Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Йолкіна Єгора Вадимовича*
(ПІБ)

академічної групи *122-18-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованої інформаційної системи для підприємства з продажу одягу в Інтернет.*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Приходченко С.Д.*			
розділів:				
спеціальний	*доц. Приходченко С.Д.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-2* (група) *Йолкіна Єгора Вадимовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованої інформаційної системи для підприємства з продажу одягу в Інтернет.*

затверджена наказом ректора НТУ «ДП» від №

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Приходченко С.Д.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Йолкін Є.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 101 с., 36 рис., 3 дод., 21 джерела.

Об'єкт розробки: веб-орієнтована інформаційна система з продажу одягу.

Мета кваліфікаційної роботи: сторення веб-орієнтованої інформаційної системи.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціювання, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні веб-орієнтованої інформаційної системи, що надає можливість перегляду товарів магазину одягу.

Актуальність даної інформаційної системи для підприємства продажу одягу визначається великим попитом, що допомогає біль оптимизувати роботу підприємства.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, МАГАЗИН, ВЕБ-ДОДАТОК, ПРОГРАММА, СИСТЕМА.

ABSTRACT

Explanatory note: 101 with., 36 Fig., 3 dod., 21 sources.

Object of development: web-oriented information system for the sale of clothing.

The purpose of the qualification work: the creation of a web-based information system.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the task.

In the first section the subject branch is analyzed, the urgency of the task and the purpose of development are determined, the statement of the task is formulated, the requirements to the software implementation, technologies and software are specified.

The second section analyzes the existing solutions, selected platforms for development, designed and developed the program, describes the program, algorithm and structure of its operation, as well as calling and loading the program, determines the input and output data, describes the parameters of hardware.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

Of practical importance is the creation of a web-based information system that allows you to view the goods of the clothing store.

The relevance of this information system for the clothing company is determined by high demand, which helps to optimize the work of the company.

List of keywords: INFORMATION SYSTEM, SHOP, WEB-APPLICATION, PROGRAM, SYSTEM.

ЗМІСТ

ПЕРЕЛІК УМНОВНИХ ПОЗНАЧЕНЬ

IT – інформаційні технології;

JS - JavaScript;

HTTP - HyperText Transfer Protocol;

API - application programming interface;

ORM - Object-Relational Mapping;

SQL - structured query language;

СУБД - Система управління базами даних;

ПК – Персональний Комп'ютер

ПЗ – Програмне забезпечення

MVC - Model-View-Controller

ВСТУП

Метою моєї кваліфікаційної роботи є створення веб-орієнтованої інформаційної системи для магазину з продажу одягу, що тісно пов'язано з моєї спеціальністю “Комп’ютерні науки”. Ця робота відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври спеціальності “Комп’ютерні науки”.

Інтернет з кожним часом стає все більш та більш невідємної частиної нашого життя. Фактично тяжко уявити собі сучасну людину двадцять першого століття, яка немала б доступу до всесвітної інформаційною павутини. Так само, як тяжко уявити підприємство з продажу будь-яких речей без свого особистого сайту. Саме сайт підприємства у сучасному інформаційному світі є невід`ємною частиної кожного бізнесу. Це є обличчям, яке потенційний клієнт бачить перший раз з знайомства із вашим бізнесом. Мати власний сайт не є великої розкішю навіть для малого підприємства, але така необхідна річь навіт на старті проекта.

Інтернет-магазин – одна з перших речей над якою треба замислитися майбутньому молодому підприємцю. Саме інтеренет магазин є ще одним з інструментів популізації свого товару не тільки в локальному значенні, а навіть в світовому. Власний інтернет магазин дає можливість реалізувати свій товар з великим показником ефективності та часу. Якщо порівнювати сайт з фізичним відкриттям магазину, то шлях праці з інтернет магазином навіть буде біль економічним рішенням.

Невід’ємною первагою сайту також є його можливість працювати 24 години у добу. У будь-який час потенційний клієнт з легкістю може здійснити покупку вашого товару з будь-якої точки країни, або світу, у будь який час доби.

Своєчасне рішення підприємства зробити свій сайт, може бути одним з факторів його майбутнього успіху на всесвітньому ринку товарів. Саме таке рішення свого часу прийняв всесвітньо відомий міл’ярдер Джефф Безос, коли у 1994 році зважився перенести торгівлю книжка в онлайн формат.

В ході планування розробки інтернет магазину було проведено аналіз предметної галузі, поставлені вимоги до інтернет магазину, також був створен алгоритм на кожному етапі роботи веб-сайту.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Ecommerce, або електронна комерція, відноситься до цифрової платформи та бізнес моделей, де ви можете купувати або продавати продукти в Інтернеті. Кожен раз коли ви купуєте продукт в Інтернеті ві берете участь в економіці електронної комерції.

Електронна комерція вперше була представлена в 1960-х роках через електронний обмін даними (EDI) у мережах з доданою вартістю (VAN). Засіб зростав із збільшенням доступу до Інтернету та появою популярних онлайн-продавців у 1990-х і на початку 2000-х років. Наприклад Amazon почав працювати як бізнес з доставки книг у гаражі Джеффа Безоса в 1995 році. EBay, який дає можливість споживачам продавати один одному в Інтернеті, запровадив онлайн-аукціони в 1995 році.

Як і будь-яка цифрова технологія або ринок покупок, орієнтований на споживачів, електронна комерція розвивалася з роками. Оскільки мобільні пристрої стали все більш популярними, мобільна комерція стала власним ринком. З появою таких сайтів, як Facebook і Pinterest, соціальні мережі стали важливим драйвером електронної комерції. Станом на 2014 рік Facebook приніс 85 відсотків продажів у соціальних мережах на платформі електронної комерції Shopify, згідно Paymill.

Ринок, що змінюється, дає величезну можливість для компаній підвищити свою актуальність і розширити свій ринок у світі онлайн. Згідно з даними Digital Commerce 360, дослідники прогнозують, що до 2022 року електронна комерція становитиме 17 відсотків роздрібних продажів у США. У 2017 році США витратить близько 460 мільярдів доларів в Інтернеті. Ці цифри продовжуватимуть зростати, оскільки використання мобільних пристроїв та Інтернету зростатиме як у США, так і на ринках, що розвиваються в усьому світі. [1]

Типи моделей електронної комерції. Електронну комерцію можна розділити на чотири основні категорії. Основою для цієї простої класифікації є сторони, які беруть участь у операціях. Отже, чотири основні моделі електронної комерції є такими:

Бізнес-до-бізнесу (Business to Business, B2B). Це бізнес-операції. Тут компанії ведуть бізнес один з одним. Кінцевий споживач не задіяний. Таким чином, в онлайн-транзакціях беруть участь лише виробники, оптові, роздрібні продавці тощо.

Споживач до споживоча (Consumer to Consumer). Споживач до споживача, де споживачі знаходяться в безпосередньому контакті один з одним. Жодна компанія не задіяна. Це допомагає людям продавати свої особисті товари та активи безпосередньо зацікавленій стороні. Зазвичай товарами, що продаються, є автомобілі, велосипеди, електроніка тощо.

Бізнес для споживача (Business to Consumer). Бізнес для споживача. Тут компанія буде продавати свої товари та/або послуги безпосередньо споживачеві. Споживач може переглядати їхні веб-сайти та дивитися товари, фотографії, читати відгуки. Потім вони розміщують замовлення, і компанія відправляє товар безпосередньо їм. Популярними прикладами є Amazon, Rozetka, Epicentr тощо.

Від споживача до бізнесу (Consumer to Business). Це протилежність B2C, це споживач для бізнесу. Отже, споживач надає фірмі товар або якусь послугу. Скажімо, вільний IT-фрілансер, який демонструє та продає своє програмне забезпечення компанії. Це буде транзакція C2B.

Не потрібно бути експертом з бізнесу, щоб знати, що електронна комерція змінила сучасний ринок за останні роки. Хоча це домінуюча модель, продаж товарів або послуг в Інтернеті має свої переваги та недоліки в порівнянні з традиційним бізнесом. [2]

Переваги електронної комерції.

Електронна комерція усуває потребу в звичайних магазинах і дозволяє підприємствам розширювати свою клієнтську базу. Крім того, що виключають можливість довгих черг, сайти електронної комерції пропонують величезну

перевагу як для покупців, так і для магазинів, які не розташовані у великих міських районах. Навіть якщо ви знаходитесь у великому місті, електронна комерція відкриває нові ринки, дозволяючи вам розробити нову бізнес-модель, орієнтовану на вашу зростаючу споживчу базу. Багато компаній досягли особливого успіху в розробці хорошої пошукової оптимізації для електронної комерції, яка спрямовує більше трафіку на сайт.

Цифрові продукти можна продавати в Інтернеті з невеликими накладними витратами. Завдяки електронній комерції споживачі можуть миттєво купувати музику, відео чи книги. Тепер магазини можуть продавати необмежену кількість копій цих цифрових товарів, не турбуючись про те, де вони будуть зберігати інвентар.

Електронна комерція також дозволяє вашому бізнесу розширюватися легше, ніж фізичним роздрібним продавцям. Коли звичайний магазин росте, йому потрібно подумати, як він обслуговуватиме більше клієнтів на тому самому маленькому просторі. Щоб пришвидшити роботу, потрібно більше співробітників, більше площі відведено для формування черг, покупці відчувають себе переповненими, оскільки клієнтська база та запаси зростають. Звичайно, логістика завжди стає жорсткішою в міру зростання бізнесу, незалежно від того, як він працює. Однак при правильному виборі стороннього постачальника логістики компанії електронної комерції можуть керувати цим зростанням, не турбуючись про аспекти фізичного магазину.

Нарешті, електронна комерція дозволяє вашому бізнесу відстежувати логістику, що є ключем до успішної компанії електронної комерції. Якщо все оцифровувати, буде легше автоматично збирати дані та аналізувати цифри. Хоча ви можете отримати вигоду від того, що краще продається, ви також можете дозволити собі йти на більший ризик щодо товарів невеликого обсягу. Звичайна стратегія роздрібної торгівлі зосереджується на складанні товарів, що швидко торгуються, але економіка електронної комерції дозволяє включати в каталог повільні й навіть застарілі товари. Зберігання коштує дешевше, а показати продукт так само просто, як додати іншу сторінку товару на свій сайт.

Недоліки електронної комерції.

Багато споживачів все ще віддають перевагу особистому контакту та стосункам, які формуються у звичайному магазині. Це може бути особливо цінним для клієнтів, які купують спеціалізовані продукти, оскільки вони можуть захотіти проконсультуватися з експертом щодо найкращого продукту для їхніх потреб. Надійна гаряча лінія обслуговування клієнтів не може замінити особисту взаємодію зі спеціалізованим торговим представником. Крім того, багато клієнтів хочуть випробувати продукт перед покупкою, наприклад, коли купують одяг.

Безпека та шахрайство з кредитними картками також є величезними ризиками під час покупок в Інтернеті. Споживачі ризикують шахрайством з ідентифікацією та подібними ризиками щоразу, коли вони вводять свої дані на сайт. Якщо ваш сайт не переконує покупців у тому, що процес оформлення замовлення безпечний, вони можуть злякатися від покупки. З іншого боку, підприємства ризикують фішинговими та іншими формами кібератак. Якщо один із ваших співробітників відкриє лише одне шкідливе посилання, це може поставити під загрозу функціональність вашого веб-сайту, фінансову інформацію або, найгірше, інформацію ваших клієнтів.

Якщо шопінг – це миттєве задоволення, то споживачі залишаються з порожніми руками. Часто їм доводиться або платити більше за прискорену доставку, або чекати кілька днів, поки товар прибуде. Очікування могло відштовхнути клієнтів. Для компаній доставка стає особливо складною, коли клієнт хоче повернути гроші. Підприємствам електронної комерції, що розвиваються, необхідно розширити свої функції зворотної логістики, тобто повернення товарів і відшкодування витрат.

Soul Partners, Baker Tilly Україна та Aequo за підтримки програми USAID «Конкурентоспроможна економіка України» представили дослідження ринку e-commerce в Україні за 2021 рік. Згідно з результатами, з початку пандемії коронавірусу обсяг ринку зріс на 41%.

Серед основних факторів, що вплинули на зростання e-commerce в країні, серед іншого, виділяють збільшення рівня проникнення інтернету та кількості користувачів смартфонів, обмеження, пов'язані з пандемією коронавірусу, а також зростання довіри до цифрових платіжних систем.

Найбільшими за обсягом та найрозвиненішими секторами електронної торгівлі в Україні є електроніка та одяг. У середньому, сектор одягу зростав на 26% з 2016 року і досягнув $291 млн у 2020-му. Частка електронної комерції у роздрібній торгівлі одягом у своїй становила 6,8%. А середній чек – $24-31.

За результатом можно зробити висновки, що електронна комерція в Україні з кожним роком набирає обертів.

1.2. Призначення розробки та галузь застосування

Призначення розробки полягає у створені повноцінної інформаційної системи управління контентом для частного підприємства з продажу одягу у інтернеті.

Ця інформаційна система є структурованим каталогом товарів, з можливостю фільтру по визначеним критеріям. Також інформаційна система виконує інформативну функцію, демонструючи новим користувачам рід діяльності фірми.

Цельовою аудиторію є чоловіки та жінки 18-40 років. Ці дані є достатньо загальними, так як основним товаром є одяг, а одяг потрібний будь-якій людині, незалежно від віку.

1.3. Підстави до розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- освітня програма 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» №__від .. р;
- завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованої інформаційної системи для підприємства з продажу одягу в Інтернет».

1.4. Постанова задачі

Метою розроблення проекту є створення функціоналу для відкриття и постійної праці інтернет-магазину з продажу одягу.

Інформаційна система повинна відповідати таким умовам як:

– дизайн сайту повинен відповідати фірмовому стилю, а саме суміші білого та чорного кольору.

– навігація сайту повинна бути максимально проста та зручна, навіть для недосвідченного користувача.

– інтерфейс користувача також має бути максимально зручним, а головне логічно інтуїтивним.

– доступ до інформації на сайту повинен бути максимально швідким.

Функції адміністратора на сайті повинні бути створені згідно таких умов:

– можливість додавати інформацію про тип одягу.

– можливість додавати інформацію про розмір одягу.

– можливість додавати інформацію про одяг.

Всі дані які будуть внеситися адміністратором мають автоматично зберігатися у базі даних інтернет-магазину.

1.5. Вимоги для програми або програмного виробу

1.5.1. Вимоги до функціоналих характеристик

Перш за все, одним з важливих речей сучасного сайту є його адаптавність під різноманітні дейвайси. Саме адаптивність є елементом зручності користувача, який наприклад користується смартфоном, планшетом тощо.

Сьогодні понад 5,19 млрд осіб нашої планети користуються мобільними телефонами (здебільшого смартфонами), приріστяких за останній рік становить 124 млн (2,4 %) людей. Згідно зі статистикою Statcounter близько 53 % усіх запитів в Інтернеті роблять саме з мобільних пристроїв, водночас 44 % від усього світового вебтрафіку становлять комп'ютери. Тому дуже важливим фактором є самє адаптивність, яка буде зручною як для користувачів ПК, так і для користувачів смартфонами.

Також не менш важливим функціоналом інформаційної системи є саме демонстрація потенційному користувачу товарних продуктів підприємства. Для більш зручною вибірки товару сайт повинен "вміти" фільтрувати інформації по певним характеристикам. Фільтр повинен мати вибірку по розміру, а також по типу одягу. Функціонал фільтрації повинен бути максимально зручним та зрозумілим для користувача.

Для адміністратора сайту повинна бути створена адмін-панель, з рівнем доступу "ADMIN". Завдяки цій панелі адміністратор може без втручання спеціаліста додавати інформацію в інтернет-магазин. Отже, адмін-панель повинна мати наступні функції:

- додавання типу одягу;
- додавання розміру одягу;
- додавання одягу;

Для цілосного інтернет-магазину повинин бути створен кошик. Користувач з рівнем доступу "User", повинен мати лише функцію додавання товару до кошика.

1.5.2. Вимогу до інформаційної безпеки

Основними вимогами для інформаційної безпеки перш за все є:

- забезпечення конфіденційність інформації;
- забезпечення цілісності даних;
- забезпечення захищенності програмного продукту від втручання зовні;

Для безпечного функціонування програмного продукту треба дотримуватися наступних вимог:

- використання лише безпечного так ліцензованого програмного забезпечення;
- забезпечення захищенності свого ПЗ від зловмисних програм;

Забезпечення інформаційною безпекою інтернет-магазина є одною з найважливіших речей при створенні сайту. Адже від цього залежить конфіденційність даних користувача, та перш за все конфіденційність даних самої компанії. Особливо важливо розуміння того, що загроза кіберзлочинності реальна, і серйозне ставлення до процесу захисту свого інтернет-магазину є важливим пунктом.

1.5.3. Вимоги до складу та параметрів технічних засобів

Технічними умовами для роботи фінальної версії програмного продукту, а саме інформаційної системи продажу одягу є:

- обсяг оперативної пам’яті не менш за 2 гб;
- операційна система Windows, Linux, MacOS, IOS, Android;
- наявність сучасного браузера, наприклад, Google Chrome, Opera, Safari;
- підтримка високошвидкістного Інтернет з’єднання.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для програмної сумісності обов’язковою вимогою є наявність останньої версію мережевого драйверу. Також не меньш важливим фактором є наявність стабільного підключення до високошвидкістного Інтернет з’єднання. Та наявність будь-якого сучасного браузера на девайсі. Наприклад: Opera, Safari, Google Chrome тощо.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Призначення інтернет-магазину для підприємства “Boresfen-Clothes”, полягає у демонстрації товарного асортименту потенційно можливому клієнту або постійному покупцю.

Для менеджер складу інтернет магазину реалізована можливість доповнювати контентом сайт, а саме додавати інформацію про: тип одягу, розмір одягу та власне додавати сам одяг на сторінку з картками товару.

Інтернет-магазин продажу одягу передбачає собою роботу 24 години на добу, виконоючи функції ознайомлення клієнтів з товаром та можливістю менеджер складу додавати інформації потрібного типу.

2.2. Опис застосованих математичних методів

Дана розробка інтернет-магазину використання спеціальних математичних методів не потребує, тому вони не будуть впроваджені в ході розробкі системи.

2.3 Опис використаних технологій та мов програмування

Робота над інтернет-магазином складається з двух невідємних частин, а саме: з клієнтської частини, та серверної частини.

Для створення клієнтської частини інтернет-магазину були обрані та використані наступні технології та мови програмування:

- React JS;
- React Bootstrap;
- Axios;
- React-router-dom;

– MobX;
– HTML;
– JS;

Даний набір технологій для клієнтської частини є доволі зручним та ефективним у використанні. Значною перевагої такого набору технологій є, його здатність до ефективної та стабільної роботи. Також невідємним пунктом для такого набору технологій є економія часу для спеціаліста, та подальшому структуруванню кода.

HTML (HyperText Markup Language – «мова гіпертекстової розмітки») – базовий будівельний блок Інтернету. Він визначає зміст та структуру веб-контенту. Усі сторінки HTML мають ряд елементів HTML, що складаються з набору тегів та атрибутів. Елементи HTML є будівельними блоками веб-сторінки. Тег повідомляє веб-браузеру, де починається і закінчується елемент, тоді як атрибут описує характеристики елемента.

Іншою важливою частиною елемента HTML є його атрибут, який має два розділи – ім’я та значення атрибута. Ім’я визначає додаткову інформацію, яку хоче додати користувач, тоді як значення атрибута дає додаткові характеристики. [3]

JavaScript є мовою сценаріїв, який працює виключно на стороні клієнта всередині браузера. Ця мова має такі ефекти, які неможливо донести ніякими іншими засобами, оскільки він запускається всередині браузера і має непосередній доступ до всіх елементів веб-документу. JavaScript є високорівневою, динамічною, нетиповою та інтерпретованою мовою програмування, яка добрепідходить для програмування в об'єктно-орієнтованому та функціональному стилях.

Тож основними перевагами використання саме JavaScipt є:

– швидкість, оскільки JavaScript є «інтерпретованою» мовою, він скорочує час, потрібний іншим мовам програмування, наприклад Java, для компіляції;
– простота, JavaScript легко зрозуміти і вивчити. Структура проста як для користувачів, так і для розробників. Це також дуже можливо реалізувати,

заощаджуючи розробникам багато грошей на розробку динамічного вмісту для Інтернету;

– популярність, оскільки всі сучасні браузери підтримують JavaScript, його можна побачити майже скрізь. Усі відомі компанії використовують JavaScript як інструмент, включаючи Google, Amazon, PayPal тощо.

– завантаження сервера, оскільки JavaScript працює на стороні клієнта, перевірка даних можлива в самому браузері, а не надсилання їх на сервер. У разі будь-яких розбіжностей весь веб-сайт не потрібно перезавантажувати. Браузер оновлює лише вибраний сегмент сторінки.

– розширена функціональність, Сторонні доповнення, такі як Greasemonkey (розширення Mozilla Firefox), дозволяють розробникам додавати фрагменти попередньо визначеного коду в свій код, щоб заощадити час і гроші. Ці додатки допомагають розробникам створювати програми JavaScript набагато швидше і легше, ніж інші мови програмування.

– універсальність, JavaScript можна розробляти як на передньому, так і на внутрішньому плані. Внутрішня розробка використовує NodeJS, тоді як багато бібліотек допомагають у розробці інтерфейсу, наприклад AngularJS, ReactJS тощо.

React JS — одна з найпопулярніших безкоштовних бібліотек JavaScript з відкритим вихідним кодом. Це надійний інструмент, який використовується для створення привабливого інтерфейсу користувача за допомогою різних компонентів інтерфейсу. Інтерфейс є важливим аспектом веб-програм, оскільки він визначає, як користувач взаємодіє та бачить передню частину вашого веб-сайту. Правильно виконаний інтерфейс забезпечує ефективну взаємодію між користувачем та додатком за допомогою контрастних візуалізацій, чистого та мінімалістичного дизайну.

Він також масштабований, що означає, що проект може використовувати стільки React, скільки потрібно, від створення лише кількох інтерактивних елементів до використання його для створення всіх сторінок. Це робить React справжнім інструментом розробки програмного забезпечення, особливо якщо

мова йде про односторінкові веб-програми, які з кожним днем стають все більш популярнішими. [4]

Тож основними перевагами використання бібліотеки ReactJS є:

– легкість у вивченні, на відміну від фреймворків JS, таких як Angular і Ember, React дуже легко і швидко вивчати навіть для тих, хто має базові знання JavaScript і HTML. Чим краще ви розумієте JavaScript, тим легше освоїти React до рівня, на якому ви можете максимально використовувати його в проектах.

– flux-архітектура, це архітектурний шаблон, що відповідає за безперебійну роботу сховища в програмах на базі JS. Використовуючи термінологію архітектури MVC, React обробляє компонент View, а Flux відповідає за модель. Дії реагують на взаємодії користувача з інтерфейсом і передають отримані дані диспетчеру, який надалі передає дані у Stores, що мають відповідний стан. Після оновлення Stores Views отримують з них відповідні дані, самі оновлюються та оновлюють дочірні Views. З умовою того що додаток стає більшим та складнішим, функціонал описаний вище, залишається стабільним та ніколи не порушується, таким чином мінімізуючи помилки в потоці даних.

– сумісність, React створений для легкої роботи з іншими бібліотеками JS і не обмежує можливості використання інших інструментів для створення веб-сайтів. Він дозволяє вибрати будь-яке відповідне розширення React, яке за допомогою зовнішніх контролерів визначатиме рівень перегляду сторінки. Іноді достатньо працювати з React над простими сторінками, але чим складніша сторінка, тим більше бібліотек їй потрібно для належного функціонування. [5]

React Bootstrap – це корисний інструмент для швидкої верстки будь-якої інтернет сторінки. Завдяки React Bootstrap, спеціаліст економить значну кількість часу, адже не доводиться витрачати час на адаптивну верстку. Також, не треба витрачати час на прописання стилів, в React Bootstrap вже є значна

кількість різноманітних об’єктів, які можно підлаштувати під себе, та використовувати у проєкті. [6]

Axios – це бібліотека, яка в проєкті використовується як інструмент запиту на сервер. Axios може працювати в браузері та node.js з тією ж самою базою кодів. На стороні сервера він використовує нативний node.js http-модуль, тоді як на стороні клієнта (браузер) він використовує XMLHttpRequests.

Якщо казати про Axios, можно виділити декілька особливостей:

- робить XMLHttpRequests запити з браузера;
- робить http запити з node.js;
- підтримує Promise API;
- перехоплює запити та відповіді;
- перетворює дані запиту та відповіді;
- скасує запити;
- автоматичне перетворення для JSON даних;
- підтримка на стороні клієнта для захисту від XSRF.

У React є своя система маршрутизації, яка дозволяє зіставляти запити до застосування з певними компонентами. Ключовою ланкою у роботі маршрутизації є модуль react-router, який містить основний функціонал роботи з маршрутизацією. Але якщо працювати у браузері, треба використовувати модуль react-router-dom. [7]

MobX — це бібліотека, що робить керування станом програми простим і масштабованим, використовуючи функціонально-реактивне програмування. React та Mobx разом – потужна комбінація. React малює стан програми, надаючи механізми для переведення його в дерево відображуваних компонентів. MobX надає механізм зберігання та оновлення стану програми, яка потім може використовувати.

React.React та MobX представляють оптимальне вирішення загальних проблем розробки програм. React дає нам можливість оптимально малювати інтерфейс за допомогою Virtual DOM. MobX дозволяє синхронізувати стан між

React-компонентами, використовуючи реактивну віртуальну залежність графічного стану, який оновлюється лише коли це дійсно потрібне.

Для створення серверної частини інтернет-магазину були обрані та використані наступні технології:

- NodeJS;
- Express;
- Sequelize;
- PostgreSQL.

Node.js — це середовище виконання JavaScript з відкритим вихідним кодом і міжплатформним. Програма Node.js працює в одному процесі, не створюючи новий потік для кожного запиту. Node.js містить у своїй стандартній бібліотеці набір асинхронних примітивів вводу-виводу, які запобігають блокування коду JavaScript, і загалом бібліотеки в Node.js написані з використанням неблокуючих парадигм, що робить поведінку блокування скоріше винятком, ніж нормою.

Node.js добре підходить для створення додатків, які вимагають будь-якої форми взаємодії користувачівабо спільної роботи в реальному часі - наприклад, сайтів-чатів або додатків, таких як CodeShare, в яких кілька користувачів можуть спостерігати в реальному часі за редагуванням документа кимось іншим.

Node.js також добре підходить для створення API, які використовуються при обробці багатьох запитів, керованих введенням виведенням (наприклад, які повинні виконувати операції з базою даних), або для сайтів, пов'язаних з потоковою передачею даних, оскільки Node.js дає можливість обробляти файли по ходу їх завантаження.

Express - це мінімалістичний та гнучкий веб-фреймворк для Node.js, що надає широкий набір функцій для мобільних та веб-додатків. Перевагою саме цього фреймворку є в його швидкості та легкості, що дозволяє робити розробку веб-додатків на Node.js більш ефективнішою, та структурованою. Також Express включає в себе різні проміжні модулі, які можно використовувати для виконання додаткових завдань, на запити та відповіді.

У якості ORM для реляційних баз даних у проекті використовується Sequelize. Sequelize також являє собою бібліотеку для додатків на Node.js, яка здійснює зіставлення таблиць у базі даних та відносин між ними з класами. Використовуючи Sequelize, не треба писати SQL-запити, у нас є можливість працювати з даними як з звичайними об’єктами, саме це робить її зручною у роботі над великим проєктом. Крім того, Sequelize може працювати з рядом СУБД - MySQL, Postgres, MariaDB, SQLite, MS SQL Server.

PostgreSQL в проекті використовується як система управління базами даних. Швидкість, безпека та надійність PostgreSQL роблять його придатним для 99% додатків, тому це чудова стартова точка для будь-якого додатка. Важливо те, що він одночасно підтримує різні типи даних в одних і тих же таблицях даних. Було доведено, що PostgreSQL має високу масштабованість як за величезною кількістю даних, якими він може керувати, так і за кількістю одночасних користувачів, які він може вмістити. У виробничих середовищах є активні кластери PostgreSQL, які керують багатьма терабайтами даних, і спеціалізовані системи, які керують петабайтами. Тож, PostgreSQL залишається актуальною системою управління базами даних навіть у 2022 році.

2.4 Опис структури системи та алгоритмів її функціонування

Інформаційная система продажу одягу складається з двух основних частин, а саме серверной частини, та клієнтської частини (рис. 2.1). В процесі створення кваліфікаційного проекту, було забезпечено логічне з'єднання усім файлам проекту, також вони були об'єднані спільним функціоналом.

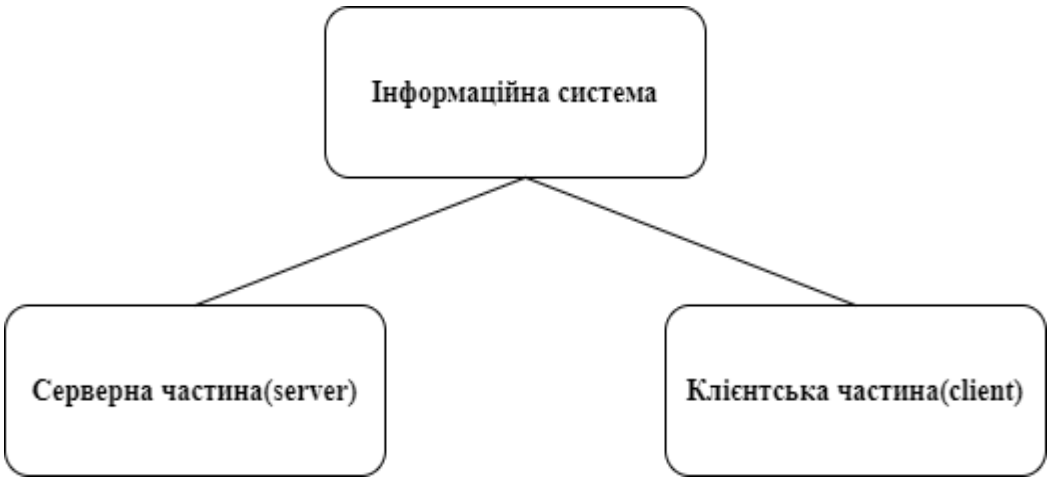

Рис. 2.1. Дві основні части проекту

Почнемо з структури серверної частини. Папка server (рис.2.2) містить в собі кореневий файл, з якого запускатиметься серверна частина програми, а саме index.js.

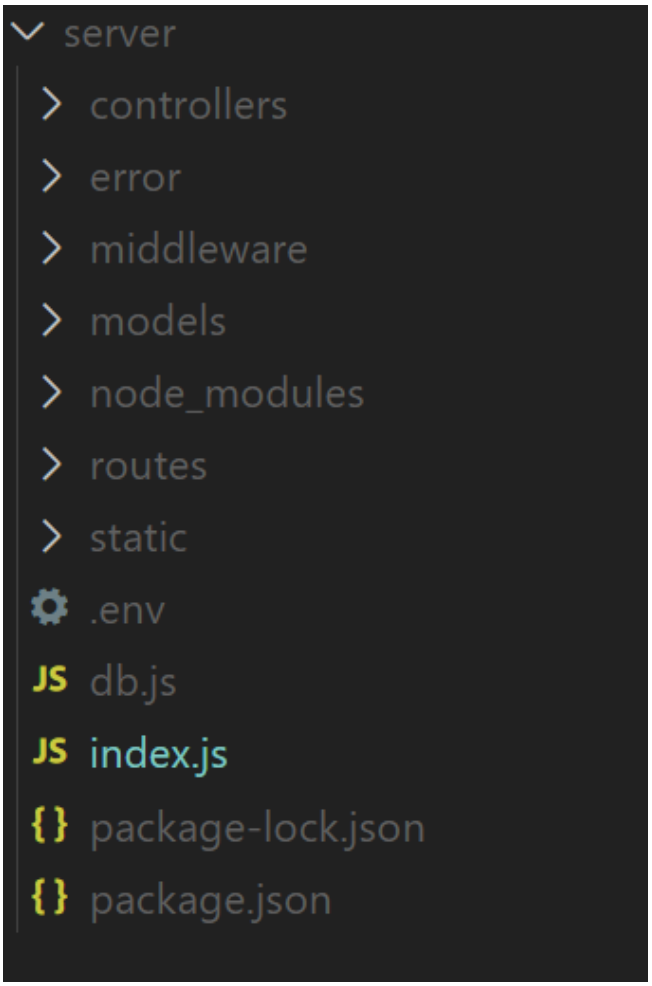

Рис. 2.2. Структура папки server

Після ініціалізації проекту у папці має з'явитися файл package.json. У package.json знаходиться опис проекту, залежності, які використовуються в проекті. Також у package.json знаходиться скрипт, який запускає додаток у режимі розробника.

Файл .env служить як змінне оточення, у нього виноситься вся конфігурація. Наприклад, порт, назва бази даних, пароль бази даних, хост тощо.

Підключення до бази даних забезпечує файл db.js, він є одним з найважливіших файлів серверної частини.

У папці models (рис.2.3), де знаходиться файл models.js описуються моделі даних, їх типи, назви та ідентифікатори поля.

Рис. 2.3. Структура папки models

Для реалізації системи маршутів була створена папка routes(рис.2.4). В якій знаходяться основні маршути, реалізовані в інформаційній системі. А також index.js, який створений для об'єднання всіх цих маршутів.

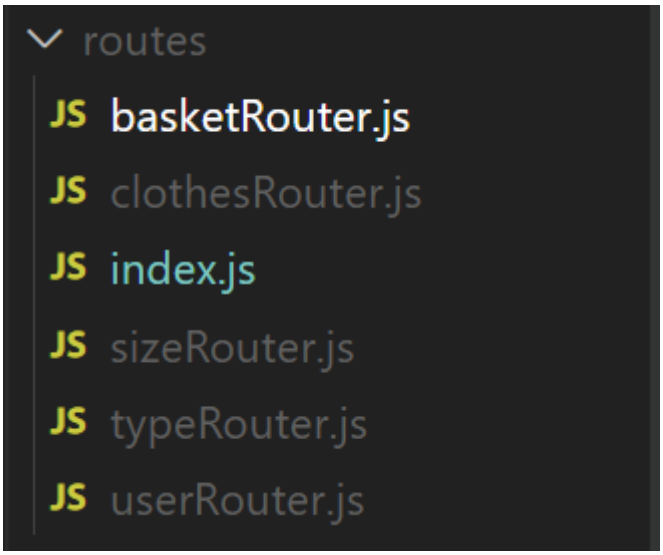

Рис. 2.4. Структура папки routes

Для реалізації функціоналу для кожної сторінки створено папку controllers (рис.2.5). Там для кожної сторінки буде створено свій функціонал, як, наприклад, у файлі userController.js створена можливо реєструвати користувача і авторизувати.

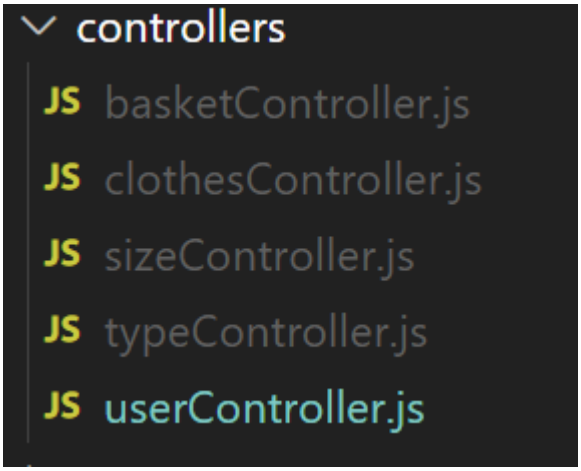

Рис. 2.5. Структура папки controllers

Папка error (рис.2.6) з файлом ErrorTool.js, допомагають фахівцю та навіть користувачеві отримувати інформацію про потрібну помилку. ErrorTool.js відловлює помилку та надає інформація, яка саме помилка виникла під час виконання роботи програми.

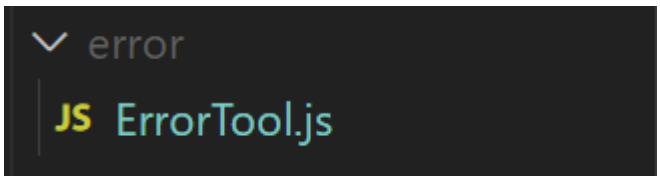

Рис. 2.6. Структура папки error

Також як інструмент для перевірки деякого функціоналу на умови була створена папка middleware (рис.2.7), всередині якої знаходиться authMiddleware – який перевіряє функції на умову авторизації, roleMiddleware – перевірка ролі при додаванні контенту та ErrorMiddleware – для повноцінної роботи ErrorTool.js

Рис. 2.7. Структура папки middleware

У папці static (рис.2.8) зберігаються зображення одягу, які надсилають із клієнта.

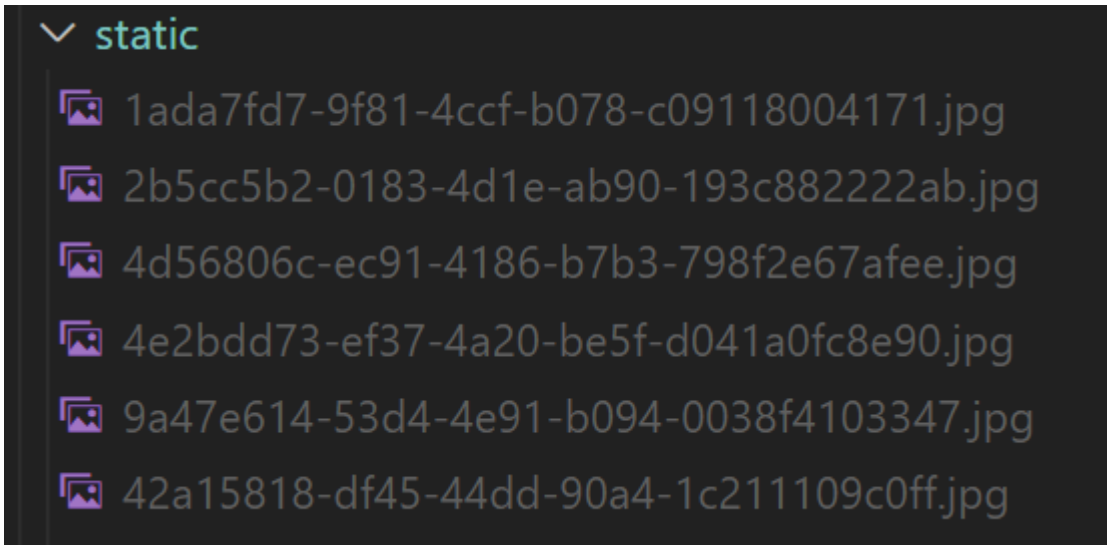

Рис. 2.8. Структура папки static

Папка node-modules потрібна для розробки, в ній знаходяться різноманітні бібліотеки, які були використані у проекті.

Перейдемо до клієнтської частини програми. Після встановлення додаткових компонентів структура папки client виглядає так (рис. 2.9).

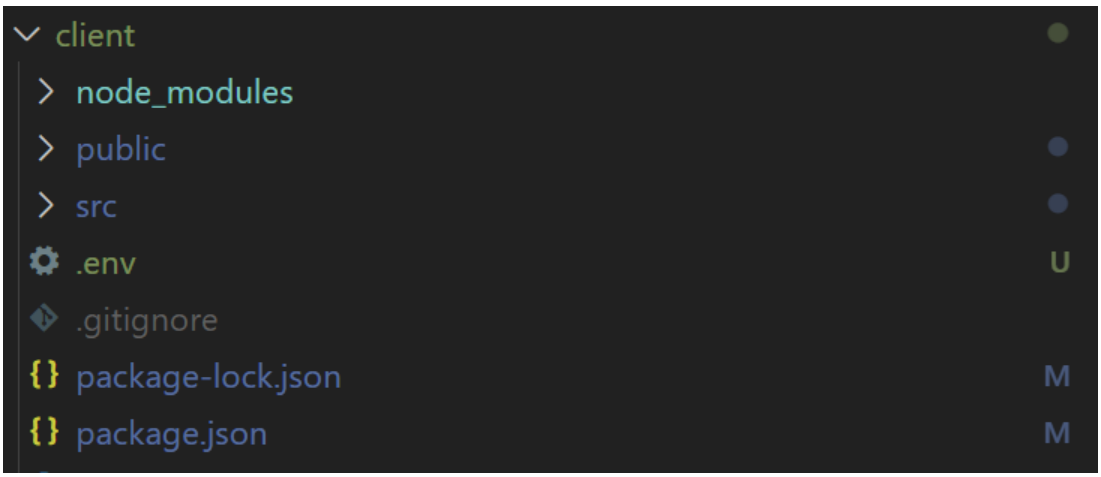

Рис. 2.9. Структура папки client

Папка src (рис.2.10) є основою клієнтської частини. App.js є основним компонентом додатку. У store проходить взаємодія з бібліотекою mobx, а також зберігаються дані про одяг, користувача та кошик. У папці page знаходяться кореневі компоненти, які є сторінками. Папка component, відповідно, є папкою для компонентів сторінки магазину. У папці routes.js описані всі маршути до конкретних сторінок, які є в нашому додатку.

Рис. 2.10. Структура папки src

Щоб не було плутанини з маршрутами в routes.js, виносимо їх як константу в папку utils (рис. 2.11), файл consts.js.

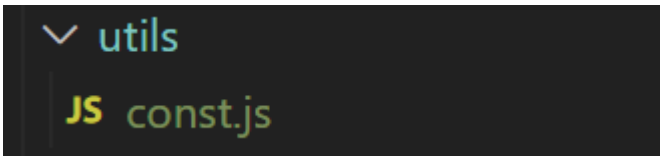

Рис. 2.11. Структура папки utils

Якщо детальніше розглянути структуру папки store (рис. 2.12), бачимо три файли сховища, BasketStore – для кошика, UserStore – для користувача, ClothesStore – для одягу.

Рис. 2.12. Структура папки store

Як і описувалося вище в папці page (рис.2.13) знаходяться компоненти, які є сторінками нашого додатка.

AdminPage.js є сторінкою адмін-панелі, з доступом рівня “ADMIN”. Завдяки цій сторінці менеджер склад може доповнювати інформаційну систему контентом.

Auth.js є сторінкою для авторизації та реєстрації. Завдяки цій сторінці, відповідно, користувач може реєструватися або авторизуватися на сайті.

ClothesPage.js сторінка, де відображається окремий елемент товару. Завдяки цій сторінці користувач може ознайомитись з одним окремим елементом товару.

BasketPage.js сторінка для корзини користувача.

MainPage.js головна сторінка сайту. Користувач її бачить щойно заходить на сайт.

Shop.js сторінка, де знаходяться всі товари магазину. На цій сторінці користувач може ознайомитись з усім асортиментом інтернет-магазину.

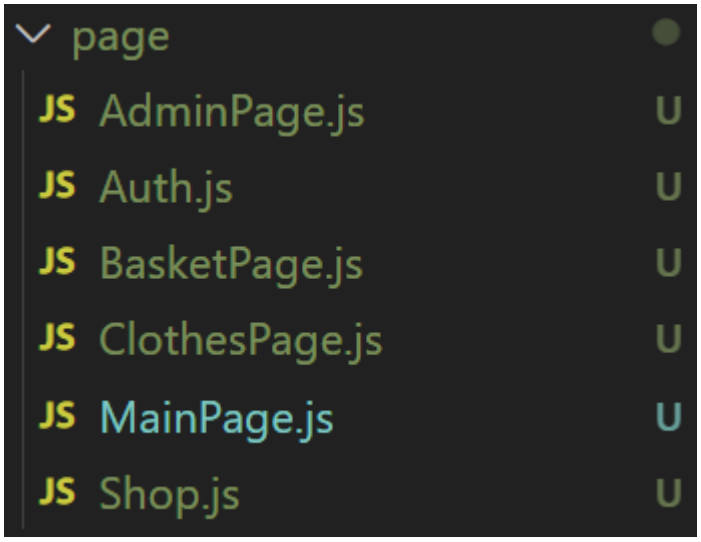

Рис. 2.13. Структура папки page

Перейдемо до детального розгляду папки component (рис. 2.14). Для реалізації навігації сторінками інтернет-магазину було створено функціонал у файлі AppRoute.js.

Basket.js є компонентом частини навбару. Він представляє картинку кошика в кутку навбару, яка динамічно змінюється в залежності від додавання користувачем нового товару в кошик.

ClothesItem.js є компонентом сторінки товару, а саме карткою з інформацією про ціну товару, назву товару, та побачить картинку з зовнішнім виглядом товару. Відповідно, завдяки цьому компоненту користувач отримає інформацію про товар, його ціну, назву, та може побачити як він виглядає. Також слід зазначити, що ClothesItem.js є піделементом ClothesList.js.

За допомогою ClothesList.js картки з одягом виводяться у рядок. А також він отримає інформацію с серверної частини, а передає дані в ClothesItem.js.

NabBar.js є інформативною частиною сайту, де зображене лого інтернет-магазину, а також виконує функції навігації, та дає змогу переходити по сторінкам сайту.

Завдяки OneClothesinBasket.js користувач може перевірити своє замовлення, після того як він обрав бажаний товар.

Pages.js реалізовано посторінковий вивод.

SizeBar.js потрібен для реалізації фільтру по розміру одягу. Завдяки цьому компоненту користувач може швидко здійснити пошук потрібного йому розміра.

TypeBar.js аналогічним компоненту SizeBar, відзнакою лише є те, що фільтрація здійснюється по типу товару.

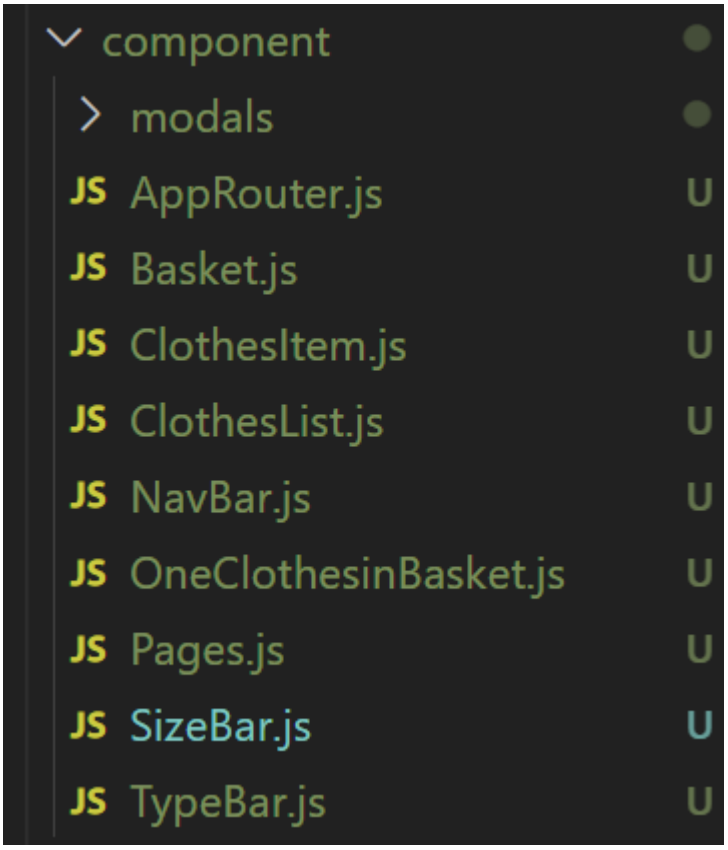

Рис. 2.14. Структура папки component

Папка modals (рис. 2.15) містить в собі три модальних вікна, які були сторені для зручності додавання інформації як тип, розмір та сам одяг.

За допомогою CreateClothes.js користувачу з рівнем доступа “ADMIN”, можно додати одяг на сторінку з одягом.

За допомогою CreateSize.js користувачу з рівнем доступа “ADMIN”, можно додати розмір одягу до компонента SizeBar.

За допомогою CreateType.js користувачу з рівнем доступа “ADMIN”, можно додати тип одягу до компонента TypeBar.

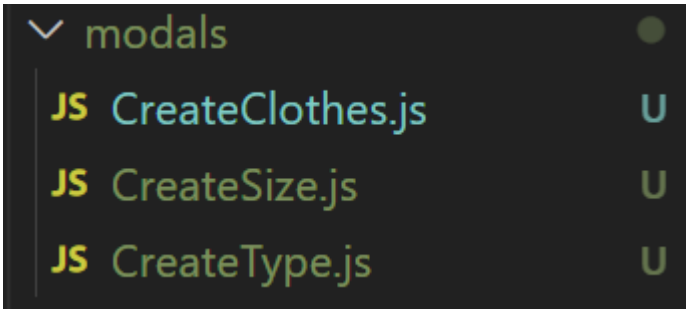

Рис. 2.15 Структура папки modals

Для реалізації можливості взаємодії з сервером було створено папку http (рис. 2.16). У файлі userApi.js реалізована функція авторизації, реєстрації та функція перевірки токена на валідність.

clothesApi.js був створений для отримання типів, розмірів та самого товару. А також для можливості створювати їх.

index.js був створений саме для з’єднання з сервером, а також для налаштування axios.

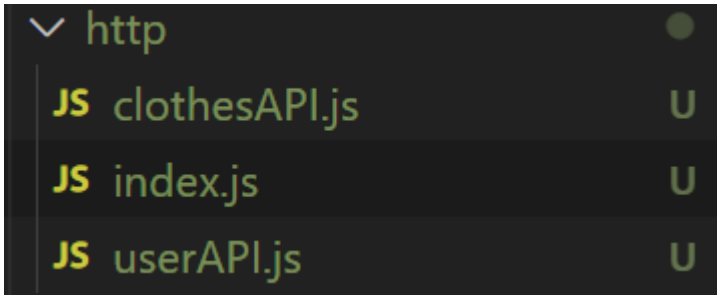

Рис. 2.16. Структура папки http

У папці assets (рис. 2.17) зберігається графічний контент для сайту у вигляді зображень різного формату.

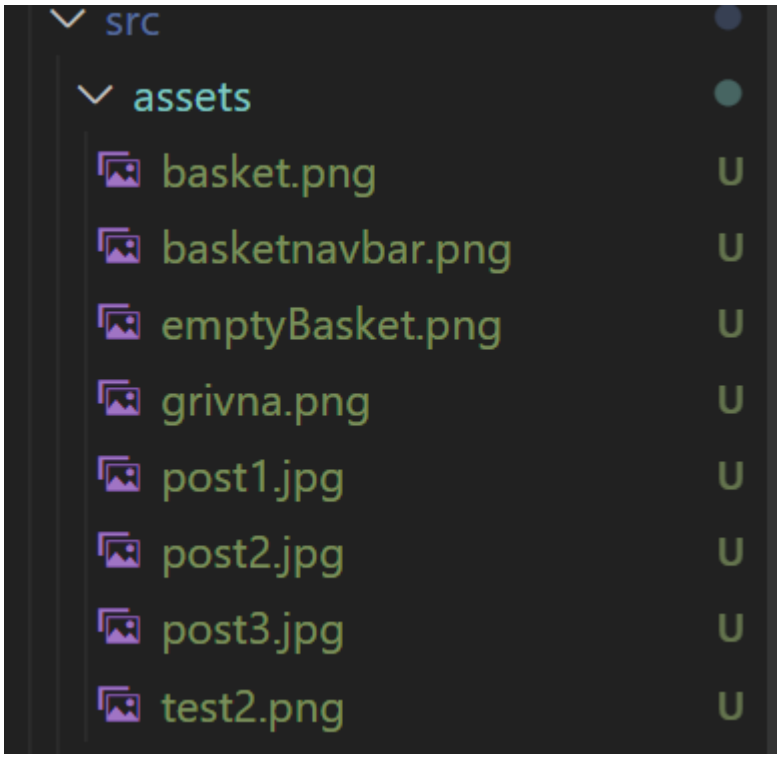

Рис. 2.17. Структура папки assets

Якщо розглядати структуру бази даних (рис. 2.18), яка була використана у розробці серверної частини, то вона буде мати наступний вигляд:

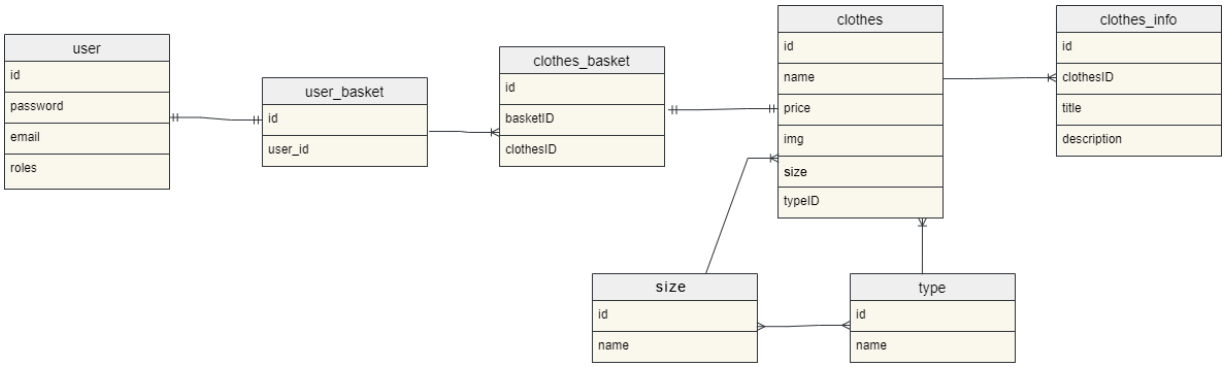

Рис. 2.18. Структура бази даних інтернет-магазину

Також, була побудована схема варіантів використання (рис. 2. 19) інформаційної системи для кожного користувача, з різними рівнями доступу.

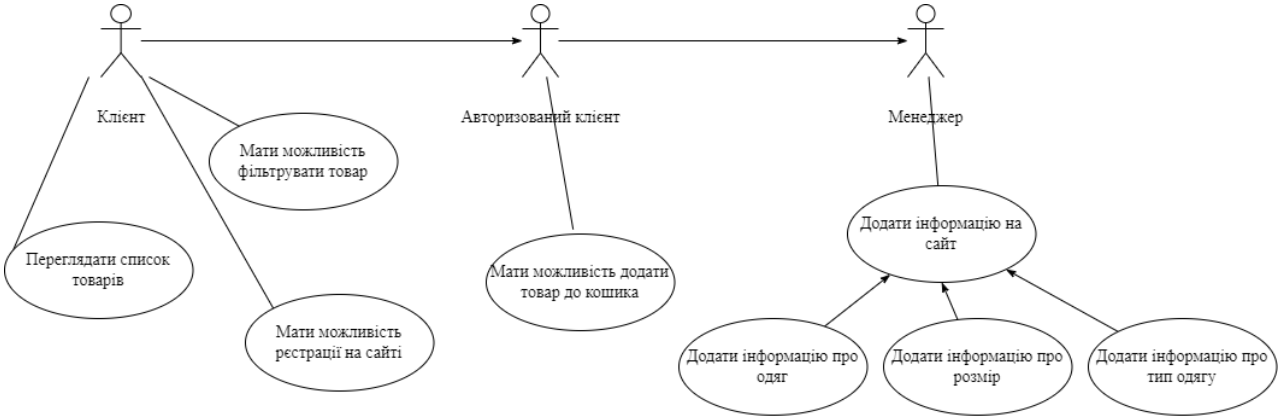

Рис. 2.19. Варіанти використання(useCase)

2.5 Обґрунтування та організація вхідних та вихідних даних програми

В даній кваліфікаційній роботі вхідними даними є ті, які користувач вводить у клієнтській частині. Наприклад дані під час реєстрації, авторизації. Також вхідними даними є інформація про одяг, розмір, та тип одягу. Вхідним даними, можуть бути дані фільтрації одягу по певним критеріям, як наприклад фільтр по тип одягу та по розміру.

Вихідними даними інформаційної системи є дані, які зберігаються у базі даних. Завдяки функціоналу клієнтської частини, вони можуть бути представлені у зручному графічному вигляді, як наприклад інформація з картками одягу.

Обмін даними між клєнтською та серверною частинами здійснюється завдяки формату JSON.

2.6 Опис розробленої системи

2.6.1 Використані технічні засоби

Для створення та тестування інформаційної системи була використана персональна ЕОМ, з наступними системними характеристиками:

- Операційна система: Windows 10 Домашняя;
- Оперативна пам'ять: 16Гб;
- Процесор: AMD Ryzen 5 1600 Six-Core Processor 3.20 GHz;
- 500Мб вільного місця на жорстокму диску;
- Монітор;
- Клавіатура та комп’ютерна миша;

2.6.2 Використані програмні засоби

Для розробки інформаційної системи я обрав Visual Studio Code.

Visual Studio Code – це легкий у використанні редактор віходного коду, який працює на багатьох операційних системах, таких як macOS, Windows, Linux. Він зроблений одразу з вбудованою підтримкою JavaScript, TypeScript та Node.js, що є великим плюсом у розробці веб-застосунку. Ключовий фактор, завдяки чому Visual Studio Code є таким зручним, це можливість встановлювати додаткові розширення для інших мов та навіть фреймворків. Завдяки цьому процесс написання коду є більш зручним для спеціаліста, та у фінальному вигляді код має більш цільну структуру.

В якості інструмента для тестування праці серверної частини, в проекті була використана программа Postman. Вона призначена для тестування роботи API, а також для відправки запитів POST і GET. У відмінності від схожих утиліт curl, вона має графічний інтерфейс, тому легко осваюється навіть новичками і є більш гнучкою у використанні. Тестування інтерфейсу API проводиться шляхом аналізу точності вихідних даних залежно від поданих під час вхідного запиту. Цим і займається Postman: він складає та відправляє їх на зазначені URL, отримує назад та зберігає у базі даних. Програма Postman дозволяє надсилати файли, а не лише текстові дані, що є душе зручною функцією на етапі тестування додавання даних про одяг. Якщо на сайті використовується захист з авторизацією за методом Basic Auth, програма Postman дозволяє перевірити її проходження.

2.6.3 Виклик та завантаження програми

Для роботи з інформаційною системою достатньо спочатку запустити северну частину, командою у терміналі. Після успішного запуску северної частини, треба запустити клієнтську, також командою у терміналі.

2.6.4 Опис інтерфейсу користувача

Після запуску інформаційної системи користувач потрапляє на головну сторінку інтернет-магазину, MainPage (рис. 2.20). На головній сторінці знаходиться NavBar який виконує функцію навігації по сторінках. Також на головній сторінці є слайдер з трьома зображеннями. Інтерфейс головної сторінки є адаптивним (рис. 2.21), його буду зручно переглядати як на звичайних моніторах, так и на мобільних пристроях.

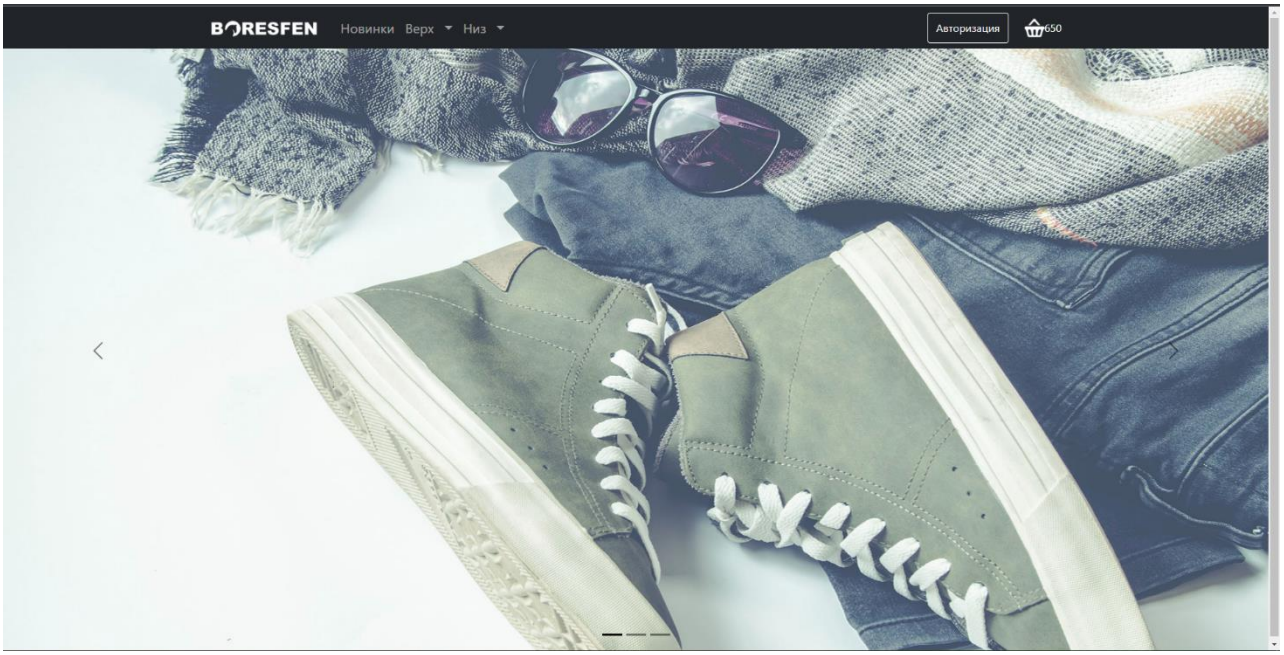

Рис. 2.20. Головна сторінка, MainPage

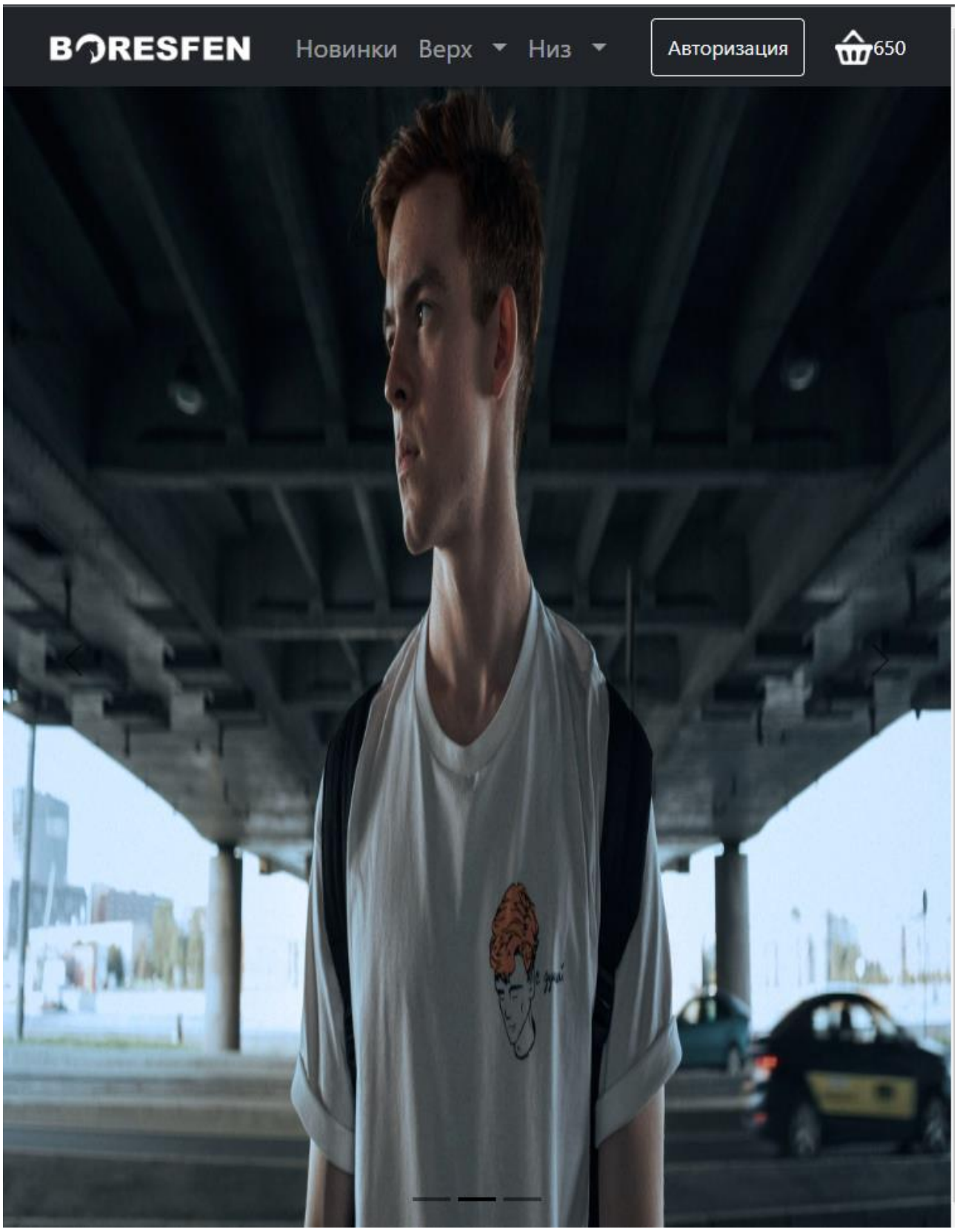

Рис. 2.21. Головна сторінка, MainPage, 768x1024

При детальнішому розгляді компонента NavBar (рис. 2.22), бачимо що було створено три посилання, з налаштованою маршутизацією на сторінку Shop,

кнопка “Авторизація” на сторінку Auth. Якщо кошик користувача порожній, компонент Basket(рис. 2.23) відрисовує нам компонент BasketPage (рис. 2.24), якщо у кошику є товар, компонент OneClothesinBasket (рис. 2.25).

Рис. 2.22. Компонент NavBar

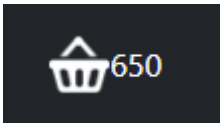

Рис. 2.23. Компонент Basket

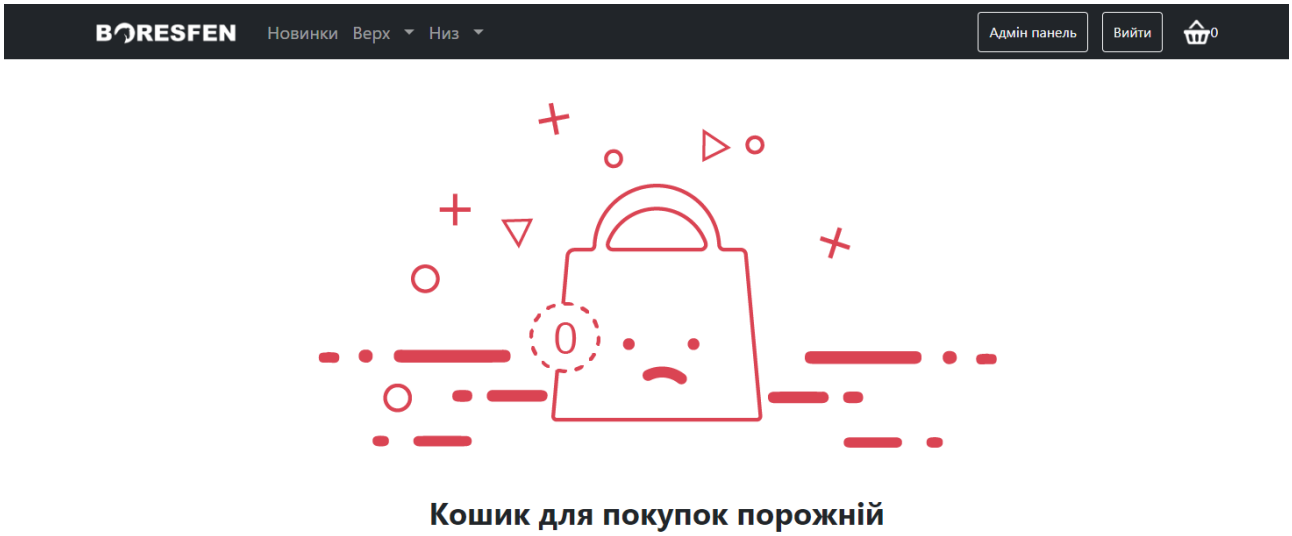

Рис. 2.24. Компонент BasketPage

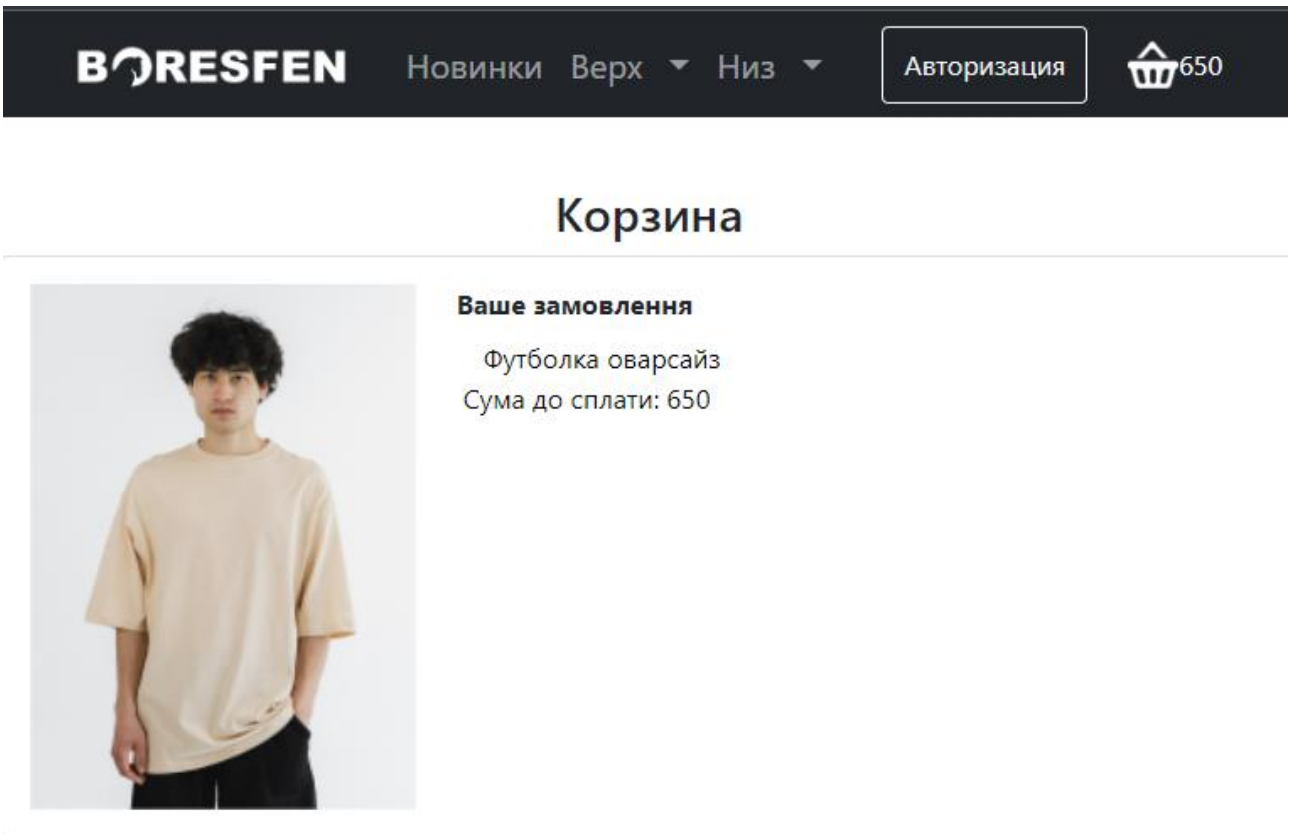

Рис. 2.25. Компонент OneClothesinBasket

На сторінці Shop (рис. 2.26) знаходиться картки з інформацією про одяг, з якою користувач може ознайомитимся, а також фільтри для пошуку потрібного одягу. Сторінка Shop також є адаптивною (рис. 2.27).

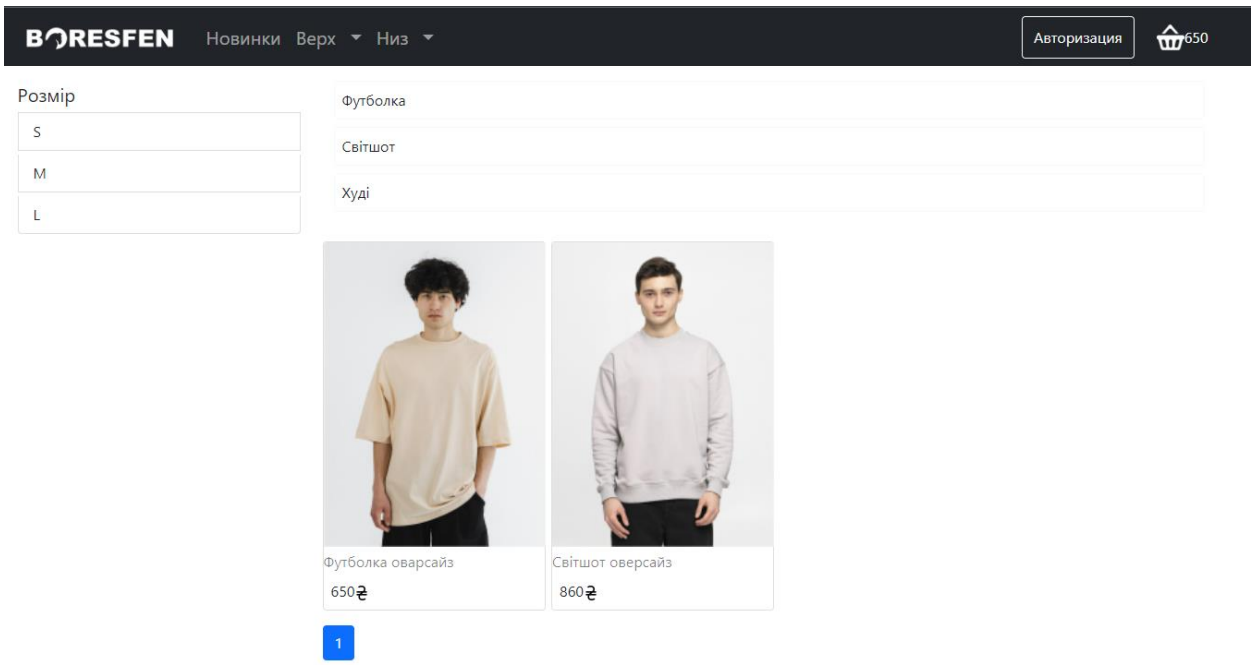

Рис. 2.26. Сторінка Shop

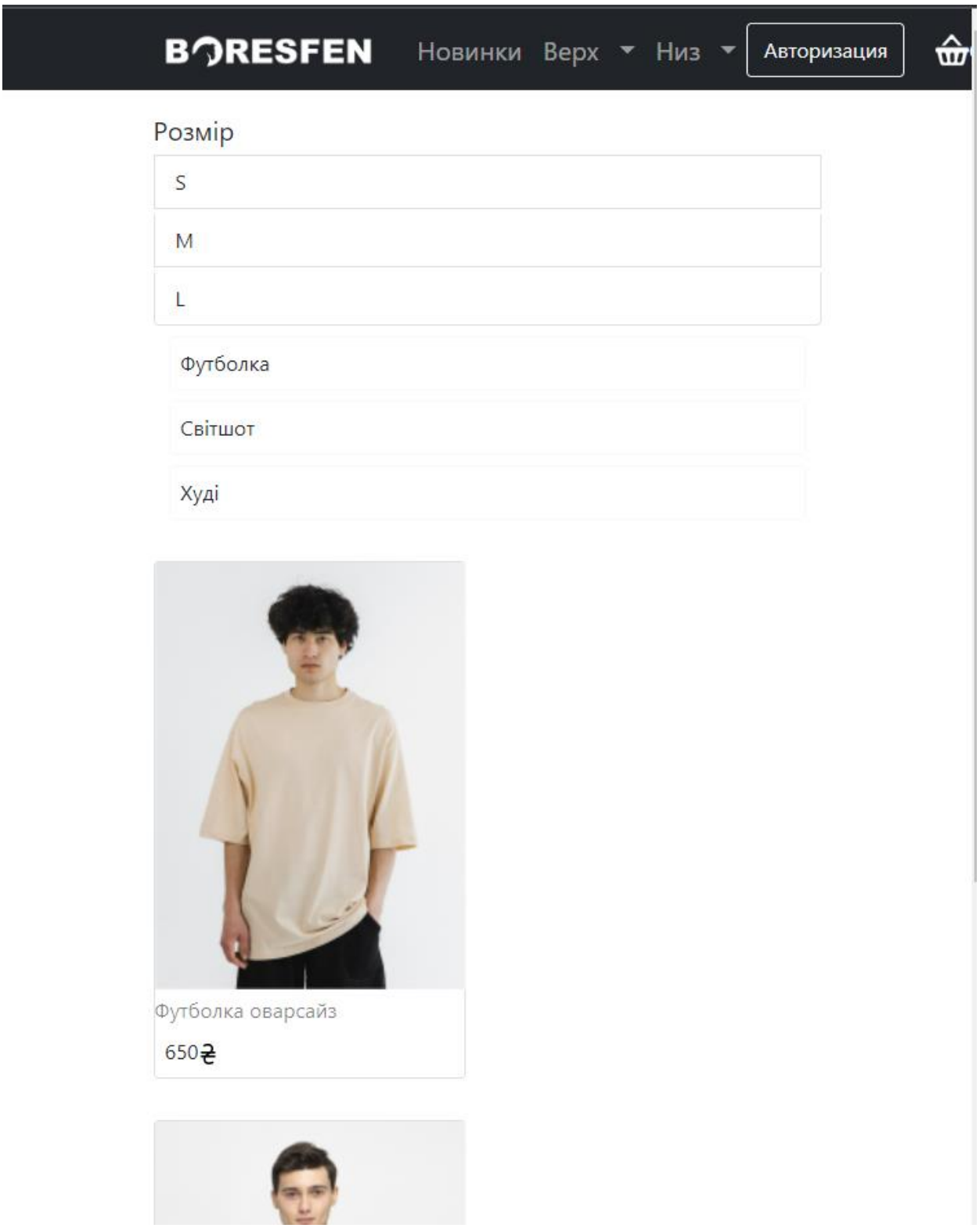

Рис. 2.27. Сторінка Shop, 768x1024

Після натискання на крапку з одягом, користувач переходить на сторінку ClothesPage (рис. 2.28), де може додати товар до кошику. Сторінка ClothesPage також є адаптивною (рис. 2.29) для мобільних пристроїв.

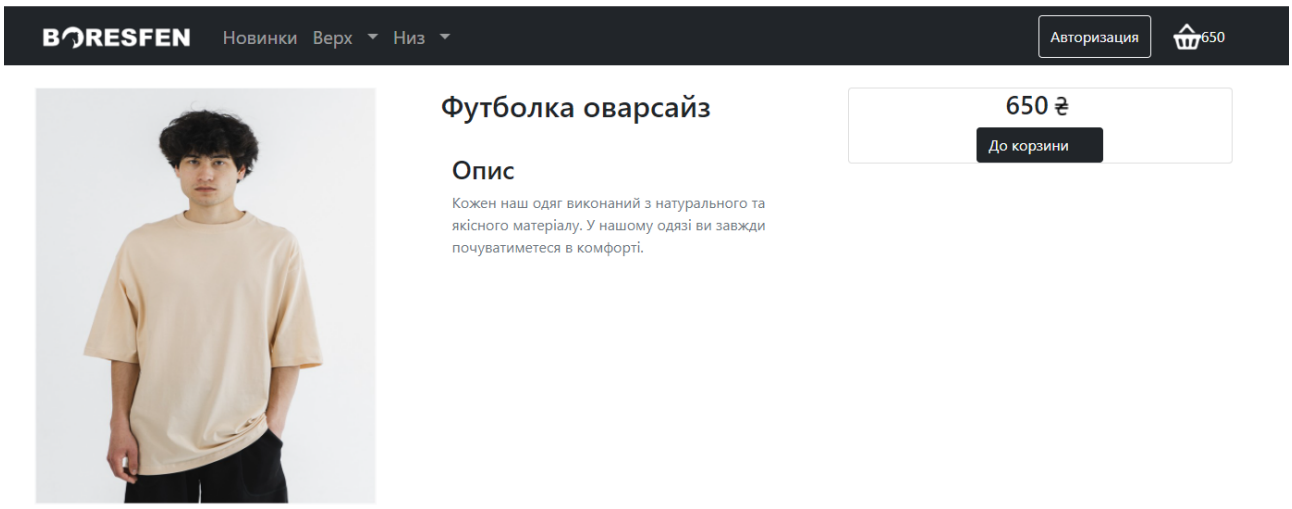

Рис. 2.28. Сторінка ClothesPage

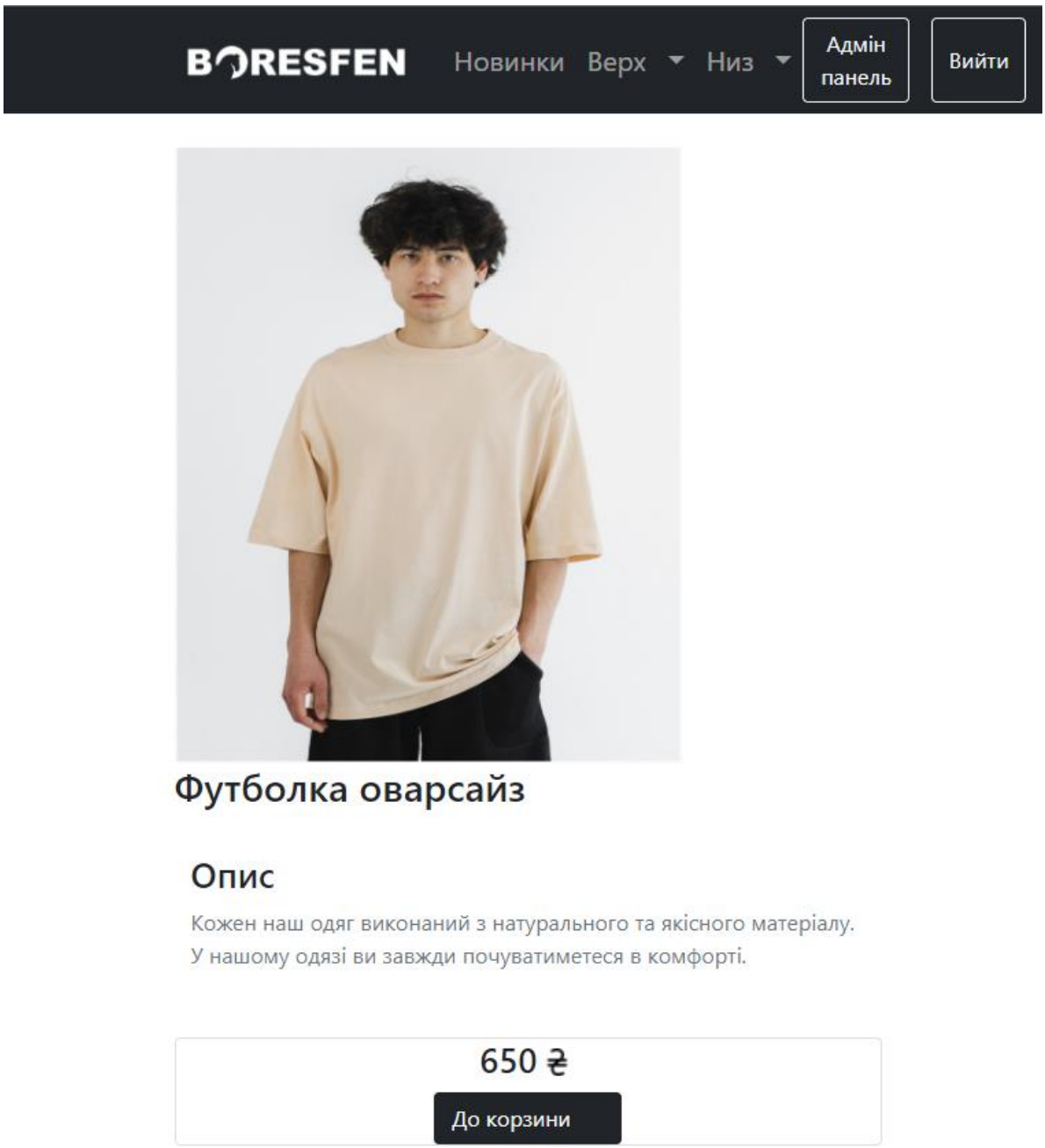

Рис. 2.28. Сторінка ClothesPage, 768x1024

Для додавання менеджмент складом нового контенту в інтернет-магазин є адмін-панель, AdminPage (рис. 2.29).

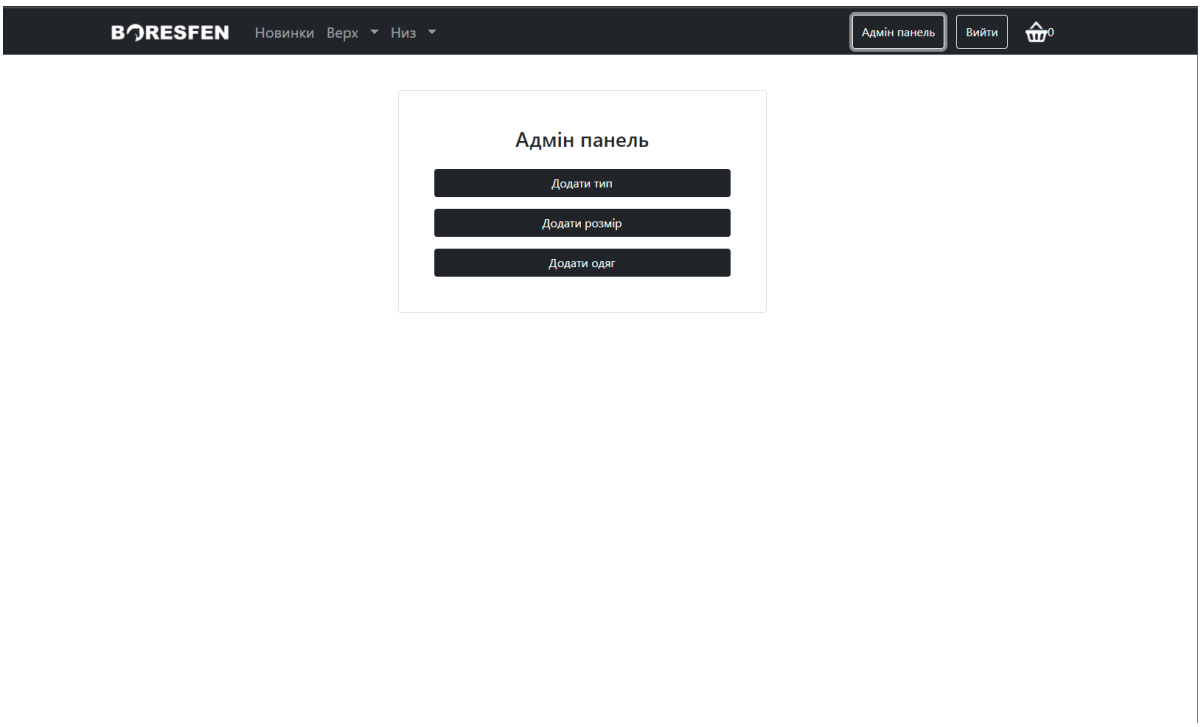

Рис. 2.29. Сторінка AdminPanel

Адмін-панель складається з трьох кнопок, CreateType (рис. 2.30) – для додавання типу одягу, CreateSize (рис. 2.31) – для додавання розміру одягу, CreateClothes (рис.2.32) – для додавання самого одягу.

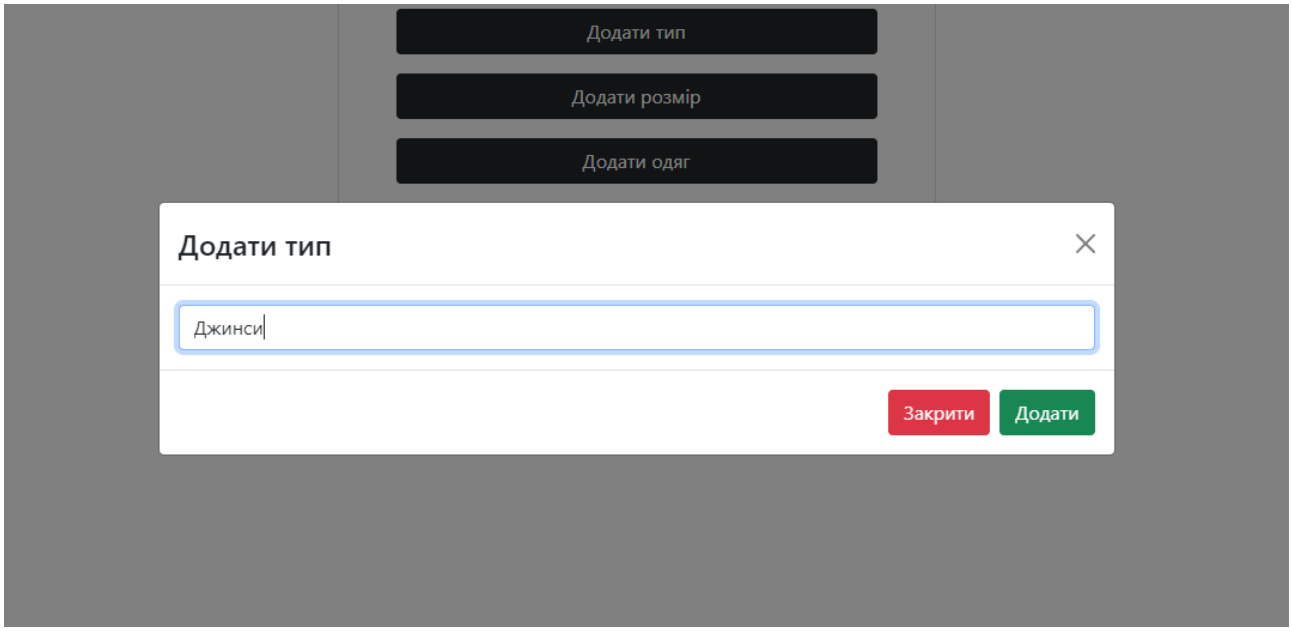

Рис. 2.30 Демонстрація додавання типу одягу

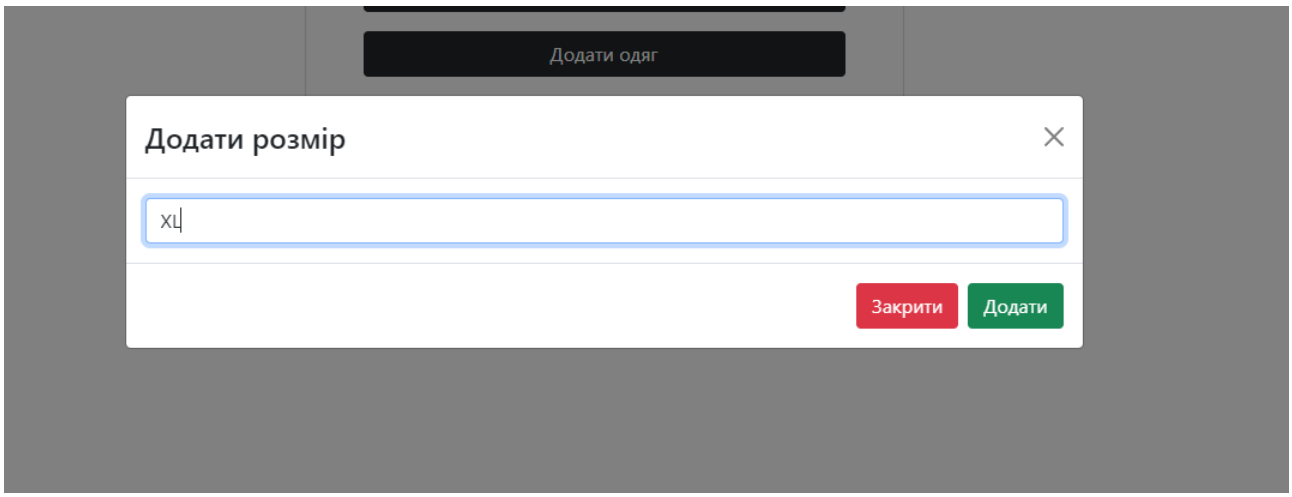

Рис. 2.31. Демонстрація додавання розміру одягу

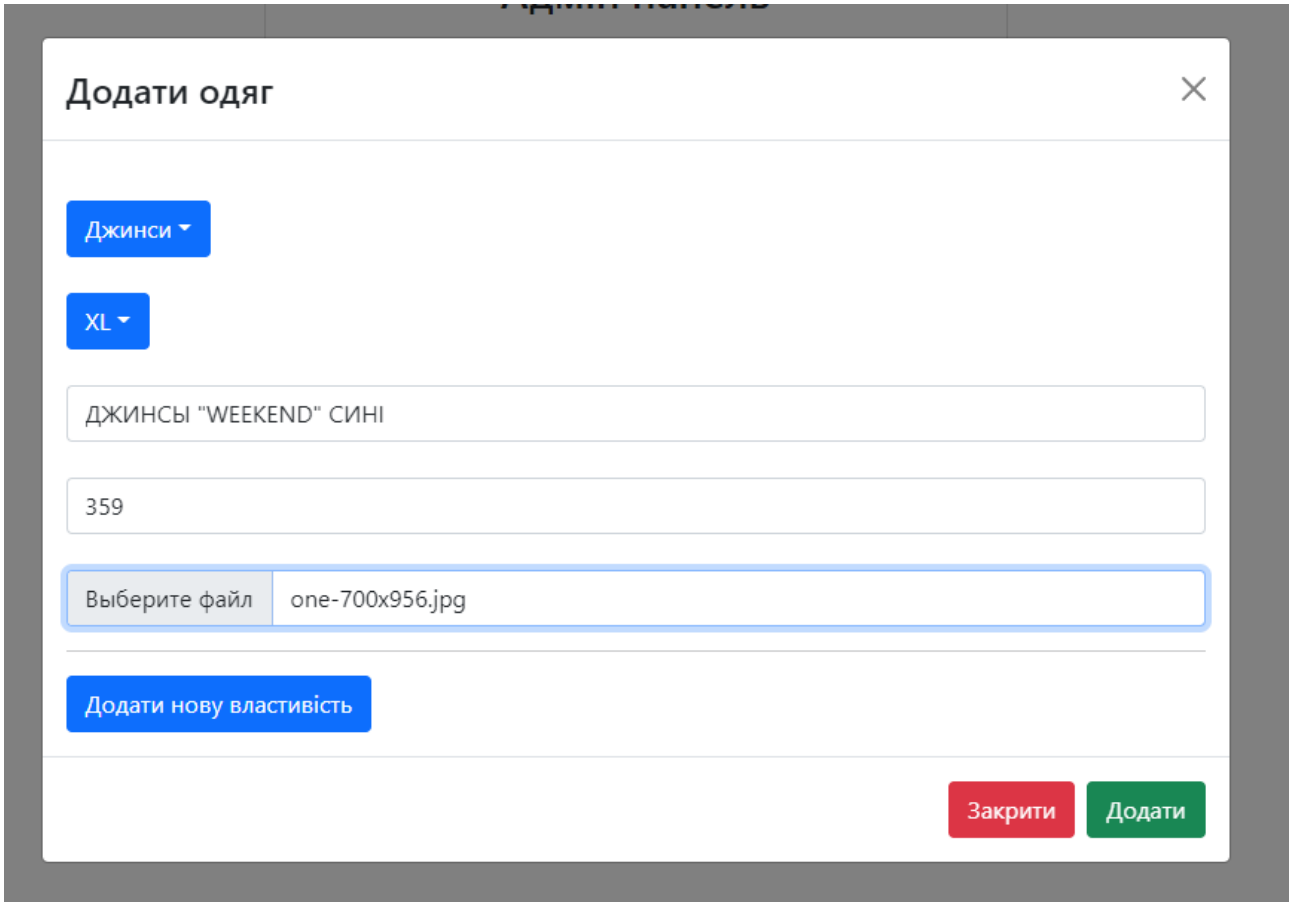

Рис. 2.32. Демонстрація додавання нового одягу

У результаті ми отримаємо новий тип, розмір та нову картку з одягом на сторінці Shop (рис.2.33).

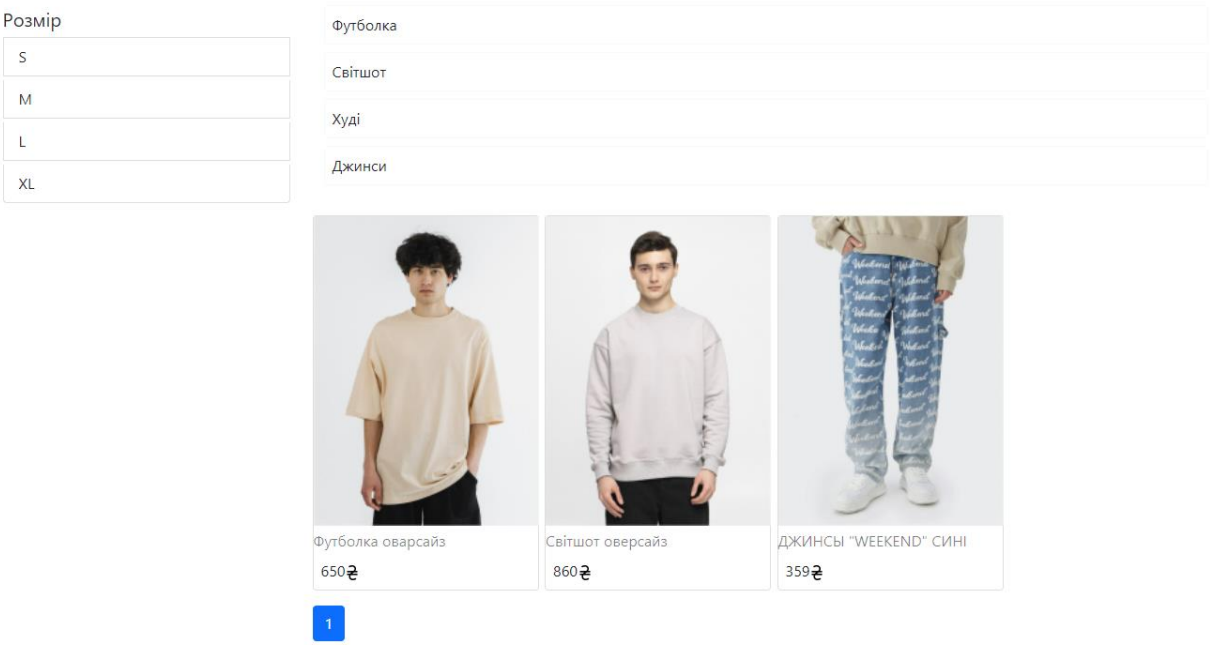

Рис. 2.33. Результат додавання типу, розміру та одягу.

Також, є можливість к фільтрації контента на сторціні користовачем, для отримання саме того одягу, який потрібен користувачеві. На прикладі фільтрація здійснється за параметрами: Розмір - S, Тип – футболка (рис.2.34). Ми отримаємо пусте поле, адже такого товару в базі даних не існує.

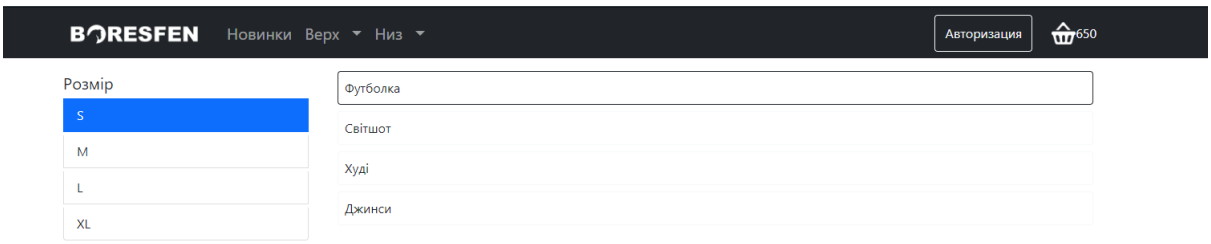

Рис. 2.34. Фільтрація за параметрами: Розмір - S, Тип – футболка

Вибираємо параметри фільтру такі, що відповідають параметрам існуючого одягу у базі даних (рис. 2.35).

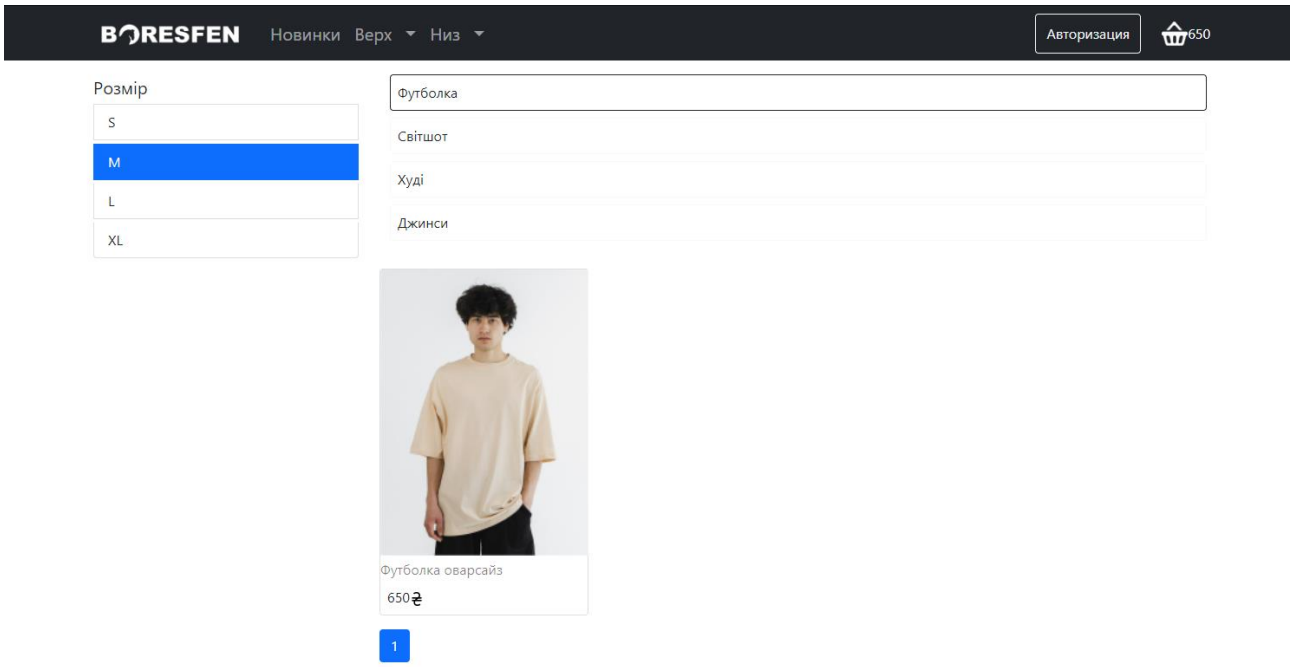

Рис. 2.35. Фільтрація за параметрами: Розмір - M, Тип – футболка

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту.

Початкові дані:

1. передбачуване число операторів програми - 1791;

2. коефіцієнт складності програми – 1,3;

3. коефіцієнт корекції програми в ході її розробки – 0,07;

4. годинна заробітна плата програміста – 94 грн/год;

Середня година зарплати на позиції Full stack developer відповідно даних сайту Work.ua [21](дата звернення 29.05.22) складає 50000 грн. на місяць. При восьмигодинному робочому дні (176 годин у місяць в середньому) середня зарплата за годину буде становити 284 грн.

5.коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,3;

7. вартість машино-години ЕОМ – 30 грн/год

Нормування праці в процесі створення веб-додатку істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки веб-додатку може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин}, \qquad (3.1)$$

де t_o– витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u– витрати праці на дослідження алгоритму рішення задачі;

t_a– витрати праці на розробку блок-схеми алгоритму;

$t_п$– витрати праці на програмування по готовій блок-схемі;

$t_{отл}$– витрати праці на налагодження програми на ЕОМ;

$t_д$– витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

Q=q·C·(1+p), де (3.2)

q– передбачуване число операторів;

C– коефіцієнт складності програми;

p– коефіцієнт кореляції програми в ході її розробки.

Q= 1791·1,3·(1+0,07)=2491,28 людино/годин

Витрати праці на дослідження алгоритму рішення задачі:

$$t_u = \frac{Q \bullet B}{(75 \ldots 85) \bullet K}$$, людино-годин (3.3)

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$t_u = \frac{2491,28 \cdot 1.2}{82 \cdot 1,3} = 28,044$ людино-годин.

Витрати праці на розробку блок-схеми алгоритму:

$$t_a = \frac{Q}{(20 \ldots 25) \square k}$$, людино-годин. (3.4)

де *Q* – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), людино-годин:

$t_a = \frac{2491,28}{20 \cdot 1,3} = 95,818$, людино-годин.

Витрати праці на програмування по завершеній блок-схемі:

$$t_п = \frac{Q}{(20…25) \cdot k}$$, людино-годин. (3.5)

$t_п = \frac{2491,28}{25 \cdot 1,3} = 76,655$, людино-годин.

Витрати праці на налагодження програми на ЕОМ:

За умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k}$$, людино-годин. (3.6)

$t_{отж} = \frac{2491,28}{5 \cdot 1,3} = 383,274$ людино-годин,

За умови комплексного налагодження завдання:

$t_{отл}^{к} = 1,2 \cdot t_{отл}$; (3.7)

$t_{отл}^{к} = 1,2 \cdot 383,274 = 459,929$ людино-годин

Витрати праці на підготовку документації:

$t_д = t_{др} + t_{до}$; (3.8)

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$t_д = \frac{Q}{(15…20) \cdot K}$; (3.9)

$t_д = \frac{2491,28}{18 \cdot 1,3} = 106,465$ людино-годин.

$t_{до}$– трудомісткість редагування, печатки й оформлення документації

$t_{до} = 0,75 \cdot t_{др}$; (3.10)

$t_{до} = 0,75 \cdot 106,465 = 79,845$, людино-годин.

$t_д = 106,465 + 79,845 = 186,31$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t =50+28,044+95,818+76,655+383,274+186,31 = 820,10 людино-годин.

За результатом ми розрахували, що в загальній складності для розробки даного програмного продукту необхідно 820,10 людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення інформаційної системи включають в себе витрати на заробітну плату виконавця і витрат машинного часу.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$– середня годинна заробітна плата програміста, грн/година

$З_{ЗП} = 820{,}10 \cdot 284 = 232908{,}40$, грн.

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год, за умови комплексного налагодження завдання.

$С_{МЧ}$– вартість машино-години ЕОМ, грн/год.

$З_{МВ} = 459{,}929 \cdot 30 = 13797{,}87$ грн.

$К_{ПО} = 232908{,}40 + 13797{,}87 = 246706{,}27$ грн.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \quad (3.14)$$

де B_k- число виконавців;

F_p– місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$T = \frac{820,10}{1 \cdot 176} = 4,66$ міс.

Висновки. На розробку даної інформаційної системи піде 820,10 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 4,66 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення інформаційної системи складатимуть 246706,27 грн.

ВИСНОВОК

За результатом виконання кваліфікаційної роботи було зроблено інформаційну систему для підприємства, яке займається продажом одягу в інтернеті. Інформаційна система містить в собі дві частини: серверну та клієнтську.

Головною метою інформаційної системи є забезпечення підприємства можливістю ведення комерційної діяльності в інтернеті. Для цього були розроблені наступні функції, які надають користувачеві виконувати наступні дії:

- можливість ознайомитись з асортиментом товару;
- можливість використовувати фільтр для пошуку потрібного товару;
- можливість додати потрібний товар до особистого кошика;

Для менеджмент-складу підприємства було розроблено адмін-панель з рівнем доступу «ADMIN». Цей функціонал дозволяє наповнювати інформацією інтернет магазин, а саме інформацією про новий тип одягу, новий розмір та новий одяг. Зручна візуальна складова дозволяє персоналу працювати більш ефективніше та швидше.

Також в проекті було взято до уваги постійне зростання мобільного трафіку, який з кожним днем має чіткий темп зростання. Роблячи висновки з цього питання, була реалізована адаптація інтернет-магазину для користувачів які користуються різноманітними пристроями з різною діагоналлю екрану.

В «Економічному розділі» було здійснено розрахунок людино-годин, які потрібні для повної розробки інформаційної системи (820,10 людино/годин). Також було визначено очікувані витрати на створення проекту (246706,27грн) і ймовірна очікувана тривалість розробки(4,66 місяці).

За результатом роботи на даним кваліфікаційним проектом було досягнуто всіх цілей, які були поставлені на початку роботи. Також були виконані всі вимоги, які були визначені в ході проектування інформаційної системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Колін Браяр, Білл Карр. Робота назад: ідеї, історії та секрети зсередини Amazon. - St. Martin's Press, 2021. – 304с.

2. Управління електронним бізнесом та електронною комерцією: стратегія, впровадження та практика 4-е видання.- Pearson College Div, 2009. – 764с.

3. Джон Дакетт. HTML і CSS. Розробка і дизайн веб-сайтів.- Litres, 2022. – 480с.

4. Джон Ларсен. React Hooks в дії: у режимі очікування та паралельного режиму. – Manning, 2021. – 375с.

5. Порчелло Єва, Бенкс Алекс. React: сучасні шаблони для розробки доданого. 2-е издание.- Питер, 2021. – 320с.

6. Харміт Сінгх, Мехул Бхатт. Навчання веб-розробці за допомогою React і Bootstrap. - Packt Publishing, 2016. – 252с.

7. Офіційна документація React. – [Електронний ресурс] – 2013 – Режим доступу: https://ru.reactjs.org/docs/getting-started.html. (дата звернення: 28.03.22)

8. Офіційна документація React-bootstrap. – [Електронний ресурс] – 2019 – Режим доступу: https://react-bootstrap.github.io. (дата звернення: 01.04.22)

9. Джон Резіг, Беар Бібо, Йосип Марас. Секрети JavaScript Ninja 2-е видання. - Manning; 2nd edition, 2016. – 464с.

10. Стоян Стефанов. JavaScript. Шаблони. - Символ-Плюс, 2018. – 272с.

11. Офіційна документація Axios. – [Електронний ресурс] – 2021 – Режим доступу: https://axios-http.com/docs/intro. (дата звернення: 25.04.22)

12. Офіційна документація MobX. – [Електронний ресурс] – 2019 – Режим доступу: https://mobx.js.org/about-this-documentation.html. (дата звернення: 01.05.22)

13. Герардус Блокдик. Веб-токен JSON: третє видання. - CreateSpace Independent Publishing Platform, 2018. – 137с.

14. Шеллі Пауерс. Навчальний Node: перехід на сторону сервера.- O'Reilly Media, 2016. – 288с.

15. Ітан Браун. Веб-розробка за допомогою Node і Express: використання стека JavaScript 1-е видання. - O'Reilly Media, 2014. – 332с.

16. Офіційна документація Express. – [Електронний ресурс] – 2015 – Режим доступу: https://expressjs.com/ru/guide/routing.html. (дата звернення: 02.04.22)

17. Офіційна документація Sequelize v6. – [Електронний ресурс] – 2019 – Режим доступу: https://sequelize.org/docs/v6/. (дата звернення: 06.04.22)

18. Регіна Обе, Лео Сюй. PostgreSQL: Початок роботи: практичний посібник із розширеної бази даних з відкритим вихідним кодом, 3-е видання. - O'Reilly Media, 2017. – 312с.

19. Офіційна документація PostgreSQL. – [Електронний ресурс] – 2013 – Режим доступу: https://www.postgresql.org/docs/. (дата звернення: 01.04.22)

20. О.Г. Вагонова, О.Б. Нікітіна, Н.Н. Романюк. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Комп’ютерні системи». - М-во освіти і науки України, ДВНЗ «Нац. гірн. ун- т». – Д.: НГУ, 2013. – 11 с.

21. Full stack developer: середня зарплата в Україні. – [Електронний ресурс] – 2022 – Режим доступу: https://www.work.ua/salary-full+stack+developer/. (дата звернення: 06.04.22)

ДОДАТОК А

КОД ПРОГРАМИ

Server/index.js

```
require("dotenv").config();
const express = require("express");
const sequelize = require("./db");
const PORT = process.env.PORT || 5000;
const app = express();
const fileUpload = require('express-fileupload')
const router = require('./routes/index')
const models = require("./models/models");
const cors = require("cors");
const ErrorMiddleware = require("./middleware/ErrorMiddleware");
const path = require('path')
app.use(cors());
app.use(express.json());
app.use(express.static(path.resolve(__dirname, 'static')))
app.use(fileUpload({}))
app.use('/api', router)
// Middleware з обробки помилок завжди йде останній.
app.use(ErrorMiddleware)
const start = async () => {
  try {
    await sequelize.authenticate();
    await sequelize.sync();
    app.listen(PORT, () => console.log(`Server started on port ${PORT}`));
  } catch (e) {
    console.log(e);
  }
};
start();
```

server/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize(
  process.env.DB_NAME, // Назва Бази даних
  process.env.DB_USER, // Назва користувача
  process.env.DB_PASSWORD, // Пароль
  {
    dialect: "postgres",
    host: process.env.DB_HOST,
    port: process.env.DB_PORT,
  }
```

);

Server/.env

```
PORT = 5000
DB_NAME= graduateWork2
DB_USER = postgres
DB_PASSWORD = *********
DB_HOST = localhost
DB_PORT = 5342
SECRET_KEY=my_shop415
```

Server/routes/basketRouter.js

```
const Router = require("express");
const router = new Router();
const BasketController = require("../controllers/basketController");
const authMiddleware = require("../middleware/authMiddleware");
router.post("/", authMiddleware, BasketController.addClothes);
router.get("/", authMiddleware, BasketController.getClothes);
module.exports = router;
```

Server/routes/clothesRouter.js

```
const Router = require('express')
const router = new Router()
const clothesController = require('../controllers/clothesController')
const roleMiddleware = require("../middleware/roleMiddleware");
router.post('/', roleMiddleware("ADMIN"), clothesController.create)
router.get('/', clothesController.getAll)
router.get('/:id', clothesController.getOne)
router.delete('/', clothesController.delete)
module.exports = router
```

Server/routes/sizeRouter.js

```
const Router = require('express')
const router = new Router()
const sizeController = require('../controllers/sizeController')
const roleMiddleware = require("../middleware/roleMiddleware");
router.post('/', roleMiddleware("ADMIN"), sizeController.create)
router.get('/', sizeController.getAll)
router.delete('/', sizeController.delete)
module.exports = router
```

Server/routes/typeRouter.js

```
const Router = require("express");
const router = new Router();
const typeController = require("../controllers/typeController");
const roleMiddleware = require("../middleware/roleMiddleware");
```

```
router.post("/", roleMiddleware("ADMIN"), typeController.create);
router.get("/", typeController.getAll);
router.delete("/", typeController.delete);
module.exports = router;
```

Server/routes/userRouter.js

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

Server/routes/index.js

```
const Router = require('express')
const router = new Router()
const clothesRouter = require('./clothesRouter')
const userRouter = require('./userRouter')
const sizeRouter = require('./sizeRouter')
const typeRouter = require('./typeRouter')
const basketRouter = require('./basketRouter')
router.use('/user', userRouter)
router.use('/type', typeRouter)
router.use('/size', sizeRouter)
router.use('/clothes', clothesRouter)
router.use('/basket', basketRouter)
module.exports = router
```

Server/models/models.js

```
const sequelize = require("../db");
const { DataTypes } = require("sequelize");
const User = sequelize.define("user", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  email: { type: DataTypes.STRING, unique: true },
  password: { type: DataTypes.STRING },
  role: { type: DataTypes.STRING, defaultValue: "USER" },
});
const Basket = sequelize.define("basket", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});
const BasketClothes = sequelize.define("basket_clothes", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  clothesId: {type: DataTypes.INTEGER},
});
const Clothes = sequelize.define("clothes", {
```

```
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true, allowNull: false },
  price: { type: DataTypes.INTEGER, allowNull: false },
  img: { type: DataTypes.STRING, allowNull: false },
});
const Type = sequelize.define("type", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true, allowNull: false },
});
const Size = sequelize.define("size", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true, allowNull: false },
});
const ClothesInfo = sequelize.define("clothes_info", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  title: { type: DataTypes.STRING, allowNull: false },
  description: { type: DataTypes.STRING, allowNull: false },
});
const TypeSize = sequelize.define("type_size", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});
User.hasOne(Basket);
Basket.belongsTo(User);
Basket.hasMany(BasketClothes);
BasketClothes.belongsTo(Basket);
Type.hasMany(Clothes);
Clothes.belongsTo(Type);
Size.hasMany(Clothes);
Clothes.belongsTo(Size);
Clothes.hasMany(BasketClothes);
BasketClothes.belongsTo(Clothes);
Clothes.hasMany(ClothesInfo, { as: "info" });
ClothesInfo.belongsTo(Clothes);
Type.belongsToMany(Size, { through: TypeSize });
Size.belongsToMany(Type, { through: TypeSize });
module.exports = {
  User,
  Basket,
  BasketClothes,
  Clothes,
  Type,
  Size,
  TypeSize,
  ClothesInfo,
};
```

Server/middleware/authMiddleware.js

```
const jwt = require('jsonwebtoken')
```

```
module.exports = function (req, res, next) {
    if (req.method === "OPTIONS") {
        next()
    }
    try {
        const token = req.headers.authorization.split(' ')[1]
        if (!token) {
            return res.status(401).json({message: "Користувач не авторизований!"})
        }
        const decoded = jwt.verify(token, process.env.SECRET_KEY)
        req.user = decoded
        next()
    } catch (e) {
        res.status(401).json({message: "Користувач не авторизований!"})
    }
};
```

Server/middleware/ErrorMiddleware.js

```
const ErrorTool = require('../error/ErrorTool')

module.exports = function (err, req, res, next) {
    if (err instanceof ErrorTool) {
        return res.status(err.status).json({message: err.message})
    }
    return res.status(500).json({message: "Непередбачена помилка."})
}
```

Server/middleware/roleMiddleware.js

```
const jwt = require('jsonwebtoken')

module.exports = function(role) {
    return function (req, res, next) {
        if (req.method === "OPTIONS") {
            next()
        }
        try {
            const token = req.headers.authorization.split(' ')[1]
            if (!token) {
                return res.status(401).json({message: "Користувач не авторизований!"})
            }
            const decode = jwt.verify(token, process.env.SECRET_KEY)
            if (decode.role !== role) {
                return res.status(403).json({message: "У вас немає доступа рівня адміністратора!"})
            }
            req.user = decode;
            next()
        } catch (e) {
            res.status(401).json({message: "Користувач не авторизований!"})
```

```
        }
    };
}
```

Server/error/ErrorTool.js

```
class ErrorTool extends Error{
    constructor(status, message) {
        super();
        this.status = status
        this.message = message
    }

    static badRequest(message) {
        return new ErrorTool(404, message)
    }

    static internal(message) {
        return new ErrorTool(500, message)
    }

    static forbidden(message) {
        return new ErrorTool(403, message)
    }
}

module.exports = ErrorTool
```

Server/controllers/basketController.js

```
const {
  Basket,
  BasketClothes,
  Clothes,
  ClothesInfo,
} = require("../models/models");
const jwt = require("jsonwebtoken");
const { Op } = require("sequelize");
class BasketController {
  async addClothes(req, res) {
    try {
      const { id } = req.body;
      const token = req.headers.authorization.split(" ")[1];
      const user = jwt.verify(token, process.env.SECRET_KEY);
      const basket = await Basket.findOne({ where: { userId: user.id } });
      await BasketClothes.create({ basketId: basket.id, clothesId: id });
      return res.json("Товар додан");
    } catch (e) {
      console.error(e);
    }
```

```
  }

  async getClothes(req, res) {
    try {
      const token = req.headers.authorization.split(" ")[1];
      const user = jwt.verify(token, process.env.SECRET_KEY);
      const { id } = await Basket.findOne({ where: { userId: user.id } });
      const basket = await BasketClothes.findAll({ where: { basketId: id } });

      const basketArr = [];
      for (let i = 0; i < basket.length; i++) {
        const basketClothes = await Clothes.findOne({
          where: {
            id: basket[i].clothesId,
          },
          include: {
            model: ClothesInfo,
            as: "info",
            where: {
              clothesId: basket[i].clothesId,
              [Op.or]: [
                {
                  clothesId: {
                    [Op.not]: null,
                  },
                },
              ],
            },
            required: false,
          },
        });
        basketArr.push(basketClothes);
      }

      return res.json(basketArr);
    } catch (e) {
      console.error(e);
    }
  }
}
module.exports = new BasketController();
```

Server/controllers/clothesController.js

```
const uuid = require("uuid");
const path = require("path");
const { Clothes, ClothesInfo } = require("../models/models");
const ErrorTool = require("../error/ErrorTool");

class ClothesController {
```

```
async create(req, res, next) {
  try {
    let { name, price, sizeId, typeId, info } = req.body;
    const { img } = req.files;
    let fileName = uuid.v4() + ".jpg";
    img.mv(path.resolve(__dirname, "..", "static", fileName));
    const clothes = await Clothes.create({
      name,
      price,
      sizeId,
      typeId,
      img: fileName,
    });

    if (info) {
      info = JSON.parse(info);
      info.forEach((i) =>
        ClothesInfo.create({
          title: i.title,
          description: i.description,
          clothesId: clothes.id,
        })
      );
    }

    return res.json(clothes);
  } catch (e) {
    next(ErrorTool.badRequest(e.message));
  }
}

async getAll(req, res) {
  let {sizeId, typeId, limit, page} = req.query
      page = page || 1
      limit = limit || 9
      let offset = page * limit - limit
      let clothes;
      if (!sizeId && !typeId) {
        clothes = await Clothes.findAndCountAll({limit, offset})
      }
      if (sizeId && !typeId) {
        clothes = await Clothes.findAndCountAll({where:{sizeId}, limit, offset})
      }
      if (!sizeId && typeId) {
        clothes = await Clothes.findAndCountAll({where:{typeId}, limit, offset})
      }
      if (sizeId && typeId) {
        clothes = await Clothes.findAndCountAll({where:{typeId, sizeId}, limit, offset})
      }
      return res.json(clothes)
}
```

```
  async getOne(req, res, next) {
    const {id} = req.params
        const clothes = await Clothes.findOne(
            {
                where: {id},
                include: [{model: ClothesInfo, as: 'info'}]
            },
        )
        return res.json(clothes)
    }


  async delete(req, res) {}
}

module.exports = new ClothesController();
```

Server/controllers/sizeController.js

```
const {Size, Type} = require('../models/models')
class SizeController {
    async create(req, res) {
      const {name} = req.body
      const size = await Size.create({name})
      return res.json(size)
    }

    async getAll(req, res) {
      const sizes = await Size.findAll();
      return res.json(sizes);
    }

    async delete(req, res) {}
  }

module.exports = new SizeController();
```

Server/controllers/typeController.js

```
const { Type } = require("../models/models");
const ErrorTool = require("../error/ErrorTool");

class TypeController {
  async create(req, res) {
    const { name } = req.body;
    const type = await Type.create({ name });
    return res.json(type);
  }
```

```
  async getAll(req, res) {
    const types = await Type.findAll();
    return res.json(types);
  }

  async delete(req, res) {}
}

module.exports = new TypeController();
```

Server/controllers/userController.js

```
const ErrorTool = require("../error/ErrorTool");
const bcrypt = require("bcrypt");
const { User, Basket } = require("../models/models");
const jwt = require("jsonwebtoken");

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

class UserController {
  async registration(req, res, next) {
    const { email, password, role } = req.body;
    if (!email || !password) {
      return next(ErrorTool.badRequest("Некоректна почта або пароль!"));
    }
    const filterUser = await User.findOne({ where: { email } });
    if (filterUser) {
      return next(
        ErrorTool.badRequest("Користувач з такою почтою вже існує!")
      );
    }
    const hashPassword = await bcrypt.hash(password, 5);
    const user = await User.create({ email, role, password: hashPassword });
    const basket = await Basket.create({ userId: user.id });
    const token = generateJwt(user.id, user.email, user.role);
    return res.json({ token });
  }

  async login(req, res, next) {
    const { email, password } = req.body;
    const user = await User.findOne({ where: { email } });
    if (!user) {
      return next(ApiError.internal("Користувач не знайден!"));
    }
    let comparePassword = bcrypt.compareSync(password, user.password);
    if (!comparePassword) {
```

```
      return next(ApiError.internal("Вказано неправильний пароль"));
    }
    const token = generateJwt(user.id, user.email, user.role);
    return res.json({ token });
  }

  async check(req, res, next) {
    const token = generateJwt(req.user.id, req.user.email, req.user.role);
    return res.json({ token });
  }
}

module.exports = new UserController();
```

Client/.env

```
REACT_APP_API_URL = 'http://localhost:5000/'
```

Client/routes.js

```
import AdminPage from "./page/AdminPage";
import BasketPage from "./page/BasketPage";
import Auth from "./page/Auth";
import MainPage from "./page/MainPage";
import Shop from "./page/Shop";
import ClothesPage from "./page/ClothesPage";
import { CLOTHES_ROUTE, SHOP_ROUTE, BASKET_ROUTE, ADMIN_ROUTE,
LOGIN_ROUTE, MAIN_ROUTE, REGISTRATION_ROUTE } from "./utils/const";


export const authRoutes = [
    {
      path: ADMIN_ROUTE,
      Component: AdminPage,
    },


  ];

  export const publicRoutes = [

    {
      path: LOGIN_ROUTE,
      Component: Auth,
    },

    {
      path: REGISTRATION_ROUTE,
      Component: Auth,
    },
```

```
    {
      path: SHOP_ROUTE,
      Component: Shop,
    },

    {
       path: MAIN_ROUTE,
       Component: MainPage,
      },

    {
      path: CLOTHES_ROUTE + '/:id',
      Component: ClothesPage,
    },

    {
      path: BASKET_ROUTE,
      Component: BasketPage,
    },
  ];
```

Client/src/index.js

```
import React, { createContext } from "react";
import ReactDOM from 'react-dom';
import App from "./App";
import UserStore from "./store/UserStore";
import ClothesStore from "./store/ClothesStore";
import BasketStore from "./store/BasketStore";

export const Context = createContext(null);
ReactDOM.render(
  <Context.Provider
    value={{
      user: new UserStore(),
      clothes: new ClothesStore(),
      basket: new BasketStore(),
    }}
  >
    <App />
  </Context.Provider>,
  document.getElementById("root")
);
```

Client/src/App.js

```
import { observer } from "mobx-react-lite";
import React, { useContext, useEffect, useState } from "react";
```

```
import { Spinner } from "react-bootstrap";
import { BrowserRouter } from "react-router-dom";
import { Context } from ".";
import AppRouter from "./component/AppRouter";
import NavBar from "./component/NavBar";
import { check } from "./http/userAPI";
import { getClothesFromBasket } from "./http/clothesAPI";

const App = observer(() => {
  const { user, basket } = useContext(Context);
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    check()
      .then((data) => {
        user.setUser(true);
        user.setIsAuth(true);
      })
      .finally(() => setLoading(false));
  }, []);

  useEffect(() => {
    if(user.isAuth === false) {
        basket.setDeleteAllClothesFromBasket();
        const savedBasket = JSON.parse(localStorage.getItem("basket"));
        for (let key in savedBasket) {
            basket.setBasket(savedBasket[key]);
        }
    } else if(user.isAuth === true){
        basket.setDeleteAllClothesFromBasket();
        getClothesFromBasket().then(data => {
            for (let key in data) {
                basket.setBasket(data[key], true);
            }
        })
    }
 }, [basket, user.isAuth]);

  if (loading) {
    return <Spinner animation={"border"} />;
  }

  return (
    <BrowserRouter>
      <NavBar />
      <AppRouter />
    </BrowserRouter>
  );
});

export default App;
```

Client/src/utils/const.js

```
export const ADMIN_ROUTE = '/admin'
export const LOGIN_ROUTE = '/login'
export const REGISTRATION_ROUTE = '/registration'
export const SHOP_ROUTE = '/shop'
export const MAIN_ROUTE = '/main'
export const CLOTHES_ROUTE = '/clothes'
export const BASKET_ROUTE = '/basket'
```

Client/src/store/BasketStore.js

```
import { makeAutoObservable } from "mobx";
import { deleteClothesFromBasket } from "../http/clothesAPI";
export default class BasketStore {
  constructor() {
    this._totalPrice = 0;
    this._basket = [];
    makeAutoObservable(this);
  }

  async setDeleteItemBasket(clothes, isAuth = false) {
    if (isAuth) {
      await deleteClothesFromBasket(clothes.id).then(() => {
        this._basket = this._basket.filter((item) => item.id !== clothes.id);
        this._totalPrice -= clothes.price * clothes.count;
      });
    } else {
      this._basket = this._basket.filter((item) => item.id !== clothes.id);
      this._totalPrice -= clothes.price * clothes.count;

      localStorage.setItem("basket", JSON.stringify(this._basket));
    }
  }

  setBasket(item, isAuth = false) {
    const checkClothesInBasket = this._basket.findIndex(
      (clothes) => clothes.id === item.id
    );
    if (checkClothesInBasket < 0) {
      this._basket = [...this._basket, { count: 1, ...item }];
      let totalPrice = 0;
      this._basket.forEach(
        (clothes) => (totalPrice += Number(clothes.price * clothes.count))
      );
      this._totalPrice = totalPrice;
    }

    if (!isAuth) {
```

```
    localStorage.setItem("basket", JSON.stringify(this._basket));
  }
}

setDeleteAllClothesFromBasket() {
  this._totalPrice = 0;
  return (this._basket = []);
}

setCountClothes(clothesId, action, isAuth = false) {
  const itemInd = this._basket.findIndex((item) => item.id === clothesId);
  const itemInState = this._basket.find(
    (clothes) => clothes.id === clothesId
  );
  if (action === "+") {
    const newItem = {
      ...itemInState,
      count: ++itemInState.count,
    };
    this._basket = [
      ...this._basket.slice(0, itemInd),
      newItem,
      ...this._basket.slice(itemInd + 1),
    ];
  } else {
    const newItem = {
      ...itemInState,
      count: itemInState.count === 1 ? 1 : --itemInState.count,
    };
    this._basket = [
      ...this._basket.slice(0, itemInd),
      newItem,
      ...this._basket.slice(itemInd + 1),
    ];
  }

  if (!isAuth) {
    localStorage.setItem("basket", JSON.stringify(this._basket));
  }

  let totalPrice = 0;
  this._basket.forEach(
    (clothes) => (totalPrice += Number(clothes.price * clothes.count))
  );
  this._totalPrice = totalPrice;
}

resetBasket() {
  this._basket = [];
  this._totalPrice = 0;
  localStorage.removeItem("basket");
```

```
  }

  get Basket() {
    return this._basket;
  }

  get Price() {
    return this._totalPrice;
  }
}
```

Client/src/store/ClothesStore.js

```
import { makeAutoObservable } from "mobx";

export default class ClothesStore {
  constructor() {
    this._types = [];
    this._sizes = [];
    this._clothes = [];
    this._selectedSize = {};
    this._selectedType = {};
    this._page = 1;
    this._totalCount = 0;
    this._limit = 3;
    makeAutoObservable(this);
  }

  setTypes(types) {
    this._types = types;
  }
  setSizes(sizes) {
    this._sizes = sizes;
  }
  setClothes(clothes) {
    this._clothes = clothes;
  }

  setSelectedSize(size) {
    this._selectedSize = size;
  }
  setSelectedType(type) {
    this._selectedType = type;
  }

  setPage(page) {
    this._page = page;
  }
  setTotalCount(count) {
    this._totalCount = count;
```

```
  }

  get types() {
    return this._types;
  }
  get sizes() {
    return this._sizes;
  }
  get clothes() {
    return this._clothes;
  }
  get selectedSize() {
    return this._selectedSize;
  }
  get selectedType() {
    return this._selectedType;
  }
  get totalCount() {
    return this._totalCount;
  }
  get page() {
    return this._page;
  }
  get limit() {
    return this._limit;
  }
}
```

Client/src/store/UserStore.js

```
import { makeAutoObservable } from "mobx";

export default class UserStore {
  constructor() {
    this._isAuth = false;
    this._user = {};
    makeAutoObservable(this);
  }

  setIsAuth(bool) {
    this._isAuth = bool;
  }
  setUser(user) {
    this._user = user;
  }

  get isAuth() {
    return this._isAuth;
  }
  get user() {
```

```
    return this._user;
  }
}
```

Client/page/AdminPage.js

```
import React, { useState } from "react";
import { Button, Card, Container } from "react-bootstrap";
import CreateSize from "../component/modals/CreateSize";
import CreateType from "../component/modals/CreateType";
import CreateClothes from "../component/modals/CreateClothes";

const AdminPage = () => {
  const [sizeVisible, setSizeVisible] = useState(false);
  const [typeVisible, setTypeVisible] = useState(false);
  const [clothesVisible, setClothesVisible] = useState(false);

  return (
    <Container className="d-flex justify-content-center align-items-center mt-5">
      <Card style={{ width: 500 }} className="p-5 ">
        <h3 className="d-flex justify-content-center">Адмін панель</h3>
        <Button
          variant="dark"
          className="mt-3"
          onClick={() => setTypeVisible(true)}
        >
          Додати тип
        </Button>
        <Button
          variant="dark"
          className="mt-3"
          onClick={() => setSizeVisible(true)}
        >
          Додати розмір
        </Button>
        <Button
          variant="dark"
          className="mt-3"
          onClick={() => setClothesVisible(true)}
        >
          Додати одяг
        </Button>

        <CreateSize show={sizeVisible} onHide={() => setSizeVisible(false)} />
        <CreateClothes
          show={clothesVisible}
          onHide={() => setClothesVisible(false)}
        />
        <CreateType show={typeVisible} onHide={() => setTypeVisible(false)} />
      </Card>
```

```
    </Container>
  );
};

export default AdminPage;
```

Client/page/Auth.js

```
import React, { useContext, useState } from "react";
import { Container, Form } from "react-bootstrap";
import Card from "react-bootstrap/Card";
import Button from "react-bootstrap/Button";
import Row from "react-bootstrap/Row";
import { NavLink, useLocation, useNavigate } from "react-router-dom";
import { REGISTRATION_ROUTE, LOGIN_ROUTE, SHOP_ROUTE } from "../utils/const";
import { login, registration } from "../http/userAPI";
import { observer } from "mobx-react-lite";
import { Context } from "../index";

const Auth = observer(() => {
  const { user } = useContext(Context);
  const location = useLocation();
  const navigation = useNavigate();
  const isLogin = location.pathname === LOGIN_ROUTE;
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const click = async () => {
    try {
      let data;
      if (isLogin) {
        data = await login(email, password);
      } else {
        data = await registration(email, password);
      }
      user.setUser(user);
      user.setIsAuth(true);
      navigation(SHOP_ROUTE);
    } catch (e) {
      alert(e.response.data.message);
    }
  };
  return (
    <div>
      <Container
        className="d-flex justify-content-center align-items-center "
        style={{ height: window.innerHeight - 40 }}
      >
        <Card style={{ width: 500 }} className="p-5">
          <h3 className="m-auto">{isLogin ? "Авторизація" : "Реєстрація"}</h3>
```

```
        <Form className="d-flex flex-column">
          <Form.Control
            className="mt-3"
            placeholder="Введіть свой email.."
            value={email}
            onChange={(e) => setEmail(e.target.value)}
          ></Form.Control>
          <Form.Control
            className="mt-3"
            placeholder="Введіть свій пароль.."
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            type="password"
          ></Form.Control>
          <Row className="d-flex justify-content-between mt-2">
            {isLogin ? (
              <div>
                Немає аккаунта?{" "}
                <NavLink
                  className="text-decoration"
                  style={{ textDecoration: "none", color: "DimGray" }}
                  to={REGISTRATION_ROUTE}
                >
                  Реєстрація
                </NavLink>
              </div>
            ) : (
              <div>
                Є аккаунт?{" "}
                <NavLink
                  className="text-decoration"
                  style={{ textDecoration: "none", color: "DimGray" }}
                  to={LOGIN_ROUTE}
                >
                  Увійти
                </NavLink>
              </div>
            )}
            <Button
              className="rounded-pill mt-3"
              variant="outline-dark"
              onClick={click}
            >
              {isLogin ? "Увійти" : "Реєстрація"}
            </Button>
          </Row>
        </Form>
      </Card>
    </Container>
  </div>
);
```

```
});

export default Auth;
```

Client/page/BasketPage.js

```
import React, { useContext } from "react";
import MAIN_ROUTE from "../utils/const";
import Nav from "react-bootstrap/Nav";
import { Button, Col, Image, Row } from "react-bootstrap";
import { Context } from "../index";
import { observer } from "mobx-react-lite";
import emptyBasket from "../assets/emptyBasket.png";
import OneClothesinBasket from "../component/OneClothesinBasket";

const BasketPage = observer(() => {
  const { basket } = useContext(Context);

  if (basket.Basket.length === 0) {
    return (
      <div className="d-flex flex-column align-items-center mt-5">
        <Image src={emptyBasket} />
        <div className="text-center mt-5" style={{ fontSize: 35 }}>
          <b>Кошик для покупок порожній</b>
        </div>
      </div>
    );
  }

  return (
    <>
      <br />
      <Row className="mt-3">
        <Col xs={12}>
          {basket.Basket.map((clothes) => (
            <OneClothesinBasket key={clothes.id} clothes={clothes} />
          ))}
        </Col>
      </Row>
    </>
  );
});

export default BasketPage;
```

Client/page/ClothesPage.js

```
import React, { useContext, useEffect, useState } from "react";
import { Button, Card, Col, Container, Image, Row } from "react-bootstrap";
import grivna from "../assets/grivna.png";
```

```
import { useParams } from "react-router-dom";
import { fetchOneClothes } from "../http/clothesAPI";
import { observer } from "mobx-react-lite";
import { Context } from "../index";

const ClothesPage = observer(() => {
  const { user, basket } = useContext(Context);
  const [clothes, setClothes] = useState({ info: [] });
  const { id } = useParams();
  useEffect(() => {
    fetchOneClothes(id).then((data) => setClothes(data));
  }, []);

  const addClothesInBasket = (clothes) => {
    if (user.isAuth) {
      addClothesInBasket(clothes).then(() => basket.setBasket(clothes, true));
    } else {
      basket.setBasket(clothes);
    }
  };

  return (
    <div>
      <Container md={4} className="mt-4">
        <Row>
          <Col md={4}>
            <Image
              className="block-example border border-light"
              width={370}
              height={450}
              src={process.env.REACT_APP_API_URL + clothes.img}
            ></Image>
          </Col>
          <Col md={4}>
            <Row>
              <h2>{clothes.name}</h2>
            </Row>
            {/* Переделать размеры */}

            <Row className="d-flex flex-column mt-4 m-0">
              <h3>Опис</h3>
              <div className="text-secondary mt-0 d-flex justify-content-between align-items-center">
                Кожен наш одяг виконаний з натурального та якісного матеріалу. У
                нашому одязі ви завжди почуватиметеся в комфорті.
              </div>
              <Row className="d-flex flex-column mt-4 m-0">
                {clothes.info.map((info) => (
                  <Row key={info.id}>
                    {info.title}: {info.desciption}
                  </Row>
```

```
            ))}
          </Row>
          <Row className="d-flex flex-column mt-4 m-0"></Row>
        </Row>
      </Col>
      <Col md={4}>
        <Card className="d-flex flex-column align-items-center justify-content-around">
          <h3>
            {clothes.price}
            <Image
              className="mt-0 m-1"
              width={20}
              height={20}
              src={grivna}
            ></Image>
          </h3>
          <Button
            variant="dark"
            onClick={() => addClothesInBasket(clothes)}
          >
            До корзини
            <Image
              className="mt-0 m-1 "
              width={16}
              height={16}
              src={basket}
            ></Image>
          </Button>
        </Card>
      </Col>
    </Row>
  </Container>
 </div>
 );
});

export default ClothesPage;
```

Client/page/MainPage.js

```
import React from "react";
import { Button, Carousel } from "react-bootstrap";
import post1 from "../assets/post1.jpg";
import post2 from "../assets/post2.jpg";
import post3 from "../assets/post3.jpg";

const MainPage = () => {
 return (
  <Carousel variant="dark">
   <Carousel.Item>
```

```
        <img
          className="d-block w-100"
          height={910}
          src={post1}
          alt="First slide"
        />
        <Carousel.Caption>

        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <img
          className="d-block w-100"
          height={910}
          src={post2}
          alt="Second slide"
        />
        <Carousel.Caption>

        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <img
          className="d-block w-100"
          height={910}
          src={post3}
          alt="Third slide"
        />
        <Carousel.Caption>

        </Carousel.Caption>
      </Carousel.Item>
    </Carousel>
  );
};

export default MainPage;
```

Client/page/Shop.js

```
import { observer } from "mobx-react-lite";
import React, { useContext, useEffect } from "react";
import { Col, Container, Row } from "react-bootstrap";
import { Context } from "..";
import ClothesList from "../component/ClothesList";
import Pages from "../component/Pages";
import SizeBar from "../component/SizeBar";
import TypeBar from "../component/TypeBar";
import { fetchClothes, fetchSizes, fetchTypes } from "../http/clothesAPI";
```

```
const Shop = observer(() => {
  const { clothes } = useContext(Context);

  useEffect(() => {
    fetchTypes().then((data) => clothes.setTypes(data));
    fetchSizes().then((data) => clothes.setSizes(data));
    fetchClothes(null, null, 1, 3).then((data) => {
      clothes.setClothes(data.rows);
      clothes.setTotalCount(data.count);
    });
  }, []);

  useEffect(() => {
    fetchClothes(
      clothes.selectedType.id,
      clothes.selectedSize.id,
      clothes.page,
      3
    ).then((data) => {
      clothes.setClothes(data.rows);
      clothes.setTotalCount(data.count);
    });
  }, [clothes.page, clothes.selectedType, clothes.selectedSize]);

  return (
    <Container>
      <Row className="mt-2">
        <Col md={3}>
          <SizeBar />
        </Col>
        <Col md={9}>
          <TypeBar />
          <ClothesList />
          <Pages />
        </Col>
      </Row>
    </Container>
  );
});

export default Shop
```

Client/http/clothesAPI.js

```
import { $authHost, $host } from ".";

export const createType = async (type) => {
  const { data } = await $authHost.post("api/type", type);
  return data;
};
```

```
export const fetchTypes = async () => {
  const { data } = await $host.get("api/type");
  return data;
};

export const createSize = async (size) => {
  const { data } = await $authHost.post("api/size", size);

  return data;
};

export const fetchSizes = async () => {
  const { data } = await $host.get("api/size");
  return data;
};

export const createClothes = async (clothes) => {
  const { data } = await $authHost.post("api/clothes", clothes);

  return data;
};

export const getClothesFromBasket = async () => {
  const { data } = await $authHost.get("api/basket");
  return data;
};

export const deleteClothesFromBasket = async (id) => {
  const { data } = await $authHost.delete(`api/basket/${id}`);
  return data;
};

export const fetchClothes = async (typeId, sizeId, page, limit = 5) => {
  const { data } = await $host.get("api/clothes", {
    params: {
      typeId,
      sizeId,
      page,
      limit,
    },
  });
  return data;
};

export const fetchOneClothes = async (id) => {
  const { data } = await $host.get("api/clothes/" + id);
  return data;
};
```

Client/http/index.js

```
import axios from "axios";

const $host = axios.create({
  baseURL: process.env.REACT_APP_API_URL,
});

const $authHost = axios.create({
  baseURL: process.env.REACT_APP_API_URL,
});

const authInterceptor = (config) => {
  config.headers.authorization = `Bearer ${localStorage.getItem("token")}`;
  return config;
};

$authHost.interceptors.request.use(authInterceptor)

export{
    $host,
    $authHost
}
```

Client/http/userAPI.js

```
import { $authHost, $host } from ".";
import jwt_decode from "jwt-decode";

export const registration = async (email, password) => {
  const {data} = await $host.post("api/user/registration", {
    email,
    password,
    role: "ADMIN",
  });
localStorage.setItem('token', data.token)
  return jwt_decode(data.token);
};

export const login = async (email, password) => {
  const {data} = await $host.post("api/user/login", {
    email,
    password,
  });
  localStorage.setItem('token', data.token)
  return jwt_decode(data.token);
};

export const check = async () => {
  const {data} = await $authHost.get("api/user/auth");
```

```
  localStorage.setItem('token', data.token)
  return jwt_decode(data.token);
};
```

Client/component/TypeBar.js

```
import { observer } from "mobx-react-lite";
import React, { useContext } from "react";
import { Button, Card, CardGroup, Col, Container, Row } from "react-bootstrap";
import { Context } from "..";

const TypeBar = observer(() => {
  const { clothes } = useContext(Context);
  return (
    <Container>
      <Row className="d-flex">
        {clothes.types.map((type) => (
          <Card
            key={type.id}
            className="p-2 mt-2"
            style={{ cursor: "pointer" }}
            onClick={() => clothes.setSelectedType(type)}
            border={type.id === clothes.selectedType.id ? "dark" : "light"}
          >
            {type.name}
          </Card>

        ))}
      </Row>
    </Container>
  );
});

export default TypeBar;
```

Client/component/SizeBar.js

```
import { observer } from "mobx-react-lite";
import React, { useContext } from "react";
import { ListGroup } from "react-bootstrap";
import { Context } from "..";

const SizeBar = observer(() => {
  const { clothes } = useContext(Context);
  return (
    <ListGroup>
      <div style={{ fontSize: 20 }} className="mt-2">
        Розмір
      </div>
      {clothes.sizes.map((size) => (
```

```
        <ListGroup.Item
          style={{ cursor: "pointer" }}
          active={size.id === clothes.selectedSize.id}
          onClick={() => clothes.setSelectedSize(size)}
          className="mt-1"
          key={size.id}
        >
          {size.name}
        </ListGroup.Item>
      ))}
    </ListGroup>
  );
});

export default SizeBar;
```

Client/component/Pages.js

```
import React, { useContext } from "react";
import { Context } from "../index";
import { observer } from "mobx-react-lite";
import { Pagination } from "react-bootstrap";

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

  for (let i = 0; i < pageCount; i++) {
    pages.push(i + 1);
  }
  return (
    <Pagination className="mt-3">
      {pages.map((page) => (
        <Pagination.Item
          key={page}
          active={clothes.page === page}
          onClick={() => clothes.setPage(page)}
        >
          {page}
        </Pagination.Item>
      ))}
    </Pagination>
  );
});

export default Pages;
```

Client/component/OneClothesinBasket.js

```
import React, { useContext } from "react";
import { Button, Card, Col, Image, Nav, Row } from "react-bootstrap";

const OneClothesinBasket = ({ clothes }) => {
  return (
    <div>
      <h2 className="d-flex justify-content-center">Корзина</h2>
      <Card key={clothes.id} style={{ width: "100%" }}>
        <Card.Body>
          <Row>
            <Col xs={4}>
              <Image
                src={process.env.REACT_APP_API_URL + clothes.img}
                style={{ width: "100%", maxWidth: 250 }}
              />
            </Col>
            <Col xs={4}>
              <Row>
                <Col xs={12}>
                  <b>Ваше замовлення</b>
                  <Nav.Link
                    href={`/clothes/${clothes.id}`}
                    style={{ color: "black" }}
                  >
                    {clothes.name}
                    <Row>Сума до сплати: {clothes.price}</Row>
                  </Nav.Link>
                </Col>
              </Row>
            </Col>
          </Row>
        </Card.Body>
      </Card>
    </div>
  );
};

export default OneClothesinBasket;
```

Client/component/NavBar.js

```
import React, { useContext } from "react";
import Navbar from "react-bootstrap/Navbar";
import Nav from "react-bootstrap/Nav";
import { Button, NavDropdown } from "react-bootstrap";
import Container from "react-bootstrap/Container";
import Image from "react-bootstrap/Image";
import logo from "../assets/test2.png";
```

```
import { Context } from "..";
import { observer } from "mobx-react-lite";
import { ADMIN_ROUTE, LOGIN_ROUTE, MAIN_ROUTE, SHOP_ROUTE } from
"../utils/const";
import { useNavigate } from "react-router-dom";
import Basket from "./Basket";

const NavBar = observer(() => {
  const { user } = useContext(Context);
  const navigate = useNavigate();

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

  return (
    <div>
      <Navbar bg="dark" variant="dark">

        <Container>
          <Navbar.Brand href={MAIN_ROUTE}>
            <Image width="180" src={logo} />
          </Navbar.Brand>

          <Nav className="me-auto" style={{ fontSize: 20 }}>
            <Nav.Link href={SHOP_ROUTE}>Новинки</Nav.Link>
            <Nav.Link href={SHOP_ROUTE}>Верх</Nav.Link>
            <NavDropdown>
              <NavDropdown.Item href={SHOP_ROUTE}>Світшот</NavDropdown.Item>
              <NavDropdown.Item href={SHOP_ROUTE}>Футболка</NavDropdown.Item>
              <NavDropdown.Item href={SHOP_ROUTE}>Худі</NavDropdown.Item>
            </NavDropdown>
            <Nav.Link href={SHOP_ROUTE}>Низ</Nav.Link>
            <NavDropdown>
              <NavDropdown.Item href={SHOP_ROUTE}>Шорти</NavDropdown.Item>
              <NavDropdown.Item href={SHOP_ROUTE}>Штани</NavDropdown.Item>
            </NavDropdown>
          </Nav>

          {user.isAuth ? (
            <Nav className="ml-auto" style={{ color: "white" }}>
              <Button
                variant={"outline-light"}
                onClick={() => navigate(ADMIN_ROUTE)}
              >
                Адмін панель
              </Button>
              <Button variant={"outline-light"} onClick={()=>logOut()} className="ms-3">
                Вийти
              </Button>
```

```
            <Basket/>

          </Nav>
        ) : (
          <Nav className="ml-auto" style={{ color: "white" }}>
            <Button
              variant={"outline-light"}
              onClick={() => navigate(LOGIN_ROUTE)}
            >
              Авторизация
            </Button>

            <Basket className="ms-3"/>

          </Nav>
        )}
      </Container>
    </Navbar>
  </div>
 );
});

export default NavBar;
```

Client/component/ClothesList.js

```
import { observer } from "mobx-react-lite";
import React, { useContext } from "react";
import { Context } from "..";
import ClothesItem from "./ClothesItem";
import { Row } from "react-bootstrap";

const ClothesList = observer(() => {
 const { clothes } = useContext(Context);
 return (
   <Row className="d-flex ">
     {clothes.clothes.map((clothes) => (
       <ClothesItem key={clothes.id} clothes={clothes}/>
     ))}
   </Row>
 );
});

export default ClothesList;
```

Client/component/ClothesItem.js

```
import React from "react";
import {Card, Col, Image } from "react-bootstrap";
```

```
import grivna from "../assets/grivna.png";
import { useNavigate } from "react-router-dom";
import { CLOTHES_ROUTE } from "../utils/const";

const ClothesItem = ({ clothes }) => {

  const navigate = useNavigate();

  return (
    <Col md={3} className={'mt-4'} onClick={() => navigate(CLOTHES_ROUTE + "/" +
clothes.id)}>
      <Card style={{width:241, cursor:'pointer'}} className='mt-2' border={'green'}>
        <Image width={240} height={330} src={process.env.REACT_APP_API_URL +
clothes.img}/>

        <div className="text-black-50 mt-1 d-flex justify-content-between align-items-center">
          <div className="d-flex align-items-center">
            <div>{clothes.name}</div>
          </div>
        </div>
        <div className="d-flex flex-row">
          <div className="p-2">
            {clothes.price}
            <Image width={16} height={16} src={grivna} />
          </div>
        </div>
      </Card>
    </Col>
  );
};

export default ClothesItem;
```

Client/component/Basket.js

```
import { observer } from "mobx-react-lite";
import React, { useContext } from "react";
import { Image, Nav } from "react-bootstrap";
import { Context } from "..";
import { BASKET_ROUTE } from "../utils/const";
import basketnavbar from '../assets/basketnavbar.png'

const Basket = observer(() => {
  const { basket } = useContext(Context);
  return (
    <div className="d-flex align-items-center mr-3">
      <Nav.Link href={BASKET_ROUTE} className="d-flex align-items-center ms-3" >
        <Image
          src={basketnavbar}
          style={{ width: "100%", maxWidth: 30 }}
```

```
          alt="basket"
        />
        <div
          className="ml-2"
          style={{ textDecoration: "none", color: "white" }}
        >
          {basket.Price}
        </div>
      </Nav.Link>
    </div>
  );
});

export default Basket;
```

Client/component/AppRouter.js

```
import React, { useContext } from "react";
import { Route, Routes } from "react-router-dom";
import MainPage from "../page/MainPage";
import { Context } from "../index";
import { authRoutes, publicRoutes } from "../routes";

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

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

      <Route path="*" element={<MainPage />} />
    </Routes>
  );
};

export default AppRouter;
```

Client/component/modals/CreateClothes.js

```
import { observer } from "mobx-react-lite";
import React, { useContext, useEffect, useState } from "react";
import {
  Button,
```

```
  Col,
  Dropdown,
  Form,
  FormControl,
  Modal,
  Row,
} from "react-bootstrap";
import DropdownItem from "react-bootstrap/esm/DropdownItem";
import DropdownMenu from "react-bootstrap/esm/DropdownMenu";
import DropdownToggle from "react-bootstrap/esm/DropdownToggle";
import { createClothes, fetchSizes, fetchTypes } from "../../http/clothesAPI";
import { Context } from "../../index";
const CreateClothes = observer(({ show, onHide }) => {
  const { clothes } = useContext(Context);
  const [name, setName] = useState("");
  const [price, setPrice] = useState(0);
  const [file, setFile] = useState(null);
  const [info, setInfo] = useState([]);

  useEffect(() => {
    fetchTypes().then((data) => clothes.setTypes(data));
    fetchSizes().then((data) => clothes.setSizes(data));
  }, []);

  const addInfo = () => {
    setInfo([...info, { title: "", description: "", number: Date.now() }]);
  };

  const deleteInfo = (number) => {
    setInfo(info.filter((i) => i.number !== number));
  };

  const selectFile = (e) => {
    setFile(e.target.files[0]);
  };

  const changeInfo = (key, value, number) => {
    setInfo(
      info.map((i) => (i.number === number ? { ...i, [key]: value } : i))
    );
  };

  const addClothes = () => {
    const formData = new FormData()
        formData.append('name', name)
        formData.append('price', `${price}`)
        formData.append('img', file)
        formData.append('sizeId', clothes.selectedSize.id)
        formData.append('typeId', clothes.selectedType.id)
        formData.append('info', JSON.stringify(info))
        createClothes(formData).then(data => onHide())
```

```
};

return (
  <Modal show={show} onHide={onHide} size="lg" centered>
    <Modal.Header closeButton>
      <Modal.Title id="contained-modal-title-vcenter">
        Додати одяг
      </Modal.Title>
    </Modal.Header>
    <Modal.Body>
      <Form>
        <Dropdown className="mt-4">
          <DropdownToggle>
            {clothes.selectedType.name || "Оберіть тип"}
          </DropdownToggle>
          <DropdownMenu>
            {clothes.types.map((type) => (
              <DropdownItem
                onClick={() => clothes.setSelectedType(type)}
                key={type.id}
              >
                {" "}
                {type.name}
              </DropdownItem>
            ))}
          </DropdownMenu>
        </Dropdown>
        <Dropdown className="mt-4">
          <DropdownToggle>
            {clothes.selectedSize.name || "Оберіть розмір"}
          </DropdownToggle>
          <DropdownMenu>
            {clothes.sizes.map((size) => (
              <DropdownItem
                onClick={() => clothes.setSelectedSize(size)}
                key={size.id}
              >
                {" "}
                {size.name}
              </DropdownItem>
            ))}
          </DropdownMenu>
        </Dropdown>
        <FormControl
          value={name}
          onChange={(e) => setName(e.target.value)}
          className="mt-4"
          placeholder="Введіть назву одягу"
        />
        <FormControl
          value={price}
```

```
          onChange={(e) => setPrice(Number(e.target.value))}
          className="mt-4"
          placeholder="Введіть ціну одягу"
          type="number"
        />
        <FormControl className="mt-4" type="file" onChange={selectFile} />
        <hr />
        <Button onClick={addInfo}>Додати нову властивість</Button>

        {info.map((i) => (
          <Row className="mt-3" key={i.number}>
            <Col md={4}>
              <FormControl
                value={i.title}
                onChange={(e) =>
                  changeInfo("title", e.target.value, i.number)
                }
                placeholder="Введіть назву властивості"
              />
            </Col>
            <Col md={4}>
              <FormControl
                value={i.description}
                onChange={(e) =>
                  changeInfo("description", e.target.value, i.number)
                }
                placeholder="Введіть опис властивості"
              />
            </Col>
            <Col md={4}>
              <Button onClick={() => deleteInfo(i.number)}>Видалити</Button>
            </Col>
          </Row>
        ))}
      </Form>
    </Modal.Body>
    <Modal.Footer>
      <Button variant="danger" onClick={onHide}>
        Закрити
      </Button>
      <Button variant="success" onClick={addClothes}>
        Додати
      </Button>
    </Modal.Footer>
  </Modal>
 );
});

export default CreateClothes;
```

Client/component/modals/CreateSize.js

```
import React, { useState } from "react";
import { Button, Form, Modal } from "react-bootstrap";
import { createSize } from "../../http/clothesAPI";
const CreateSize = ({ show, onHide }) => {
  const [value, setValue] = useState("");
  const addSize = () => {
    createSize({ name: value }).then((data) => {
      setValue("");
      onHide();
    });
  };

  return (
    <Modal show={show} onHide={onHide} size="lg" centered>
      <Modal.Header closeButton>
        <Modal.Title id="contained-modal-title-vcenter">
          Додати розмір
        </Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form>
          <Form.Control
            value={value}
            onChange={(e) => setValue(e.target.value)}
            placeholder={"Введіть розмір"}
          ></Form.Control>
        </Form>
      </Modal.Body>
      <Modal.Footer>
        <Button variant="danger" onClick={onHide}>
          Закрити
        </Button>
        <Button variant="success" onClick={addSize}>
          Додати
        </Button>
      </Modal.Footer>
    </Modal>
  );
};

export default CreateSize;
```

Client/component/modals/CreateType.js

```
import React, { useState } from "react";
import { Button, Form, Modal } from "react-bootstrap";
import { createType } from "../../http/clothesAPI";
```

```
const CreateType = ({ show, onHide }) => {
  const [value, setValue] = useState("");
  const addType = () => {
    createType({ name: value }).then((data) => {
      setValue("");
      onHide();
    });
  };

  return (
    <Modal show={show} onHide={onHide} size="lg" centered>
      <Modal.Header closeButton>
        <Modal.Title id="contained-modal-title-vcenter">Додати тип</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form>
          <Form.Control
            value={value}
            onChange={(e) => setValue(e.target.value)}
            placeholder={"Введіть назву типу"}
          ></Form.Control>
        </Form>
      </Modal.Body>
      <Modal.Footer>
        <Button variant="danger" onClick={onHide}>
          Закрити
        </Button>
        <Button variant="success" onClick={addType}>
          Додати
        </Button>
      </Modal.Footer>
    </Modal>
  );
};

export default CreateType;
```

Всі інші файли роботи можна буде переглянути на магнітному носії, на якому буде архів з готовим проектом.

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

Перелік файлів на диску

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Йолкін.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Йолкін.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Йолкін кваліфікаційна робота.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Презентація Йолкін.ppt	Презентація кваліфікаційної роботи