МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка інтернет магазину для продажу мобільних телефонів»**

Виконав: студент 2 курсу групи Іт-22сп

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Лисейко Юрій Іванович.
(Прізвище та ініціали)

Керівник: к.т.н., в.о. доцента Падюка Р.І.
(Прізвище та ініціали)

Рецензент: к.т.н., доцент Шарибура А.О.
(Прізвище та ініціали)

ДУБЛЯНИ-2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА
ІНФОРМАЦІЙНИХ ТЕХНЕОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А. М. Тригуба
«____» _____2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Лисейко Юрію Івановичу

1. Тема роботи: «Розробка інтернет магазину для продажу мобільних телефонів»

Керівник роботи Падюка Роман Іванович, в.о. доцента
затверджені наказом по університету від 30.12.2022 року № 453/к-с.

2. Строк подання студентом роботи 10.06.2021 р.
3. Вихідні дані до роботи: *вимоги до проектування інтернет-магазинів; методика проектування інформаційних систем; технічне завдання на проектування інтернет-магазину з продажу мобільних телефонів .*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз предметної області.
2. Постановка задачі.
3. Проектування інтернет-магазину з продажу мобільних телефонів .
4. Охорона праці.
Висновки та пропозиції.
Список використаної літератури.
5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень): *Огляд існуючих платформ з продажу мобільних телефонів, аналіз існуючих технологій зі створення інтернет-магазинів, етапи створення інтернет магазину і пов'язані із цим технології, структурна схема проектованої інформаційної системи, сценарії використання магазину та загальний вигляд сторінки та її фунціоналу*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3	*Падюка Р.І., в.о. доцента кафедри ІТ*		
4	*Городецький І.М., доцент кафедри управління проектами та безпеки виробництва*		

7. Дата видачі завдання 2 січня 2023 р.

Календарний план

№ з/п	Назва етапів дипломного проекту	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*2.01-02.02.23*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*03-28.02.23*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*01.03-30.04.23*	
4.	*Написання розділу «Охорона праці»*	*01-15.05.23*	
5.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*15-30.05.23*	
6.	*Завершення роботи вцілому*	*01 - 10.06.23*	

Студент _____ Лисейко Ю.І.
(підпис)

Керівник роботи _____Падюка Р.І.
(підпис)

УДК 004.9 : 631.1

Розробка інтернет магазину для продажу мобільних телефонів.

Лисейко Ю.І. Кафедра ІТ – Дубляни, Львівський НУП, 2023.

Кваліфікаційна робота: 67 с. текст. част., 56 рис., 10 арк. ілюстраційного матеріалу, 32 джерел.

Здійснено аналіз предметної області дослідження, визначено поняття та основні характеристики електронної комерції, особливості концепції інтернет-магазинів як одного з основних типів веб-сайтів, проаналізовано основні переваги та недоліки інтернет-магазинів, визначено класифікацію та характеристики методів розробки.

Детально пояснені різні технології, які поєднуються для формування стеку MERN, що вони демонструють, як вони функціонують та з'єднуються. Крім того здійснено аналіз, інших інструментів, які допомагають керувати даними додатків, обробкою помилок та аутентифікацією користувача. Обговорено та вивчено шляхи реалізації кінцевої програми – інтернет магазину з продажу мобільної техніки

Обрано модель *SPA*, оскільки вона дає швидкість і простоту використання. Під цю модель обрано базу даних *mongoDb* і зберігання цієї бази даних на серверах *mongoDb*. Написано модулі для хедера, адаптованість якого забезпечується наявністю мобільного і десктопного хедера, які вмикаються в залежності від ширини екрана. Реалізовано сторінки з купівлею товарів та сторінки для улюблених товарів і кошика.

Розроблено заходи щодо охорони праці.

Ключові слова: інтернет-магазин, мобільні телефони, сайт, стек MERN, модель SPA.

Key words: online store, mobile phones, website, MERN stack, SPA model.

ЗМІСТ

ВСТУП

В сучасному світі мобільні пристрої стали невід'ємною частиною нашого повсякденного життя. Зростаюча популярність та високий попит на мобільні пристрої створюють великі можливості для бізнесу. Розробка інтернет-магазину, спеціалізованого на продажі мобільних пристроїв, є ключовим елементом успішного ведення такого бізнесу.

Мета цієї кваліфікаційної роботи полягає в розробці інтернет-магазину, який забезпечує зручне та просте замовлення мобільних пристроїв для покупців, а також надає надійну та безпечну платформу для торгівлі. Ми будемо розглядати різні аспекти розробки, такі як веб-дизайн, функціональність, безпека, оптимізація швидкості завантаження.

Під час розробки інтернет-магазину ми використовуємо сучасні технології та інструменти, такі як стек MERN та бази даних і фреймворки, які входять до його складу. Застосування цих інструментів допомагає нам створити зручне та привабливе користувацьке середовище для наших клієнтів, забезпечуючи їм позитивний досвід під час покупки мобільних пристроїв.

Кваліфікаційна робота також розгляне питання безпеки та захисту даних у інтернет-магазині. Ми звернемо особливу увагу на захист персональної інформації покупців, використовуючи шифрування та інші заходи безпеки для запобігання несанкціонованому доступу до даних.

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття електронної комерції

Електронна комерція, також відома як «e-commerce», стосується купівлі та продажу товарів або послуг через Інтернет, а також передачі коштів і даних для здійснення цих операцій. Електронна комерція часто використовується для позначення продажу фізичних товарів через Інтернет, але вона також може описувати будь-який тип комерційної операції через Інтернет. Тим часом електронна комерція стосується транзакцій товарів і послуг, оскільки електронна комерція стосується всіх аспектів онлайн-бізнесу. З кожним роком цей вид ділових відносин захоплює все більшу частку ринку, збільшуючи продажі товарів і об'єднуючи нові напрямки і сфери діяльності. Електронна комерція може означати бронювання та виконання замовлень, операції через банківські послуги або системи електронних грошей [32].

Існує чотири основні типи моделей електронної комерції, які можуть описати майже будь-яку транзакцію між споживачем і компанією [29].

Рис. 1.1 Основні типи моделей електронної комерції

Business-to-consumer (B2C): коли компанія продає продукт або послугу окремому споживачеві (наприклад, ви купуєте пару взуття в інтернет-магазині);

Business-to-business (B2B): коли одна компанія продає продукт або послугу іншій компанії (наприклад, компанія продає програмне забезпечення як послугу для використання іншими компаніями);

Consumer-to-consumer (C2C): коли один споживач продає продукт або послугу іншому споживачеві (наприклад, ви продаєте вживані меблі іншому споживачеві на eBay);

Consumer-to-business (C2B): коли споживачі продають свої товари чи послуги компанії чи організації (наприклад, впливова особа пропонує своїй аудиторії в Інтернеті в обмін на плату, або фотограф ліцензує свої фотографії для використання в бізнесі) [29].

Насьогодні охоплено більшість ринків B2B і B2C. Однак електронна комерція повинна містити наступні компоненти:

- Інтернет-призначення (наприклад, веб-сайт, обліковий запис, онлайн-магазин або цільова сторінка);
- Канали залучення трафіку (такі як SEO, SMM, контекстна реклама, таргетована реклама);
- Системи обробки замовлень і обслуговування клієнтів
- CRM, офіси продажів і служби підтримки

Послуги охоплюють закупівлю, постачання, доставку та повернення.

Електронна комерція має широкий спектр можливостей, починаючи від транзакцій між бізнесом і споживачем і закінчуючи обміном різними товарами в таких транзакціях.

Роздрібна торгівля, як прямий продаж товарів від підприємства покупцеві без залучення посередників.

Оптова торгівля означає продаж товарів у великих кількостях, як правило, роздрібним продавцем, який згодом продає їх безпосередньо споживачам.

Дропшиппінг передбачає продаж продуктів, які виробляються та доставляються замовнику третьою стороною.

Краудфандинг передбачає збір коштів від споживачів до запуску продукту, як засіб забезпечення необхідного початкового капіталу для виходу на ринок.

Підписка означає процес придбання продукту чи послуги, який автоматично поновлюється на регулярній основі, доки абонент не вирішить скасувати її.

Продукти, які підпадають під категорію фізичних продуктів, це ті, які потребують поповнення запасів і замовлення виконуються фізичними засобами під час продажу [32].

Цифрові елементи, такі як моделі, курси для завантаження та медіафайли, перед використанням потрібно придбати, що класифікує їх як цифрові продукти.

Послуги стосуються певних навичок або набору навичок, які пропонуються за плату. Клієнти можуть придбати час у постачальника послуг за плату.

Здійснення онлайн-транзакцій вимагає створення спеціальних ресурсів, які спрощують процес для всіх залучених сторін. Різні форми електронної комерції підтримують кілька типів сайтів:

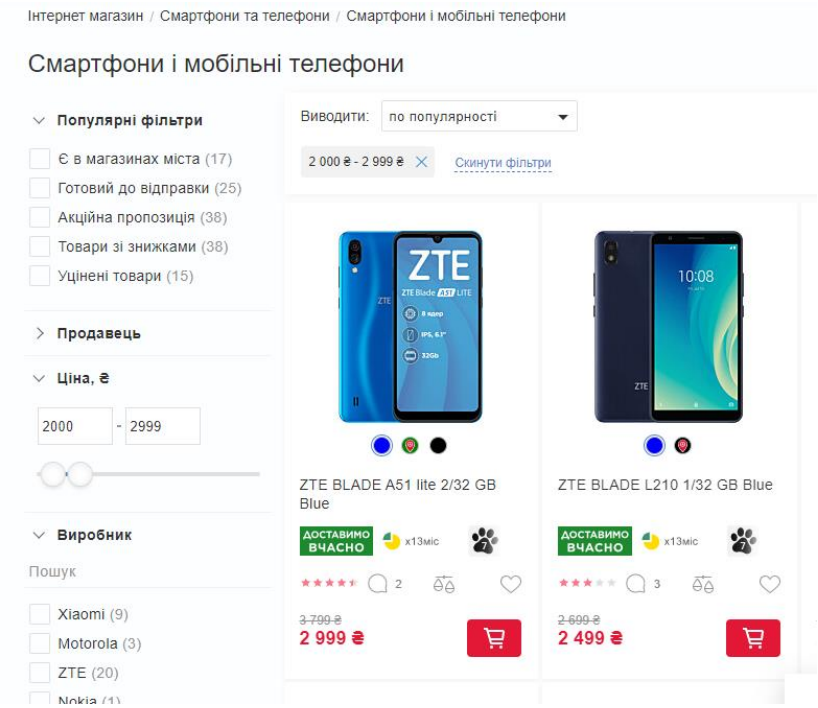

Рис. 1.2 Сортування пропозицій мобільних телефонів на прикладі інтернет-магазину мережі «АЛЛО»

Сервіси оголошень. Ці веб-сайти зосереджені на відносинах C2C і пропонують рекламні послуги. Користувачі можуть створювати картки, що рекламують їхні товари чи послуги на платформі, включно з умовами покупки та класифікацією їх для зручності пошуку потенційними покупцями (як показано на рис. 1.2).

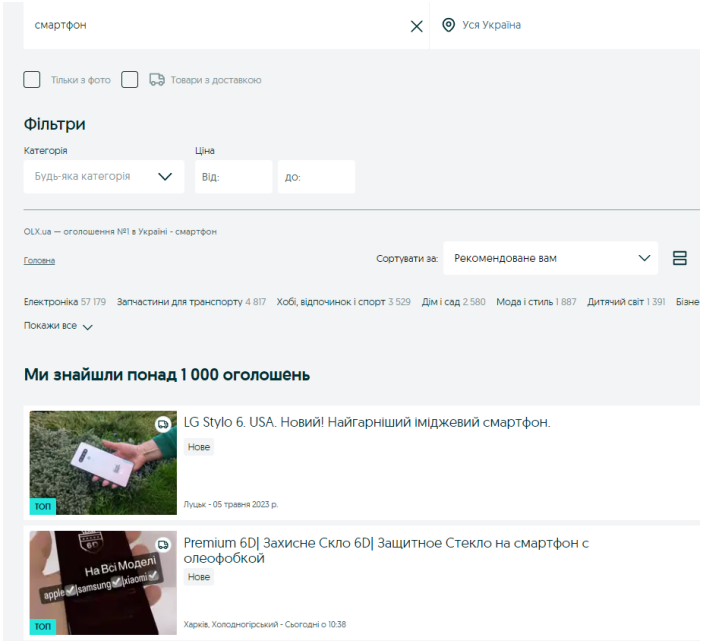

Рис. 1.3 Веб-сайт сервісу оголошень OLX

Сайти знижок. Веб-сайти, які спеціалізуються на знижках, збирають пропозиції від численних постачальників і окремих осіб з однією загальною вимогою: промокод, який надає покупцям знижену ціну (рис. 1.4).

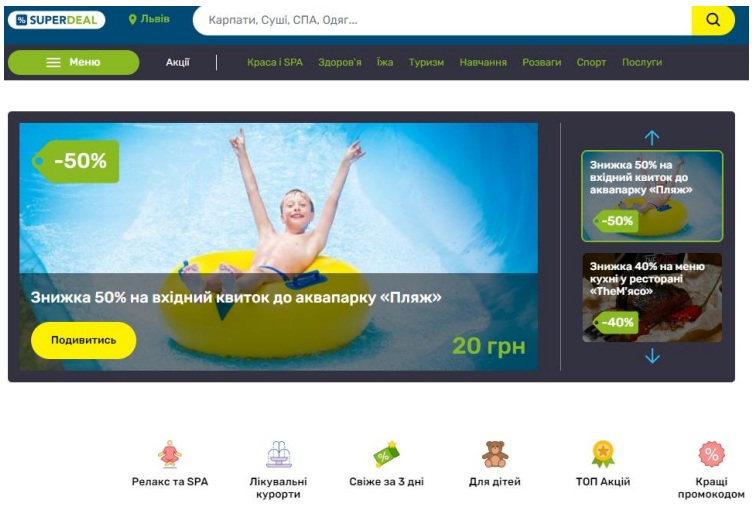

Рис. 1.4 Веб-сайт сервісу знижок SUPERDEAL

Втіленням класичного розуміння електронної комерції є інтернет-магазин. Розглянемо поняття та основні характеристики інтернет-магазину більш детально.

1.2 Огляд характеристик інтернет-магазину як різновиду веб-сайтів

Продаж своїх товарів та послуг через Інтернет є основним способом комерційного використання сайтів для більшості виробничих та торгових компаній. Компанія створює веб-сайт і публікує інформацію про свою продукцію та послуги, ціни та гарантії для клієнтів. Інтернет-магазини стають одним з найнеобхідніших та найефективніших інструментів для збільшення продажів, підвищення обороту та покращення іміджу.

Існує кілька визначень поняття "інтернет-магазин".

Інтернет-магазин – спеціалізований вид бізнесу, заснований на дистанційному обслуговуванні, що дозволяє продавцю та покупцю здійснювати угоди з продажу певного виду товарів через Інтернет.

Інтернет-магазин – це інтерактивний веб-сайт, який рекламує товар або послугу, приймає замовлення на покупку та пропонує користувачеві вибір варіантів оплати, наприклад, отримання замовлення та рахунки на оплату.

Інтернет-магазин – набір програм, які працюють на сайті та дозволяють покупцеві дистанційно вибрати товар із каталогу та оформити замовлення.

Інтернет-магазин є посередником між покупцем та складом виробника, якому достатньо мати інтернет-ресурс та телефон.

Логіка роботи інтернет-магазину

Відвідувач заходить на сайт та вибирає перелік товарів, після чого оформляється замовлення. Якщо відвідувач вже купував товари на цьому сайті, достатньо ввести логін, пароль та вказати адресу доставки. Якщо ви не придбали

товар, вам доведеться запровадити додаткову контактну інформацію. Потім вибирається спосіб доставки та формується кінцева вартість замовлення (товар + доставка).

Наступний крок – вибір форми оплати. Зазвичай відбувається перехід на сайт платіжної системи, де користувач входить до системи та оплачує покупку.

Інтернет-магазин має ряд переваг у порівнянні зі звичайними магазинами:

- інтернет-магазин працює 24 години на добу, 365 днів на рік, без перерви на обід, вихідні та свята;
- не обов'язково мати товар у наявності. Але в даному випадку вам потрібна мережа постачальників, яка працює за принципом "точно вчасно". Це гарантує доставку товару у мінімальні заздалегідь встановлені терміни;
- не потрібно приміщення для продажів. Вітрини, полиці, прилавки, квадратні метри та орендна плата – все це може заощадити багато грошей;
- час та витрати на створення інтернет-магазину незрівнянно менші, ніж на створення звичайного магазину;
- свобода пересування для продавця. Оскільки інтернет-магазин – це торгова точка в кіберпросторі, ви можете керувати ним із будь-якої точки світу, де є доступ до Інтернету;
- за допомогою інтернет-магазину ви маєте можливість розширити географію свого бізнесу до глобальних ринків. З застереженням, що зміст сайту буде зрозумілим і іноземцям;

Професійно розроблений інтернет-магазин може працювати повністю автономно.

Однак крім переваг ведення торгівлі через інтернет-магазин є і суттєві ризики:

- хакерські атаки;
- "жучки". Це помилки у програмному забезпеченні, створені самими розробниками. Помилки є завжди. Це може призвести до того, що в якийсь момент, за певних умов, ваш магазин почне "глючити" або навіть "відмовлятися"

працювати. Служба технічної підтримки, що добре функціонує, у розробника допомагає усунути помилки;

- клієнти легко приходять та йдуть. На щастя, не обов'язково "їхати" до інтернет-магазину: важливо знати адресу. Але, на жаль, щоб залишити його, достатньо одного кліка. Все, що ви можете зробити для залучення клієнтів на свій сайт – це оригінальний дизайн, хороший бізнес, хороші текстові описи товарів.

Однак, незважаючи на існуючі недоліки, які можна усунути, доклавши трохи зусиль, інтернет-магазини завойовують дедалі більшу довіру та популярність і, безперечно, мають велике майбутнє.

Головна особливість інтернет-магазину полягає в тому, що більша частина взаємодії продавця та покупця відбувається у режимі онлайн.

Багато етапів купівлі-продажу цілком успішні на відстані. Це велика перевага. Але водночас це одна з труднощів, яку має подолати будь-який інтернет-магазин: викликати довіру у відвідувача та поставити її перед фактом покупки.

1.3 Класифікація інтернет-магазинів

Сьогодні є велика кількість класифікацій інтернет-магазинів. Розглянемо основні їх класифікації. Інтернет-магазини можна розділити на такі типи [32]:

- за моделлю бізнесу;
- за обсягами продажів;
- за видами продажів;
- за способами отримання доходу;
- за товарним асортиментом;
- за відношенням з постачальниками;
- за подачею товарів у каталозі.

За моделлю бізнесу:

- Чисто онлайновий магазин.
- Поєднання офлайнового бізнесу з онлайновим (коли інтернет-магазин був створений на основі вже діючої реальної торгової структури).
- На аутсорсингу. При цьому виді інтернет-магазину ви самі безпосередньо не займаєтеся прийомом, доставкою, зберіганням комплектацією замовлень, а передаєте на аутсорс сторонній компанії, вирішуючи лише організаційні бізнес питання.
- Продаж за системою дропшипінг (коли товару немає, а сайт магазину “продає” продукцію постачальників, відправляючи замовнику готові замовлення або купуючи у нього від імені свого покупця).

За обсягами продажів [32]:

- роздрібний продаж;
- оптовий продаж.

За видами продажів:

- B2B - продаж або надання послуг іншим комерційним підприємствам;
- B2C - продаж або надання послуг кінцевому споживачеві.

За способами отримання доходу [32]:

Продаж через сайт товарів і послуг від виробника або офіційного представника.

Продаж товарів і послуг за партнерською програмою.

Продаж інформації (контенту). Для продажу інформації вже немає необхідності у витратах на папір, тиражі, точки продажу. Будь-яка веб-сторінка

може бути доступна мільйонам користувачів з усіх країн світу. Основний спосіб реалізації цього виду інтернет-магазину - платні електронні розсилки та сайти з платним доступом.

По відношенню з постачальниками [32]:

- мають власний склад (наявність реальних товарних запасів);
- працюють за договорами з постачальниками (відсутність значних власних запасів).

Види інтернет-магазинів за подачею товарів у каталозі: інтернет-вітрини, інтернет магазини, онлайн-аукціони.

Інтернет-вітрина – швидше це рекламний сервер. На вітрині викладають інформацію про товари, яку постійно оновлюють. Витрати на її створення та адміністрування можуть бути досить низькими, а практична користь такої вітрини очевидна.

Але це ще не торгівля. Потенційний покупець, відвідавши вітрину, повинен зателефонувати на фірму, оплатити товар, домовитися про доставку.

Тому інтернет-вітрина виправдана в тих випадках, коли покупця треба познайомити зі складною продукцією, на вивчення якої в торговому залі у нього піде дуже багато часу.

Інтернет-вітрина може бути розміщена де завгодно - на власному сервері, на сервері провайдера, на сервері, що надає безкоштовні сторінки.

Для роботи з вітриною достатньо мати підключення через телефонну лінію і мінімум навичок роботи з HTML.

Каталог інтернет-магазину буде містити багаторівневе логічне дерево розділів і підрозділів, всередині яких будуть перебувати списки з картками товарів. Він повинен бути добре структурованим і дуже логічним, щоб в розмаїтті товарів відвідувачі знайшли те, що їм дійсно потрібно. Кінцевою точкою переміщення по каталогу повинна стати картка товару, що містить його опис, ознаку наявності в продажу і ціну.

Онлайн-аукціон або інтернет-аукціон - це аукціон, який проводиться за допомогою мережі Інтернет. Він є видом інтернет-магазину, так як має в своєму функціоналі прийом онлайн-платежів. Момент закінчення інтернет-аукціону заздалегідь призначається самим продавцем при постановці товару на торги [1].

1.4 Аналіз основних вимог до створення інтернет-магазину

Сучасний Інтернет стрімко розвивається, і те, що ще вчора було неможливо, сьогодні доступне кожному. Нещодавно розробка та створення інтернет-магазину було заняттям для всіх, а сьогодні інтернет-магазин може створити практично кожен.

Існує кілька умовних базових вимог, які допоможуть створити якісний та ефективний інтернет-магазин.

Вибір CMS. Система управління контентом пропонує зручність обслуговування клієнтів, вартість використання програмного забезпечення, функціональність майбутнього проекту, особливості його дизайну та багато іншого.

Вибір оптимальної платформи ґрунтується на багатьох параметрах. До них відносяться вартість використання, функціональність, технічні характеристики, зручність інтерфейсу та наявність технічної підтримки. До цього списку можна додати особисті переваги та характеристики діяльності. Однак спочатку необхідно з'ясувати, чи підходить обрана платформа для створення інтернет- магазину.

Справа в тому, що існує ряд двигунів, які мають певні обмеження, що не дозволяють розробити якісну торгову платформу. Наприклад, відсутність кошика, платіжних систем та інших компонентів. Крім того, є платформи, призначені тільки для розробки форумів або візитних карток. Такі CMS не підходять апріорі.

Архітектура сайту – систематизація інформації та навігація по ній щоб допомогти відвідувачам ефективніше знаходити потрібні їм дані, це логічний

поділ сайту на блоки, розташовані в ієрархічному порядку. Двома словами, це схема позиціонування розділів продукції, інформаційних сторінок, товарних листів, спеціальних сторінок. Переглядаючи структуру на папері або моніторі комп'ютера, можна з першого погляду оцінити загальну картину роботи інтернет-магазину. Добре продумана та грамотна архітектура сайту гарантує, що користувачі витрачають менше часу на пошук потрібної інформації. Приклад структури сайту інтернет-магазину представлено на рис. 1.5 [32].

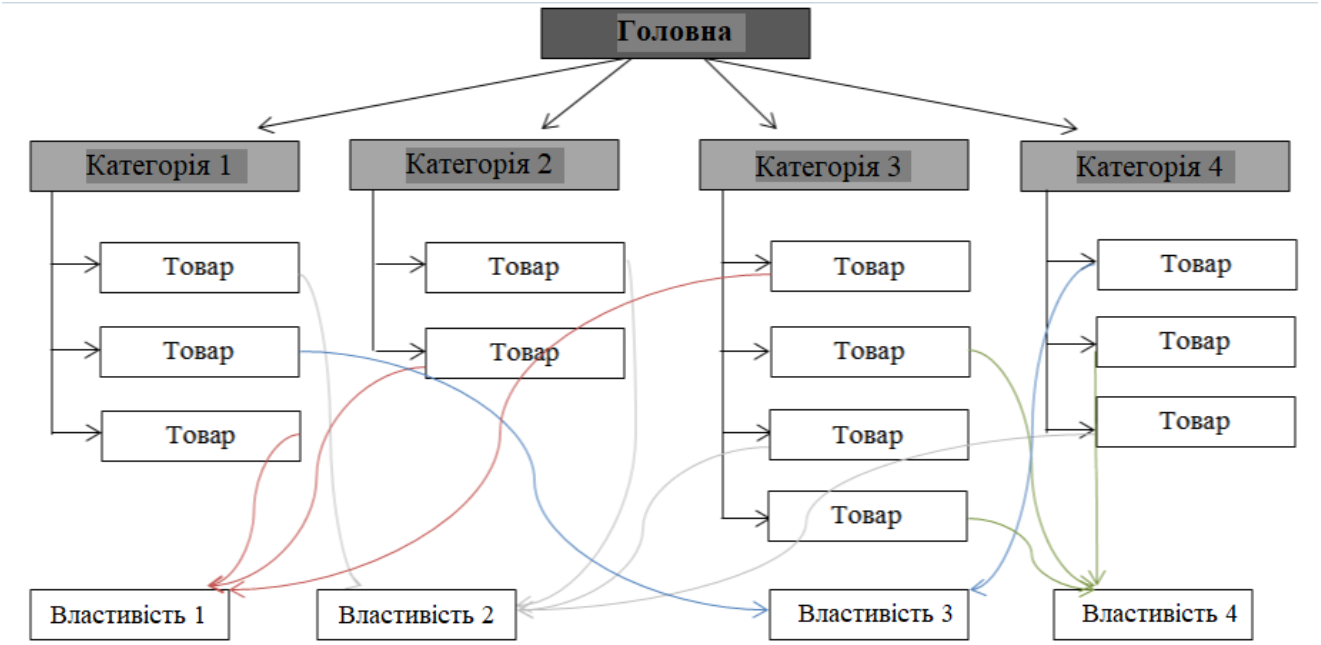

Рис. 1.5 Приклад структури сайту інтернет-магазину

Структура сайту має бути зрозумілою, відвідувач повинен за кілька секунд зрозуміти, як зробити покупку, де каталог товарів, де кошик. Це проблема не тільки для інтернет-магазинів, але й для багатьох інших сайтів, де навігація найчастіше не дуже зручна.

Адаптивний дизайн. Все більше користувачів використовують для придбання та пошуку інформації різні гаджети, що мають доступ до Інтернету: смартфони, планшети, ноутбуки з різною шириною екрану. Щоб користувачам не доводилося збільшувати окремі елементи сайту, що дуже незручно, необхідно адаптувати сайт під будь-який тип екрану. Приклад адаптивного дизайну представлено на рис. 1.6.

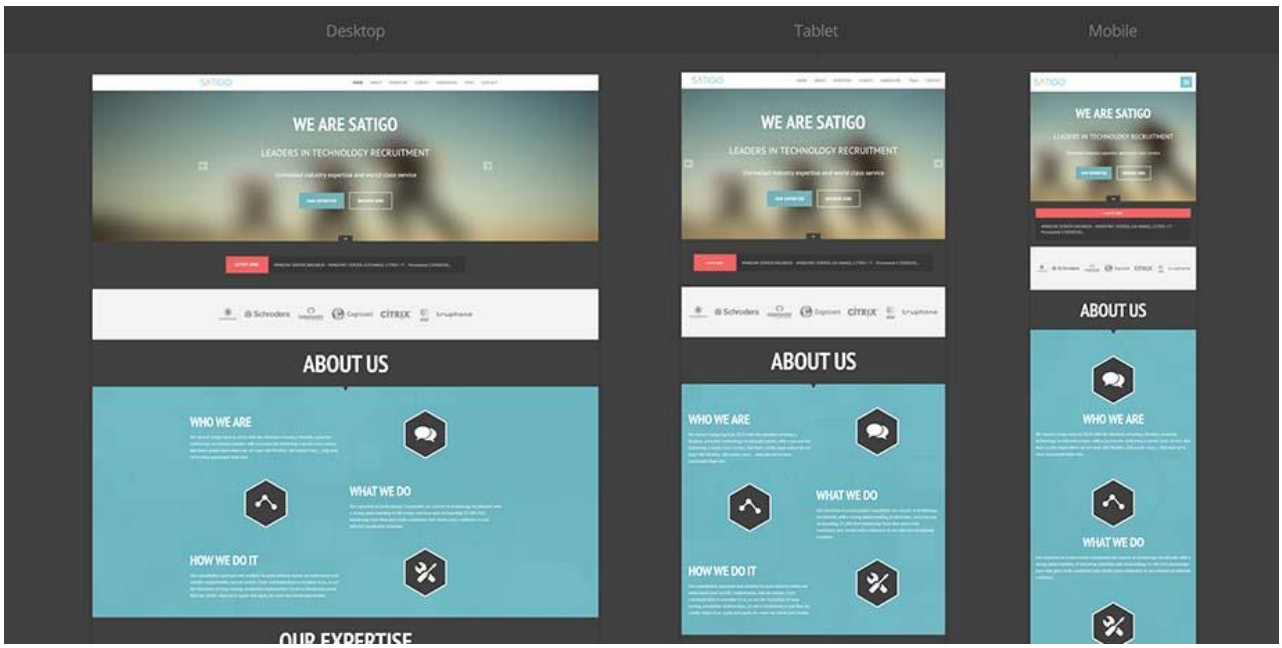

Рис. 1.6 Приклад адаптивного дизайну сайту під різні види гаджетів

Якісний дизайн. Дизайн інтернет-магазину відіграє не менш важливу роль, ніж його зміст. Адже саме візуальна оболонка формує перше сприйняття веб-ресурсу: вона притягує чи навпаки відштовхує. Адже перше враження від завантаженої сторінки сайту може спонукати користувача до здійснення покупки або навпаки - змусить його "залишити" інтернет-магазин.

Зверніть увагу, що дизайн сайту інтернет-магазину повинен починатися з визначення цілей ресурсу. Для будь-якого бізнесу електронної комерції це продаж. Але грамотний дизайн – один із способів досягти цього.

Висновки до розділу 1

У першому розділі було проаналізовано предметну область дослідження, визначено поняття та основні характеристики електронної комерції, особливості концепції інтернет-магазинів як одного з основних типів веб-сайтів, проаналізовано основні переваги та недоліки інтернет-магазинів, визначено класифікацію та характеристики методів розробки.

Інтернет-магазини мають складну класифікаційну структуру і мають ряд основних переваг: час роботи - 24 години на добу, 365 днів на рік, без перерв на

обід, без вихідних та свят, немає необхідності мати товар, не потрібно приміщення для продажів, час та вартість створення інтернету-магазину незрівнянно менше, ніж звичайного магазину, свобода переміщення продавця, з інтернет-магазином з'являється можливість розширити географію бізнесу на світові ринки, професійно створений інтернет-магазин може функціонувати повністю автономно.

2. ПОСТАНОВКА ЗАДАЧ ТА ПЛАНУВАННЯ ПРОЕКТУ

2.1. Мета і задачі проекту

Метою даного проекту є створення інтернет-магазину з продажу мобільних телефонів та іншої портативної техніки, використовуючи сучасні технології та засоби розробки веб-сайтів.

Для реалізації даного проекту необхідно в ході роботи над даною кваліфікаційною роботою виконати наступні завдання:

- Проаналізувати основні поняття електронної комерції та основні характеристики і класифікацію інтернет-магазинів.
- Обґрунтувати та здійснити вибір засобів реалізації проекту, серед яких основну увагу приділити сучасним технологіям для створення односторінкових веб-додатків.
- Забезпечити безпеку користувачів завдяки використанню сучасних засобів аутентифікації та авторизації
- Розробити концепції веб-сайту проектованого інтернет-магазину
- Здійснити розробку веб-сайту та забезпечити адаптацію його інтерфейсу для користувачів, які користуватимуться сайтом з різних видів гаджетів.

2.2. Аналіз основних понять стеку технологій для створення односторінкових веб-додатків.

У цьому розділі будуть детально пояснені різні технології, що поєднуються для формування стеку MERN. що вони демонструють, як вони функціонують та з'єднуються. Крім того, інші інструменти, які допомагають керувати даними додатків, обробкою помилок та аутентифікацією користувача. Буде обговорена та вивчена реалізація кінцевої програми – веб сайту з продажу мобільних гаджетів.

MERN

MERN - це стек, який базується на основі стеку MEAN, який вперше було представлене командою інженерів, що працює в MongoDB, у 2013 році. Стек MEAN - це абревіатура комбінацій мов та фреймворків: "M" для MongoDB, "E" для Express, "A" для AngularJS та "N" для Node. Замінивши популярний фреймворк AngularJS на бібліотеку React для покриття зовнішнього інтерфейсу та об'єднали її як стек MERN, React може супроводжувати інші технології для створення програм, орієнтованих на Javascript та JSON. [15]

У стеці MERN, MongoDB виступає як документо-орієнтована база даних, Express - це веб Framework та веб-сервер, React працює як клієнтська бібліотека, а Node - це середовище виконання. Малюнок нижче пояснює архітектуру стеку MERN.

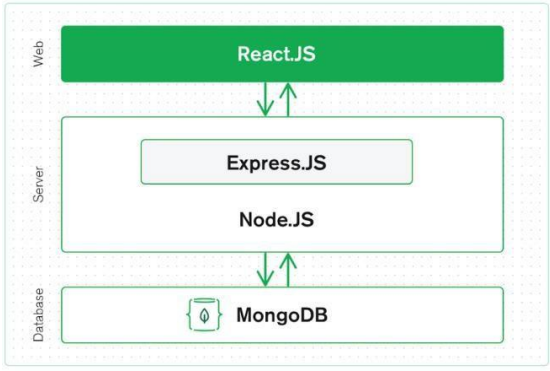

Рисунок 2.1. Трирівнева архітектура (клієнт, сервер, база даних) [2].

На основі рисунку 2.1 видно, що стек MERN є повнорозмірним рішенням для розробки додатків, що використовує ідею давно встановленої 3-рівневої архітектури. Стек MERN складається з рівня відображення клієнта з React, рівня програми з Express і Node, а також рівня бази даних з MongoDB. [13]

MongoDB

Спочатку публічно представлений у 2007 році, MongoDB поступово виділився, як зручна технологія баз даних для розробників [13]. На відміну від

SQL, який відомий як мова структурованих запитів, MongoDB є представником сімейства NoSQL загалом та мовних дерев на основі документів [13]. Термінологія NoSQL іноді згадується як не SQL або не тільки SQL. Тим не менше, більшість дійшла висновку, що бази даних NoSQL гнучко зберігають дані у форматі JSON-подібних документів на відміну від тих, що збираються реляційними базами даних [22].

Хоча база даних SQL буде збирати дані у поєднанні рядків і стовпців, база даних NoSQL буде впорядковувати інформацію в термінах документа, що містить масиви та об'єкти. Малі проєкти не можуть добре відображати різницю у використанні цих двох баз даних. Проте програма середнього розміру могла б реально продемонструвати членам команди, які безпосередньо керують та розробляють програму, переваги та труднощі кожного типу баз даних [22].

Express

Як зазначено в документації на офіційному веб-сайті Express, Express є мінімальною структурою Node, яка може забезпечити надійний набір функціональних можливостей як для веб, так і для мобільних додатків [4]. Express сам по собі справді мінімальний, оскільки робить мало, але покладається на проміжне програмне забезпечення і реалізує елементарний рівень на особливостях програми [4].

Проміжне програмне забезпечення відверто демонструє програмне забезпечення, що працює посередині життєвого циклу відповіді на запит. Один або кілька фрагментів проміжного програмного забезпечення виконуються для виконання точних завдань, таких як аутентифікація запитів або синтаксичний аналіз тіла запиту. Конвеєр завдань можна часто запускати з першим проміжним програмним забезпеченням, що викликається для обробки запиту. Це перше проміжне програмне забезпечення може закінчити запит і надіслати відповідь користувачам або викликати наступне проміжне програмне забезпечення, щоб продовжити запит. Той самий процес буде продовжуватися, як і кожен

наступний. Проміжне програмне забезпечення бере результат попереднього як аргумент до останнього проміжного програмного забезпечення конвеєра [18].

На малюнку нижче описаний процес від запуску запиту до остаточної функції, що надсилає відповідь.

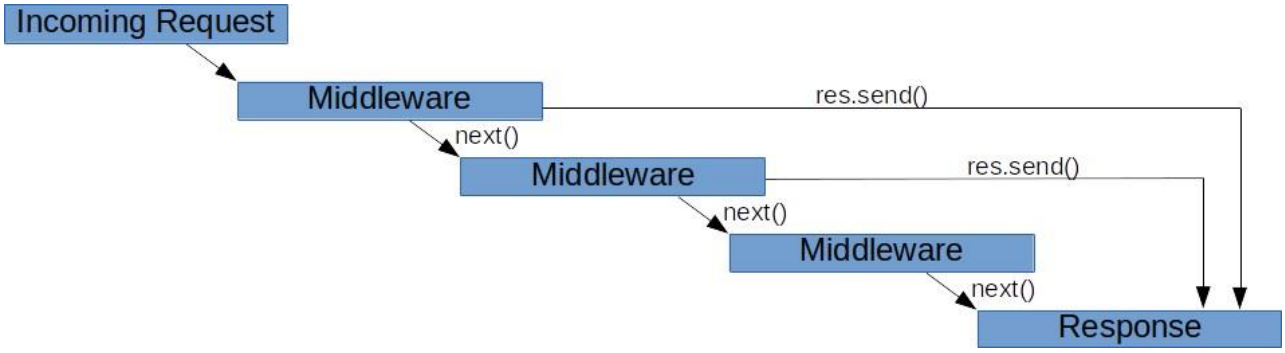

Рисунок 2.2 Запит-відповідь через проміжне програмне забезпечення [18]

Посилаючись на проміжне програмне забезпечення, точніше, воно стосується не лише функцій, які здатні доступатися не тільки до запитів HTTP та повернених об'єктів, але й до послідовних дій у циклі програми запит- відповідь [21]. Вище точно визначено, маршрутизація і фреймворк проміжного програмного забезпечення, такий як Express, може виконувати будь-які рядки коду, змінювати запит і результуючі об'єкти, відпишіться від циклу запит-відповідь і викличте наступний метод проміжного програмного забезпечення, що стоїть у стеці. Якщо є такий випадок, що поточниа функція проміжного програмного забезпечення не може завершити цикл, користувачі можуть слідувати інструкціям на веб-сайті Express, який має викликати функцію, записану як next(), щоб пройти контроль до наступного входження проміжного програмного забезпечення, щоб уникнути незавершеного циклу [18, 21].

Існують різні типи проміжного програмного забезпечення. Одним із найважливіших типів є проміжне програмне забезпечення рівня маршрутизатора, яке може використовуватися для обробки маршрутизації в Express. Воно схоже на будь яке інше проміжне програмне забезпечення, але воно обмежується лише екземпляром експрес-маршрутизатора [21].

React

React - це бібліотека Javascript, випущена в 2013 році Facebook. Спочатку створена для вирішення складних великомасштабних користувацьких інтерфейсів зі зміною даних та прив'язкою їх у режимі реального часу, React продовжує зростати у розробці SPA програм та вдосконаленні інтерфейсних службових програм для всіх рівнів програмістів. Однією з функцій багатого функціоналу React, яка задовольняє великі виробничі додатки, які повинні бути швидкими та продуктивними, є віртуальний DOM або VDOM. Ця концепція представляє віртуальне представлення інтерфейсу користувача, яким React може швидко маніпулювати, не торкаючись реального інтерфейсу, використовуючи цей віртуальний об’єкт, щоб визначити, що потрібно зробити з реальним деревом DOM, та синхронізувати ці віртуальні та реальні дерева відповідно [3].

Поєднуючись з основними ідеями React, інші надійні функції, запроваджені командою інженерів Facebook для вирішення інших проблем будь яких веб або мобільних додатків, що вимагають адаптивного розміщення та масштабованості зростаючих даних користувачів. також отримали схвалення від розробників у всьому світі. У межах даної курсової роботи лише деякі основні атрибути React згадані нижче у списку React Компоненти і React Хуки.

React компоненти

Компоненти - основна концепція React, де розробникам пропонується розбити інтерфейс користувача на незалежні та багаторазові розділи. Компонент React може бути написаний двома способами: функціональний компонент або компонент класу, і найбільш легкий підхід до складання компонента - це написання функціонального компоненту, як функції Javascript, або використання класу ES6 для ілюстрації компонента. На думку React, ці дві методики, які продемонстровані нижче - ідентичні. [12]

```
function Welcome(props) {
  return (
    <div>
      <p>Hello {props.name} </p>
    </div>
  );
}
```

(а)

```
class Welcome extends React.Component {
  render() {
    return (
      <div>
        <p>Hello {this.props.name}</p>
    </div>
    )
  }
}
```

(б)

Рисунок 2.3 Функціональний компонент (а) проти компонента класу (б) в React

Як показано на рисунку 2.3, функціональні компоненти та компоненти класу мають однакові результати. Відображається компонент "p" зі словом "Hello...", а також пропс "name", передані в компонент.

Компоненти можуть посилатися один на одного, оскільки один компонент може бути батьківським компонентом, який містить безліч інших дочірніх компонентів, без обмеження на будь який рівень будь яких деталей. Незалежно від того, чи це клас, чи функціональний компонент, вони обидва дотримуються одного строгого правила, встановленого React: усі компоненти React - це чисті функції, які не змінюють свої властивості. Пропс - це набір входів, переданих як параметри компоненту, тоді як чиста функція ілюструє випадок, коли функція виконує логіку без зміни аргументів. Отже, компонент React працює як чиста функція, яка поважає свої вхідні дані і завжди повертає однакові результати для тих самих пропсів. [19]

React Хуки

Передача глобального стану за допомогою Redux виглядає продуктивно, незважаючи на це, управління локальним станом у складі компонента Redux вважається надмірним та непотрібним. Класи з React з самого початку добре виконують свою роботу з підтримання локальних станів з чіткою синтаксичною структурою. Це продовжує працювати і сьогодні, коли застосовується незалежно від масштабу проекту. Також, додатковим варіантом, який робить те саме, що і класи React є React Хуки, які були представлені Софі Алперт і Даном Абрамовим на React Conf 2018. [19]

Ця зміна є плавним переходом від класичних класів React, оскільки хуки не замінюють і не включають будь-які нові концепції React, наприклад, що стосуються властивостей, стану та життєвих циклів компонентів.

Вступ Хуків також не означає відмову від класів React. Розробники та менеджери проєктів можуть вільно вирішувати, чи хочуть вони спробувати щось нове, і рухатися вперед, або залишатись із тим самим синтаксисом, з яким працювали до цього. Спочатку хуки можуть викликати заплутане враження, але врешті-решт, логіка та цілі не далекі від основних ідей класів. На практиці можна сказати, що React Hooks зменшили кількість рядків коду, а також тимчасово усунули використання ключового слова "this". Насправді є більше відмінностей, ніж просто зменшення загальної кількості рядків коду та зміна вигляду програми.

React Hooks представляє хуки стану, які також відомі як useState хуки, які обробляють управління станом рівня компонентів. Якщо бути точнішим, useState - це хук, який переходить у стан React шляхом ініціалізації змінної стану, яка зберігається React-ом. Цей хук отримує та надсилає два значення як результати: поточний стан та функцію для його зміни. За допомогою хука useState стан компонента можна легко ініціалізувати, використовувати та оновлювати. [19]

Ще одним ключовим хуком, на який потрібно звернути увагу, є хук ефектів, який більш відомий як useEffect. Поки useState має справу зі станом,

useEffect допомагає програмістам обробляти життєві цикли компонентів. Проблема розбиття логіки та даних на кілька життєвих циклів класу, наприклад, componentDidMount, componentDidUpdate, componentWillUnmount, була добре висвітлена в Effect Hook. Компонент React може охоплювати кілька ефектів, щоб відокремити проблеми маніпулювання даними. [19]

Одне, чого не вистачає класам React, в той час як Hooks мають відповідь - це спільна логіка функціональності. Щоб поділитися логікою стану, візуальною або невізуальною логікою раніше ніж Хуки, програмісти вибирають компоненти вищого порядку (HOC), або відтворюють шаблони пропсів. Що веде за собою відповідне налаштування ієрархії компонентів та робить програму більш складною для підтримання. З іншого боку, хуки дозволяють розробникам повторно використовувати логіку, не змінюючи структуру компонента. [19]

Node

Незважаючи на те, що технологія "народилася" 12 років тому, Node. (або Node.js) зарекомендував себе як життєво важливе середовище виконання JavaScript, що використовує популярність серверного JavaScript [1]. Середовище виконання не є ні мовою, ні структурою. Тим не менш, це потужний інструмент, побудований на двигуні V8 Chrome, який також працює з Javascript [21]. Використовуючи керовану подіями асинхронну та неблокувальну модель I/O, Node підтримує розробників ще одним варіантом створення легких додатків у реальному часі, крім стандартного шляху очікування та обслуговування запитів [21].

Для розробників, які в основному зосереджуються на Javascript, NodeJS надає переваги у створенні програм, а саме сервера та клієнта. Щоб докладніше розповісти про те, які ще переваги надає Node, програмісти доцільно виділяють менеджер пакетів Node або npm, який надає доступ до сотень тисяч пакетів, зареєстрованих у системі Node [1]. Реєстр npm вважався одним із найбільших у світі реєстрів пакетів, у 2017 році було зафіксовано понад триста п'ятдесят тисяч

пакетів, багато з яких є відкритими кодами та розроблені розробниками по всьому світу [1].

2.3 Обґрунтування вибору стеку MERN, як засобу реалізації проекту

Стек MERN - не єдина назва у списку комбінованих технологій для розробки веб-додатків. Можна назвати декілька, це MEAN (MongoDB, Express, Angular, NodeJS), LAMP (Linux, Apache, MySQL або MongoDB, PHP), Django (Python, Django, Apache, MySQL).

Існують різні причини, чому стек MERN є популярним та добре схваленим стеком технологій для створення веб-додатків. Стек MERN фокусується на одній базі коду з використанням Javascript та JSON, що дозволяє розробникам глибше копати в певній мові програмування та покращити командну роботу та робочий процес у різних частинах цілого веб- додатку. Послідовність стеку також вказує на менший час для створення програми, менші зусилля для подальшого прогресу, легке розширення та підтримання програми. Не менш важливим є те, що кожен фрагмент стеку MERN пов’язаний із великою громадою, яка робить внесок у розвиток технологій та підтримку програмістів на будь-якому рівні. Доступно багато матеріалів з відкритим кодом, таких як документація, спеціальні пакети, додаткові бібліотеки. [14.]

Mongoose

Mongoose не є частиною MongoDB, а скоріше бібліотекою відображення об'єктних документів (ODM) для поліпшення роботи з Node та MongoDB [16]. Він не тільки забезпечує перевірку схем та управління взаємозв’язком даних, але також пов’язує об’єкти в коді та об’єкти в MongoDB [16]. Рисунок 2.4, нижче, описує зв’язок між Mongoose та іншими програмами:

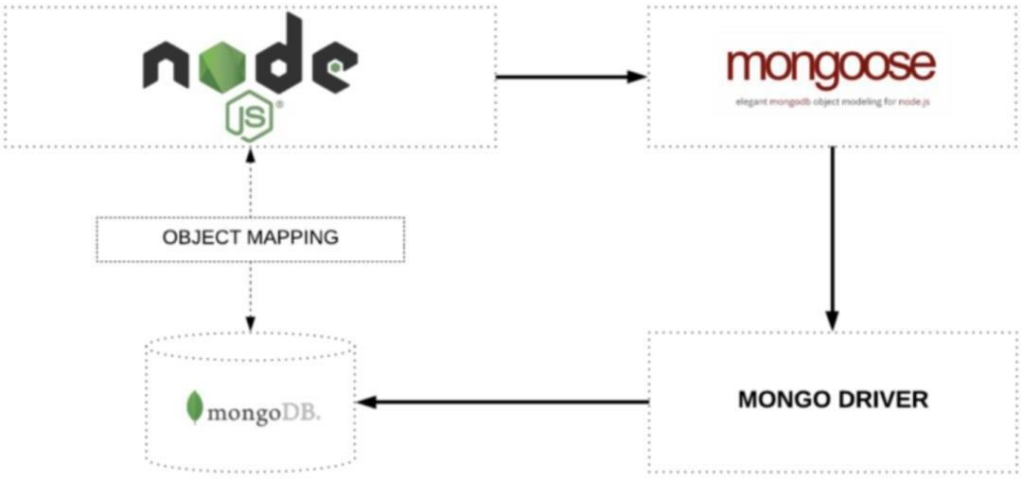

Рисунок 2.4 Mongoose відображає об'єкти між Node та MongoDB [16].

Як показано на рисунку 2.4, Mongoose використовується для Node для підключення до MongoDB за допомогою зіставлення об'єктів. Потім Mongoose підключається до MongoDB через драйвер Mongo. Через зв’язок між Mongoose, Node та MongoDB забезпечується можливість потоку даних. [16.]

Подібно до інших бібліотек ODM, першим кроком, з якого починається Mongoose, є схема [16]. Схема роз'яснює структуру даних та інші практики, перелічені на сторінці документації Mongoose: методи екземпляру, складений індекс, методи статичної моделі та проміжні засоби. Після завершення першого кроку, створені схеми відображатимуться у колекціях MongoDB та формуватимуть документи даних кожної колекції [16]. Другий крок, який програмістам потрібно виконати - це створення моделі Mongoose. Моделі - це компільовані конструктори схем, основними обов'язками яких є генерація та сканування документів бази даних Mongo. Інші можливості моделей, про які варто згадати - це запити, видалення та оновлення документів у базі даних [17].

Redux

Управління станами, мабуть, не є найбільш пріоритетною проблемою для дрібних проєктів, коли дані користувачів, відповіді серверів та кешовані дані не уповільнюють або порушують обмеження, які може обробляти веб-браузер.

Незважаючи на це, коли додаток зростає, управління станом стає справжнім випробуванням для подальшого розвитку та налагодження програми, особливо коли змішуються два поняття асинхронності та мутації. Асинхронність визначає кілька змін в асинхронній послідовності, хоча мутація уточнює зміни стану програми. Ці дві концепції, як правило, складаються разом, коли в гру вступають непередбачувані події або час очікування відповіді, які, можливо, можуть видати несподівану поведінку. [17]

Сконструктований і розроблений Даном Абрамовим у 2015 році Redux, пропонував вирішити вищезазначену проблему шляхом перетворення передбачуваної мутації, не впливаючи на переваги асинхронності [17]. Якщо детальніше, то Redux зберігає стан в одному джерелі і вимагає суворої структури того, як може відбуватися модифікація стану, де відіграють багаторазові та чисті функції. Замість того, щоб змінювати значення даного об'єкта, чисті функції повертають ті самі нові значення на основі наданих аргументів, де б вони не були викликані. Таким чином, Redux дозволяє більш простий процес налагодження, тестування, підтримування коду та більш плавний розвиток досвіду розробки особисто чи в команді [17].

Ще одним визначним фактором, який було обрано Redux-ом, є його простота у використанні об'єктів і функцій Javascript. Він може добре поєднуватися з React та іншими клієнтськими бібліотеками або фреймворками, наприклад, AngularJS, Angular, VueJS, Polymer, Ember. Щоб додати більше можливостей до своєї гнучкості, Redux працює в різних середовищах, таких як браузер та сервер . [17]

Основна концепція Redux обертається навколо двох слів: редюсер та дія. Для спрощення термінів кожна мутація стану називається дією, а редюсер - це чиста функція, яка пов’язує стан та пов’язані з нею дії. Щоб запустити процес Redux, розробникам потрібно надіслати дію з мутацією стану, який записується як об'єкт Javascript, включаючи ім'я дії та іншу інформацію (якщо потрібно). для більш детального опису дії. Друге, що потрібно обробити - це написати редюсер, який приймає стан і дію як параметри і повертає новий стан. В результаті чистої

функції повернутий результат редюсерів завжди має однакове очікуване значення, що робить стан незмінним і дотримується суворихвказівок, запропонованих Redux. [17]

2.4 Аутентифікація та авторизація користувачів за допомогою JWT

Аутентифікація - це процедура з'ясування того, ким є користувач, тоді як авторизація - це процедура визначення того, до чого користувач може отримати доступ. Зазвичай аутентифікація проводиться перед авторизацією. Після встановлення того, ким є користувач, доступ до якогось ресурсу дозволений або відхилений. [20.]

Існують різні способи реалізації аутентифікації та авторизації для веб-додатків, де найбільш перевіреною в житті, і стандартною технікою є використання сеансів для обробки статусу користувача як на клієнті, так і на сервері [23]. Однак опублікована пізніше технологія JSON Web Token (JWT) реалізувала нову концепцію механізму без збереження стану, в якому зберігається статус користувача [23]. Рисунок 2.5, нижче, пояснює загальний процес механізму:

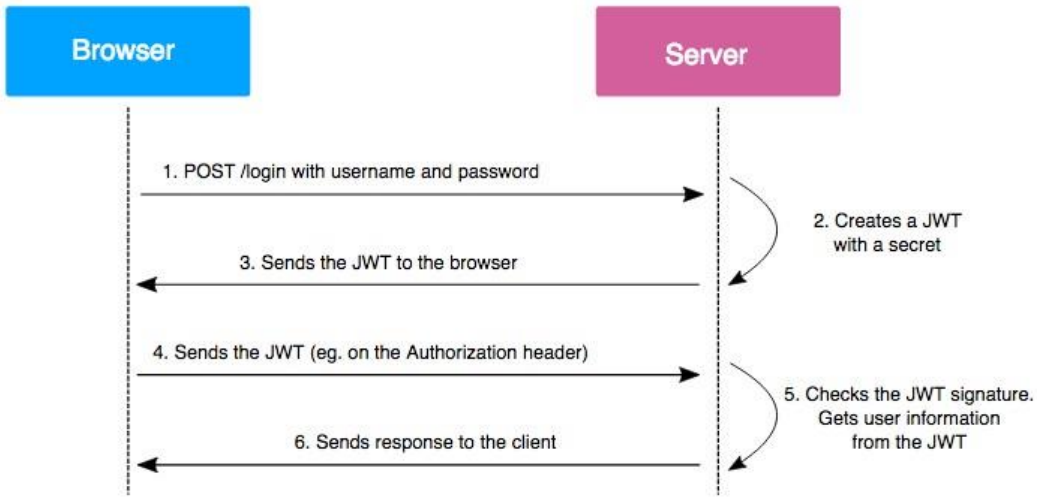

Рисунок 2.5 Процес авторизації за допомогою JWT [5]

Як показано на рисунку 2.5, по-перше, коли авторизується вхідний

сервер, сервер створює токен JWT, підписаний за допомогою введених даних користувача та секретного ключа. Потім токе буде відправлений клієнту для зберігання у файлі cookie або локальному сховищі, таким чином передаючи клієнту підтримку стану користувача. Після успішного входу, в будь-який запит на захищені кінцеві точки, токен повинен бути приєднаний до заголовка запиту авторизації у форматі "Авторизація: Bearer <JSON веб токен>". Коли сервер обробляє запити до захищених кінцевих точок, токен пройде перевірку його дійсності. У цьому випадку доступ надається користувачеві, а в іншому випадку виникає помилка. [5]

Висновки до розділу 2.

В цьому розділі було проаналізовані та детально пояснені різні технології, які поєднуються для формування стеку MERN. Що вони демонструють, як вони функціонують та з'єднуються. Крім того здійснено аналіз, інших інструментів, які допомагають керувати даними додатків, обробкою помилок та аутентифікацією користувача. Обговорено та вивчено шляхи реалізації кінцевої програми – інтернет магазину з продажу мобільної техніки.

3. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ З ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ

3.1. Розробка концепції веб-сайту інтернет-магазину

Сайт має бути швидким, однак через те що він повинен мати велику кількість елементів і функцій, це може серйозно сповільнювати роботу. Одне з потенційних рішень є використання сучасних фреймворків, які будуть частково оптимізувати роботу сайту за нас.

По-перше концепція *SPA* дає можливість після того як сайт завантажився швидко переводити по його сторінках, оскільки код для їх відображення вже завантажився у перший раз, тож потрібно лише дочекатися завантаження самих товарів з серверу, а це краще, ніж чекати доки весь сайт завантажиться.

По-друге фреймфорки можуть мати гарний функціонал для кушевання, що знову ж скорочує час очікування.

Оскільки сайт буде розроблюватися без сторонніх *CMS*, вартно обрати базу даних і хостінг для зберігання самого сайта.

Серед сучасних *NoSQL* баз даних можна виокремити *mongoDb*, оскільки вона має простий інтерфейс для початку роботи з базам даних, а також зберігання у хмарі. Тобто не потрібно розміщавати у себе базу даних і прописувати повудінку для під'єднання, все це автоматично розбить *mongoDb*, якщо ми користуємось її хмарними технологіями. Також перевага у тому, що тип бази даних *NoSQL,* оскілки так простіше описувати об'єкт товару, він фактично описаний в одному *JSON* об'єкті, а не збирається із 10 таблиць. Через це простіше описувати інтерфейси, оскільки ми одразу розуміємо що це інтрефейс відповідає за певну сутність, і містить всі поля, що їй належать.

Розглянемо основні концептуальні рішення створюваного веб-сайт інтернет магазину.

Ціль: Створити повноцінний сайт інтернет магазин мобільних просторів. Він має відповідати сучасних напрямкам дизайну, бути швидким і зручним у

користуванні. Приклад сторінки з товарами (рис. 3.1).

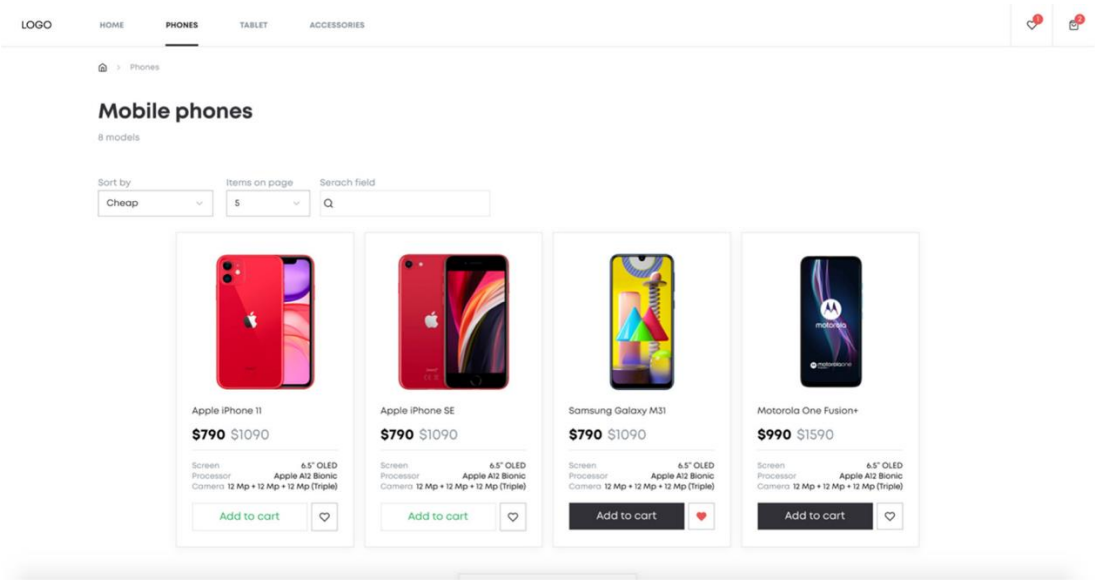

Рис. 3.1. Сторінка з товарами

Аудиторія: Зважаючи на специфіку товару потенційна аудиторія доволі широка, це люди віком від 12 до 50. Оскільки інтернет магазину загалом мають схожі функції, по типу фільтрації, сортування товарів, додавання у кошик або в улюблене, то складнощів у використанні людей, зо знайому з хоча ю один інтернет магазином виникнути не повинно (рис. 3.2).

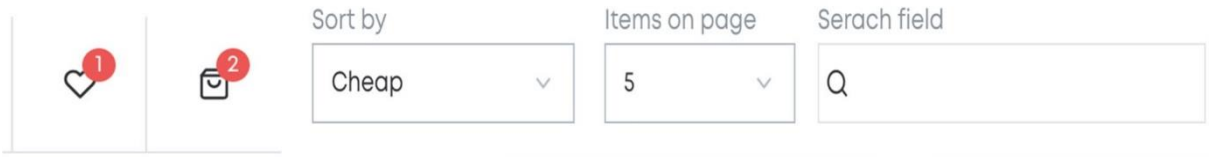

Рис. 3.2. Основні елементи інтернет магазинів

Структура: Сайт має стандарті елементи, хедер, футер, саме тіло, що розмірюється між ними. Хедер містить навігацію по сайту (рис. 3.3)

Рис. 3.3. Хедер

Він є адаптованим, а точніше для різних розмірів екрану є два окремих

хедера, це зроблено для простоти (рис. 3.4)

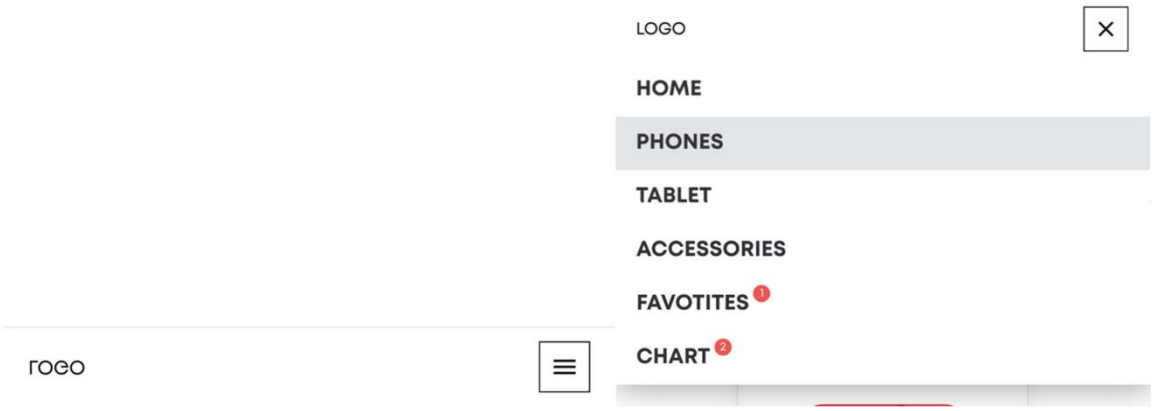

Рис. 3.4. Мобільний хедер

Типографіка: Використовувався шрифт *Mont* з різними нарисами і товщиною (рис. 3.5)

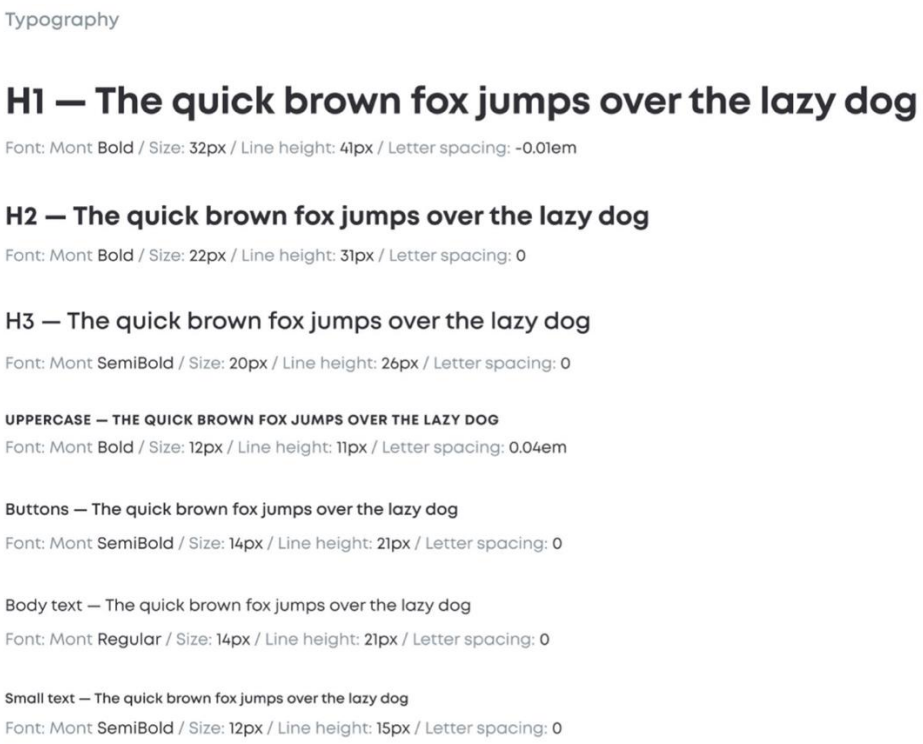

Рис. 3.5. Типографіка в проєкті

Колірне рішення (рис. 3.6): Загалом сайт зроблено в чорно-білому варіанті, з певними градаціями, виключенням являються ключові елементі для

привернення уваги, наприклад кількість товарів у кошику, або доданий товар в “улюблене”, чи ні. Це яскраво виокремлює важливі елементи.

Рис. 3.6. Колірне бачення проекту

Композиційне рішення

Особливості використовуваних елементів: З пере використовуваних елементів варто виділити кнопки, та інтерактивні (клікабельні) іконки (рис. 3.7). Щоб показати що елемент не активний, змінимо його колір на більш світлий, в порівняння з активним кольором, це виглядає так наче він перестає бути інтерактивним, і на нього не можна клікнути

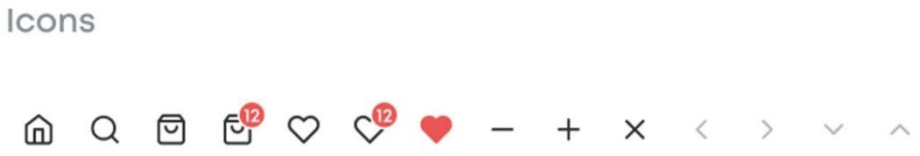

Рис. 3.7. Клікабельні іконки

Кнопки, та елементи, які змінюють свій стан при наведенні курсора (рис. 3.8). Вони можуть бути у таки станах:

1. Неактивні, коли курсор поза ними
2. Активні, коли курсор наведено на елемент
3. Вимкнені, коли неважливо наведений курсор чи ні, стан не змінюється

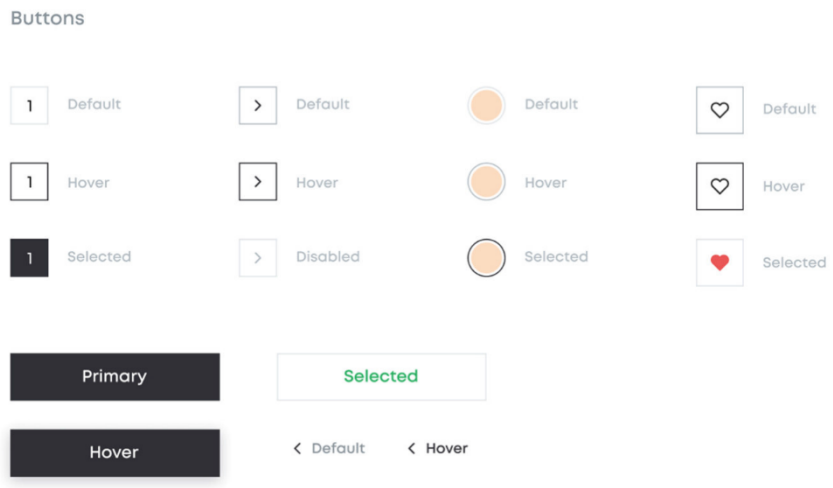

Рис. 3.8. Елементи, що змінюють свої стилі при наведенні

Оскільки в браузерах є стандартні селекти (рис. 3.9), можна використовувати їх, одна проблема в тому що вони практично не кастомізуються, тому часто роблять власні побідні елементи вже згідно стилю сайту. Вони повинні мати стилі для таких положень:

1. Неактивний, на нього не клацали мишкою
2. Активний, клацнули і випадає меню
3. Вибір з випадаючого меню певного елементу

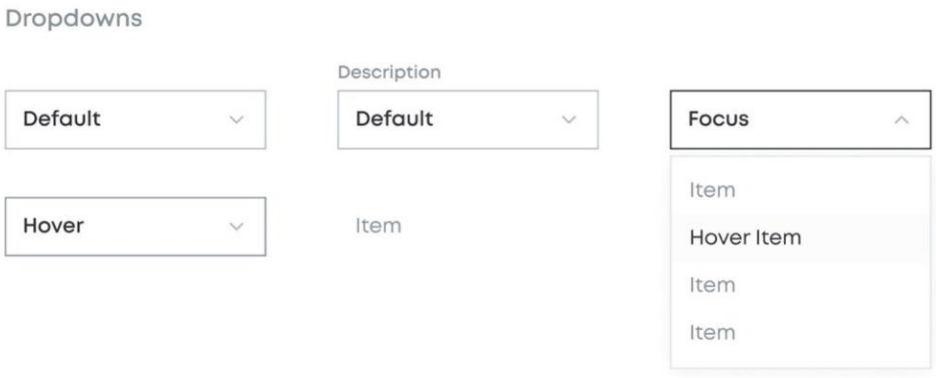

Рис. 3.9. Кастомні селекти

3.2. Програмна реалізація проекту інтернет-магазину

Для реалізації фронт частини було обрано сучасний фреймфорк *ReactJS* через те, що він дозволяє робити *SPA* застосунки, що збільшує швидкість роботи сайту і має зручний підхід у реалізації.

Першим кроком було вибір дизайну проекту сайту інтернет-магазину дизайну і його ТЗ. Після цього було почато розробку самого проекту. Оглядаючи дизайн сайту на швидкоруч було виділено основні компоненти, на які можна розділити сайт.

Критерії для виокремлення фрагменту, як компонента:

- Відносна простота, має не багато елементів
- Використовується в декількох місцях

Після такого розділення, а такого самого дизайну і ТЗ було отримано знання, яку архітектуру краще закласти в сайт. В якості бази даних було обрано *MongoDb* як *NoSQL* базу даних, через її простоту, можливість безкоштовного використання на початку роботи (розробки), а також крутого допоміжного ПЗ *compass*.

Після обрання бази даних і вибору шляху будівництва архітектури, було створено сам проект, для цього було використано готовий шаблон реакту за допомогою команди *npx create-react-app phone-store*

phone-store – це назва проекту. Попередньо для цього було встановлено *node.js* https://nodejs.org/en/. Після цієї команди було отримано шаблон для початку роботи (рис. 3.10).

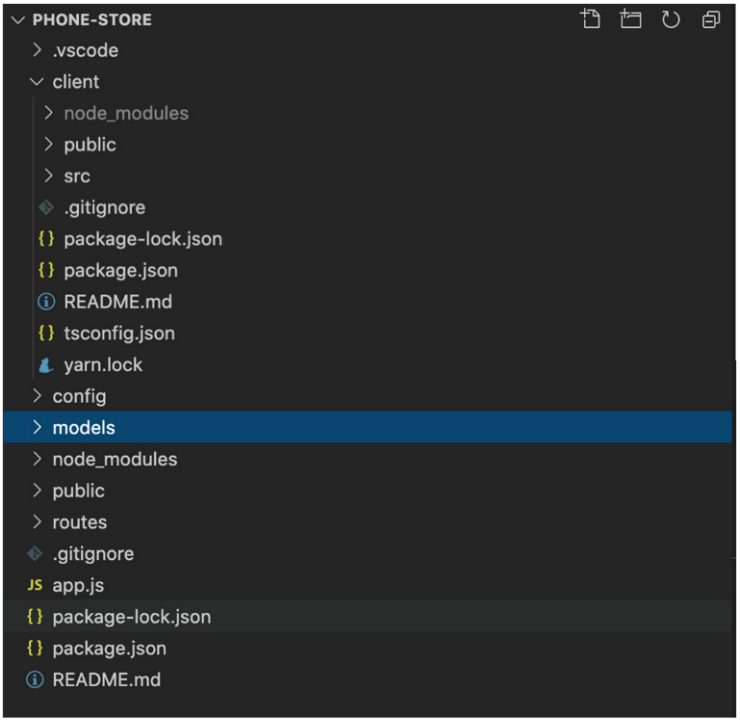

Рис. 3.10. Структура проекту

Після цього було створено базу даних підключено її до проекту. Щоб створити базу даних було зареєстровано користувача на сайті https://www.mongodb.com/ і реєструю там нову базу даних, після цього, копіюю адресу, за якою до неї можна достукатися і під'єднаю *compass* (рис. 3. 11).

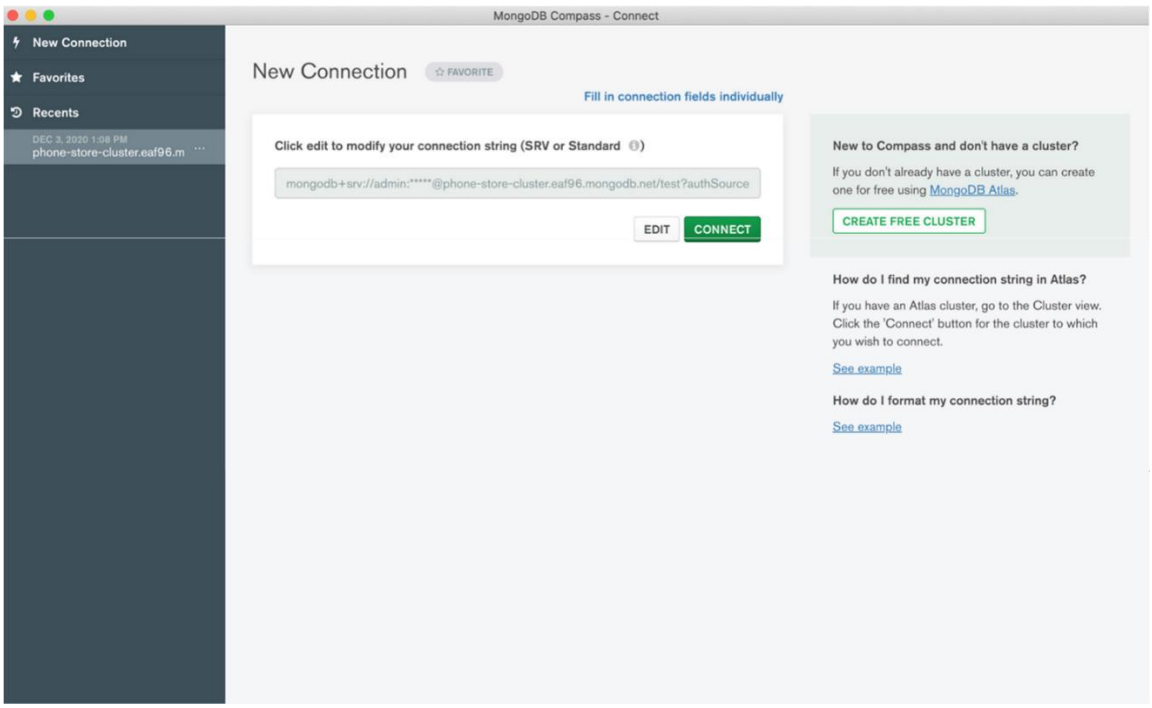

Рис. 3.11. Головний екран *compass*

Далі створюю колекції для телефонів, планшетів, для гарячих пропозицій і для кращих товарів (рис. 3.12). Колекції – це відокремлені логічно і змістовно групи в одній базі даних. Вони можуть бути пов’язані за допомогою id, або будь-якого іншого поля.

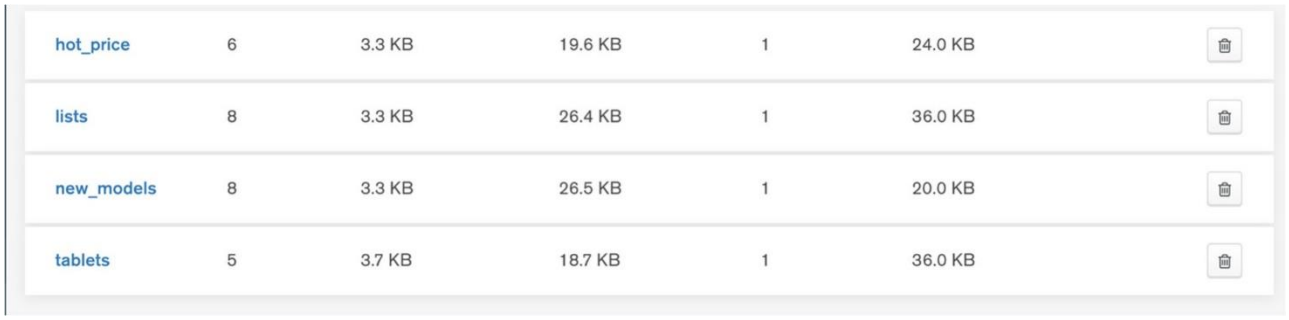

Рис. 3.12. Колекції для груп товарів

Приклад товарів у колекції list, в яких знаходяться списки всіх телефонів що доступні на сайті. Їх представлено у вигляді об’єктів, для кращого сприйняття і редагування (рис. 3.13)

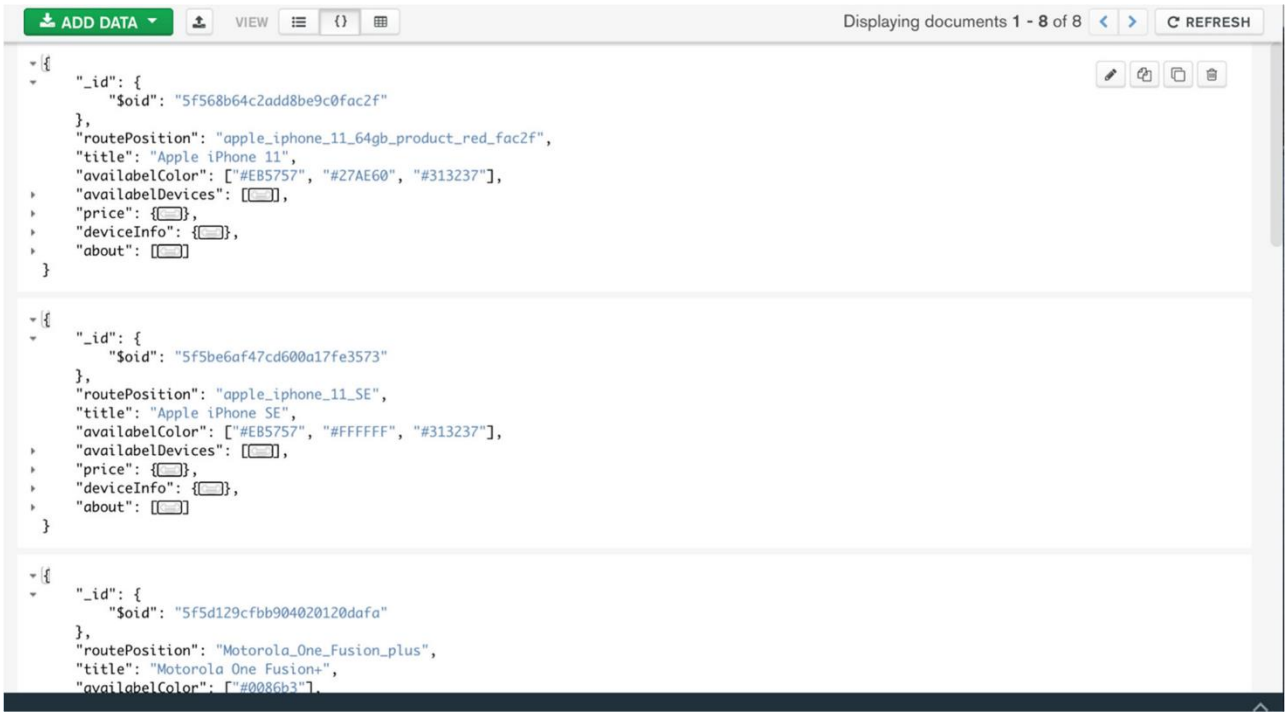

Рис. 3.13. Вигляд як об’єкти

Також можна представити їх у вигляді таблиці або списку (рис. 3.14)

Рис. 3.14. Вигляд списку

Або у вигляді таблиць (рис. 3.15).

Рис. 3.15 Вигляд таблиць

Далі під’єднав базу даних у коді, для цього використав залежність *config*, щоб зручно зберігати якість константні дані. Далі створив папку *config* і додав туди адресу для отримання свої бази даних, номер порту, на якому буде розміщуватися сайт, доки він в розробці, і секретний ключ на випадок, якщо доведеться шифрувати особисті дані (рис. 3.16)

config > {} default.json > ...

```
{
    "port": 5000,
    "jwtSecret": "SenyaSecretKey",
    "mongoUri": "mongodb+srv://admin:admin@phone-store-cluster.eaf96.mongodb.net/phone-list?retryWrites=true&w=majority"
}
```

Рис. 3.16. Базові константи

Для того щоб використовувати *mongoDb* потрібно описати типи моделей, які будуть в нашій базі даних. Приклад опису моделі для мобільного телефону (рис. 3.17).

```
const phonePatternSchema = {
    // _id: String,
    routePosition: String,
    title: String,
    availabelColor: [String],
    availabelDevices: {
        red: {
            availableRAM: [String],
            images: {
                main: String,
                other: [String]
            },
            color: String
        },
        green: {
            availableRAM: [String],
            images: {
                main: String,
                other: [String]
            },
            color: String
        },
        black: {
            availableRAM: [String],
            images: {
                main: String,
                other: [String]
            },
            color: String
        },

    },
    price: {
        current: String,
        old: String,
    },
    deviceInfo: {
        screen: String,
        resolution: String,
        processor: String,
        camera: String,
        zoom: String,
        cell: String,
    },
    about: [{
        title: String,
        description: String
    }]

};
```

Рис. 3.17. Схема телефону для бази даних

Тепер створюємо саму схему і експортуємо її (рис. 3.18).

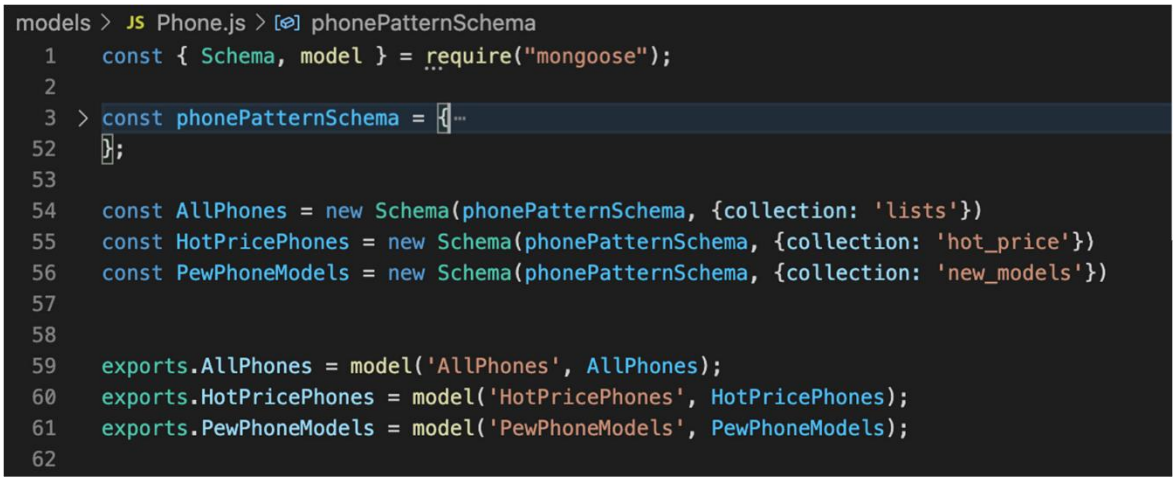

Рис. 3.18. Ініціалізація схеми

Тепер за допомогою бібліотеки *express* налаштовуємо роути на стороні бекенду (рис. 3.19)

```
app.use("/public", express.static("public"));

app.use('/api/auth', require('./routes/auth.routes'))

app.use('/api/phone', require('./routes/phone.routes'))
app.use('/api/tablet', require('./routes/tablets.routes'))
```

Рис. 3.19. Налаштування основних роутів

Тепер база даних готова, а роути для запитів на сервер написані, отже залишається реалізація фронтової частини. Фронтову частину починаємо писати з підєднання *redux* в проект, оскільки він допомагає керувати сховищем даних для сайту. Після цього потрібно додати екшети і редюсери щоб керувати станом редаксу (рис. 3.20 – рис. 3.21)

```
export const phoneLoading = (loading: boolean): AppStateActionTypes => ({
    type: PHONE_LIST_LOADING,
    loading
});
```

Рис. 3.20. Приклад екшену для завантаження моделі телефону

```
export const phonesState = (state = initialState, action: AppStateActionTypes) => {
    switch (action.type) {
        case PHONE_LIST_LOADING:
            return { ...state, loading: action.loading }

        case PHONE_LIST_SUCCESS:
            return { ...state, phoneList: action.phoneList }

        case PHONE_LIST_ERROR:
            return { ...state, error: action.error }

        case PHONE_LIST_STATE:
            return { ...state, phoneListState: action.phoneListState }

        case PHONE_ITEM_SUCCESS:
            return { ...state, currentModel: action.currentModel }

        default:
            return state
    }
}
```

Рис. 3.21. Приклад редюсера для зберігання стейта телефонів

Тепер можна перейти до основних компонентів, а саме хедеру, футеру і карток товарів.

3.3 Розробка адаптивного інтерфейсу інтернет-магазину.

Одразу робимо два варіанти, один для версії з великим екраном, більше 700 px, інший менше, для мобільних пристроїв і планшетів (рис. 3.21).

Рис. 3.22. Структура мобільного і десктопного хедеру

Після цього була написана функція яка відстежує ширину екрану в даний момент і передає його в редакс, тож весь додаток миттєво дізнається, якщо ширина змінилася і це дає змогу нам визначати коли який хедер показати (рис. 3.22 – рис. 3.24).

```
// check device screen width
const checkDeviceSize = (event?: any) => {
  const screenWidth = event ? event.currentTarget.innerWidth : window.innerWidth;

  store.dispatch(setDeviceScreen({
    value: screenWidth,
    name: screenWidth > 0 && screenWidth <= 500
      ? 'phone'
      : screenWidth > 500 && screenWidth <= 900
      ? 'tablet'
      : 'desktop'
  }))
}
```

Рис. 3.23. Відстеження розміру екрана

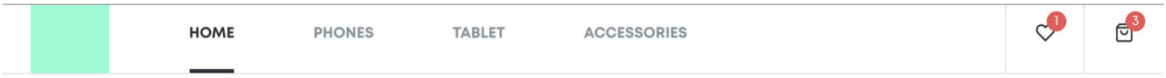

Рис. 3.24. Десктопний хедер

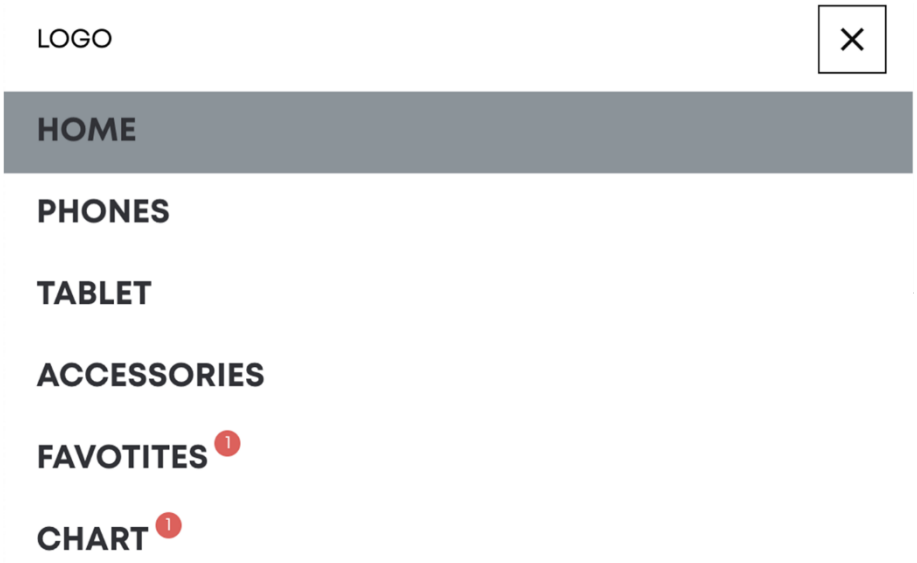

Рис. 3.25. Мобільний хедер

Важливим елементів є пошук потрібних товарів на сайті, тому потрібно вмонтовати пошукове меню у хедер сайта. Однак враховуючи адаптивність сайта і довжину пошукового меню, потрібно зробити декілька варіантів розміщення поля пошуку і хедері, для маленьких розмірів екрану і для великих (рис. 3.25 – рис. 3.30).

Рис. 3.26. Елемент пошуку

Рис. 3.27. Хедер для великих екранів

Рис. 3.28. Хедер для середніх екранів (планшетів)

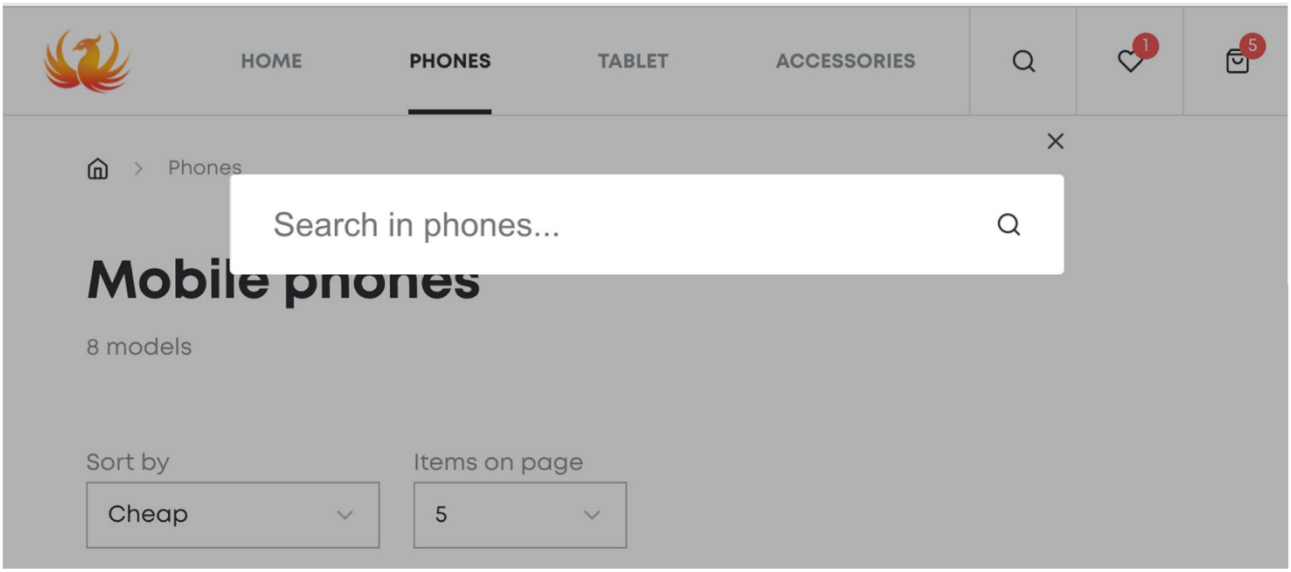

Рис. 3.29. Відкриття додатково поля для пошуку на середніх екранів

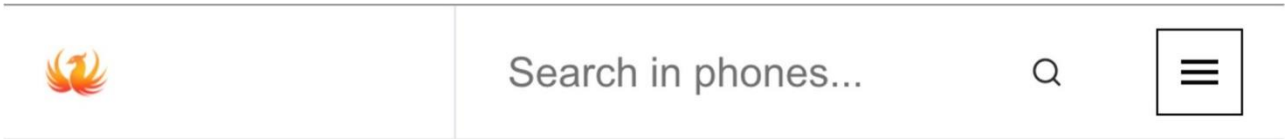

Рис. 3.30. Для маленьких пристроїв (телефонів)

Тепер час для футера, в цьому випадку не потрібно створювати два окремих варіанти для мобільного розширення і для великих моніторів, адже тут не так багато елементів і вони всі еластичні, тож можна просто попрацювати над адаптивністю (рис. 3.30 – рис. 3.31).

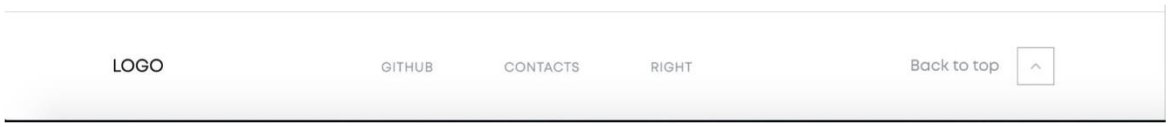

Рис. 3.31. Футер на повну ширину

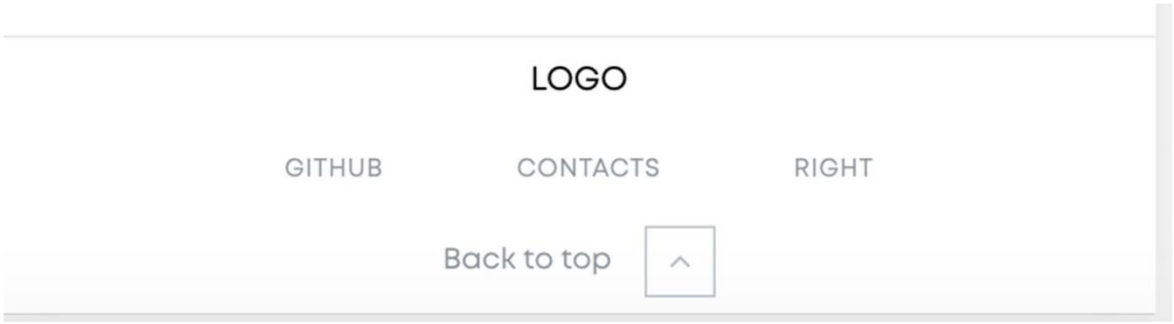

Рис. 3.32. Адаптований футер під малу ширину

На головному екрані має розміщуватися сладйер, однак не такий, як для товарів, а просто сладйер для зображень, тож потрібно трохи змінити висоту кнопок вперед / назад (рис. 3.33).

Рис. 3.33. Слайдер для зображень

В дизайні на головній сторінці присутні слайдери, тож варто створити

шаблон для цього елемента, однак перед цим потрібно створити компонент картки товару (рис. 3.34).

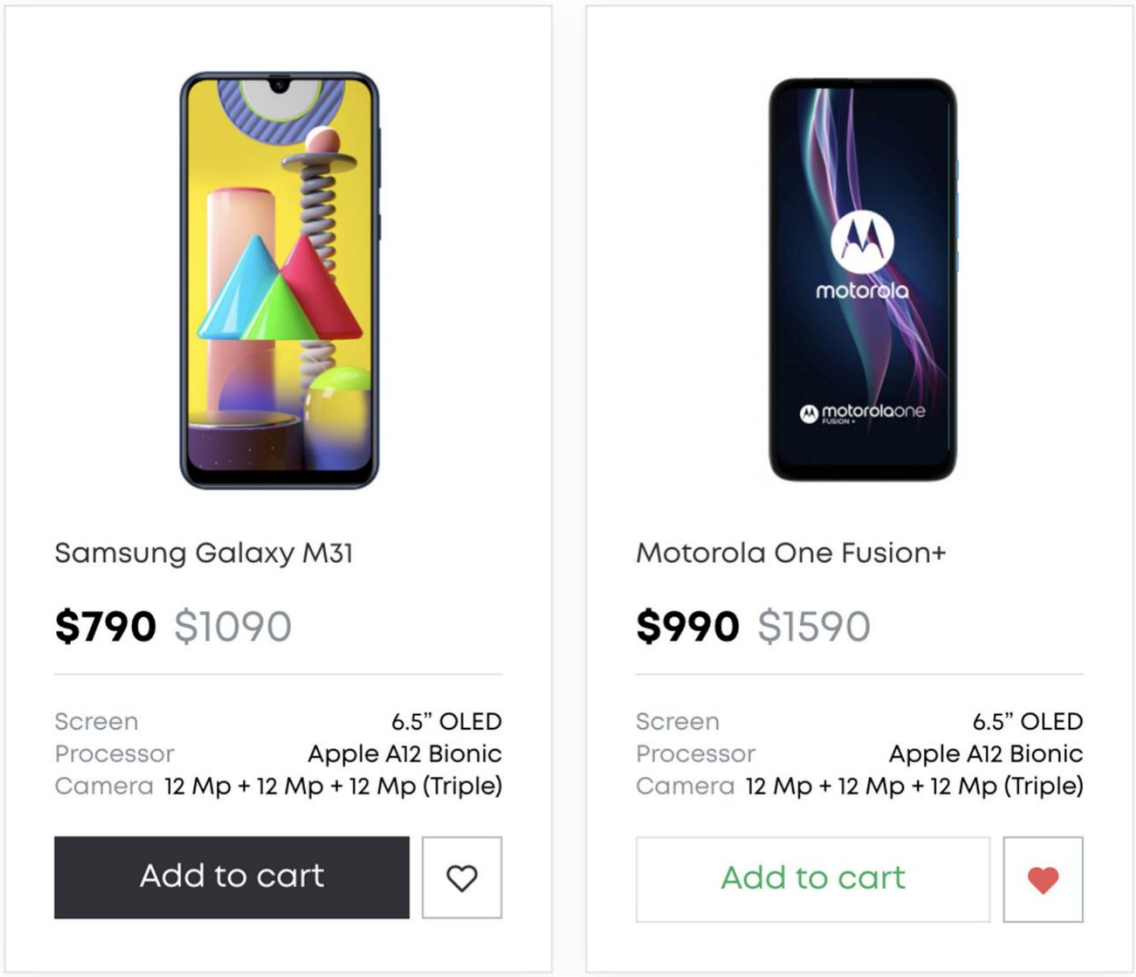

Рис. 3.34. Картки товару

Для того щоб з цих карток зробити карусель (слайдер) було використано бібліотеку *slick-slider*. По суті просто передаючи в *slick-slider* дані для рендера карток карусель готова, однак для зручності використання потрібно додати клавіші для керування і отримаємо повноцінний слайдер (рис. 3.34).

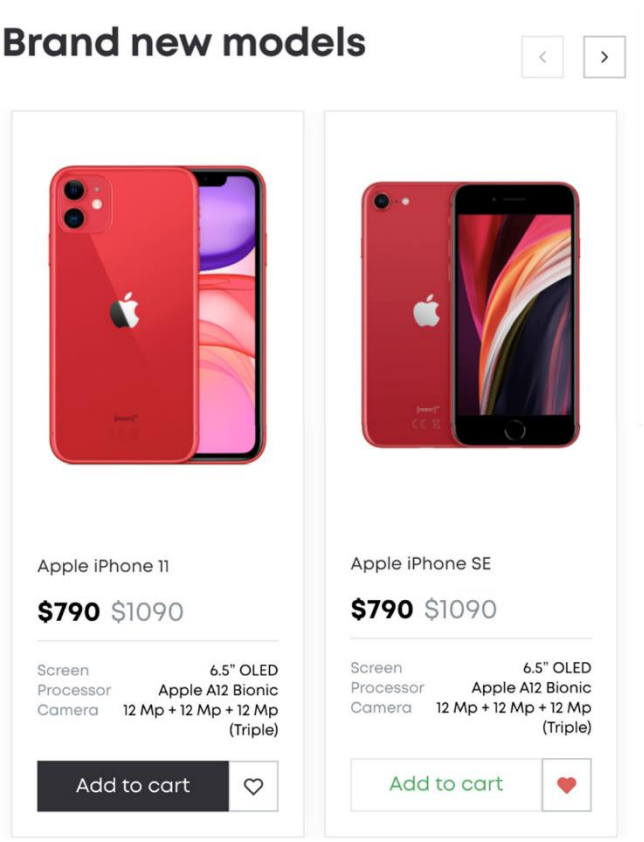

Рис. 3.35. Слайдер

Окрім слайдеру за дизайном на головній сторінці сайту має бути список з плиток категорій товарів (рис. 3.36)

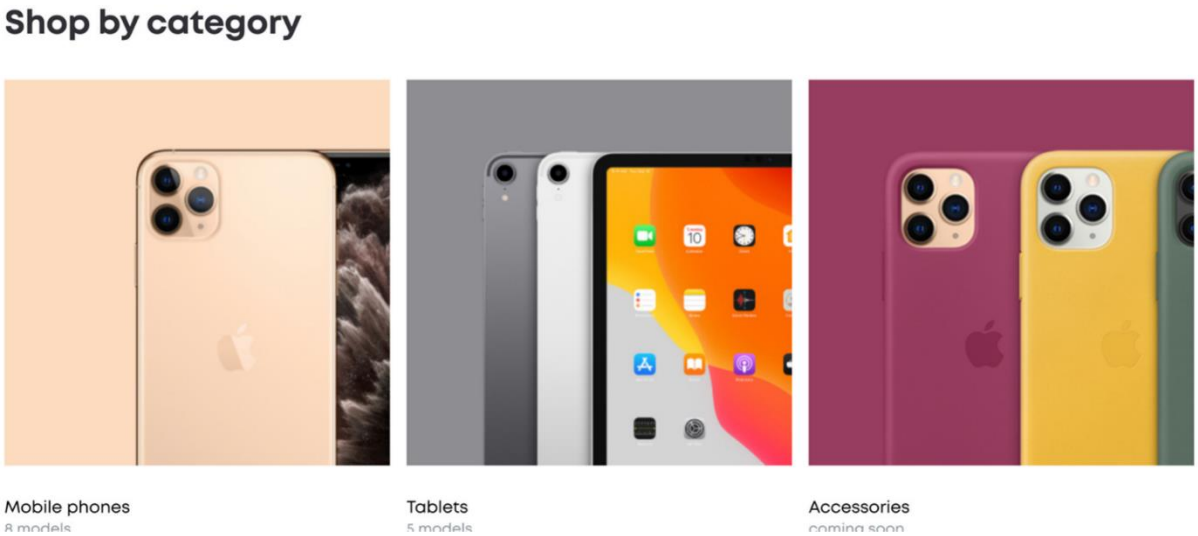

Рис. 3.36. Плитки на головній сторінці

Сторінка для улюблених товарі, що користувач вподобав під час користування сайтом буде містити хедер, футер, а також список у вигляді

плиток самих товарів (рис. 3.37)

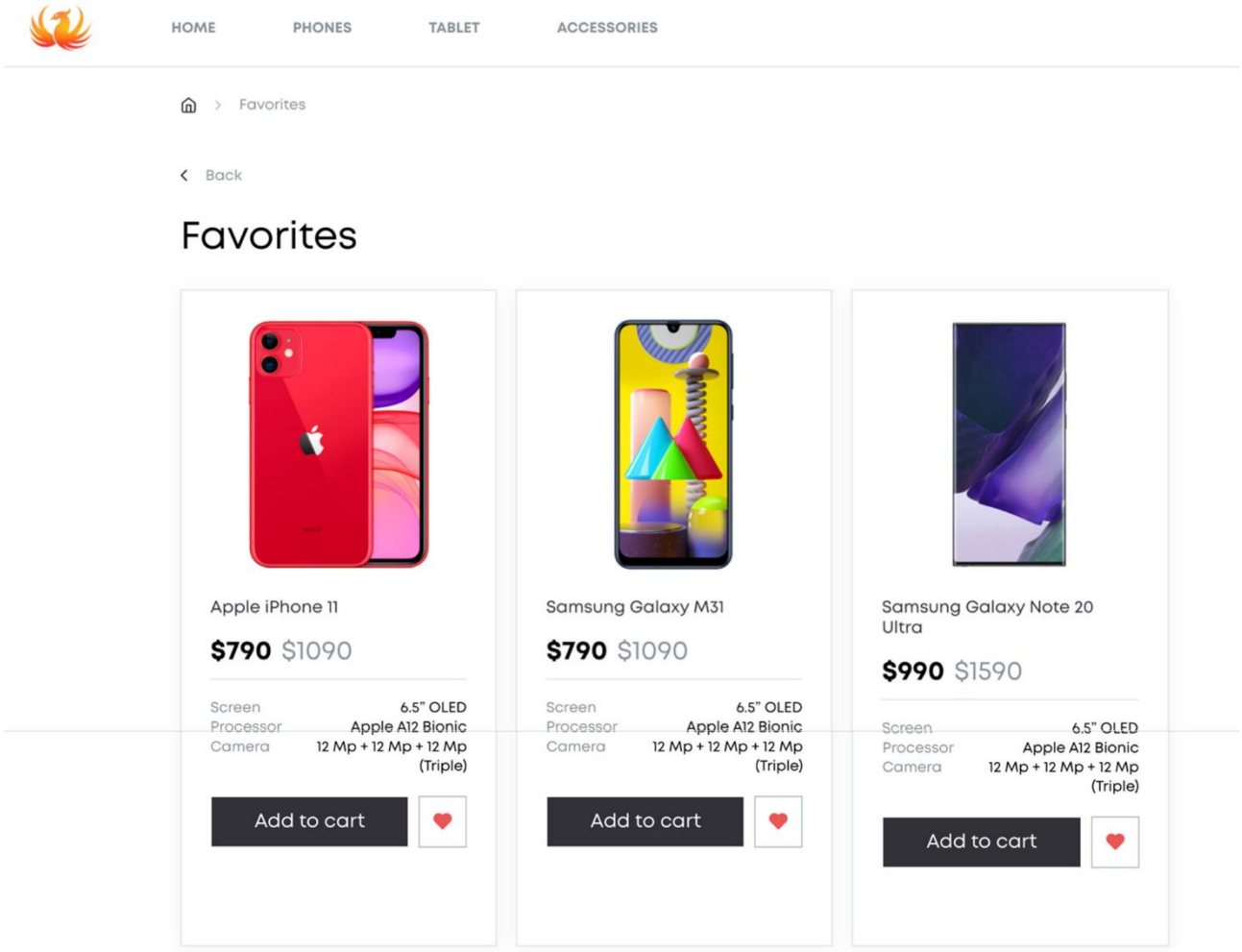

Рис. 3. 37. Сторінка вподобаних товарів

Окрім цього для кращої навігації створимо міні карту для того, аби юзер розумів де саме він знаходиться. Початковою точкою буде головна сторінка, а далі той шлях, який було пройдено. На рисунку 3.37 показано приклад коли юзер перейшов на сторінку з усіма телефонами і обрав там телефон Sansub Galaxy Note 20 і перейшов на сторінку конкретного товару. Як видо з рисунка остання точка шляху неактивна, оскільки користувач і так знаходиться на ній, а решта активні, і клікнувши на них можна перейти на цю сторінку.

Рис. 3.38. Маленька навігація

Тепер створимо сторінку для планшетів, вона буде такою самою як і сторінка для телефонів, і схожою на сторінку для аксесуарів (рис. 3.39).

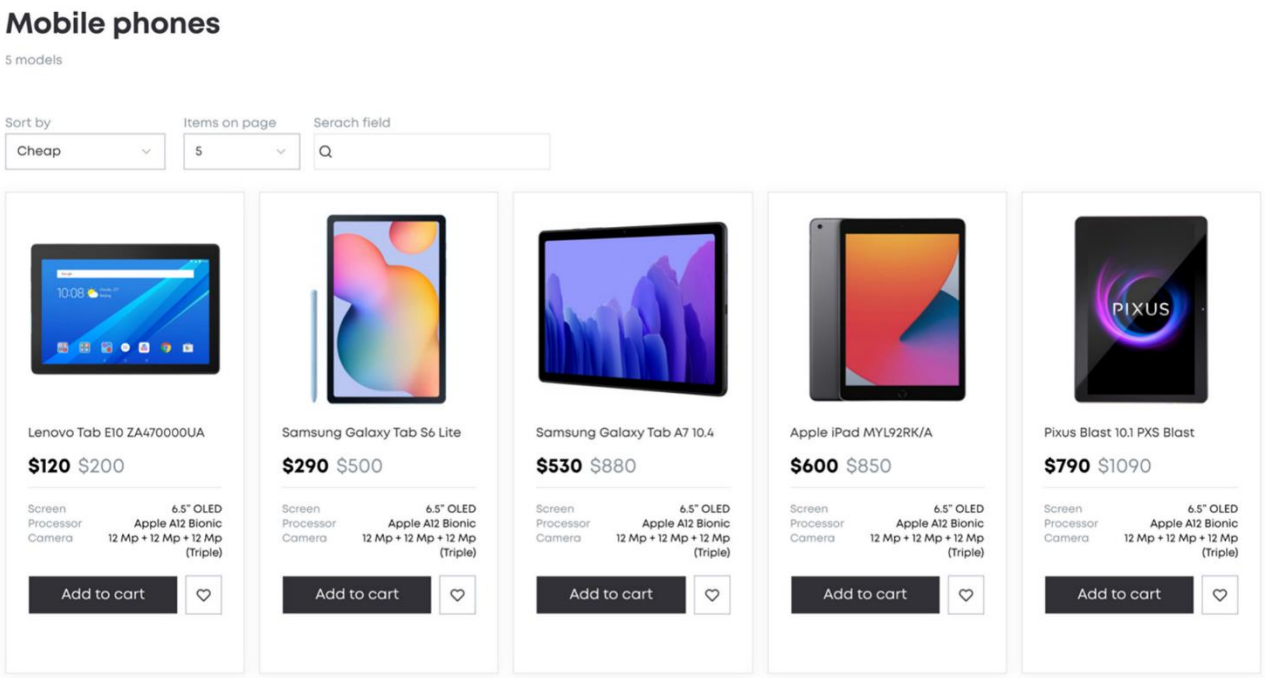

Рис. 3.39. Сторінка планшетів

Коли користувач тільки зайшов на сторінку, нам потрібно завантажити список товарів, однак про це потрібно якось повідомити юзера, тому додаємо *preloader*, що буда повернитися, доки сторінка не завантажиться (рис. 3.40). Це стандартний елемент кожного сайту, десь він реалізований в більшій мірі, десь в меншій.

Рис. 3.40. Preloader

Тепер створимо сторінку для конкретного товару (рис. 3.40), вона

скрадатиметься з двох основних елементів, перший – це фото товару і можливі варіації кольору та інших опціональних характеристик.

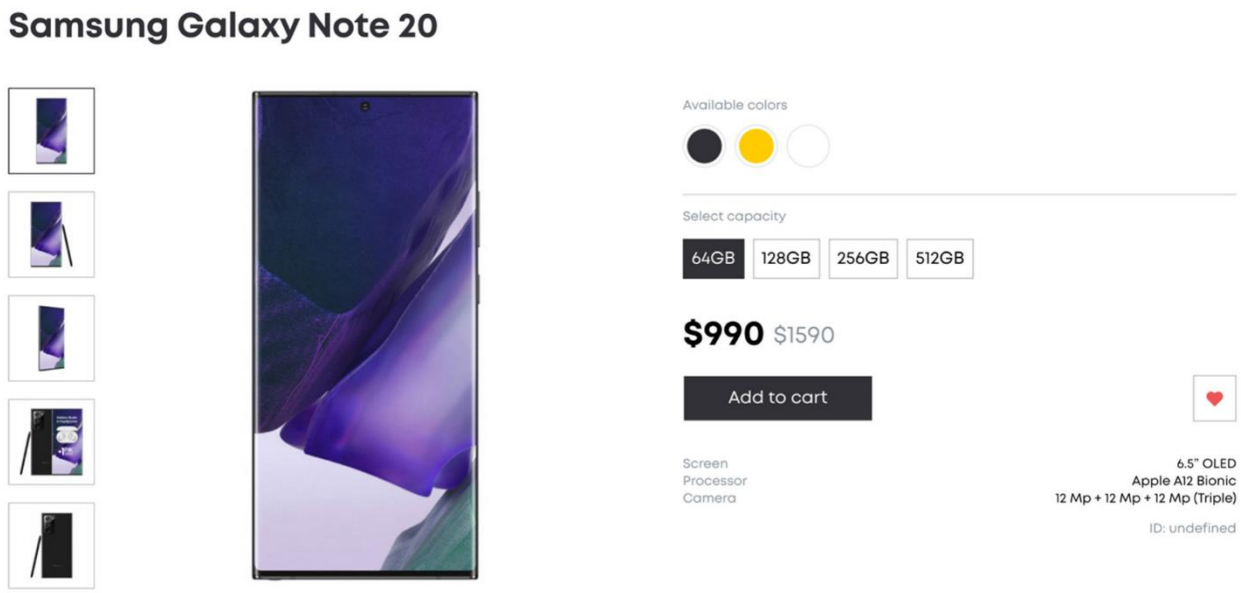

Рис. 3.40. Перша частина картки товару

Другий – це опис телефона або будь-кого іншого пристрою, а також перелік усіх інших характеристик, що вписані для цієї моделі товару (рис. 3.41).

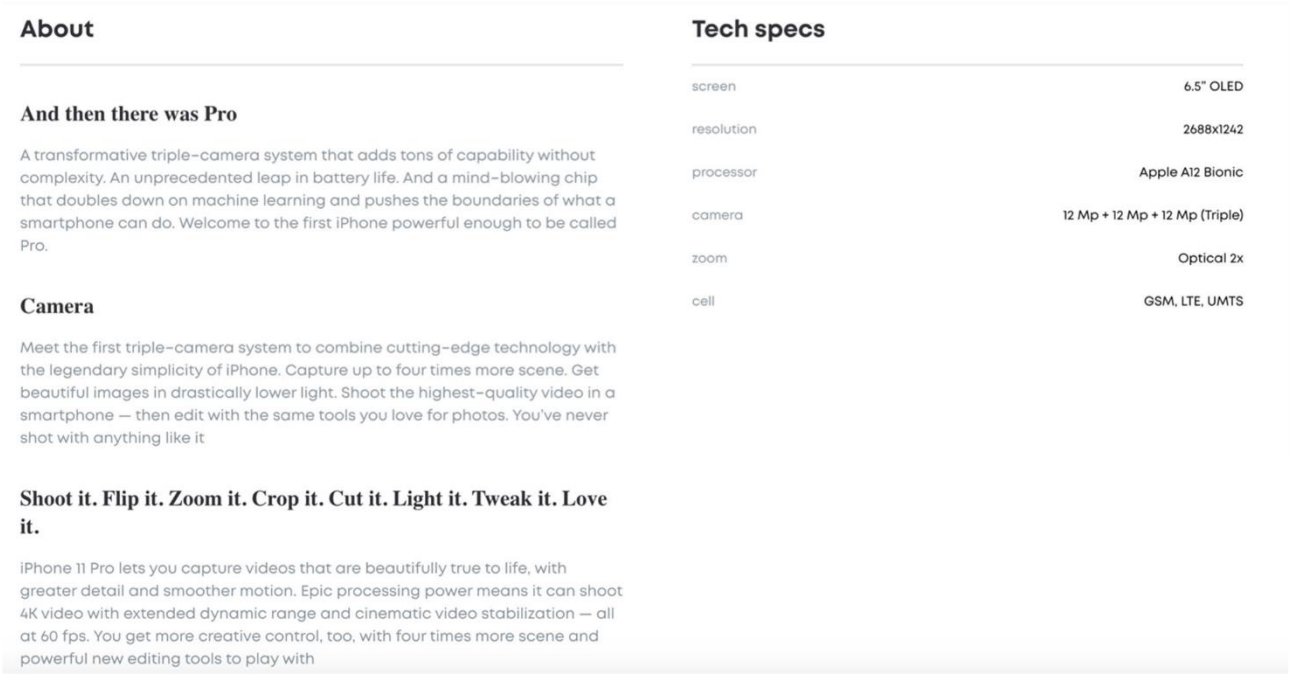

Рис. 3.41. Друга частина картки товару

В кінці була реалізована функція кошика (рис. 3.42), тобто людина просто

клацає на іконку "додати в кошик в будь-якому місці де є ця іконка і товар автоматично додається до кошика. Додавання до кошика реалізовано у два етапи, перший - додання у динамічну пам’ять, тобто під час перезавантаження вона зникне, але це швидкий обмін даними. Другий етап – це додавання цих даних в *LocalStorage*, це сховище не залежить від перезавантаження сторінки чи вимикання комп'ютера. Тож якщо людина додала якісь товари вони будуть там доти, доки вона їх сама не видалить, або не обновить кеш і куки.

Перший елемент цієї сторінки – це сама картка доданого товару, на ній має обов’язково бути фото товара, його назва, блок регуляції кількості і сума.

Рис. 3.42. Товар, що було додано до кошику

Після цього йде блок де можна здійснити саму покупку, тут вказана кінцева сума, тобто якщо товарів декілька то тут буде результат суми всіх цін, а також сумарна кількість одиниць товару (рис. 3.43).

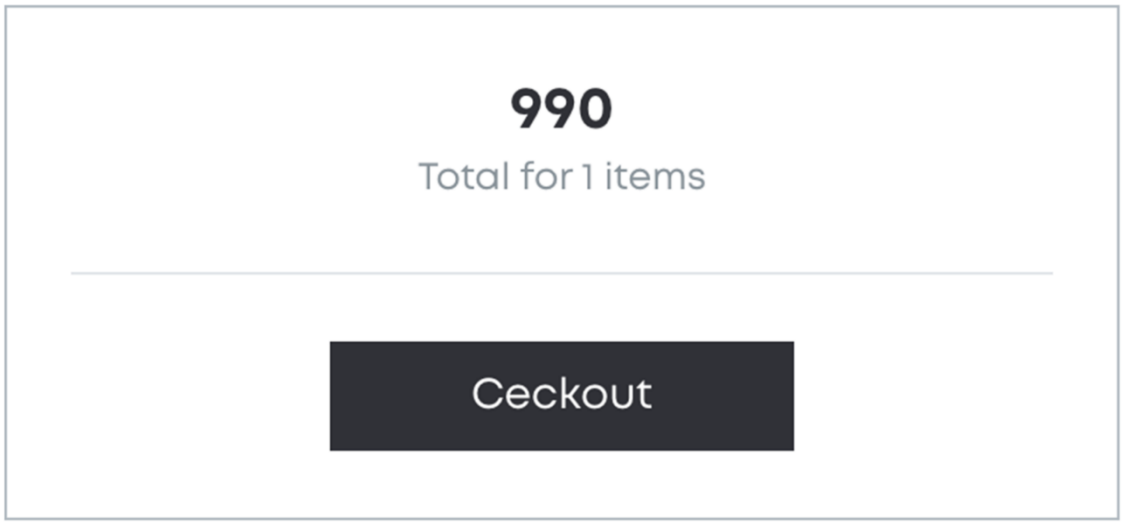

Рис. 3.43. Кнопка для покупки

На рисунку 3.44 представлено як вся конструкція виглядатиме на середньому і великому екрані.

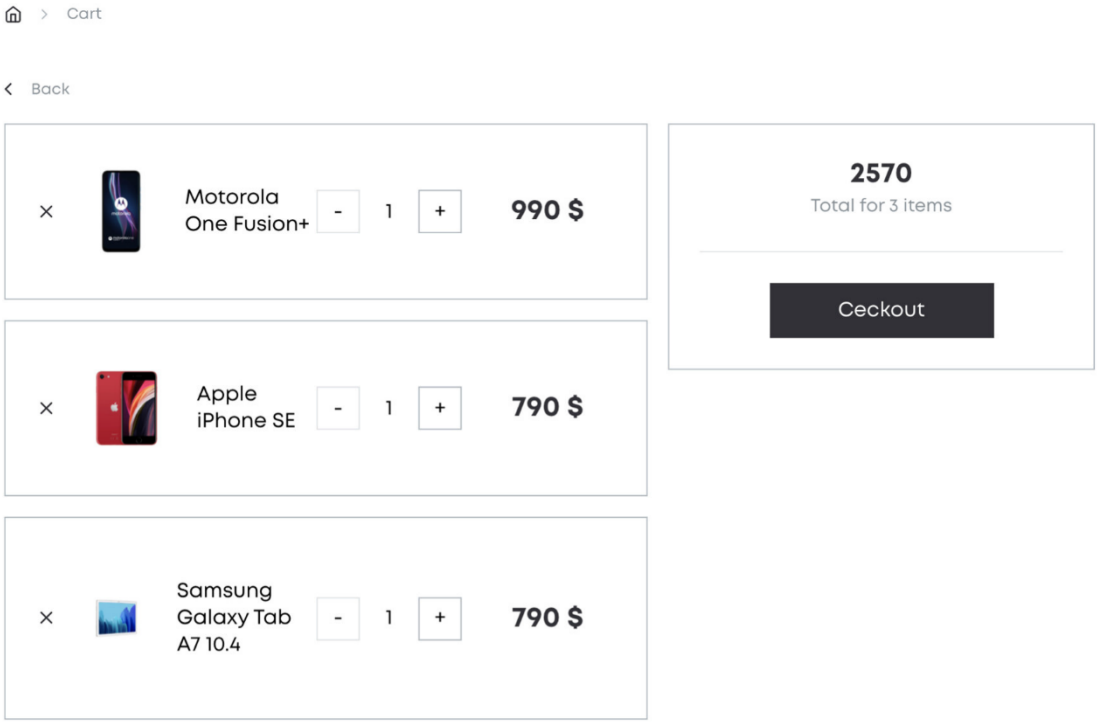

Рис. 3.44. Сторінка покупки товару для великих екранів

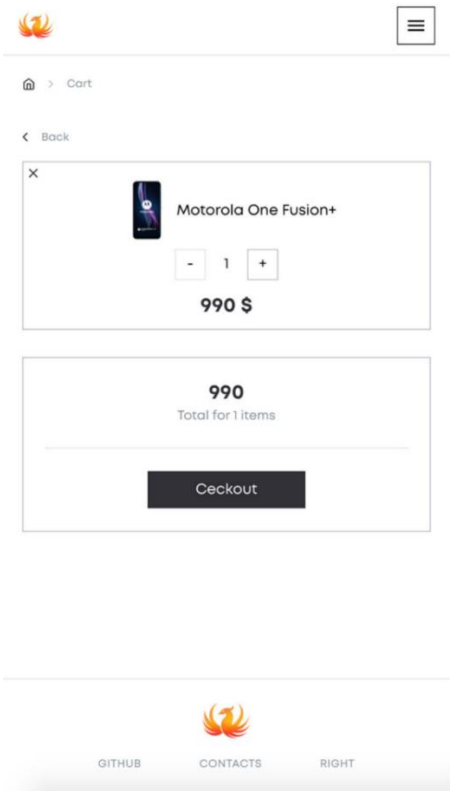

Рис. 3.45. Сторінка покупки товару для маленьких екранів

Оскільки таку конструкцію вкрай важко масштабувати під маленькі пристрої, то просто переверстаємо під більш відповідну модель (рис. 3.45).

Вся розробка вимагає багато часу і багато коду. Тому очевидно що під час розробки будуть виникати проблеми і непривальні рішення. Які доведеться перероблювати. Тому для покращення цього моменту використовувалась система контроль версій *git*, а також сервіс *git-hub*.

Висновки до розділу 3

Обрано модель *SPA*, оскільки вона дає швидкість і простоту використання. Під цю модель обрано базу даних *mongoDb* і зберігання цієї бази даних на серверах *mongoDb*. Створено роути під необхідні групи товарів і експортовано їх як модулі для обробки простоти написання.

Створено інтерфейси під групи товарів, а також під екшени в редаксі.

Написано модулі для хедера, адаптованість якого забезпечується не пружністю верстки, а наявністю мобільного і десктопного хедера, які вмикаються в залежності від ширини екрана.

Створено модель картки товару, а об’єднано їх в список товарів. Додано слайдер, щоб на головній сторінці ті ж самі картки можна було прокручувати як у слайдері.

Створено сторінки з товарами мобільних пристрої та таку саму для планшетів. Реалізовано сторінки з купівлею товарів та сторінки для улюблених товарів і кошика.

4. ОХОРОНА ПРАЦІ

4.1 Аналіз небезпеки під час роботи комп'ютера

З'ясувалося, що під час роботи з комп'ютером найбільшому ризику піддаються зорова, опорно-рухова, нервово-психічна система, достовірно невідомо, що саме порушує її – випромінювання або постійна статична поза.

Дисплей – головне джерело небезпеки. Він випускає випромінювання декількох видів: рентгенівське, ультрафіолетове, інфрачервоне, електромагнітне. Для кожного з цих випромінювання розроблені гранично допустимі норми, проте вони досить умовні й різняться в кожній країні. Норми передбачають, що опромінюється весь організм людини, тоді як на ділі впливу піддається лише верхня частина тулуба. Згадані норми встановлені з розрахунку на кожен вид опромінення в окремо, хоча реально всі поля діють одночасно, а їх комплексний вплив досі не досліджено.

Крім того, відео-дисплейний термінал порушує рівновагу між позитивно і негативно зарядженими іонами в повітрі. Електростатичне поле дисплея притягає негативні іони, порушуючи тим самим загальний баланс атмосфери. Це також шкодить здоров'ю. Вже через годину роботи біля монітора спостерігається майже повне зникнення негативних іонів. Ось чому необхідно, щоб до робочого місця за комп'ютером проникав свіже повітря. У зв'язку з усіма цими небезпеками досить чітко регламентовані розміри столу і стільця для роботи з комп'ютером. Отже непорушна постава шкідливо впливає на скелетно-м'язову систему. Стіл повинен бути просторим, зі спеціальною підставкою для ніг, а робочий стілець повинен мати відрегульовану висоту, певний кут нахилу сидіння і спинки.

Джерел випромінювання є два. Системний блок і монітор.

1. Системний блок створює тільки електромагнітне поле (випромінювання). Правда є ще й шум від вентиляторів, але ця тема всім

зрозуміла і не вимагає пізнань електроніки. Шкода від електромагнітного поля однозначно є при високому рівні поля. Однак поле комп'ютер створює набагато менше, ніж мобільний телефон.

2. Монітор має два основних шкідливих фактори. Бета-випромінювання (а простіше, потік електронів), яке власне кажучи створює картинку на екрані, і висока напруга (як і в будь-якому телевізорі, воно досягає 16-20 кіловольт), викликає іонізацію повітря. Бета-випромінювання поширюється монітором в двох напрямках – вперед і назад. У старих телевізорах і моніторах випромінювання досягало одного або двох метрів від екрану (всі пам'ятають рекомендаційне сидіти ближче двох, а то й трьох метрів від телевізора). Тобто виходив отакий потужний прожектор, що стріляє в нас шквалом електронів. По дорозі вибиваючи електрони з молекул повітря, перетворюючи їх на позитивні іони, так шкідливі для людини. На даний момент монітори мають дуже низький рівень бета-випромінювання, тобто електрони вилітають за межі екрану на пару сантиметрів. Основне випромінювання монітора направлено назад. Тому «зона ураження» поширюється на метр-півтора. Ось її і слід уникати. Висока напруга примудряється відхоплюватиму молекул повітря електрони, також перетворюючи молекули під шкідливі позитивні іони. До виробників моніторів і телевізорів пред'являються все більш жорсткі вимоги щодо використання високих напруг, і це не може не радувати.

4.2 Розрахунок, схема освітлення, вентиляції в робочому приміщенні

За правилами, світло при роботі з комп'ютером повинне падати зліва, а відстань від очей до екрана має бути близько 50 сантиметрів. Крім того, крісло слід відрегулювати так, щоб очі були на одному рівні з центром монітору. Фахівці говорять, що сааме очі найбільш страждають при роботі з комп'ютером. Виявляється, коли довго дивишся на екран, перестаєш моргати. Тому очі червоніють, сльозяться, а значить, знижується зір. Невелику відстань до екрану, дрібний шрифт, мерехтіння, різне освітлення призводять, у кінцевому рахунку,

до короткозорості. Якщо очі червоніють, сльозяться, з'являється печіння, починає боліти голова – це вже ознаки того, що очі втомилися, і треба відпочити. Але краще, звичайно, до такого стану себе не доводити.

4.3 Інструкція з охорони праці під час роботи з комп'ютером

Персонал, що працює на комп'ютері зобов'язаний дотримуватися вимог інструкції, розробленої на підставі Санітарних норм і правил, а також нести особисту відповідальність за дотримання вимог безпеки своєї праці та за створення небезпечного або шкідливого виробничого фактора для інших працюючих і поломки комп'ютера.

При роботі з комп'ютером шкідливими і небезпечними факторами є:

• електростатичні поля;

• електромагнітне випромінювання;

• наявність потужних іонізуючих випромінювання;

• локальне стомлення, загальне стомлення;

• стомлюваність очей;

• небезпека ураження електричним струмом;

• пожежонебезпека.

Режими праці та відпочинку при роботі з комп'ютером повинні організовуватися в залежності від виду та категорії трудової діяльності. Види трудової діяльності поділяються на3 групи:

• Група А – робота з зчитування інформації з екрана комп'ютера з попереднім запитом;

• Група Б – робота з введення інформації;

• Група В – творча робота в режимі діалогу.

За основну роботу з комп'ютером слід приймати таку, що займає не менше 50% часу протягом часу роботи **комп'ютером**. Для видів трудової діяльності встановлюється 3 категорії тяжкості і напруженості роботи з комп'ютером, які визначаються:

для групи А – по сумарному числу прочитуються знаків за час роботи з комп'ютером, але не більше 60 000 знаків;

для групи Б – по сумарному числу зчитуються або вводяться знаків за час роботи з комп'ютером, але не більше 40000 знаків;

для групи В – по сумарному часу безпосередньої роботи з комп'ютером, але не більше 6 годин за час роботи з комп'ютером. Для забезпечення оптимальної працездатності і збереження здоров'я протягом часу роботи з комп'ютером повинні встановлюватися регламентовані перерви.

Перед початком роботи необхідно переконатися, що монітори комп'ютера мають анти блокове покриття (крім групи А) з коефіцієнтом відображення не більше 0,5. Покриття повинне також забезпечувати зняття електростатичного заряду з поверхні екрана, іскріння і накопичення пилу. Корпус монітора повинен забезпечувати захист від іонізуючих та неіонізуючих випромінювань.

Необхідно перевірити робоче положення комп'ютера відстань між стіною з віконними прорізами і столом повинно бути не менше 0,8 м. Відстань між робочими столами повинна бути не менше 1,2 м. Не допускається знаходження другого робочого місця з боку задньої сторони.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

1. Проаналізовано предметну область дослідження, визначено поняття та основні характеристики електронної комерції, особливості концепції інтернет-магазинів як одного з основних типів веб-сайтів, проаналізовано основні переваги та недоліки інтернет-магазинів, визначено класифікацію та характеристики методів розробки.

2. Встановлено, що інтернет-магазини мають складну класифікаційну структуру і час створення інтернету-магазину незрівнянно менше, ніж звичайного магазин. З’являється свобода переміщення продавця, з інтернет-магазином з'являється можливість розширити географію бізнесу на світові ринки і професійно створений інтернет-магазин може функціонувати повністю автономно.

3. Проаналізовані та детально пояснені різні технології, які поєднуються для формування стеку MERN. Крім того здійснено аналіз, інших інструментів, які допомагають керувати даними додатків, обробкою помилок та аутентифікацією користувача. Обговорено та вивчено шляхи реалізації кінцевої програми – інтернет магазину з продажу мобільної техніки.

4. Обрано модель *SPA*, оскільки вона дає швидкість і простоту використання. Під цю модель обрано базу даних *mongoDb* і зберігання цієї бази даних на серверах *mongoDb*. Створено роути під необхідні групи товарів і експортовано їх як модулі для обробки простоти написання.

5. Написано модулі для хедера, адаптованість якого забезпечується не пружністю верстки, а наявністю мобільного і десктопного хедера, які вмикаються в залежності від ширини екрана. Створено модель картки товару, а об’єднано їх в список товарів. Додано слайдер, щоб на головній сторінці ті ж самі картки можна було прокручувати як у слайдері.

6. Створено сторінки з товарами мобільних пристроїв та таку саму для планшетів. Реалізовано сторінки з купівлею товарів та сторінки для улюблених товарів і кошика.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. A brief history of Node.js [Електронний ресурс] // NodeJS. – 2021. – Режим доступу: https://nodejs.dev/learn/a-brief-history-of-nodejs.
2. Bachuk A. Redux · An Introduction [Електронний ресурс] / Alex Bachuk // Articles. – 2016. – Режим доступу: https://www.smashingmagazine.com/2016/06/an-introduction-to-redux/.
3. Components and Props [Електронний ресурс] // React. – 2021. – Режим доступу: https://reactjs.org/docs/components-and-props.html.
4. Express [Електронний ресурс] // Express. – 2021. – Режим доступу: https://expressjs.com.
5. Garcia R. JWT tokens and security – working principles and use cases [Електронний ресурс] / Romain Garcia // VAADATA. – 2016. – Режим доступу: https://www.vaadata.com/blog/jwt-tokens-and-security-working-principles-and-use-cases/.
6. Git Handbook [Електронний ресурс] // GitHub. – 2021. – Режим доступу: https://guides.github.com/introduction/git-handbook/.
7. Goldwater M. An abbreviated history of JavaScript package managers [Електронний ресурс] / Matt Goldwater // JavaScript. – 2019. – Режим доступу: https://javascript.plainenglish.io/an-abbreviated-history-of-javascript- package-managers-f9797be7cf0e.
8. Historical trends in the usage statistics of server-side programming languages for websites [Електронний ресурс]. – Режим доступу: https://w3techs.com/technologies/history_overview/programming_language
9. Hoque S. Full-stack React Projects Second Edition. Birmingham, UK [Електронний ресурс] / Shama Hoque // Packt Publishing. – 2020
10. Horowitz E. A Founder's Reflections on 10 Years of MongoDB [Електронний ресурс] / Eliot Horowitz // MongoDB. – 2017. – Режим доступу: https://www.mongodb.com/mern- stack.
11. IDE features [Електронний ресурс] // JetBrains. – 2021. – Режим

доступу:https://www.jetbrains.com/webstorm/features/ide-features.html.

12. Introducing Hooks [Електронний ресурс] // React. – 2021. – Режим доступу: https://reactjs.org/docs/hooks-intro.html.

13. Karnik N. Introduction to Mongoose for MongoDB [Електронний ресурс] / Nick Karnik // freeCodeCamp. – 2018. – Режим доступу: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/.

14. Karpov V. The MEAN Stack: MongoDB, ExpressJS, AngularJS and Node.js [Електронний ресурс] / Valeri Karpov // MongoDB. – 2017. – Режим доступу: https://www.mongodb.com/blog/post/the-mean-stack-mongodb-expressjs-angularjs-and

15. MERN Stack [Електронний ресурс] // MongoDB. – 2021. – Режим доступу: https://www.mongodb.com/mern-stack.

16. Models [Електронний ресурс] // mongoose. – 2021. – Режим доступу:https://mongoosejs.com/docs/models.html.

17. Motivation [Електронний ресурс] // Redux. – 2021. – Режим доступу: https://redux.js.org/understanding/thinking-in-redux/motivation.

18. Using middleware [Електронний ресурс] // Express. – 2021. – Режим доступу: https://expressjs.com/en/guide/using-middleware.html.

19. Using the State Hook [Електронний ресурс] // React. – 2021. – Режим доступу: https://reactjs.org/docs/hooks-state.html.

20. Virtual DOM and Internals [Електронний ресурс] // React. – 2021. – Режим доступу: https://reactjs.org/docs/faq-internals.html.

21. What is middleware? How it works in nodeJS? A Simple implementation [Електронний ресурс] // medium. – 2019. – Режим доступу: https://medium.com/dataseries/what-is-middleware-how-it-works-in-nodejs-a-simple-implementation-485bcb9c3d53.

22. Wodehouse C. 5 SQL vs. NoSQL Databases: What's the Difference? [Електронний ресурс] / Carey Wodehouse // upwork. – 2019. – Режим доступу: https://www.upwork.com/resources/sql-vs-nosql-databases-whats- the- difference?

utm_source=google&utm_campaign=SEM_GGL_INTL_NonBrand_

23. Zuraida A. Authorization and Authentication [Електронний ресурс] / Audira Zuraida // Meduim. – 2018. – Режим доступу: https://medium.com/@audira98/authorization-and-authentication-cd0a7c994278.

24. Документація бази даних mongoDb. [Електронний ресурс] – Режим доступу: https://www.mongodb.com/cloud

25. Документація мови програмування javaScript. [Електронний ресурс] – Режим доступу: https://javascript.info/

26. Документація мови програмування typeScript. [Електронний ресурс] – Режим доступу: https://www.typescriptlang.org/

27. Документація фреймфорку ReactJs. [Електронний ресурс] – Режим доступу: https://reactjs.org/

28. Іванкевич О.В. Інформаційні системи та структури даних / О.В. Іванкевич, Г.М. Кременецький, В.І. Мазур // Навч.посібник. –К.: НАУ, 2006. –232с.

29. Класифікація сучасних інтернет-магазинів в електронній комерції [Електронний ресурс] – Режим доступу: https://sebweo.com/klasifikatsiya-suchasnih-internet-magaziniv-v-elektronnij-komertsiyi/

30. Проникнення Інтернету в Україні [Електронний ресурс]. Жовтень, 2019 Автор статистичне агентство Factum Group. – Режим доступу: https://inau.ua/sites/default/files/file/1910/dani_ustanovchyh_doslidzhen_iii_kvartal_2019_roku.pdf

31. Райчев І.Е. Принципи проектування відкритих розподілених систем / І.Е.Райчев, О.Г.Харченко, В.В.Замковий // Навч. посіб. –К.: Вид-во Нац. авіац. ун-ту “НАУ-друк”, 2015. – 240 с.

32. Шалева О. І. Електронна комерція. Навч. посіб. – К.: Центр учбової літератури, 2011. – 216 с.

ДОДАТКИ

Додаток А. Код компонентів ProductList.js та ProductItem.js

```
// src/components/ProductList.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const ProductList = () => {
  const [products, setProducts] = useState([]);

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const response = await axios.get('/api/products');
        setProducts(response.data);
      } catch (error) {
        console.error(error);
      }
    };

    fetchProducts();
  }, []);

  return (
    <div>
      <h1>Магазин мобільних телефонів</h1>
      <ul>
        {products.map((product) => (
          <li key={product._id}>{product.name}</li>
        ))}
```

```
      </ul>
    </div>
  );
};

export default ProductList;

// src/components/ProductItem.js
import React from 'react';

const ProductItem = ({ product }) => {
  return (
    <div>
      <h3>{product.name}</h3>
      <p>{product.description}</p>
      <img src={product.image} alt={product.name} />
      <p>Ціна: {product.price}</p>
    </div>
  );
};

export default ProductItem;
```

Додаток Б. Вміст файла server.js

```
// backend/server.js
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');

const app = express();
const port = process.env.PORT || 5000;

app.use(cors());
app.use(express.json());

// MongoDB configuration
const uri = 'mongodb://localhost:27017/mern_shop'; // При необхідності, замініть URL MongoDB
mongoose.connect(uri, { useNewUrlParser: true, useCreateIndex: true, useUnifiedTopology: true });
const connection = mongoose.connection;
connection.once('open', () => {
  console.log('Connected to MongoDB database');
});

// Define product schema and model using Mongoose
const productSchema = new mongoose.Schema({
  name: String,
  description: String,
  image: String,
  price: Number
});
```

```
const Product = mongoose.model('Product', productSchema);

// API endpoints
app.get('/api/products', async (req, res) => {
  try {
    const products = await Product.find();
    res.json(products);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal server error' });
  }
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running on port: ${port}`);
});
```