет-магазину, він має змогу спершу побачити найцікавіші та найпопулярніші товари, на думку продавця, що є у ньому. Вони не просто розкидані собі у «вільному плаванні» по магазину, а згруповані у так звані категорії чи розділи.

Раптом ці товари не зацікавили відвідувача, то як раніше було згадано, він спокійно гортає сайт нижче і бачить основні категорії наявних товарів. Не пильним оком видно, що їх всього є п'ять, а це досить не велика кількість і зручна у користуванні. Якщо ж захотілося поміркувати, які ж категорії можуть бути у магазині автозапчастин, то я одразу відповім, що це буде поділ по марці автомобіля. Так, автозапчастин є нереально велика кількість і було б досить логічно та стандартно поділити їх на категорії по типу самих автозапчастин. Але такий підхід є досить стандартним, всюди використовуваним і не уніфікованим. Тому для зацікавлення саме у моєму інтернет-магазині, було прийнято рішення зробити нестандартний розподіл по марках автомобілів. Одразу, мабуть, думається, що тут нічого особливого немає. Ну відповідають категорії маркам автомобілів та й що? Напевно далі будуть йти подібні підкатегорії, але вже по стандартному типу автозапчастин.

Як ви можете побачити на структурній схемі, в даному рішенні ніяких підкатегорій для товарів немає. Не дивлячись на важливість розподілення товару, оскільки це займає менше часу при пошуку товару по категоріям було прийнято рішення замість звичайних підкатегорій використати фільтрацію наявних товарів, які відповідають конкретній категорії. Оскільки ми вже вибрали якусь категорію, то слід зауважити що мова йде про марку автомобіля, то далі нам необхідно обрати тип автозапчастини, яку ми шукаємо, а також обов'язково крім марки повинна бути і модель автомобіля. Відповідно до цієї логіки і було розроблено фільтрацію у інтернет-магазині. Коли користувач зайшов у якусь із категорій, він просто за допомогою цих фільтрів обирає тип автозапчастини і

модель свого автомобіля і отримує список всіх наявних товарів, що вже точно будуть відповідати його запиту. А це в свою чергу зробить просто максимально швидкий пошук потрібного товару і цим самим є великим плюсом для мого магазину.

Що ж можна сказати про товари у нашому інтернет-магазині автозапчастин «Auto Tech»: за структурною схемою стає зрозуміло, що самих товарів є не один чи два, а якась чимала кількість. Це проясняється у тому, що після структурного елемента «Товар 2» йде не структурний елемент під назвою «Товар 3», а для зручності цей елемент позначено як «Товар...n», що означає якусь n-ну кількість товару (а зазвичай так позначається доволі вагома кількість чогось). Більше ніякої додаткової інформації про ці товари за структурною схемою ми, на жаль, отримати не в змозі [1].

Далі йде етап розробки веб-сторінок, які будуть присутні на моєму сайті. Для початку потрібно уявляти які вони бувають, а вже тоді буде можливість продумати які саме сторінки будуть у проєкті. На рисунку 2.2 наведено деякі приклади загального розподілення різних веб-сторінок.

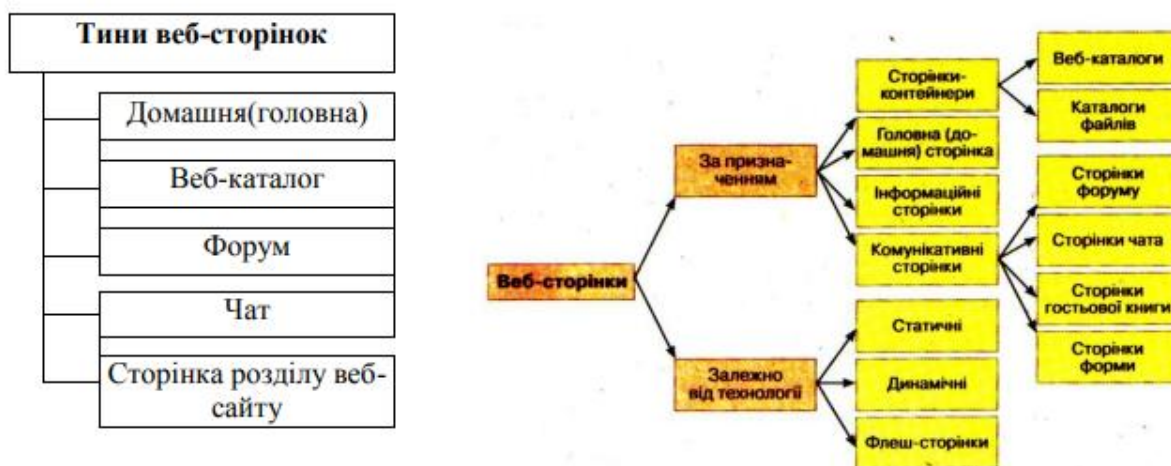


Рисунок 2.2 – Приклади розподілення веб-сторінок

Так як у інтернет-магазині планується написання двох різних частин – клієнтської та адміністративної, то у ньому відповідно буде і розподіл всіх написаних веб-сторінок за цими двома типами.

Отож, почнемо з розробки веб-сторінок для клієнтської частини. Як і будь-якому сайті у нас буде домашня сторінка сайту. У ній буде поєднано всі необхідні компоненти, такі як «Navbar», «Slider», «Categories», «Products», «Newsletter», «Footer» між собою і за допомогою цього створювати загальний дизайн сайту. Це буде головна сторінка сайту, на яку можна буде повернутися з будь-якої його клієнтської частини. Далі на нашому веб-сайті обов'язково будуть дві форми – реєстрації та логіну, і це повинні бути дві окремі сторінки, вони завжди є в будь-якому інтернет-магазині, тут додаткових пояснень, мабуть, не потрібно. Наступною сторінкою клієнтської частини буде корзина, яка призначена для зберігання та відображення обраних користувачем товарів, підрахунку сумарної кількості грошей, які потрібно заплатити за обрані товари, ну і звичайно безпосередню їх оплату. Для відображення успішного замовлення та оплати буде розроблена сторінка під назвою «Success». За допомогою неї ми будемо знати, що замовлення було успішно оформлено. І в решті-решт дві чи не головні сторінки для клієнта – список всіх наявних у магазині товарів та при виборі якогось з них буде відображатися сторінка його опису та певної інформації про нього. В сумі у нас повинно бути 7 веб-сторінок.

Далі буде йти розробка адмін-частини, яка призначена для роботи адміністратора, зокрема перевірка різних статистик, додавання, редагування чи видалення товарів, створення, видалення або редагування користувачів. Які ж веб-сторінки будуть міститися у моїй адмін-частині? Оскільки адміністраторів може бути декілька, то відповідно у кожного буде свій акаунт, тому було прийнято рішення створення простої форми для логіну адміністратора. Далі, як і в клієнтській частині, тут буде існувати домашня сторінка, яка буде поєднувати в собі всі наявні компоненти між собою, а також показувати певні статистики, які будуть розроблені пізніше. Коли адміністратор буде залогінений і на домашній сторінці, він зможе використовувати вищеописаний функціонал. Тому, для додавання нового товару було прийнято рішення створити окрему сторінку на якій буде можливість це зробити. Також окрему сторінку буде створено і для

редагування наявних товарів. Щоб побачити всі наявні товари також буде окрема сторінка, на ній буде список всіх товарів і ряд функцій, які можна з ними зробити, а саме – редагувати та видалити.

Для створення нового користувача та редагування вже існуючого буде також створено окремо дві подібні сторінки. А також слід не забувати про видалення цих користувачів, але це вже буде виконуватися за логікою, подібною до логіки видалення існуючих товарів, тобто буде створена веб-сторінка, яка буде відображати всіх користувачів, а також ряд функцій, які можна виконати з ним, а саме – редагувати або ж видалити. В сумі ми повинні створити 8 веб-сторінок.

2.2 Створення та верстка сторінок сайту

Верстка сайту — це в принципі є один з найперших та найважливіших етапів розробки практично будь-якого веб-сайту чи веб-додатку. На цьому етапі ми будемо розробляти дизайн всіх сторінок, які будуть в нашому дипломному проєкті. При верстці не буде створено жодної функціональності сайту, а лише його оболонка, тобто зовнішній вигляд [2].

Зазвичай верстку сайту роблять за допомогою HTML та CSS, але точно не в нашому випадку. Ці методи верстки звичайно хороші і достатньо стандартні, але все ж таки вони не є досконалими або так би мовити «з великими можливостями». Цілком природньо і логічно, що завжди хочеться покращувати свої навички, покращувати свою роботу, а також не відставати від сучасності, що означає впроваджувати нові, сучасні технології у свої проєкти. Тому при верстці мого інтернет-магазину автозапчастин «Auto Tech» було використано компоненту верстку React, тобто верстку з компонентним підходом, використовуючи бібліотеку Styled-components встановлену, звичайно за допомогою пакетного менеджера npm. Styled Components - це суміш інновацій зі

звичністю. Вона є однією з таких бібліотек, яка досить швидко привернула до себе увагу мас.

Оскільки, було вирішено використовувати React при створенні свого сайту, то однозначно потрібно користуватися таким хорошим заміником та використовувати CSS уже в контексті мови Javascript та бібліотеки React .

Із переваг такого методу верстки, хочеться відзначити, що така верстка завжди буде коректно відображатися на різних пристроях та з різним розширенням при вертикальному або горизонтальному перегляді.

Отож, розпочнемо з верстки знову ж таки саме клієнтської частини. Перша сторінка, яку нам потрібно зверстати – домашня сторінка. Їй не потрібно задавати будь-яких стилів за допомогою styled-components, а лише в правильному порядку розташувати відображення наших компонентів як було задумано раніше. Щоб детальніше ознайомитися з кодом домашньої сторінки, можна подивитися на рисунок 2.3, який знаходиться нижче.

```

1  import React from "react";
2  import Announcement from "../components/Announcement";
3  import Categories from "../components/Categories";
4  import Footer from "../components/Footer";
5  import Navbar from "../components/Navbar";
6  import Newsletter from "../components/Newsletter";
7  import Products from "../components/Products";
8  import Slider from "../components/Slider";
9
10 const Home = () => {
11   return (
12     <div>
13       <Announcement/>
14       <Navbar/>
15       <Slider/>
16       <Categories/>
17       <Products/>
18       <Newsletter/>
19       <Footer/>
20     </div>
21   );
22 };
23
24 export default Home;

```

Рисунок 2.3 – Код домашньої сторінки

Наступну будемо верстати сторінку реєстрації користувача. Оскільки це вже буде окрема та повноцінна сторінка, то її необхідно оформляти вже з використанням вищеописаних styled-components. Тому для створення сторінки

реєстрації користувача було розроблено не дуже великий ряд наступних компонентів: «Title», «Form», «Input», «Agreement» та «Button». Такий тип написання компонентів буде використовуватися і для подальшої розробки всіх необхідних для нашого дипломного проекту веб-сторінок. Наступним етапом їм було задано відповідні стилі, які дозволяють створити компоненти відповідного дизайну, прийнятого для даного проекту. Після цього ці компоненти було використано і розміщено в правильному та послідовному порядку для коректного відображення нашої форми реєстрації. Для ознайомлення з невеликою частинкою верстки цієї форми реєстрації, можна звернути увагу на рисунок 2.4, а для розуміння, як вона буде виглядати на сайті – рисунок 2.5.

```

17 const Container = styled.div`
18   width: 100vw;
19   height: 100vh;
20   background: linear-gradient(rgba(255, 255, 255, 0.5),
21     rgba(255, 255, 255, 0.5)),
22   url("https://cdn.motor1.com/images/mgl/lQzjw/s1/4x3/bmw-m4-competition-kith-design-study-edition-exterior.webp") center;
23   background-size: cover;
24   display: flex;
25   align-items: center;
26   justify-content: center;
27 `;
28
29 const Wrapper = styled.div`
30   width: 40%;
31   padding: 20px;
32   background-color: white;
33   ${mobile({ props: {width: "75%"}})}
34 `;

```

Рисунок 2.4 – Фрагмент верстки форми реєстрації

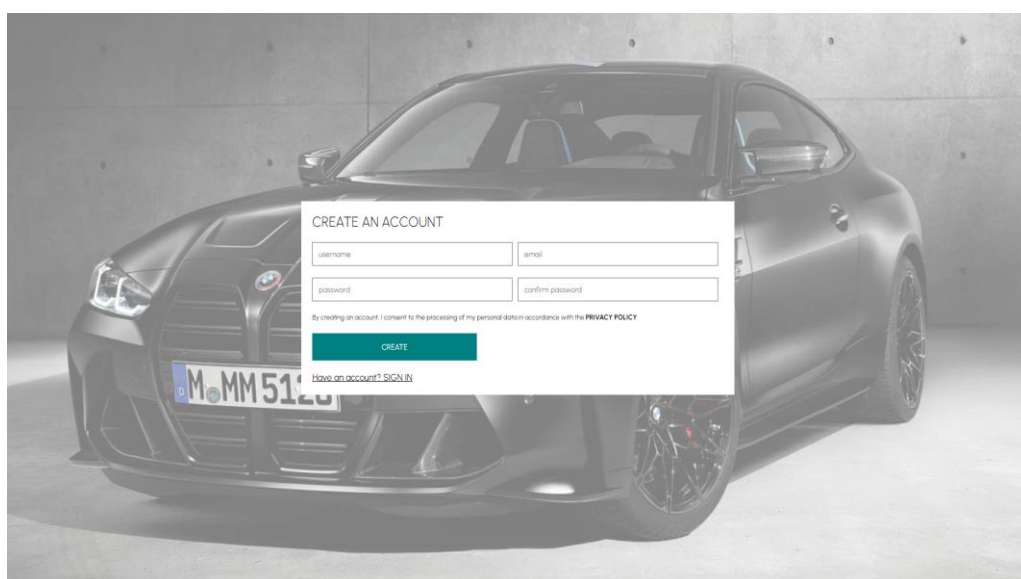


Рисунок 2.5 – Вигляд форми реєстрації

Логічно, що раз у нас уже є сторінка реєстрації, то наступним кроком буде верстання сторінки логіну. Там метод верстання буде аналогічний до верстання форми реєстрації, тобто будуть створені невеликі компоненти, їм буде задано стилі і тоді їх буде використано та розміщено в правильному порядку для коректного відображення. Малий фрагмент коду можна переглянути на рисунку 2.6, а побачити вигляд форми логіну можна на рисунку 2.7.

```

45  const Button = styled.button`
46    width: 40%;
47    border: none;
48    padding: 15px 20px;
49    background-color: teal;
50    color: white;
51    cursor: pointer;
52    margin-bottom: 10px;
53    &:disabled {
54      color: green;
55      cursor: not-allowed;
56    }
57  `;
58
59
60  const Error = styled.span`
61    color: red;
62  `;
63
64  const Agreement = styled.span`
65    font-size: 12px;
66    margin: 20px 0px;
67  `;

```

Рисунок 2.6 – Фрагмент коду форми логіну

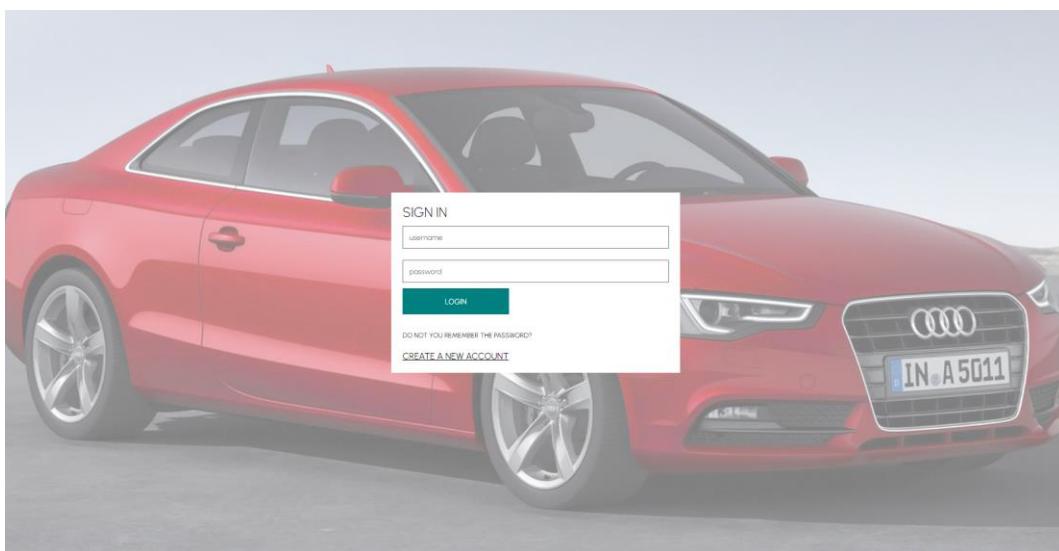


Рисунок 2.7 – Вигляд форми логіну

Наступним кроком, ми будемо верстати сторінку, для відображення списку наших товарів. І знову ж таки, ми використовуємо наші styled-components для цієї реалізації. На цій сторінці будуть відображатися всі наявні товари у нашому магазині автозапчастин. Готові компоненти зі стилями було використано та розміщено в правильному порядку для коретного відображення. Для мінімального ознайомлення з кодом цієї сторінки, ми маємо можливість подивитися на рисунок 2.8, а вже зрозуміти як це виглядає на сайті нам допоможе рисунок 2.9.

```

11  const Container = styled.div`
12
13  const Title = styled.h1`
14    margin: 20px;
15  `;
16
17  const FilterContainer = styled.div`
18    display: flex;
19    justify-content: space-between;
20  `;
21
22  const Filter = styled.div`
23    margin: 20px;
24    ${mobile(props: {width: "0px 20px", display: "flex", flexDirection: "column"})}
25  `;
26
27  const FilterText = styled.span`
28    font-size: 20px;
29    font-weight: 600;
30    margin-right: 20px;
31    ${mobile(props: {marginRight: "0px"})}
32  `;
33
34  const Select = styled.select`
35    padding: 10px;
36    margin-right: 20px;
37    ${mobile(props: {margin: "10px 0px"})}
38  `;
39  const Option = styled.option`

```

Рисунок 2.8 – Фрагмент коду сторінки відображення товарів



Рисунок 2.9 – Вигляд сторінки відображення товарів

Оскільки у нас є список всіх наявних товарів, то слід зверстати сторінку для одного товару, це буде динамічна сторінка, яка буде представляти будь-який обраний товар. На етапі верстки ми тільки можемо подивитися на сам вигляд даної сторінки. Розроблялася вона згідно з тією ж логікою, що і список всіх товарів, тобто створення компонентів, задання їм стилю та правильного, послідовного розміщення. Для детальнішого розуміння, ми можемо ознайомитися з малесеньким фрагментом коду, що зображено на рисунку 2.10, а для того, щоб побачити це, ми можемо звернути увагу на рисунок 2.11.

```

23  const ImgContainer = styled.div`
24    flex: 1;
25  `;
26
27  const Image = styled.img`
28    width: 100%;
29    height: 70vh;
30    object-fit: cover;
31    ${mobile( props: {height: "40vh"})};
32  `;
33
34  const InfoContainer = styled.div`
35    flex: 1;
36    padding: 0px 50px;
37    ${mobile( props: {padding: "10px"})};
38  `;
39
40  const Title = styled.h1`
41    font-weight: 200;
42  `;
43
44  const Desc = styled.p`
45    margin: 20px 0px;
46  `;
47
48
49  const Price = styled.span`
50    font-weight: 100;
51    font-size: 40px;
52  `;

```

Рисунок 2.10 – Фрагмент коду сторінки товару

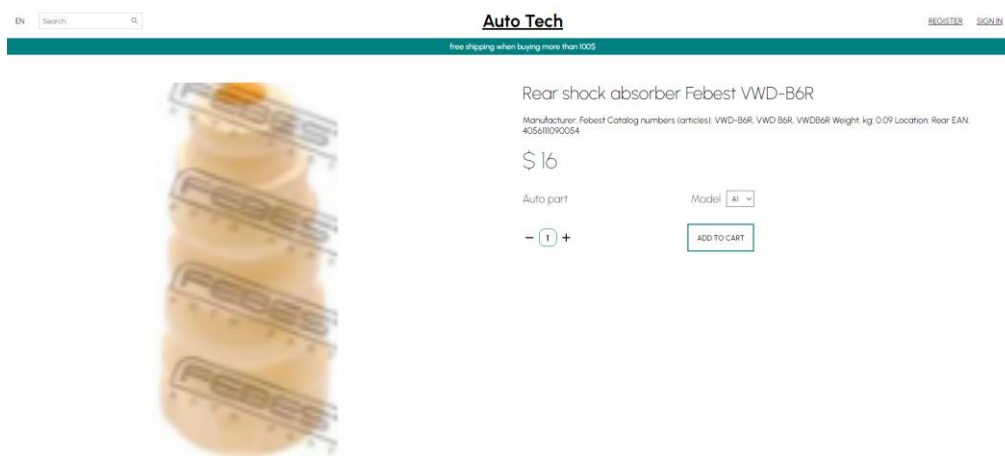


Рисунок 2.11 – Вигляд сторінки відображення товару

Залишилося зверстати останню сторінку з нашої клієнтської частини, а саме корзину користувача. Вона також буде верстатися за тією ж технологією, створимо деякі компоненти, задамо їм необхідні стилі і розташуємо в правильному порядку. Для ознайомлення з фрагментом коду, дивимось на рисунок 2.12, а для розуміння вигляду на сайті дивимось на рисунок 2.13.

```
const KEY = process.env.REACT_APP_STRIPE;

const Container = styled.div``;

const Wrapper = styled.div`
padding: 20px;
${mobile(props) {padding: "10px"}}
`;

const Title = styled.h1`
font-weight: 300;
text-align: center;
`;

const Top = styled.div`
display: flex;
align-items: center;
justify-content: space-between;
padding: 20px;
`;

const TopButton = styled.button`
padding: 10px;
font-weight: 600;
cursor: pointer;
border: ${(props) => props.type === "filled" && "none"};
background-color: ${(props) =>
  props.type === "filled" ? "black" : "transparent"};
color: ${(props) => props.type === "filled" && "white"};
`;
```

Рисунок 2.12 – Фрагмент коду сторінки корзины

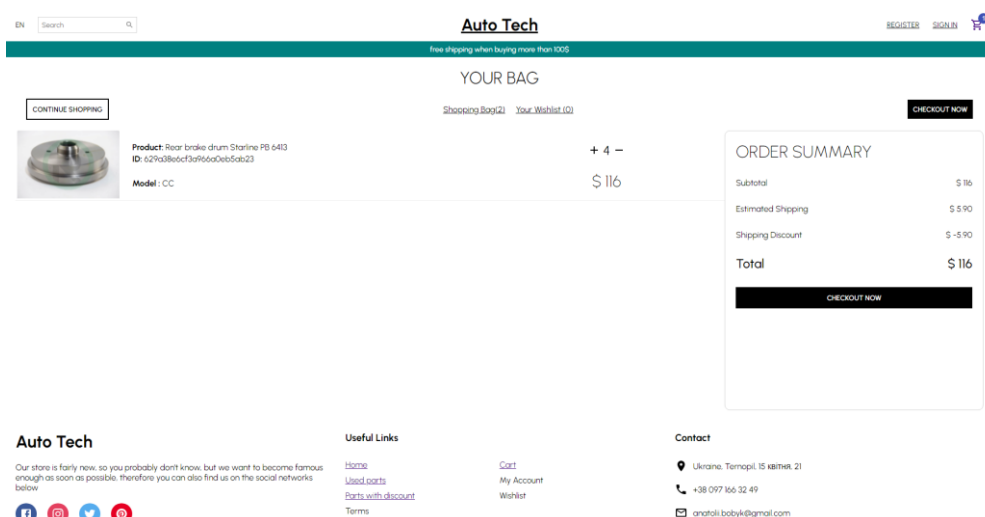


Рисунок 2.13 – Вигляд сторінки корзины на сайті

Верстання клієнтської частини закінчено, а отже, це означає, що ми

можемо сміливо переходити до верстки адміністративної частини. Розпочнемо цю верстку також зі створення домашньої сторінки, яка просто буде поєднувати в собі всі інші сторінки та компоненти, які ми створимо. Для ознайомлення з кодом домашньої сторінки, ми можемо подивитися на рисунок 2.14, а вигляд цієї сторінки можна побачити на рисунку 2.15.

```
return (
  <div className="home">
    <FeaturedInfo />
    <Chart
      data={userStats}
      title="User Analytics"
      grid
      dataKey="Active User"
    />
    <div className="homeWidgets">
      <WidgetSm />
      <WidgetLg />
    </div>
  </div>
);
```

Рисунок 2.14 – Фрагмент коду домашньої сторінки

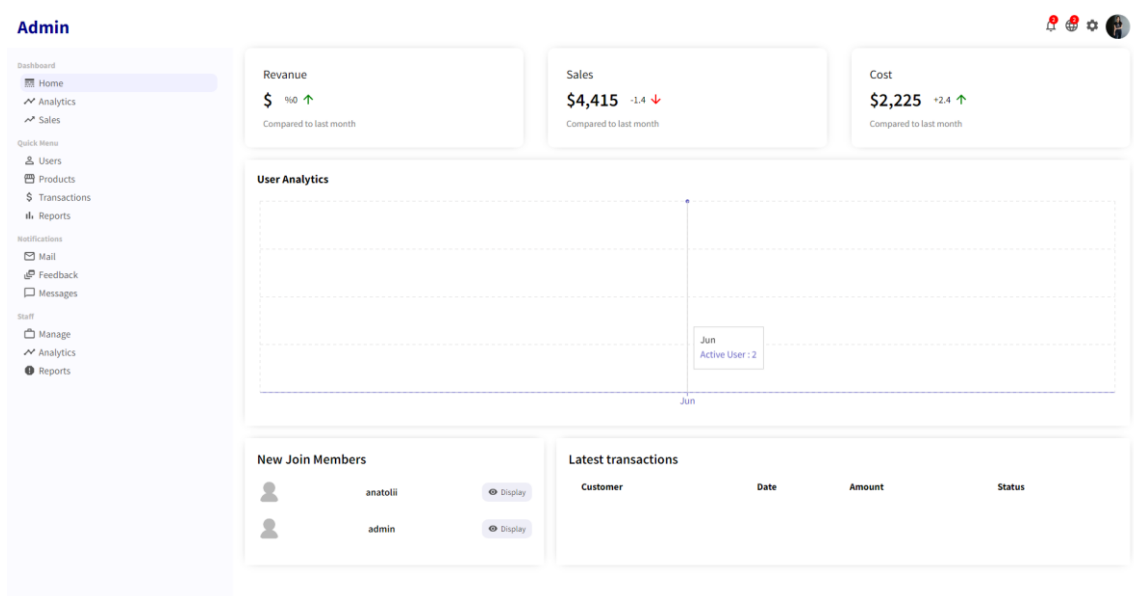


Рисунок 2.15 – Вигляд домашньої сторінки

Далі ми зверстаємо максимально примітивну форму логіну в нашу адмін-панель, оскільки адміністраторів може бути кілька і у кожного буде свої дані для входу в систему. Тут нічого не було ускладнено, оскільки ця формочка потрібна

лише декільком людям-адміністраторам. Фрагмент коду цієї простої формочки ми можемо побачити на рисунку 2.16, а за бажанням її візуальне зображення – зображено на рисунку 2.17.

```
return (
  <div
    style={{
      height: "100vh",
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
      justifyContent: "center",
    }}
  >
    <input
      style={{padding: 10, marginBottom: 20}}
      type="text"
      placeholder="username"
      onChange={(e : ChangeEvent<HTMLInputElement> ) => setUsername(e.target.value)}
    />
    <input
      style={{padding: 10, marginBottom: 20}}
      type="password"
      placeholder="password"
      onChange={(e : ChangeEvent<HTMLInputElement> ) => setPassword(e.target.value)}
    />
    <button onClick={handleClick} style={{padding: 10, width: 100}}>
      Login
    </button>
  </div>
);
```

Рисунок 2.16 – Фрагмент коду форми реєстрації

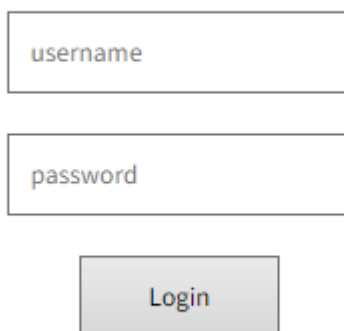


Рисунок 2.17 – Вигляд форма логіну на сайті

Наступним кроком буде верстання сторінки всіх товарів, які наявні в нашому інтернет-магазині. Тут спершу нам потрібно зверстати список, який буде складатися зі всіх наявних товарів, відображатися він буде за допомогою ID товару, його зображення, найменування та ціни. Тоді потрібно додати справа від

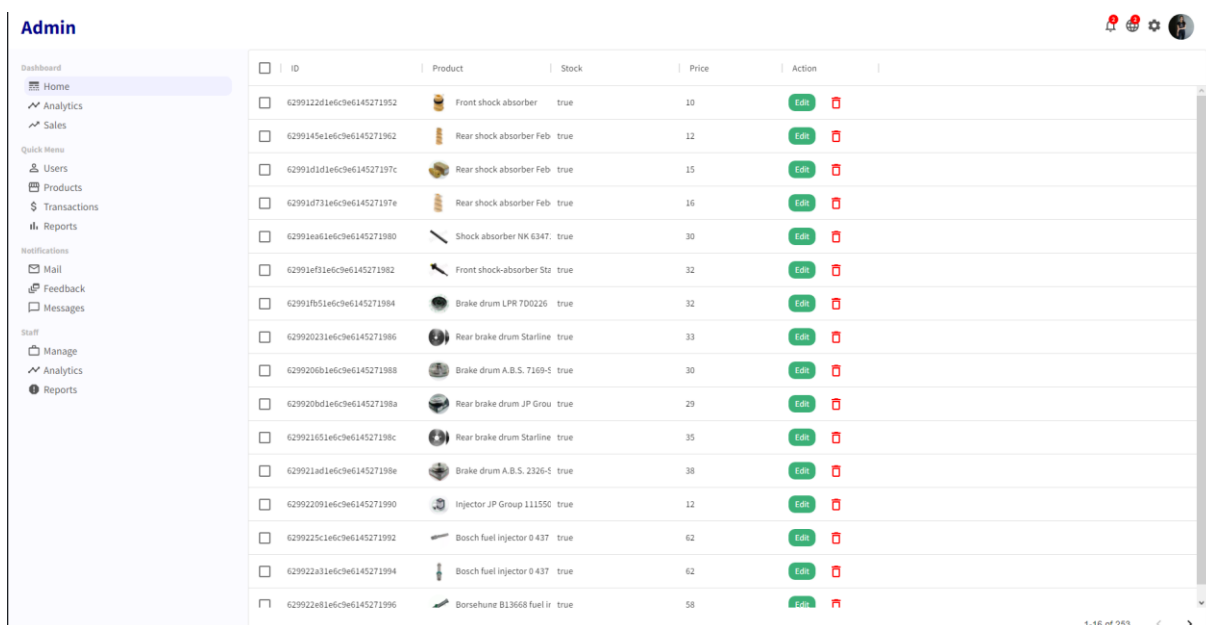
цих товарів дві кнопки – редагування та видалення цього товару. Кнопка редагування буде зеленого кольору, а видалення – червоного. Пізніше вони будуть виконувати свої функції, але поки що це тільки їх дизайн. Щоб ознайомитися з фрагментом коду даної сторінки, можна подивитися на рисунок 2.18, а щоб побачити вигляд цієї сторінки на сайті, то дивимося на рисунок 2.19.

```

1  .productList{
2    flex: 4;
3  }
4
5  .productListItem{
6    display: flex;
7    align-items: center;
8  }
9
10 .productListImg {
11   width: 32px;
12   height: 32px;
13   border-radius: 50%;
14   object-fit: cover;
15   margin-right: 10px;
16 }
17
18 .productListEdit{
19   border: none;
20   border-radius: 10px;
21   padding: 5px 10px;
22   background-color: #3bb077;
23   color: white;
24   cursor: pointer;
25   margin-right: 20px;
26 }
27
28 .productListDelete{
29   color: red;
30   cursor: pointer;
31 }

```

Рисунок 2.18 – Фрагмент коду сторінки відображення товарів



ID	Product	Stock	Price	Action
6299122d1e6c9e6145271952	Front shock absorber	true	10	Edit Delete
6299145e1e6c9e6145271962	Rear shock absorber Feb	true	12	Edit Delete
62991d1d1e6c9e614527197c	Rear shock absorber Feb	true	15	Edit Delete
62991d731e6c9e614527197e	Rear shock absorber Feb	true	16	Edit Delete
62991ea61e6c9e6145271980	Shock absorber NK 6347	true	30	Edit Delete
62991ef31e6c9e6145271982	Front shock-absorber Sta	true	32	Edit Delete
62991fb51e6c9e6145271984	Brake drum LPR T00226	true	32	Edit Delete
629920231e6c9e6145271986	Rear brake drum Starline	true	33	Edit Delete
6299206b1e6c9e6145271988	Brake drum A.B.S. 7169-5	true	30	Edit Delete
629920bd1e6c9e614527198a	Rear brake drum JP Grou	true	29	Edit Delete
629921651e6c9e614527198c	Rear brake drum Starline	true	35	Edit Delete
629921ad1e6c9e614527198e	Brake drum A.B.S. 2326-5	true	38	Edit Delete
629922091e6c9e6145271990	Injector JP Group 111550	true	12	Edit Delete
6299225c1e6c9e6145271992	Bosch fuel injector 0 437	true	62	Edit Delete
629922a31e6c9e6145271994	Bosch fuel injector 0 437	true	62	Edit Delete
629922e81e6c9e6145271996	Borsehune B13668 fuel in	true	58	Edit Delete

Рисунок 2.19 – Вигляд сторінки відображення товарів на сайті

Тепер нам потрібно створити сторінку редагування товару. Для цього нам потрібно зверстати невеличку форму, в яку буде заповнюватися все про даний товар та кнопку, яка буде означати оновлення товару. Поглянути на малий фрагмент коду цієї сторінки можна на рисунку 2.20, а для розуміння вигляду цієї сторінки на сайті – рисунок 2.21.

```

1  .product {
2    flex: 4;
3    padding: 20px;
4  }
5
6  .productTitleContainer {
7    display: flex;
8    align-items: center;
9    justify-content: space-between;
10 }
11
12 .productAddButton {
13   width: 80px;
14   border: none;
15   padding: 5px;
16   background-color: teal;
17   color: white;
18   border-radius: 5px;
19   font-size: 16px;
20   cursor: pointer;
21 }

```

Рисунок 2.20 – Фрагмент коду форми редагування

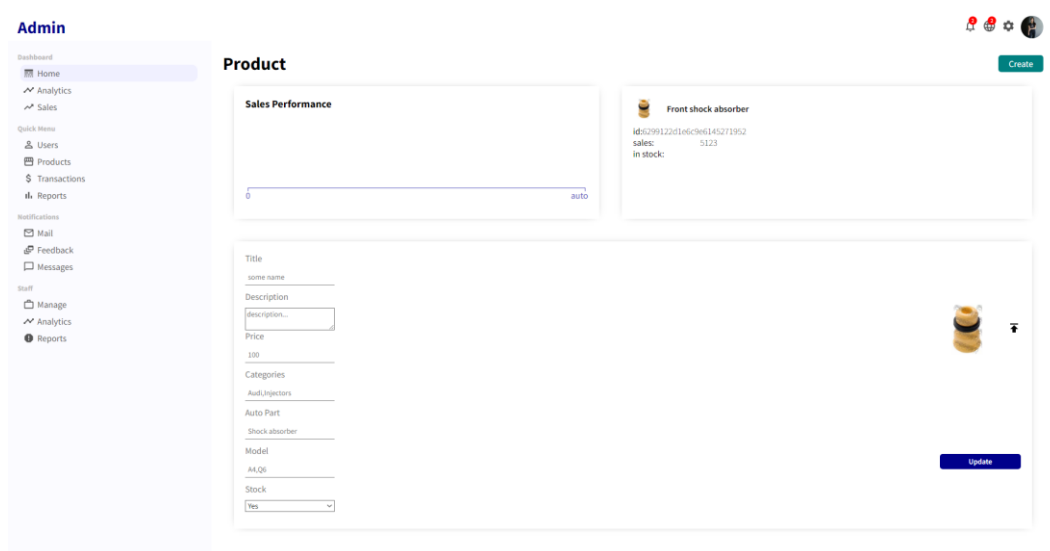


Рисунок 2.21 – Вигляд сторінки редагування товару на сайті

Далі нам потрібно зверстати сторінку створення товару. Собою вона буде

представляти знову ж таки форму з аналогічними полями як у формі редагування товару, що цілком логічно, і вона буде відрізнятися лише тим, що в неї інакша кнопка та її виконувана функція. Для ознайомлення з кодом цієї сторінки, дивимося на рисунок 2.22, а для можливості побачити її вигляд на сайті нам слугує рисунок 2.23.

```
.newProduct {
  flex: 4;
}

.addProductForm {
  margin-top: 10px;
}

.addProductItem {
  width: 250px;
  display: flex;
  flex-direction: column;
  margin-bottom: 10px;
}

.addProductItem > label {
  color: gray;
  font-weight: 600;
  margin-bottom: 10px;
}

.addProductItem > input {
  padding: 10px;
}
```

Рисунок 2.22 – Фрагмент коду форми створення товару

The screenshot shows an 'Admin' dashboard with a sidebar menu on the left containing sections like Dashboard, Quick Menu, Notifications, and Staff. The main content area is titled 'New Product' and contains the following form fields:

- Image:** A button labeled 'Вибрати файл' (Choose file) with the text 'Файл не вибрано' (File not selected).
- Title:** A text input field containing 'some name'.
- Description:** A text input field containing 'description...'.
- Price:** A text input field containing '100'.
- Categories:** A text input field containing 'AutoInjectors'.
- Auto Part:** A text input field containing 'Shock absorber'.
- Model:** A text input field containing 'AA-Q8'.
- Stock:** A dropdown menu with 'Yes' selected.
- Create:** A blue button at the bottom of the form.

Рисунок 2.23 – Вигляд сторінки створення товару на сайті

Тепер ми будемо розробляти сторінку, подібну до сторінки зі всіма товарами, тільки тепер це буде сторінка зі всіма користувачами. Написання в неї буде дуже подібне, просто тепер вона буде відображати всіх наявних користувачів, а не товари. Для того, щоб мати можливість побачити фрагмент коду цієї сторінки достатньо подивитися на рисунок 2.24, а для уявлення вигляду цієї сторінки на сайті – рисунок 2.25.

```

.userList {
  flex: 4;
}

.userListUser {
  display: flex;
  align-items: center;
}

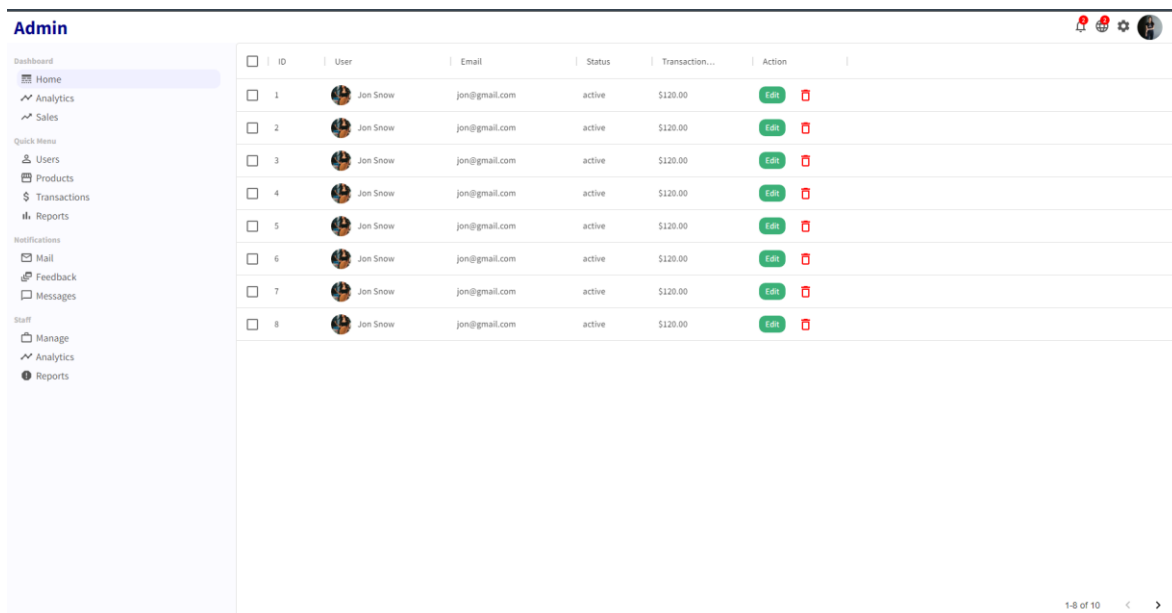
.userListImg {
  width: 32px;
  height: 32px;
  border-radius: 50%;
  object-fit: cover;
  margin-right: 10px;
}

.userListEdit {
  border: none;
  border-radius: 10px;
  padding: 5px 10px;
  background-color: #3bb077;
  color: white;
  cursor: pointer;
  margin-right: 20px;
}

.userListDelete {
  color: red;
  cursor: pointer;
}

```

Рисунок 2.24 – Фрагмент коду сторінки відображення користувачів



ID	User	Email	Status	Transaction...	Action
1	Jon Snow	jon@gmail.com	active	\$120.00	Edit Delete
2	Jon Snow	jon@gmail.com	active	\$120.00	Edit Delete
3	Jon Snow	jon@gmail.com	active	\$120.00	Edit Delete
4	Jon Snow	jon@gmail.com	active	\$120.00	Edit Delete
5	Jon Snow	jon@gmail.com	active	\$120.00	Edit Delete
6	Jon Snow	jon@gmail.com	active	\$120.00	Edit Delete
7	Jon Snow	jon@gmail.com	active	\$120.00	Edit Delete
8	Jon Snow	jon@gmail.com	active	\$120.00	Edit Delete

Рисунок 2.25 – Вигляд сторінки відображення користувачів на сайті

Знову ж таки аналогічно до сторінки редагування товарів, ми будемо створювати подібну сторінку редагування користувачів. Тут немає нічого надзвичайного, просто взяли логіку роботи зі сторінки редагування товарів, замінили назви полів, картинки та задали трішки інші параметри. Фрагмент коду створення дизайну цієї форми ми можемо побачити на рисунку 2.26, а вигляд даної сторінки на рисунку 2.27.

```
.user {
  flex: 4;
  padding: 20px;
}

.userTitleContainer {
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.userAddButton {
  width: 80px;
  border: none;
  padding: 5px;
  background-color: teal;
  border-radius: 5px;
  cursor: pointer;
  color: white;
  font-size: 16px;
}

.userContainer {
  display: flex;
  margin-top: 20px;
}
```

Рисунок 2.26 – Фрагмент коду форми редагування користувачів

The screenshot shows an 'Admin' dashboard with a sidebar menu. The main section is titled 'Edit User' and displays the profile of 'Anna Becker', a Software Engineer. The profile includes fields for Username (annabeck99), Full Name (Anna Becker), Email (annabeck99@gmail.com), Phone (+1 123 456 67), and Address (New York | USA). There are 'Create' and 'Update' buttons.

Рисунок 2.27 – Вигляд сторінки редагування користувачів на сайті

І в решті-решт потрібно зверстати сторінку створення нового користувача, і знов ж таки за аналогом зі сторінки створення товарів. Міняємо поля форми на такі як у формі редагування користувачів, вносимо трішки змін і отримуємо бажаний результат. Невеликий фрагмент коду сторінки створення користувачів ми будемо бачити на рисунку 2.28, а вигляд цієї сторінки на сайті – рисунок 2.29.

```
.newUser {
  flex: 4;
}

.newUserForm {
  display: flex;
  flex-wrap: wrap;
}

.newUserItem {
  width: 400px;
  display: flex;
  flex-direction: column;
  margin-top: 10px;
  margin-right: 20px;
}

.newUserItem > label {
  margin-bottom: 10px;
  font-size: 14px;
  font-weight: 600;
  color: rgb(151, 150, 150);
}
```

Рисунок 2.28 – Фрагмент коду форми створення користувачів

The screenshot shows an 'Admin' dashboard with a sidebar menu on the left. The main content area is titled 'New User'. The form contains the following fields and controls:

- Username:** Text input with the value 'john'.
- Full Name:** Text input with the value 'John Smith'.
- Email:** Text input with the value 'john@gmail.com'.
- Password:** Text input with the value 'password'.
- Phone:** Text input with the value '+1123 456 78'.
- Address:** Text input with the value 'New York | USA'.
- Gender:** Radio buttons for 'Male', 'Female', and 'Other'.
- Active:** A dropdown menu with the value 'Yes'.
- Create:** A blue button to submit the form.

The sidebar menu includes sections for Dashboard, Quick Menu, Notifications, and Staff, with various sub-items like Home, Analytics, Sales, Users, Products, Transactions, Reports, Mail, Feedback, Messages, Manage, and Reports.

Рисунок 2.29 – Вигляд сторінки створення користувача на сайті

Ось ми і завершили верстання наших сторінок. Загалом такий дизайн мені до вподоби, насправді його створення забрало дуже багато часу і ми отримали такий результат. Я сподіваюся, що він виявиться достатньо привабливим кожному, хто буде на нього дивитися, а краще – користуватися. Звичайно, у ньому ще багато чого можна доробити, але то вже на майбутнє.

2.3 Розробка структури бази даних сайту

Давайте розглянемо, що розуміємо під базою даних. База даних (скорочено БД) – це в нашому розумінні сукупність всіх даних, які організовані між собою відповідно до концепції, яка буде описувати характеристику цих даних і взаємозв'язки між їх елементами [3,4].

Пригадаємо основну інформацію про бази даних, розглянувши рисунок 2.30, де зображено прекрасні моделі баз даних та що вони означають, тобто кількість зв'язків між об'єктами, кількість об'єктів, рівень об'єктів, головні та підлеглі об'єкти, у якому вигляді і яку інформацію вони містять.



Рисунок 2.30 – Моделі баз даних

Тож якщо під базою даних ми розуміємо сховище даних, то під СКБД – засоби для роботи з самим цим сховищем даних. Коротка основна інформацію про СКБД наведена на рисунку 2.31.

Далеко не всі розуміють важливість роботи бази даних на будь-якому веб-сайті, але насправді саме база даних є однією з багатьох найголовніших його складових. Вона завжди використовується при розробці динамічних веб-сайтів чи інтернет-магазинів для зберігання різнопланової інформації. В свою чергу ця вся інформація буде динамічно відображатися на сайті, чого ніколи не зробить статика. Якщо взяти трішки простіше, то бази даних являють собою якийсь набір взаємозалежних одна від одної таблиць. Розміри таблиць у базах бувають різними, а їх кількість абсолютно довільною, це буде залежати тільки від вас.



Рисунок 2.31 – структура СКБД

Для інтернет-магазину автозапчастин «Auto Tech» було використано базу даних MongoDB. MongoDB являє собою No-SQL базу даних, яка зберігає в собі дані у вигляді колекцій в форматі JSON. Як мені відомо, колекції можуть бути індексовані, а саме це покращує швидкість вибірки та сортування даних. Також це дозволяє швидко та легко вносити необхідні мені зміни в об'єкти. Хочеться сказати, що перевагою цієї бази даних для мого магазину є той момент, що на відмінну від SQL баз даних в ній можна зберігати об'єкти, масиви, а саме за

допомогою цього я зміг скорити необхідну кількість таблиць для мого інтернет-магазину. На рисунку 2.32 ми можемо бачити невеличке порівняння SQL баз даних з MongoDB.

При проектуванні початково було прийнято рішення використовувати SQL базу даних, тому що це давно знайомі бази даних. Але виявляється, що для кожного проекту підхід буде індивідуальний. В результаті використання SQL БД виявилось, що для мого проекту потрібно створити вісім різних таблиць. Коли ж було виконано аналіз як якось скорити цю кількість, то не знайшов іншого варіанту як використати об'єкти та масиви, але як ми знаємо в SQL такого немає і тут на допомогу прийшла та сама БД під назвою MongoDB.

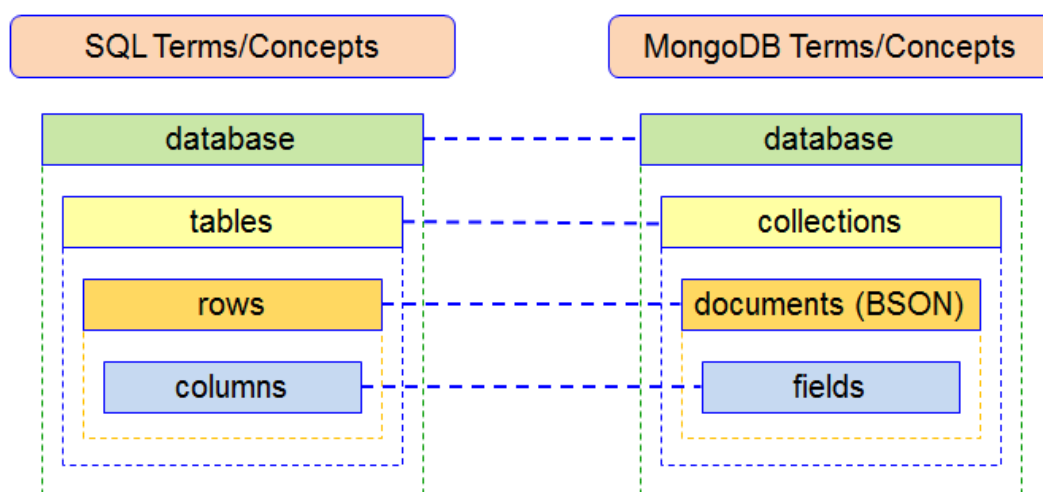


Рисунок 2.32 – Порівняння SQL баз даних з MongoDB

У ній я зміг замінити таблиці на масиви об'єктів і додати ці масиви в поля інших таблиць, зрозуміло, щоб воно відповідало очікуванню. В результаті замість створення окремих таблиць для типів автозапчастини, моделей автомобілів, опису товарів і т.д, було створено додаткові поля в самій моделі товару і вже в ті поля поміщено ці масиви. В результаті таких, можливо, трішки хитрих дій кількість необхідних для роботи мого інтернет-магазину таблиць зменшилась до чотирьох, що є однозначно «плюсом», тому як чим менше таблиць – тим менше часу потрібно на перебирання їх елементів.

Отже, ми розібралися яку базу даних було обрано для магазину і чому прийнято саме таке рішення. А це означає, що настав час зобразити базу даних у вигляді ER-діаграми.

Взявши інтернет-магазин автозапчастин «Auto Tech», можна сказати, що його база даних після значних скорочень і переході з SQL на MongoDB складається, як зазначено раніше тільки з чотирьох таблиць, а саме: таблиця зареєстрованих користувачів, таблиця корзини зареєстрованого користувача, замовлення, яке зробив зареєстрований користувач і яке потрапило у корзину, а також чи не найголовніше – самі товари, які будуть продаватися у нашому магазині. Оскільки в нас так мало таблиць, то розробка діаграми багато часу не буде займати. Найважче, що там потрібно було продумати – це розташування відповідності зв'язків для правильного функціонування магазину. Щоб побачити цю ER-діаграму до нашого інтернет-магазину автозапчастин «Auto Tech», нам всього лиш потрібно подивитися на рисунок 2.33, який знаходиться трішки нижче.

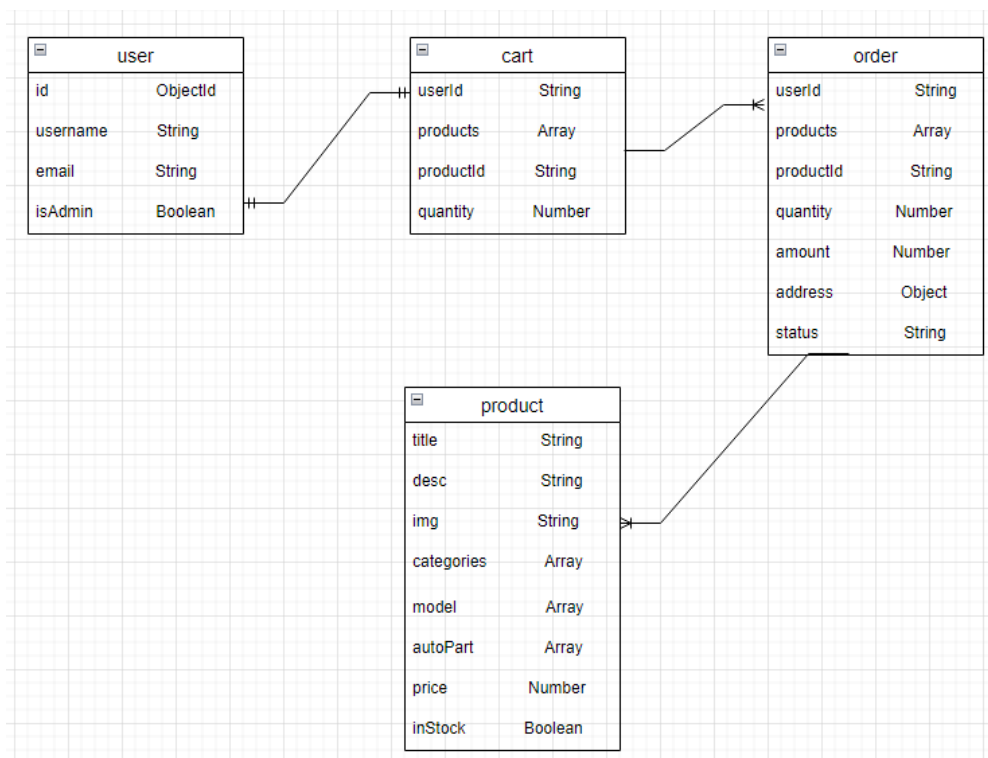


Рисунок 2.33 – ER-діаграма бази даних

Отже, відповідно до нашої діаграми бази даних, ми прямо можемо бачити, що відповідно до неї, користувач може мати лише одну власну корзину, що цілком логічно і відповідає будь-якому інтернет-магазину, для цього було використано зв'язок один до одного. В корзині користувача може знаходитися одне або більше замовлень, також вони можуть бути зроблені в різний час; корзина та замовлення з'єднані зв'язком один до багатьох відповідно, а в самому замовленні може знаходитися один товар або багато різних товарів, що логічно, і це також підтверджується зв'язком один до багатьох. Покращення такого варіанту бази даних також обов'язково буде розглядатися в майбутньому.

2.4 Програмування сайту

2.4.1 Розробка структури сайту і веб-сторінок

У клієнтської частини архітектури розробляється максимально зручна і доступна робота потенційного клієнта на сторінках інтернет-магазину. Розробка привабливого інтерфейсу, доступні і зрозумілі діалогові вікна, зручні системи вибору категорії, вибору товару та його властивостей, а також зручні системи оплати і доставки товарів [6,7,9].

Проаналізувавши роботу багатьох вже працюючих інтернет-магазинів, був зроблений висновок про те, що обов'язково буде реалізовано в проекті.

1. Вітрина магазину буде оформлена так, щоб покупець без проблем міг знаходити необхідний для нього товар і мати можливість отримати про нього потрібну інформацію (опис у вигляді тексту плюс фотографія).

2. Товари будуть розділені по групах, тобто буде створено категорії до яких буде належати певна кількість товарів, що забезпечить можливість пошуку необхідних товарів якомога швидше. Для кожного товару буде передбачено опис і звичайно фотографія.

3. Для наочності будуть додані спеціальні розділи, що містять товари, що згруповано за маркетинговими ознаками. Припустимо:

- «Все, що вам необхідно» (всі товари, які є в магазині);
- «Спеціальні пропозиції» (товари, на які з яких-небудь причин знижені ціни);
- «Використовувані товари» (товари, які вже не є новими та використовувалися раніше іншими автомобілями);

4. Про той чи інший товар зареєстрований користувач зможе залишити відгук.

5. При оплаті замовлення покупець вносить контактну інформацію: email, ім'я, адресу доставки, а потім звичайно і картку та пароль до неї. Після оформлення покупцеві буде відправлятися по електронній пошті лист з даними резервування.

Наведемо приклад коду, в якому можна побачити використання цих технологій. В цьому прикладі показано, як розроблялось відображення одного конкретного товару, який буде вибраний користувачем на сайті. Для цього дивимося на рисунок 2.34.

```

125 const Product = () => {
126   const location = useLocation();
127   const id = location.pathname.split( separator: "/" )[2];
128   const [product, setProduct] = useState( initialState: {} );
129   const [quantity, setQuantity] = useState( initialState: 1 );
130   const [autoPart, setAutoPart] = useState( initialState: "" );
131   const [model, setModel] = useState( initialState: "" );
132   const dispatch = useDispatch();
133
134   useEffect( effect: () => {
135     const getProduct = async () => {
136       try {
137         const res = await publicRequest.get( url: "/products/find/" + id );
138         setProduct( res.data );
139       } catch {
140       }
141     };
142     getProduct();
143   }, [ id ] );
144
145   const handleQuantity = ( type ) => {
146     if ( type === "dec" ) {
147       quantity > 1 && setQuantity( value: quantity - 1 );
148     } else {
149       setQuantity( value: quantity + 1 );
150     }
151   };
152
153   const handleClick = () => {
154     dispatch(
155       addProduct( state: { ...product, quantity, autoPart, model } )
156     );
157   }

```

Рисунок 2.34 – Код відображення вибраного користувачем товару

Далі наведемо рисунок 2.35 для розуміння, що все працює коректно.

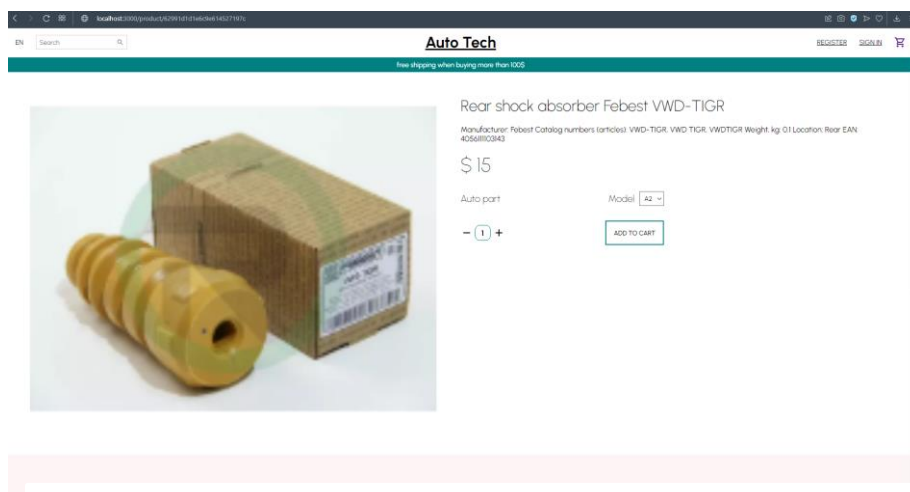


Рисунок 2.35 – Відображення обраного товару

Отже, ми бачимо, що за допомогою цього коду ми дійсно можемо отримати вибраний користувачем товар. Повноцінний текст підпрограми сортування та відображення товарів зображено на листі. Таких прикладів можна навести дуже багато, адже подібні технології використовувалися по всьому веб-сайті, тому показувати всього коду змісту немає і ми не будемо затримуватися над цим питанням, а просто пам'ятаємо, що ми оживили сайт за допомогою React та Redux.

2.4.2 Написання admin- частини

Адміністративна частина також є дуже важливою складовою інтернет-магазину, вона повинна бути розроблена для зручного користування нею власника даного магазину або інших довірених осіб. Саме адміністрування містить інструменти управління інтернет - магазином і включає в себе як загальні установки магазину, так і спеціальні настройки [6,7,9]. В адмініструванні будуть міститися основні налаштування інтернет - магазину:

- перегляд зареєстрованих користувачів;
- видалення або редагування зареєстрованих користувачів;
- додавання товарів;
- редагування товарів;
- видалення товарів;
- перегляд прибутку за останній місяць;
- перегляд кількості продаж за останній місяць;
- перегляд статистики активності користувачів;
- перегляд останніх транзакцій;
- управління оформленими замовленнями, управління зареєстрованими клієнтами;
- додавання, видалення, змін курсів валют;
- статистичні звіти про роботу інтернет-магазину;

Поки що у нас є ось така адміністративна панель, далі планується її покращення, збільшення функцій і можливостей для адміністратора.

Прикладом для адмін.-частини наведемо код товару, написаний з використанням Redux, який дозволяє робити нам CRUD операції з нашими товарами. Фрагмент такого коду наведено на рисунку 2.36.

```
export const productSlice = createSlice({ options: {
  name: "product",
  initialState: {
    products: [],
    isFetching: false,
    error: false,
  },
  reducers: {
    //GET ALL
    getProductStart: (state: Draft<State>) => {
      state.isFetching = true;
      state.error = false;
    },
    getProductSuccess: (state: Draft<State>, action: PayloadAction<any>) => {
      state.isFetching = false;
      state.products = action.payload;
    },
    getProductFailure: (state: Draft<State>) => {
      state.isFetching = false;
      state.error = true;
    },
    //DELETE
    deleteProductStart: (state: Draft<State>) => {
      state.isFetching = true;
      state.error = false;
    },
    deleteProductSuccess: (state: Draft<State>, action: PayloadAction<any>) => {
      state.isFetching = false;
      state.products.splice(
        state.products.findIndex((item: T) => item._id === action.payload),
        deleteCount: 1
      );
    },
  },
});
```

Рисунок 2.36 – Приклад коду з використанням Redux технологій

Знову ж таки все оживлення адмін.-панелі також виконувалося за допомогою того самого підходу, що і клієнтська частина, тому тут не варто затримуватися і наводити багато прикладів коду.

2.5 Тестування веб-сайту

Любий розроблений програмний продукт потрібно перевіряти на справність. При розробці можливе використання юніт тестування. Для готового програмного продукту ми скористаємося можливостями тестування методом чорного ящика. Таке тестування позиціонує себе як швидкий показ потенційних помилок у даному програмному забезпеченні, які необхідно негайно виправити і тільки тоді оприлюднювати свою розробку.

Завжди веб-систему необхідно перевіряти повністю від початку до кінця, до того як вона почне працювати за назначенням для кінцевих користувачів.

Отже, розпочинаємо тестування нашого інтернет-магазину з форми реєстрації та логіну у систему. За нашою задумкою після того як користувач натисне на кнопку реєстрування, він повинен додаватися в базу даних і мати змогу логіну. Для тестування цього переходимо на форму реєстрації сайту і вводимо дані в поля, як ми можемо бачити на рисунку 2.37.

CREATE AN ACCOUNT

denys	denys@gmail.com
****	****

By creating an account, I consent to the processing of my personal data in accordance with the [PRIVACY POLICY](#)

CREATE

[Have an account? SIGN IN](#)

Рисунок 2.37 – Заповнені поля форми реєстрації

Далі ми тиснемо на кнопку «CREATE» і користувач повинен зберігатися в базу, перевіряємо, а результат зображуємо на рисунку 2.38.

```
> {
  _id: ObjectId("62acb90eec2abdfb290c458a")
  username: "denys"
  email: "denys@gmail.com"
  password: "U2FsdGVkX19pEeISjq8CK6ZqbJ9jXvgzDXt6J/X2LNg="
  isAdmin: false
  createdAt: 2022-06-17T17:25:34.181+00:00
  updatedAt: 2022-06-17T17:25:34.181+00:00
  __v: 0
}
```

Рисунок 2.38 – Результат реєстрації користувача

Отже, як видно, користувач успішно додався в нашу базу даних, а це означає, що тестування пройшло звичайно, що успішно і все працює коректно. Далі ми будемо тестувати форму логіну користувача. Для цього ми переходимо на нашу сторінку логіну і вводимо в існуючу на ній форму наші дані, а саме ім'я користувача та його пароль відповідно, як зображено на рисунку 2.39. Якщо все пройде успішно, то нас буде переадресовано на головну сторінку сайту і з неї вже можливості потрапити на форму реєстрації або логіну у нас не буде. Результат тестування логіну буде зображено на рисунку 2.40.

SIGN IN

DO NOT YOU REMEMBER THE PASSWORD?

[CREATE A NEW ACCOUNT](#)

Рисунок 2.39 – Введені дані в форму логіну

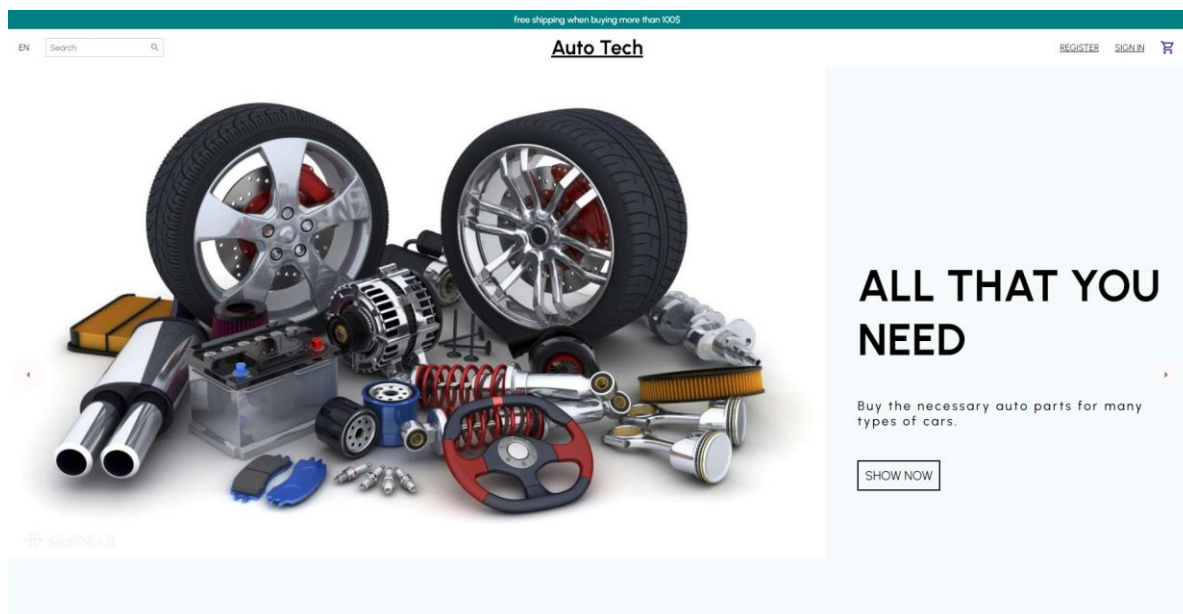


Рисунок 2.40 – Результати логіну

Після натискання на кнопку «LOGIN» нас перемістило на головну сторінку веб-сайту. Тоді було здійснено спробу переміститися на форму реєстрації, а також на форму логіну, але нас все ж таки переадресувало та залишало на головній сторінці, а це означає, що все працює коректно.

Тепер, коли ми потрапили на головну сторінку нашого веб-сайту, необхідно протестувати його слайдову частину, яку можна одразу побачити на головному екрані, при потраплянні на головну сторінку сайту. На кожному слайді знаходиться різні великі та стилізовані зображення, праворуч від зображень є дуже короткий опис, що цей слайд представляє, а також кнопка, яка є абсолютно ідентичною з вигляду на всіх слайдах, і при натисканні на неї, вона повинна відкривати користувачу список всіх товарів зазначеної на слайді категорії. Ці кнопки працюють за абсолютно однаковим принципом, а лише шлях у них прописаний різний, тому тут достатньо протестувати хоча б одну з них, і якщо в результаті такого невеликого тестування вона буде працювати коректно та відкривати список товарів за категорією, яка описана на слайді, то це буде означати, що всі інші кнопки також виконують свою функцію коректно. Результати цього не важкого тестування буде зображено на рисунку 2.41.

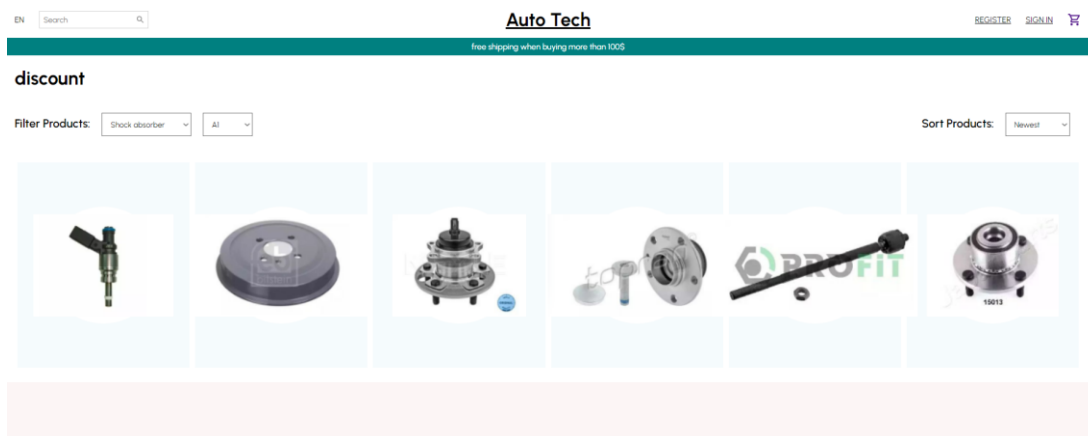


Рисунок 2.41 – Результати тестування слайдової частини

При натисканні на кнопку слайдової частини нам відкрився список товарів, назва цього списку відповідає опису про запчастини на слайді. Висновок з цього тестування буде наступним: все працює правильно, можна переходити далі.

Тепер необхідно протестувати відповідність товарів нашим категоріям і взагалі чи працює поділ товарів на категорії. Для цього ми виберемо одну з існуючих категорій на нашому сайті, натиснемо відповідну кнопку, результатом чого нам повинен відкритися список товарів, які належать саме до неї. Результати тестування будуть зображені на рисунку 2.42.

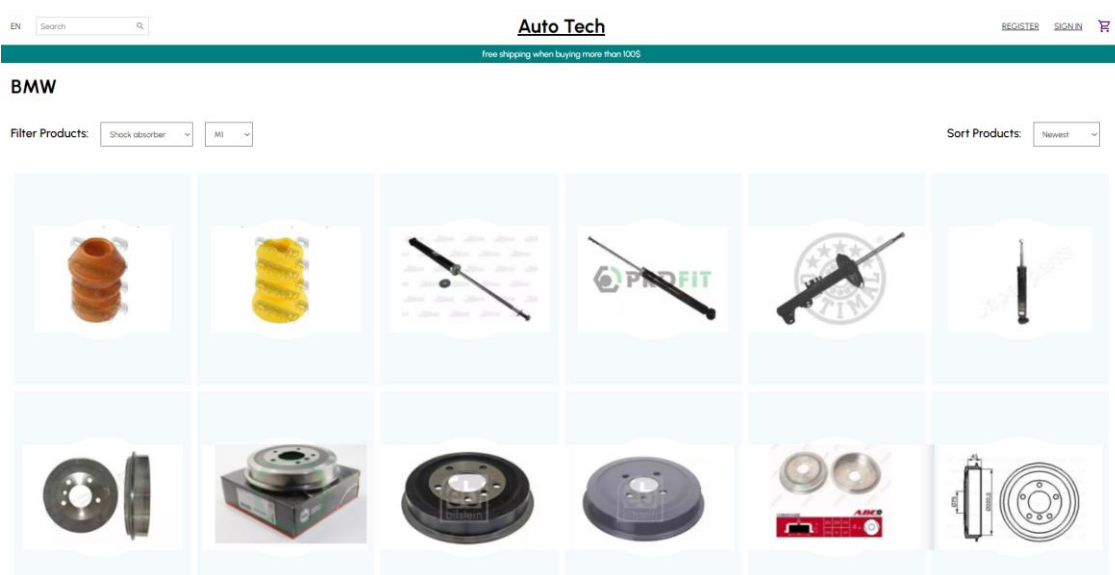


Рисунок 2.42 – Результат тестування поділу товарів по категоріям

Отож, після натискання кнопки на нашій категорії під назвою «BMW», нам відкрило список товарів, які до неї належать, саму ж категорію також вказало перед фільтрацією товарів. Все працює чітко. Перевіряти кожну категорію окремо змісту немає, тому що всі вони працюють по однаковій логіці, і робота однієї означає справне виконання всіх.

Гаразд, маємо ми товари в категорії змішані всі на купу. Ще з самого початку було відзначено як це не зручно і що це займає надто багато часу, бо прийдеться передивлятися всі товари. Як ви мабуть помітили, під назвою категорії знаходяться певні фільтри, а саме: тип автозапчастини та модель автомобіля. Таким чином зроблено цікаве розділення всієї цієї великої кучі різних автозапчастин. Просто вказавши тип автозапчастини або модель свого автомобіля, або і того, і того (тут можна комбінувати по своєму бажанню) ви отримаєте список всіх наявних товарів у магазині за максимально короткий термін. Що ж, звучить наче б то не погано, але тепер потрібно перевірити це на практиці, а чи працює воно взагалі. Тому виберемо тип автозапчастини і вкажемо модель автомобіля, а результати сортування зображу вам на рисунку 2.43.

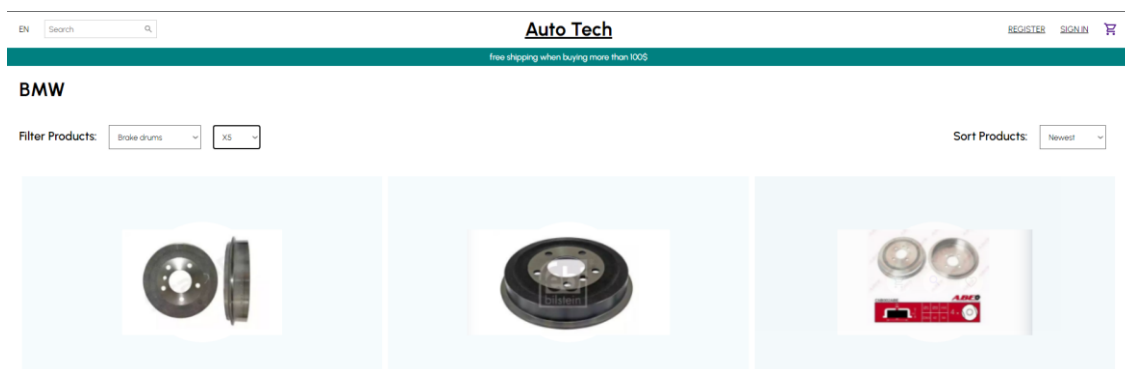


Рисунок 2.43 – Результати тестування фільтрації

Для тестування я вказав, що шукаю гальмівні диски моделі X5, і як ми можемо спостерігати із всієї великої кількості автозапчастин залишилося всього три товари. Вони і будуть відповідати нашим потребам, все працює коректно.

Також, як ви мабуть замітили, щось цікаве і подібне до фільтрації у нас є і з правої сторони. Але на цей раз це не фільтрація товару, адже ми вже отримали весь товар за нашим запитом. Цього разу ми будемо використовувати сортування товару, і звичайно що за ціною. При виборі сортування ціни за спаданням буде один порядок, а за зростанням інший. Також є сортування по давнині додавання в базу даних, але це не так цікаво. Результати тестування товару за спаданням ціни буде зображено на рисунку 2.44.

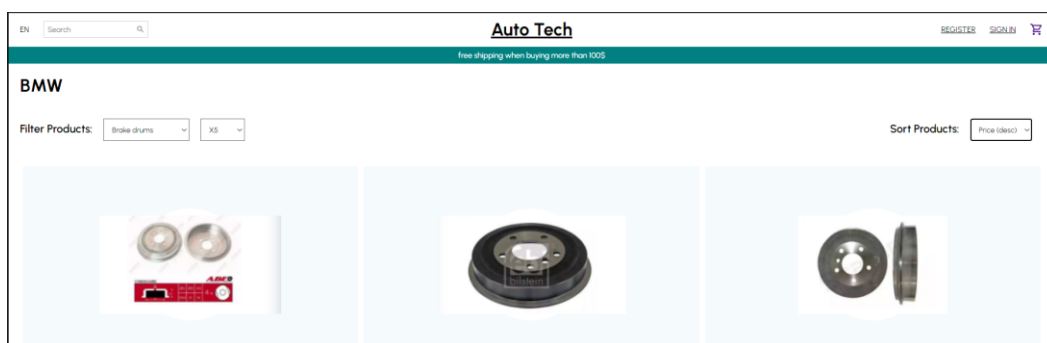


Рисунок 2.44 – Результати тестування сортування товару

Ті самі товари, які ми бачили раніше розмістилися дещо інакше. Якщо зайти на кожен товар окремо, то буде чітко видно, що вони сортуються за спаданням ціни, а це означає, що наше тестування успішне.

Наступне тестування ми проведемо з замовленням товару і його попаданням безпосередньо в саму корзину. Тут, на сторінці товару, при кліку на наявну кнопку «ADD TO CART» товар повинен потрапляти в корзину користувача, а на самій емблемі корзини має з'являтися біла цифра обведена у синьому кружечку, ця цифра буде означати кількість доданих товарів в неї. Як ви встигли помітити на рисунках раніше, емблема корзини знаходиться праворуч від кнопки реєстрації та логіну і на ній не було нічого написано і відповідно в самій корзині також не було жодного товару. Зараз ми будемо пробувати додавати товар в неї, результат буде зображено на рисунку 2.45.

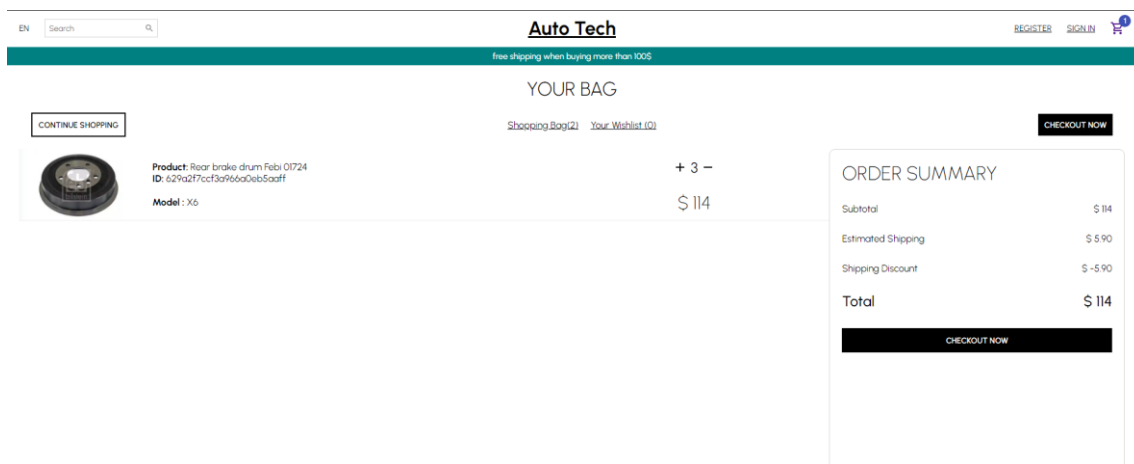


Рисунок 2.45 – Результати тестування корзини

Прекрасно видно, що товар потрапив до корзини зі всіма вказаними характеристиками, на самому значку корзини появилася циферка 1, тобто кількість доданих товарів в неї. Під значком корзини ми можемо бачити таблицю розрахунку вартості за всі товари, які є в корзині, а це означає, що все працює коректно. Тестування оплати можна також зробити, але воно проводилося дещо раніше, тому зараз його тут показувати не буду, просто повірте на слово, що воно працює.

Висновок до всіх проведених тестувань буде такий: сайт працює коректно.

3 РОБОТА З САЙТОМ В ІНТЕРНЕТІ

3.1 Інструкція з розміщення сайту в інтернеті

На даному етапі ми вже маємо повністю готовий інтернет-магазин автозапчастин «Auto Tech». Для адміністратора, якому він створювався обов'язково потрібні пояснення та інструкція про розміщення даної розробки в інтернеті.

Оскільки для даного проекту використовувалася база даних, яка знаходиться в хмарі, то з її встановленням проблем не буде. Необхідно просто зайти на сайт MongoDB та залогінитися. З цього моменту буде можливість керування даними. Для переносу всіх даних буде створено дамп бази даних і при потребі вони будуть мати можливість імпортуватися. Також, якщо потрібно внести будь-які зміни в базу даних, їх всіх можна буде легко зробити.

Якщо ж раптом MongoDB не буде підтримуватися певним хостингом, який було обрано для розміщення даного веб-сайту, то будуть виникати проблеми. Адже вона достатньо відрізняється від SQL баз даних і просто перейти з неї на останні не получится. В такому разі доведеться міняти майже всі існуючі моделі структури даних та додавати нові, крім цього у коді програми також потрібно буде зробити чимало змін, тому не варто робити такий різкий перехід. У випадку, якщо користування даної базою даних стане платним, можна перейти на інший варіант NoSQL бази даних, яка на даний час також вважається найкращою та на рівні з використовуваною – це Redis. У такому випадку не потрібно буде змінювати моделі структури даних, лише невеличкі зміни в коді і все буде працювати коректно. Детальніше ознайомитися з цією базою даних ви можете за посиланням: <https://redis.io>. Інших варіантів розвитку подій не передбачено. Але для уникнення можливих проблем, потрібно обирати хостинг для розміщення сайту, який буде підтримувати саме базу, на якій було започатковано дану розробку, тобто хостинги з підтримкою MongoDB.

Як всім відомо, більшість сайтів протягом свого створення розробляються на власних комп'ютерах, використовуючи їх як сервери. Це робить розробку продукту максимально зручною, адже тоді розробник не отримує жодних збоїв чи падінь мережі під час роботи зі своїм творінням, а також він економить мережевий трафік. Коли ж він остаточно закінчив розробляти свій веб-сайт, то його можна переносити на справжній сервер.

Хостинги бувають фізичними та віртуальними, а також також платними або безкоштовним. В нашому випадку буде використовуватися безкоштовний хостинг Heroku – Platform as a Service. Це означає, що дана платформа працює як сервіс, тобто дає нам можливість користувачу використовувати певні функції та її можливості, при цьому її інфраструктура повністю прихована. Особливу популярність Heroku набула через свою простоту у використанні та досить широкий спектр пропонованих сервісів. У даній платформі є декілька невеликих вимог, а саме:

- потрібно зареєструватися на офіційному сайті Heroku та вибрати тариф;
- скачати консольну утиліту The Heroku Command Line Interface, для роботи якої необхідно мати встановлений Git;
- Heroku підтримує Ruby, Python, PHP, Node.js, Java, Go, Scala і Clojure.

Для цього в першу чергу переходимо на сайт даного хостингу, його можна знайти за посиланням: <https://www.heroku.com>. Реєструємося на ньому, заповнивши форму. Далі скачуємо архів Heroku CLI для Windows 64-bit та встановлюємо його на комп'ютер. Оскільки ми використовували середовище розробки WebStorm, то заходимо в нього, відкриваємо консоль і для перевірки на успішне встановлення вводимо команду: `heroku --help`. Якщо це нам видасть версію heroku та список різних команд, значить встановлення CLI пройшло успішно. Тепер за допомогою команди `heroku login` входимо в стистему. Тепер ми можемо створити новий проект за допомогою команди `heroku create`. Якщо ж

зайти на сайт `dashboard.heroku`, то побачимо, що цей проект там появиться.

Після цього проініціалізуємо наш проект в `git`, якщо цього не було зроблено раніше за допомогою команди `git init`, а також додати всі файли за допомогою команди `git add`. Після цього робимо коміт даного проекту, не забуваючи до цього винести не потрібні файли в `gitignore`. Далі додаємо деякі віддалені репозиторії, за допомогою команди `heroku git:remote -a + назва проекту`. Наступною командою буде: `git push heroku master`. Стає видно, що віддалений сервер підхопив наш проект, встановлює всі необхідні залежності для нього, чистить кеш, будує пакет `node_modules` та запускає його і в результаті отримаємо посилання за яким можемо перейти або прописавши команду `heroku open`.

Одразу видно, що нам автоматично відкрилася вкладка в браузері по даній силці і тут ми вже можемо бачити наш проект, тільки тепер ми знаходимося не на локальному хості, а на віддаленому сервері, що може підтвердити адреса даного сайту, що зображено на рисунку 3.1.

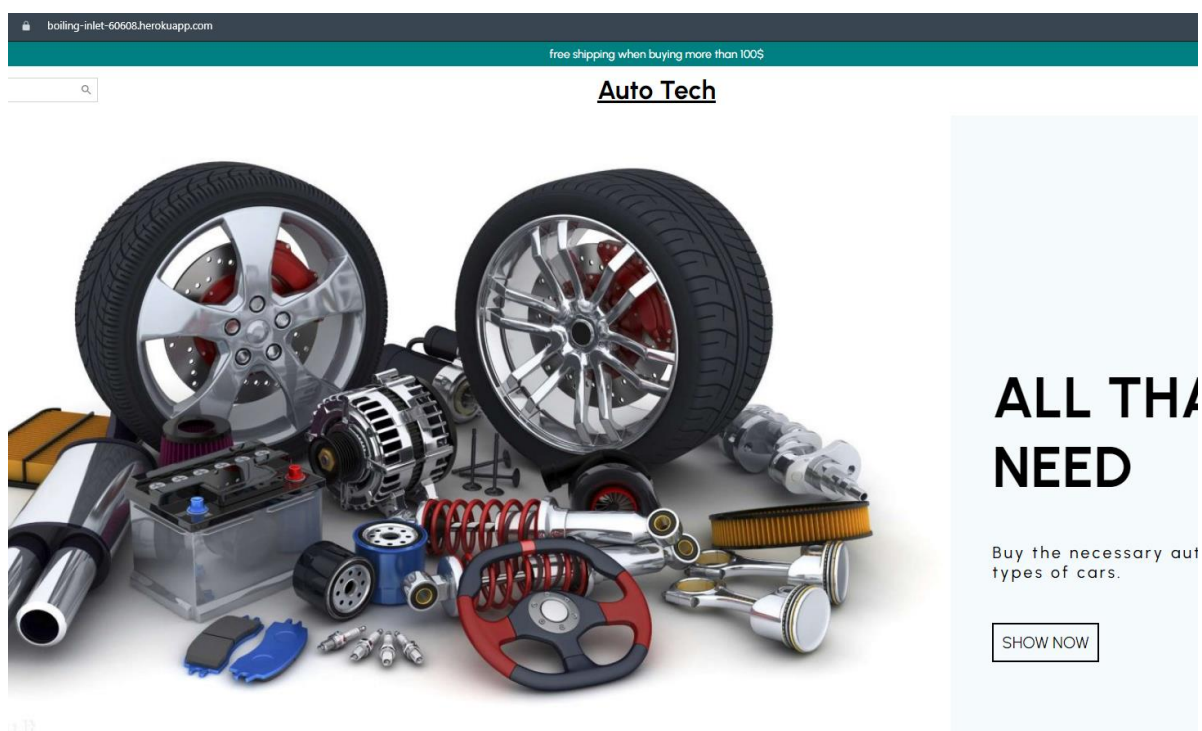


Рисунок 3.1 – Розміщення сайту на хостингу heroku

Отже, за допомогою цієї інструкції ми перенесли та розмістили наш веб-

сайт з локального хостингу на віддалений, тобто розмістили його в інтернеті.

3.2 Інструкція з обслуговування та наповнення сайту

Так зване обслуговування сайту та наповнення являється достатньо важливим моментом створення будь-якого веб-проєкту. Але тут йдеться мова про обслуговування та наповнення саме нашого інтернет-магазину автозапчастин «Auto Tech».

Наповнення даного магазину не буде відрізнятися від того, як його було наповнено раніше. Мається на увазі те, що вигляд всього його наповнення буде абсолютно такий самий як він є. Щоб щось змінити, потрібно лізти в код і вже там вирішувати певні аспекти, що можна змінити. Звичайно, в майбутньому це буде робитися, але зараз мова йде про наповнення саме вже існуючого інтернет-магазину з вже раніше прийнятими рішеннями його дизайну.

Тож якщо адміністратор виявить бажання додати якийсь товар у даний магазин, то йому потрібно виконати наступні кроки:

- зайти у свою адміністративну панель;
- ліворуч у функціоналі вибрати меню під назвою «Products»;
- далі він буде бачити список всіх наявних у магазині товарів, які відображаються в порядку їх додавання у магазин по часу, старіші товари будуть йти першими. Опис кожного товару буде відображатися у такій послідовності: ID, зображення товару, назва товару, ціна товару. Після ціни товару створено дві кнопки – «Edit» та «Delete»; Йому потрібно клікнути на кнопку «Edit»;
- далі відкриється сторінка редагування товару, а на ній в правому верхньому куті буде кнопка «Create», яка відповідає за створення нового товару, клікаємо на неї;
- тоді буде відкрита сторінка створення нового товару, яка зображена

на рисунку 3.2, на якій знаходиться лише одна формочка, яку потрібно заповнити. Коли адміністратор буде наповнювати сайт новим товаром, йому потрібно вказати його назву, опис, категорію до якої він належить, тип автозапчастини та модель автомобіля, також йому потрібно буде вказати зображення, яке представляє візуальний вигляд цього товару;

The screenshot shows the 'New Product' form in the Admin interface. The form is titled 'New Product' and has a sidebar on the left with navigation links: Home, Analytics, Sales, Users, Products, Transactions, Reports, Mail, Feedback, Messages, Manage, Analytics, and Reports. The form fields are: Image (with a 'Choose image' button), Title (text input), Description (text input), Price (text input), Categories (text input), Auto Part (text input), Model (text input), and Stock (dropdown menu). A 'Create' button is located at the bottom of the form.

Рисунок 3.2 – Форма для додавання нового товару

- коли всі поля форми заповнені, потрібно натиснути на кнопку «Create» і товар буде додано до цього інтернет-магазину і він буде відображатися у списку всіх товарів, результат таких маніпуляцій добавлень зображено на рисунку 3.3.

The screenshot shows the 'Admin' interface with a list of products. The table has columns for ID, Product, Stock, Price, and Action. The list contains 16 items, including various car parts like shock absorbers, brake drums, and injectors.

ID	Product	Stock	Price	Action
621912221a6c9e6452713952	Front shock absorber	true	10	edit delete
6219149a6c9e6452713952	Rear shock absorber Feb	true	12	edit delete
6219149a6c9e645271397c	Rear shock absorber Feb	true	15	edit delete
6219147731a6c9e645271397e	Rear shock absorber Feb	true	16	edit delete
621914a6c9e6452713980	Shock absorber NK 6347	true	30	edit delete
621914f11a6c9e6452713982	Front shock absorber Stz	true	32	edit delete
621918f01a6c9e6452713984	Brake drum LPR T0526	true	32	edit delete
621912021a6c9e6452713986	Rear brake drum Startline	true	35	edit delete
621912061a6c9e6452713988	Brake drum A.B.S. T269-5	true	30	edit delete
621912091a6c9e645271398a	Rear brake drum JP Grou	true	29	edit delete
6219121a1a6c9e645271398c	Rear brake drum Startline	true	35	edit delete
6219121a1a6c9e645271398e	Brake drum A.B.S. 2326-5	true	38	edit delete
621912201a6c9e6452713990	Injector JP Group 131356	true	12	edit delete
621912251a6c9e6452713992	Bosch fuel injector 0 437	true	42	edit delete
6219122a1a6c9e6452713994	Bosch fuel injector 0 437	true	42	edit delete
6219122d1a6c9e6452713996	Bosch fuel injector 0 437	true	58	edit delete

Рисунок 3.3 – Результат додавання товарів

Після виконання цих кроків товар успішно з'явиться на сайті.

У випадку, якщо було написано неправильну інформацію про той чи інший товар або інформація оновилася і на сайті знаходиться ще старий її варіант, адміністратор може скористуватися функцією редагування існуючих товарів, щоб успішно оновити вже існуючий товар, слід виконати наступні кроки:

- зайти у свою адміністративну панель;
- ліворуч у функціоналі вибрати меню під назвою «Products»;
- далі він буде бачити список всіх наявних у магазині товарів, які відображаються в порядку їх додавання у магазин по часу, старіші товари будуть йти першими. Опис кожного товару буде відображатися у такій послідовності: ID, зображення товару, назва товару, ціна товару. Після ціни товару створено дві кнопки – «Edit» та «Delete»; Йому потрібно клікнути на кнопку «Edit»;
- відкриється сторінка редагування товару, на ній буде знаходитися форма з подібними полями як у формі створення товару. Знову ж таки адміністратору потрібно її заповнити, не пропустивши жодне поле і натиснути на кнопку «Edit».

Після виконання подібних до створення товару кроків, товар буде успішно оновлено і це можна буде бачити всередині інтернет-магазину.

Якщо ж товар був випадково доданий у магазин, перестав більше продаватися даним магазином або сталася будь-яка інша причина, по якій його присутність більше не потрібна на сайті, адміністратор може скористатися функцією видалення товару. Для цього йому знову ж таки потрібно виконати наступне:

- зайти у свою адмін-панель;
- у функціоналі вибрати список товарів;
- знайти необхідний для видалення товар;
- справа від нього натиснути на кнопку видалення.

Після виконання цих кроків товар буде успішно видалено із магазину і він більше не буде відображатися на сайті.

Якщо потрібно виконати додавання, редагування чи видалення користувачів, то адміністратору слід виконати ряд аналогічних дій, але в адміністративній панелі обрати меню «Users».

Для розуміння того, що товар було замовлено, адміністратор у своїй панелі, на домашній сторінці справа знизу має список відображення замовлень. Тому для перевірки замовлень, йому просто потрібно зайти у свою панель та поглянути на список цих замовлень. Вони будуть сортуватися по часу, останні замовлення будуть йти першими.

Також для адміністратора було створено невеличку аналітику просування його магазину, тобто розроблено невеличкі таблиці статистики, де він може бачити кількість активних користувачів, які зареєструвалися на сайті в певний місяць, а також бачити скільки він прибутку отримав за останній місяць. Ця статистика не завжди є потрібною для продавців, але для тих власників, хто хоче ретельно проводити і підбивати статистику продажу та використання свого магазину загалом буде дуже актуальною.

Насамкінець, слід сказати, що нічого надзвичайного при наповненні даного магазину не прийдеться виконувати. Щоб він був завжди актуальним просто потрібно наповнювати його завжди. Як тільки появився якийсь новий товар – одразу додавати його в наш магазин, і тоді користувачі будуть бачити, що власник дійсно займається своїм сайтом, старається зробити його якомога насиченим товарами та корисним для всіх. Також для більшого привернення уваги та бажання купувати саме у цьому магазині автозапчастин слід давати максимальний опис кожного товару, щоб потенційні покупці одразу бачили всі характеристики і розуміли чи підходить їм цей товар, чи ні. Якщо таке питання зацікавить адміністратора, то він просто повинен відредагувати кожен товар, вказавши максимально наявний опис кожного.

3.1 Інструкція з популяризації та підтримки сайту

На сьогоднішній день не буде достатньо просто створити інтернет-магазин з товаром, нікому нічого не сказавши, і сподіватися, що він буде приносити хоч якийсь прибуток. Тут мається на увазі не те, що ваш сайт поганий, а лише те, що спочатку свого існування про нього ніхто не буде знати, а в гіршому випадку не знати взагалі. Задля поширення даної розробки, слід покращувати її SEO оптимізацію. Тут мова йде про оптимізацію сайту під різні запити пошукових систем. Потрібно вживати комплекси різних заходів, які будуть спрямовані на підвищення власного ресурсу при багатьох різних пошукових запитах користувача [5].

Тому з цього моменту було розпочато вищесказану оптимізацію сайту. Заходимо в настройки сайту, там необхідно вказати заголовок сторінки, META опис, а також META ключові слова. Над заголовком сторінки довго задумуватися не приходить, він буде наступний: «Магазин автозапчастин «Auto Tech». Щодо META опису, то вкажемо наступне: автозапчастини до багатьох автомобілів, автозапчастини зі знижкою, б/у автозапчастини. Що буде стосуватися META ключових слів, то також нічого надзвичайного – автозапчастини, авто, запчастини.

Наступним кроком буде завантаження бейджику для соціальних мереж. Він повинен містити заголовок, опис та картинку. Заголовок буде використано той самий, що і в попередньому пункті, опис відповідно також, а картинку на цьому моменті ми підгружаємо.

Надалі потрібно зробити читабельну адресу сторінки, тобто достатньо зрозумілий url для нашого сайту. Оскільки ми користуємося безкоштовною оптимізацією, то у нас є можливість зробити лише домен третього рівня, тим не менше робимо його людським: auto-tech.

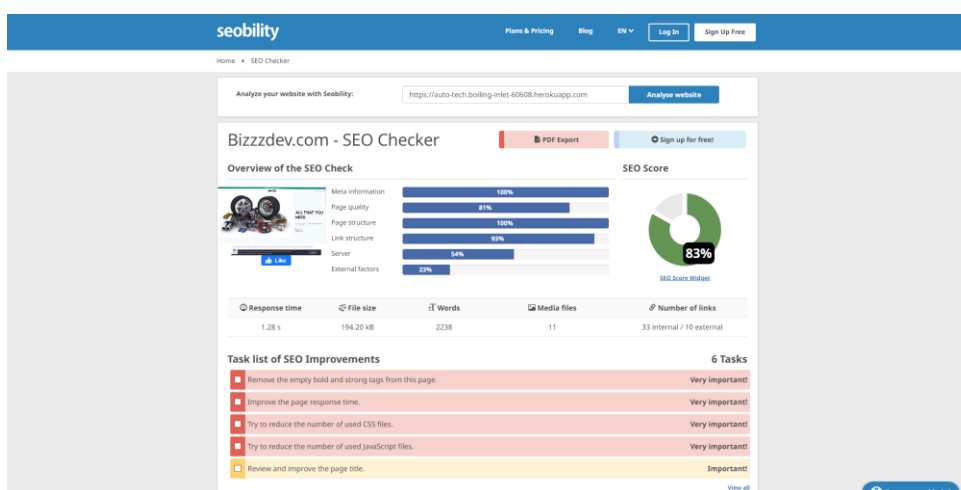
Тепер обов'язково потрібно додати заголовки за допомогою тегів H1, H2, H3. Тут є такий момент, що якщо в нас одразу є розподіл заголовків з використанням всіх цих трьох тегів, то цей пункт автоматично виконано, якщо ж ні то, за допомогою редактора змінюємо, щоб відповідало вимогам.

На наступному етапі слід встановити іконку для нашого сайту. Для цього просто заходимо в настройк сайту та вибираємо пункт з меню «іконка сайту» та підгружаємо наш малюнок туди.

В будь-якому робочому сайті завжди є сторінка, яка видає 404 помилку. Тут нам не треба над нею задумуватися, тому що при створенні сайту із шаблону вона автоматично створюється.

Далі потрібно налаштувати індексацію сторінок. Загалом для новачків краще це робити вибравши варіант «генерувати автоматично», але також є можливість це зробити вручну. Також є можливість вибрати деякі додаткові сторінки, які також будуть індексуватися. Налаштовуємо це і нажимаємо на кнопку «зберегти».

Зараз ми можемо зробити аналіз нашого сайту за допомогою онлайн-перевірки, прикладом такої перевірки можна навести сайт за посиланням: <https://www.seobility.net/en/seocheck>. Копіюємо посилання нашого сайту і вставляємо його в онлайн-аналізпатор. Він нам покаже на скільки оптимізований наш сайт та що забули вказати, як зображено на рисунку 3.4.



Риснок 3.4 – Результати онлайн-тестування сайту

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Основні причини електротравматизму. Електробезпека користувача ВДТ

Електротравми можуть бути наслідком технічних, організаційно-технічних, організаційних і організаційно-соціальних причин.

До технічних причин належать: недосконалість конструкції електроустановки і засобів захисту, допущені недоліки при виготовленні, монтажі і ремонті електроустановки. Крім перерахованих, технічними причинами електротравм можуть бути несправності електроустановок, що виникають в процесі їх експлуатації, несправність захисних засобів, невідповідність будови електроустановок і захисних засобів умовам їх застосування, використання електрозахисних засобів із простроченою датою чергових випробувань.

До організаційно-технічних причин належать: невиконання вимог чинних нормативів щодо контролю параметрів та опосвідчення технічного стану електроустановок; помилки в знятті напруги з електроустановок при виконанні в них робіт без перевірки відсутності напруги на електроустановці, на якій працюють люди; відсутність огорожень або невідповідність конструкції і розміщення вимогам чинних нормативів та відсутність необхідних плакатів і попереджувальних та заборонних написів; помилки в установленні і знятті переносних заземлень, або їх відсутність.

До основних організаційних причин електротравм належать:

- відсутність (непризначення наказом) на підприємстві особи, відповідальної за електрогосподарство або невідповідність кваліфікації цієї особи чинним вимогам;
- недостатня укомплектованість електротехнічної служби працівниками відповідної кваліфікації;

- відсутність на підприємстві посадових інструкцій для електротехнічного персоналу та інструкцій із безпечного обслуговування та експлуатації електроустановок;

- недостатня підготовленість персоналу з питань електробезпеки, несвоєчасна перевірка знань, невідповідність групи з електробезпеки персоналу характеру робіт, що виконуються;

- недотримання вимог щодо безпечного виконання робіт в електроустановках за нарядами-допусками, розпорядженнями та в порядку поточної експлуатації;

- неефективний нагляд, відомчий і громадський контроль за дотриманням вимог безпеки при виконанні робіт в електроустановках та їх експлуатації.

До основних організаційно-соціальних причин електротравм належать: змушене виконання не за спеціальністю електронебезпечних робіт; негативне ставлення до виконуваної роботи, обумовлене соціальними чинниками; залучення працівників до понадурочних робіт; порушення виробничої дисципліни; залучення до роботи осіб віком до 18 років.

Візуальний дисплейний термінал (ВДТ) включає в себе візуальний дисплей, клавіатуру і друкуючий пристрій (лазерний, струменевий або матричний). Робота з ВДТ потребує відповідні умови для забезпечення безпеки і гігієни праці і відповідно дотримання правил роботи з ними.

Правила безпеки при роботі з візуальними дисплеями:

- природне і штучне освітлення мають відповідати встановленим нормам. Природне освітлення забезпечується через світлові прорізи, орієнтовані на північ або північний схід, з коефіцієнтом природного освітлення не менше 1,5%

- висота екрану, кут нахилу та відстань до очей працівника мають бути оптимальними для комфортної роботи

- працівникам необхідно дотримуватися встановлених режимів праці та відпочинку для запобігання перенапруженню. Наприклад, після кожної години роботи з візуальним дисплеєм рекомендується робити 10-15-хвилинні перерви

- робочі місця з ВДТ повинні розміщуватися в окремих приміщеннях, що забезпечують необхідну площу та об'єм для кожного працівника. Площа одного робочого місця має бути не менше 6 м², об'єм – не менше 20 м³

- робочі місця повинні бути облаштовані з урахуванням ергономічних вимог, що дозволяють мінімізувати зорове та нервово-емоційне навантаження.

Правила безпеки при роботі з клавіатурою:

- клавіатура повинна розташовуватися на спеціально призначеній для цього поверхні, висота якої дозволяє підтримувати передпліччя і кисті рук у зручному положенні, щоб уникнути зайвого навантаження на м'язи і суглоби

- використання підставок для зап'ясть може знизити навантаження на кисті рук і запобігти розвитку тунельного синдрому

- регулярні перерви та виконання вправ для кистей рук і передпліч зменшують ризик виникнення професійних захворювань

Правила безпеки при роботі з друкуючими пристроями:

- приміщення, де розташовані друкуючі пристрої, повинні мати достатню вентиляцію для видалення озону та інших шкідливих речовин, що виділяються під час друку

- принтери і копіювальні апарати слід встановлювати на стійких поверхнях, на достатній відстані від робочих місць для зменшення впливу шуму та вібрації

- регулярне технічне обслуговування та своєчасна заміна картриджів і інших витратних матеріалів забезпечують безперебійну роботу пристроїв і знижують ризик виникнення аварійних ситуацій

Ці правила поширюються на умови й організацію праці при роботі з ВДТ усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок, що використовуються в ЕОМ колективного використання та ПЕОМ.

4.2 Протипожежне водопостачання

Система протипожежного водопостачання - це комплекс інженерно-технічних пристроїв, що виконують важливу роль у забезпеченні пожежної безпеки об'єктів та населених пунктів.

Під протипожежним водопостачанням слід розуміти таке водопостачання, коли вода подається цілодобово і у такій кількості, яка необхідна для гасіння пожеж ззовні та всередині будівель і споруд.

Водопроводи розраховують на безперебійну подачу води для виробничих, господарських та протипожежних потреб, іноді проектують спеціальні протипожежні водоводи. У деяких випадках допускається зберігання пожежного об'єму води у спеціальних резервуарах чи відкритих водоймах.

Протипожежні потреби складаються з розрахункових витрат води на зовнішнє пожежогасіння через гідрант і внутрішнє пожежогасіння через пожежні кран-комплекти, спринклерні, дренчерні та інші системи та установки пожежогасіння.

Водопроводи бувають високого та низького тиску. У водопроводах низького тиску, тиск води для гасіння забезпечується автомашинами, при водопроводах високого тиску - гасіння пожеж відбувається подачею води безпосередньо від водопровідної мережі.

Водопроводи протипожежного призначення не проектуються у виробничих будівлях I та II ступенів вогнестійкості з виробничими категоріями за пожежною небезпекою Г і Д незалежно від їх об'єму і у будівлях III ступеня вогнестійкості тієї ж пожежної небезпеки, але за умови, що їх об'єм не перевищує 1000 м³. В нормативних документах (СНиП) визначені умови, за яких влаштування водопроводів протипожежного призначення у будівлях є обов'язковим.

Для отримання води з мережі на протипожежні потреби у колодязях

встановлюють підземні або наземні пожежні гідранти, як правило за кільцевою системою, яка дозволяє у випадку аварії гідранта з одного боку магістралі подавати воду з іншого. До цехів або приміщень, що розташовані окремо, прокладають тунельні водопровідні лінії.

Пожежні гідранти на території підприємства встановлюють уздовж доріг та проїздів на розрахунковій відстані один від одного, але не далі 150 м та педалі як за 5 м від стін виробничого приміщення й поблизу перехрестя доріг. При встановленні гідрантів поза проїжджою частиною їх розташовують не далі як за 2 м від її краю. На стіні будівлі, біля місця розміщення гідранта, вивішують знак, який освітлюється у нічну пору доби. Протипожежний трубопровід має забезпечувати тиск не менше як 4 атм і не більше як 10 атм при витраті води не менш як 5 л/с.

Для надання струменю води необхідного напрямку, збільшення дальності дії і розпилення використовують ручні й лафетні стволи. Дальність струменя води залежить від параметрів ствола і тиску. Лафетні стволи призначені для отримання потужних водяних струменів. Живлення лафетних стволів здійснюється по 2-4 пожежних рукавах. Для надання струменю дальності й циліндричної форми служать насадки стволів, а для отримання конуса дрібно розпиленої води - застосовують стволи різних конструкцій.

Від мережі зовнішнього водопроводу живиться також внутрішній протипожежний водопровід з одним або двома вводами і внутрішніми пожежними кран-комплектами, які розміщуються у коридорах або сходових клітках на висоті 1,35 м від підлоги. Кран-комплект закривається у шафу і обладнується пожежним рукавом довжиною 20 м і пожежним стволом. На дверцятах шафи має бути позначка ПК з номером. Відстань між внутрішніми кран-комплектами залежить від довжини пожежного рукава, дальності дії струменя води, кількості необхідних пожежних струменів і розміщення технологічного обладнання. Розміщення пожежних кран-комплектів має бути таким, щоб гарантувати зрошення кожної точки приміщення не менше ніж двома

струменями води по 2,5 л/с кожен.

Зовнішню мережу протипожежного водопроводу виконують, як правило, кільцевого типу. Розміщують її на відстані не ближче 5 м від будівель і не більше 2,5 м від узбіччя автомобільних доріг. На мережі встановлюють водозабірні пожежні гідранти в такій кількості, щоб забезпечити пожежогасіння будь-якої будівлі, споруди, об'єкта або його частини не менше ніж від двох гідрантів (при потребном витраті води на зовнішнє пожежогасіння 15 л / с і більше) або від одного гідранта (менше 15 л / с) з урахуванням максимальної довжини прокладаються рукавних ліній 100-200 м (залежно від виду підключеної до гідранта пожежної техніки).

Пожежні гідранти повинні завжди перебувати в справному стані, в зимовий час їх утеплюють і очищають від снігу і льоду. У гідрантів і по напрямку руху до них повинні бути встановлені відповідні покажчики (об'ємні зі світильником або плоскі зі світловідбивним покриттям, стійкі до атмосферних впливів). На них повинні бути нанесені цифри, що вказують відстань до гідранта.

Якщо отримання розрахункової кількості води безпосередньо з джерела водопостачання (свердловина з насосною станцією) неможливе або економічно недоцільне, то в системі водопостачання передбачають спеціальні резервуари, штучні водойми (не менше двох), в кожному з яких має знаходитися не менше 50% потрібного (розрахункового) об'єму води. Для цих цілей також використовують наявні поруч природні водойми (озера, річки). Їх обладнають під'їздом з майданчиками (пірсами) з твердим покриттям розмірами не менше 12 × 12 м для встановлення на них пожежних автомобілів в будь-який час року.

Пожежні резервуари або штучні водойми розміщують виходячи з умови, що кожен пожежа повинна гаситися не менше, ніж з двох сусідніх резервуарів з урахуванням радіуса обслуговування: 200 м, якщо воду на гасіння пожежі подають автонасосами (пожежними автомобілями), і 100 м - мотопомпами.

ВИСНОВКИ

В даній дипломній роботі було розроблено інтернет-магазин автозапчастин «Auto Tech» на мові програмування JavaScript використовуючи такі її технології як React, Node.js, Redux, а також сховищем даних тут виступила база даних MongoDB.

Оскільки зараз інтернет-магазини стали надзвичайно популярними і практично вся молодь користується саме ними, то дана розробка буде дуже актуальна. Особливо зважаючи на пандемію в теперішній час є дуже багато людей, які не мають бажання потрапляти в багатолюдні місця, а магазини - одні з них, і саме таким людям це також буде дуже цікаво.

Звичайно, він не є ідеальним. Завжди є що покращувати і необхідність покращувати для хорошого існування та конкурентоспроможності на ринку з іншими подібними інтернет-магазинами. Надалі його буде тільки покращено, додано нові функції і можливості, також можливо буде покращена і сама подача інформації у ньому та дизайн.

Варто зауважити, що розроблена програма відповідає усім поставленим вимогам технічного завдання.

Під час виконання дипломної роботи було закріплено і поглиблено знання, одержанні за час вивчення таких предметів як «Web-технології та web-дизайн», а також «Технології та мови створення веб-сайтів», більш детально вивчено мову програмування JavaScript з використанням її технологій React та Redux, а також серверний JavaScript, тобто Node.js.

Для розробки даного програмного забезпечення використовувались принципи і концепції об'єктно-орієнтованого програмування. Використане середовище програмування WebStorm, його особливі, можливості і переваги, що дозволило швидко конструювати програмний засіб. Під час виконання даної дипломної роботи було використано все, що дозволило в значній мірі пришвидшити розробку даної програми.

ПЕРЕЛІК ПОСИЛАНЬ

1. Поняття, структура та різновиди веб-сайтів. Автоматизоване розроблення веб-сайтів. URL: <http://www.ndu.edu.ua/liceum/web.pdf>.
2. Знайомство з Styled components. Порівняння його з CSS. URL: https://habr.com/company/productivity_inside/blog/321804/.
3. Початок роботи з MongoDB. Основні переваги та недоліки. URL: <https://freexbcodes.com/sisadmin/pochatok-roboti-z-mongodb-dlja-pochatkivciv/>.
4. Основи MongoDB. URL: <https://codeguida.com/post/519>.
5. Початок просування сайту. URL: <https://bizzzdev.com/how-to-independently-promote-your-website/>.
6. React початок роботи. URL: <https://reactjs.org/docs/getting-started.html>.
7. React – Хуки: вебсайт. URL: <https://reactjs.org/docs/hooks-faq.html#gatsby-focus-wrapper>.
8. NodeJs документація. URL: <https://nodejs.org/uk/docs/>.
9. Основи Redux. URL: <https://rajdee.gitbooks.io/content/>.
10. NodeJs швидкий курс всього за одну годину. URL: <https://www.youtube.com/watch?v=3aGSqasVPsI&t=2150s>.
11. NodeJs повний базовий курс від А до Я. URL: <https://www.youtube.com/watch?v=nu4PiyjAmAE&t=8540s>.
12. React повний базовий курс від А до Я. URL: <https://www.youtube.com/watch?v=GNrdg3PzpJQ&t=1s>.
13. М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – “Інженерія програмного забезпечення” для усіх форм навчання [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.
14. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник ІВ

Бойко, МР Петрик, Г Цуприк – 2020

15. Дискретні структури (Алгебраїчні та числові системи, комбінаторний аналіз) : навчально-методичний посібник для студентів спеціальності 121 «Інженерія програмного забезпечення», аспірантів та викладачів вищих навчальних закладів / Укладач : Бойко І.В., Петрик М.Р., Цуприк Г.Б. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2017 – 64 с.

16. Моделювання та видобуток даних (висопродуктивні обчислення у великих алгебраїчних та числових системах, комбінаторному аналізі): навчальний посібник. Тернопіль: : ТНТУ 2019 – 62 с.

17. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.

ДОДАТКИ

ДОДАТОК А – Лістинг файлу «App.jsx»

```
import Product from "../pages/Product";
import Home from "../pages/Home";
import ProductList from "../pages/ProductList";
import Register from "../pages/Register";
import Login from "../pages/Login";
import Cart from "../pages/Cart";
import {
  BrowserRouter as Router,
  Switch,
  Route,
  Redirect,
} from "react-router-dom";
import Success from "../pages/Success";
import {useSelector} from "react-redux";

const App = () => {
  const user = useSelector((state) => state.user.currentUser);
  return (
    <Router>
      <Switch>
        <Route exact path="/">
          <Home/>
        </Route>
        <Route path="/products/:category">
          <ProductList/>
        </Route>
        <Route path="/product/:id">
          <Product/>
        </Route>
        <Route path="/cart">
          <Cart/>
        </Route>
      </Switch>
    </Router>
  );
};
```

```

        </Route>
        <Route path="/success">
            <Success />
        </Route>
        <Route path="/login">
            {user ? <Redirect to="/" /> : <Login />}
        </Route>
        <Route path="/register">
            {user ? <Redirect to="/" /> : <Register />}
        </Route>
    </Switch>
</Router>
)
};

export default App;

```

ДОДАТОК Б – Лістинг файлу «Products.jsx»

```

import styled from "styled-components";
import {popularProducts} from "../data";
import Product from "../Product";
import {useEffect, useState} from "react";
import axios from "axios";

const Container = styled.div`
  padding: 20px;
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
`;

const Products = ({cat, filters, sort}) => {
  const [products, setProducts] = useState([]);
  const [filteredProducts, setFilteredProducts] = useState([]);

  useEffect(() => {
    const getProducts = async () => {
      try {
        const res = await axios.get(
          cat
            ?
            `http://localhost:5000/api/products?category=${cat}`
            : "http://localhost:5000/api/products"
        );
        setProducts(res.data);
      } catch (err) {
      }
    };
    getProducts();
  });

```

```

    }, [cat]));

    useEffect(() => {
      cat &&
      setFilteredProducts(
        products.filter((item) =>
          Object.entries(filters).every(([key, value]) =>
            item[key].includes(value)
          )
        )
      );
    }, [products, cat, filters]);

    useEffect(() => {
      if (sort === "newest") {
        setFilteredProducts((prev) =>
          [...prev].sort((a, b) => a.createdAt - b.createdAt)
        );
      } else if (sort === "asc") {
        setFilteredProducts((prev) =>
          [...prev].sort((a, b) => a.price - b.price)
        );
      } else {
        setFilteredProducts((prev) =>
          [...prev].sort((a, b) => b.price - a.price)
        );
      }
    }, [sort]);

    return (
      <Container>
        {cat
          ? filteredProducts.map((item) => <Product item={item}
            key={item.id}/>)
        }
      </Container>
    );
  }
}

```

```
        : products
          .slice(0,12)
          .map((item) => <Product item={item} key={item.id}/>)}
      </Container>
    );
  };

export default Products;
```

ДОДАТОК В – Лістинг файлу «product.js»

```
const router = require("express").Router();
const verifyTokenAndAuthorization =
require("../middleware/verifyTokenAndAuthorization")
const verifyTokenAndAdmin = require("../middleware/verifyTokenAndAdmin")
const Product = require("../models/Product");
const ApiError = require("../error/ApiError")
```

```
//CREATE
```

```
router.post("/", verifyTokenAndAdmin, async (req, res) => {
  const newProduct = new Product(req.body);

  try {
    const savedProduct = await newProduct.save();
    res.status(200).json(savedProduct);
  } catch (err) {
    res.status(500).json(err);
  }
});
```

```
//UPDATE
```

```
router.put("/:id", verifyTokenAndAdmin, async (req, res) => {

  try {
    const updatedProduct = await Product.findByIdAndUpdate(
      req.params.id,
      {
        $set: req.body,
      },
```

```

        {new: true}
    );
    res.status(200).json(updatedProduct);
} catch (err) {
    res.status(500).json(err);
}
});

```

//DELETE

```

router.delete("/:id", verifyTokenAndAdmin, async (req, res) => {
    try {
        await Product.findByIdAndDelete(req.params.id);
        res.status(200).json("Товар було видалено");
    } catch (err) {
        res.status(500).json(err);
    }
});

```

//GET PRODUCT

```

router.get("/find/:id", async (req, res) => {
    try {
        const product = await Product.findById(req.params.id);
        res.status(200).json(product);
    } catch (err) {
        res.status(500).json(err);
    }
});

```

//GET ALL PRODUCTS

```

router.get("/", async (req, res) => {
    const qNew = req.query.new;
    const qCategory = req.query.category;
    try {
        let products;

```

```

        if (qNew) {
            products = await Product.find().sort({ createdAt: -1
}).limit(1);
        } else if (qCategory) {
            products = await Product.find({
                categories: {
                    $in: [qCategory],
                },
            });
        } else {
            products = await Product.find();
        }
        res.status(200).json(products);
    } catch (err) {
        res.status(500).json(err);
    }
});

module.exports = router;

```

ДОДАТОК Г – Лістинг файлу «NewProduct.jsx»

```
import {useState} from "react";
import "./newProduct.css";
import {
  getStorage,
  ref,
  uploadBytesResumable,
  getDownloadURL,
} from "firebase/storage";
import app from "../../firebase";
import {addProduct} from "../../redux/apiCalls";
import {useDispatch} from "react-redux";

export default function NewProduct() {
  const [inputs, setInputs] = useState({});
  const [file, setFile] = useState(null);
  const [cat, setCat] = useState([]);
  const [part, setPart] = useState([]);
  const [model, setModel] = useState([]);
  const dispatch = useDispatch();

  const handleChange = (e) => {
    setInputs((prev) => {
      return {...prev, [e.target.name]: e.target.value};
    });
  };

  const handleCat = (e) => {
    setCat(e.target.value.split(","));
  };

  const handlePart = (e) => {
    setPart(e.target.value.split(","));
  };
}
```

```

const handleModel = (e) => {
    setModel(e.target.value.split(","));
};

const handleClick = (e) => {
    e.preventDefault();
    const fileName = new Date().getTime() + file.name;
    const storage = getStorage(app);
    const storageRef = ref(storage, fileName);
    const uploadTask = uploadBytesResumable(storageRef, file);

    // Register three observers:
    // 1. 'state_changed' observer, called any time the state changes
    // 2. Error observer, called on failure
    // 3. Completion observer, called on successful completion
    uploadTask.on(
        "state_changed",
        (snapshot) => {
            // Observe state change events such as progress, pause, and
resume
            // Get task progress, including the number of bytes
uploaded and the total number of bytes to be uploaded
            const progress =
                (snapshot.bytesTransferred / snapshot.totalBytes) *
100;

            console.log("Upload is " + progress + "% done");
            switch (snapshot.state) {
                case "paused":
                    console.log("Upload is paused");
                    break;
                case "running":
                    console.log("Upload is running");
                    break;
                default:

```

```

    }
  },
  (error) => {
    // Handle unsuccessful uploads
  },
  () => {
    // Handle successful uploads on complete
    // For instance, get the download URL:
    https://firebasestorage.googleapis.com/...
    getDownloadURL(uploadTask.snapshot.ref).then((downloadURL)
=> {
      const product = {...inputs, img: downloadURL,
categories: cat, autoPart: part, model: model};
      addProduct(product, dispatch);
    });
  }
);
};

return (
  <div className="newProduct">
    <h1 className="addProductTitle">New Product</h1>
    <form className="addProductForm">
      <div className="addProductItem">
        <label>Image</label>
        <input
          type="file"
          id="file"
          onChange={(e) => setFile(e.target.files[0])}
        />
      </div>
      <div className="addProductItem">
        <label>Title</label>
        <input

```

```

        name="title"
        type="text"
        placeholder="some name"
        onChange={handleChange}
    />
</div>
<div className="addProductItem">
    <label>Description</label>
    <textarea
        name="desc"
        type="text"
        placeholder="description..."
        onChange={handleChange}
    />
</div>
<div className="addProductItem">
    <label>Price</label>
    <input
        name="price"
        type="number"
        placeholder="100"
        onChange={handleChange}
    />
</div>
<div className="addProductItem">
    <label>Categories</label>
    <input type="text" placeholder="Audi, Injectors"
onChange={handleCat}/>
</div>
<div className="addProductItem">
    <label>Auto Part</label>
    <input type="text" placeholder="Shock absorber"
onChange={handlePart}/>
</div>

```

```
      <div className="addProductItem">
        <label>Model</label>
        <input          type="text"          placeholder="A4,Q6"
onChange={handleModel}/>
      </div>
      <div className="addProductItem">
```

ДОДАТОК Д – Диск із кваліфікаційною роботою бакалавра