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 *Ніколаєнка Артема Віталійовича*
(ПІБ)

академічної групи *122-19ск-1*
(шифр)

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

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

на тему: *Розробка веб-орієнтованого додатку з продажу одягу*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
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 *122-19ск-1* *Ніколаєнка Артема Віталійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованого додатку з продажу одягу*

затверджена наказом ректора НТУ «ДП» від «18» травня 2022 р. № 268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної 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. Спірінцев В.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Ніколаєнко А.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

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

РЕФЕРАТ

Пояснювальна записка: 102 с., 46 рис., 3 дод., 23 джерел.

Об'єкт розробки: система функціонування та адміністрування інтернет-магазину одягу .

Мета кваліфікаційної роботи: розробка та адміністрування веб-орієнтованого додатку Інтернет магазину з продажу одягу.

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

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано платформу для розробки, описано проектування і розробку програми, наведено опис алгоритму і структури функціонування системи, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описано виклик та завантаження застосунку, описана робота програми.

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

Практичне значення полягає у створенні системи, що забезпечує відвідувачам сайту доступ до перегляду каталогу товарів інтернет-магазину (усіх, нових, або відсортованих за категоріями), надає можливість додавання та редагування товарів у кошику, та оформлення замовлень, що забезпечує комфортне користування інтернет-магазином. Адміністративна панель надає можливість адміністратору магазину додавати та редагувати товари (надавати актуальну інформацію щодо наявності та вартості товару) та їх категорії, а також здійснювати перегляд та редагування наявних замовлень.

Актуальність інтернет-магазину визначається великим попитом на онлайн послуги, що дозволяють оптимізувати та спростити обслуговування клієнтів, а також тих, хто не має фізичної можливості відвідати магазин, усе ці фактори сприяють створенню та закріпленню позитивного іміджу бренду або компанії.

Список ключових слів: ІНТЕРНЕТ-МАГАЗИН, АДМІНІСТРАТИВНА ПАНЕЛЬ, WEB-САЙТ, WEB-ДОДАТОК, WEB-РОЗРОБКА.

ABSTRACT

Explanatory note: 102 p., 46 figs., 3 apps, 23 sources.

Development object: the system of functioning and administration of the online clothing store.

The purpose of the qualification work: development and administration of a web-oriented application of an online clothing store.

The introduction considers the analysis and 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 problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and the purpose of development are determined, the task statement is developed, the requirements to the software implementation, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, describes the design and development of the program, describes the algorithm and structure of the system, determines the input and output data, provides characteristics of the parameters of technical means, describes calling and loading applications, describes the program.

The economic section determines the complexity of the developed information subsystem, calculates the cost of work to create an application and calculates the time for its creation.

The practical significance is to create a system that provides site visitors with access to view the catalog of online store products (all, new, or sorted by category), provides the ability to add and edit items in the cart, and place orders, which provides comfortable use of the online store. The administrative panel allows the store administrator to add and edit products (provide up-to-date information on the availability and value of goods) and their categories, as well as view and edit existing orders.

The relevance of the online store is determined by the high demand for online services that optimize and simplify customer service, as well as those who do not have the physical ability to visit the store, all these factors contribute to creating and consolidating a positive brand or company image.

Keywords: ONLINE STORE, ADMINISTRATIVE PANEL, WEBSITE, WEB APPLICATION, WEB-DEVELOPMENT

ЗМІСТ

ПЕЛЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК - персональний комп’ютер;

БД - база даних;

СКБД - система керування базами даних;

WWW - World Wide Web;

IT - інформаційні технології;

MVC - Model – View – Controller;

CSS - Cascading Style Sheets;

HTML - Hypertext Markup Language;

JS - Java Script;

PHP - Hypertext Preprocessor;

SQL - Structured Query Language;

URL - Uniform Resource Locator;

GPL - General Public License.

ВСТУП

Інформаційні технології призвели до значних змін у повсякденному житті людини, відтворивши значну частину звичних нам фізичних або матеріальних товарів та послуг у віртуальному світі завдяки комп'ютерним технологіям та мережі Інтернет. Завдяки цьому сучасна людина має змогу відвідати банк, магазин або навчальний заклад, при цьому навіть фізично не покидаючи межі свого будинку. Таким чином, як клієнти так і надавачі послуг отримали у своє розпорядження потужний інструмент взаємодії, що при правильному використання надає найрізноманітніші можливості у покращенні якості послуг, починаючи від їх швидкості та закінчуючи ефективністю, адже автоматизація «виробничих» процесів дозволяє мінімізувати витрати часу, збільшити їх ефективність та прибрати потенційні помилки, що можуть бути обумовлені людським фактором.

У рамках даної кваліфікаційної роботи нас цікавить саме сфера продажу товарів через мережу, тобто така річ як інтернет-магазин. На даний момент сфера продажу товарів та послуг через мережу Інтернет є однією з найбільш розвинутих та поширених як на стороні клієнтів-покупців, так і на стороні продавців (у даному контексті позначає увесь пласт людей, задіяних у замовленні, розробці та адмініструванні інтернет-магазинів). Отже, для вдалого здійснення торгівельної діяльності клієнт потребує інтернет-магазин, що надаватиме йому змогу зручно ознайомитися з товаром та зробити замовлення, а продавець у свою чергу потребує інтернет-магазин як інформаційно-торгівельну систему, що дозволить йому зручно взаємодіяти з магазином на стороні продавця, тобто матиме на сайті інструменти для автоматизації роботи з товарами, замовленнями тощо.

Задачею даної роботи є створення веб-орієнтованого додатку з продажу одягу, що включає в себе як клієнтську, так і адміністраторську частини інтернет-магазину та надає можливість власнику та клієнту ефективно взаємодіяти між собою.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Історія та передумови виникнення мережі «Інтернет»

Перша електронна обчислювальна машина була представлена світу у 1943-1946 роках. Майже через 30 років активного прогресу та вдосконалення наявної технічної бази 1 вересня 1969 року у Лос-Анджелесі встановлюється перший сервер – ARPANET – на той момент ще секретна розробка, яка стала проривом в області комп’ютерних комунікацій та відіграла значну роль у розвитку такої області як комп’ютерні мережі та мережева комунікація. Мережа ARPANET стала першою глобальною мережею у яку було успішно інтегровано усі сучасні мережні розробки та технології, що використовуються і по нинішній час. 1 січня 1983 року мережа ARPANET перейшла з протоколу NCP на протокол TCP/IP, який досі успішно використовується для об'єднання мереж. Саме у 1983 році за мережею ARPANET закріпився термін «Інтернет».

Інтернет, здатний зв'язувати мільйони пристроїв та людей по всьому світу, можна по праву назвати одним з найважливіших досягнень людства. Використання інтернету зробили можливим вільний обмін інформацією між користувачами по усьому світу, тим самим забезпечивши людство якісно новим засобом зв’язку та комунікації. Проте, перші версії інтернету мали обмежений функціонал та були орієнтовані виключно на пересилку файлів і неформатованого тексту. Зрештою фізики Тім Бернерс-Лі і Роберт Кайо (із Женевського ЦЕРНа вирішили розробити інфраструктуру, що дозволить обмінюватися результатами досліджень через інтернет у такому вигляді, яким ми знаємо його наразі - відформатованого й ілюстрованого тексту, що містить посилання. Так було започатковано World Wide Web (WWW) - Всесвітню інформаційну павутину, яка на сучасному етапі охопила своїми мережами практично весь комп'ютерний світ і зробила Інтернет доступним і привабливим

для мільйонів користувачів. Сьогодні Інтернет являє собою об'єднання великої кількості мереж, кожна з яких складається з десятків і сотень серверів. Сервери сполучені між собою різноманітними лініями зв'язку: кабельними, наземним радіозв'язком, супутниковим радіозв'язком. До кожного серверу підключається велика кількість комп'ютерів і локальних комп'ютерних мереж, що є клієнтами мережі [1].

Створення та вдосконалення технологій та електронних обчислювальних пристроїв на призвели до однієї з ключових революцій у історії людства – цифрової та інформаційної. Процес переходу до цифрового простору призвів до стрімкого розвитку та створення нових технологій в різних сферах наукової та практичної діяльності. На сьогоднішній день, інтернет містить близько 1 мільярду 600 мільйонів сайтів (дані на травень 2022 року) та ним користується понад 5 мільярдів користувачів по всьому світу, що складає понад 63% від загального населення.

Наразі майже неможливо уявити сучасний світ без інтернету, настільки тісно він увійшов у наше життя, знайшовши місце майже у будь-якій сфері людської діяльності: починаючи від високотехнологічних виробництв і процесів та до найбільш побутових та соціальних, таких як навчання, спілкування або розваги.

Не буде помилкою, якщо ми скажемо, що інтернет має незліченну кількість застосувань, тому не дивно, що абсолютна більшість сервісів та послуг з фізичного світу знайшли своє місце в цифровому всесвіті - мережі «Інтернет». Так, наразі у цифрову площину частково або навіть повністю перейшли більшість звичних нам сервісів, таких як, наприклад, робота та освіта: фріланс та віддалена робота; усі можливі грошові операції: банкінг, купівля та продаж товарів та послуг онлайн; соціально-орієнтовані: спілкування та обмін повідомленнями; а також розважальні: змога дивитися або розміщувати власні матеріали будь-якого виду та формату, від текстової інформації до фото- та відео- контенту, відеоігри та інші [2].

Прямо зараз можна стверджувати, що інтернет відіграє ключову роль, у певному роді, в еволюції освітніх процесів, дозволивши ефективно адаптувати освітній процес під реалії сьогодення, такі як пандемія або військові дії, по всьому світу. Також використання інтернету дозволяє більш ефективно навчати та навчатися, завдяки наявності доступу до великих обсягів необхідної інформації. Можна сказати, що сучасна людина завдяки інтернету за кілька хвилин здатна знайти значний об’єм необхідної інформації, на пошук якої сто років тому назад могли б знадобитися роки.

Також активне застосування інтернет знайшов у сфері бізнесу. Більшість сучасних бізнесів тим чи іншим чином залежні від інтернету та використовують його у своїй роботі, будь то реклама своєї продукції та послуг, спілкування з клієнтами та постачальниками, отримання виробничої аналітики, слідкування за трендами та новинами у сфері їх діяльності або повноцінне розгортання бізнесу у віртуальній площині, адже наразі можна побачити крупні ринки та сервіси, уся діяльність яких є цілком цифровою та не вимагає від бізнесу виготовлення та надання фізичних товарів або послуг. Сюди ж можна віднести і онлайн-банкінг, який у значній мірі змінив використання банківської сфери. Наразі, маючи доступ до мережі Інтернет, люди мають змогу виконувати усі можливі банківські операції, такі як відкриття, перевірка та налаштування власного клієнтського рахунку, пересилання та отримання грошей, а також зв'язок з технічною підтримкою. Завдяки інтернету усе це можна зробити, навіть фізично не відвідуючи відділення банку, не витрачаючи час на подорож, черги та отримання необхідної допомоги від співробітника банку. Окрім того, що використання сучасних технологій у значній мірі спрощує значну кількість процесів, воно також і робить їх більш безпечними та надійними, адже зменшує потенційний негативний людський вплив.

1.1.2. Основні поняття з теми web-орієнтованих додатків

WEB-браузер

Для перегляду необхідної інформації користувач повинен спочатку відкрити відповідну web-сторінку, група таких сторінок називається web-сайтом. Web-сайт – це сукупність загальнодоступних, взаємопов'язаних web-сторінок, які мають єдине доменне ім'я. Web-сторінкою є окремий файл, написаний відповідними мовами програмування, такими як наприклад HTML або PHP, залежно від мети та задач які покладаються на web-сторінку; більшість web-сторінок мають відповідні розширення: «.html» або «.php». WEB-сторінки формуються за допомогою мови гіпертекстової розмітки HTML (Hyper-Text Markup Language), документи HTML складаються з тексту і коду, який містить інструкції, що вказують браузеру як саме відображати дані на екрані користувача [3].

Інформація, що повинна бути відображена на web-сторінці зберігається на web-сервері та подається користувачу у певному вигляді. Для перегляду web-сайтів користувач на своєму пристрої (комп'ютер, телефон тощо) використовує спеціальні програми, які називаються браузерами.

Робота web-браузера полягає у тому, щоб зробити відповідний запит до web-сервера на якому зберігається необхідна web-сторінка, отримати її зміст у формі спеціального програмного коду та відобразити у вигляді, зрозумілому та зручному для кінцевого користувача. На жаль, не всі розробники браузерів вирішують питання інтерпретації отриманого коду web-сторінки однаково. Для користувачів це означає, що web-сайт може виглядати та функціонувати по-різному. Створення узгодженості між браузерами, щоб будь-який користувач міг користуватися Інтернетом, незалежно від обраного браузера, називається web-стандартами.

Коли web-браузер отримує дані з сервера, підключеного до інтернету, він використовує програмне забезпечення, яке називається механізмом візуалізації, щоб перевести ці дані в текст і зображення. Ці дані записані мовою

гіпертекстової розмітки (HTML), і web-браузери зчитують цей код, щоб створити та відобразити користувачу web-сторінку у такому вигляді, якою ми звикли їх бачити в Інтернеті. Гіперпосилання дозволяють користувачам перейти на інші сторінки або сайти в інтернеті [4]. Кожна web-сторінка, зображення та відео мають свій унікальний уніфікований локатор ресурсів (URL), який також відомий як web-адреса. Коли браузер отримує відповідь від сервера, останній повідомляє браузеру url-адресу, за якою можна знайти та переглянути необхідний web-сайт.

Також за необхідністю браузер може виконувати інші запити до сервера, якщо останній передбачає відповідний функціонал для обробки даних запитів. У випадку успішного виконання запиту web-сервер поверне браузеру відповідь у якій буде міститися результат запиту, або у випадку невдалого звернення до web-серверу поверне повідомлення про помилку, яке буде надано кінцевому користувачу.

WEB-сервер

WEB-сервером називається як програмне забезпечення, виконуючу функцію web-сервера, так і безпосередньо комп'ютер, на якому працює програмне забезпечення. Основним завданням web-сервера є відображення вмісту web-сайту шляхом зберігання, обробки та доставки web-сторінок користувачам. У такому випадку робота web-сервера полягає у відповіді на клієнтські запити, зроблені до конкретного сервера через мережу інтернет: коли web-користувачу потрібна web-сторінка, розміщена на web-сервері, браузер запитує файл за допомогою протоколу HTTP. Коли web-сервер отримує запит, він передається на HTTP-сервер, який у свою чергу знайде вміст необхідної сторінки або файлу відповідно до запиту і надішле його назад до браузера користувача з використанням спеціальних протоколів передачі, таких як, наприклад, HTTP (Hypertext Transfer Protocol) – протокол для передачі текстової інформації, при використанні якого кінцевому користувачу інформація надається у вигляді звичної web-сторінки. Крім HTTP, web-сервери також підтримують SMTP (Протокол передачі пошти) і FTP (Протокол передачі

файлів), які використовуються для електронної пошти, передачі файлів і зберігання [5].

Апаратне забезпечення web-сервера підключено до внутрішньої (локальної) та зовнішньої (глобальна мережа «Інтернет») мереж і дозволяє обмінюватися даними з іншими підключеними пристроями, тоді як програмне забезпечення web-сервера контролює, як користувач отримує доступ до файлів, розміщених на сервері. Процес web-сервера є прикладом моделі клієнт/сервер. Усі комп'ютери, на яких розміщуються web-сайти, повинні мати програмне забезпечення web-сервера. Web-сервери використовуються у web-хостингу, або для розміщення даних для web-сайтів, web-додатків або web-програм. Один сервер може зберігати та обробляти запити до безлічі web-ресурсів, будь то сторінки, сайти, медіа-файли тощо, єдиним реальним обмеженням у такому випадку є тільки апаратна складова, яка визначає потужність, обсяг доступної пам'яті та інші корисні характеристики, які має у своєму розпорядженні комп'ютер, що виконує роботу сервера.

База даних

Базою даних у загальному значенні можна вважати будь-який впорядкований набір даних, проте, у сфері інформаційних технологій, бази даних являють собою більш формалізовану, яка будується відповідно до визначених правил та підтримує роботу з нею інших додатків. Отже, базою даних зветься організована сукупність структурованої інформації або даних, які зберігаються в електронній формі в комп'ютерній системі. База даних зазвичай контролюється системою керування базами даних (СКБД). Разом дані та СКБД разом із додатками, які з ними пов'язані, називають системою баз даних, або ще у більш вживаній формі - бази даних [6].

Дані в найпоширеніших типах баз даних, які використовуються сьогодні, зазвичай зберігаються у вигляді таблиць, які складаються з рядків та стовпців, що робить їх обробку та запити до цих баз даних більш ефективними та зручними. Дані які зберігаються таким чином можна легко отримувати,

змінювати їх, оновлювати, та впорядковувати відповідно до потреб системи у якій вони використовуються.

1.2. Призначення розробки та галузь застосування

Як об'єкт розробки адміністраторської та клієнтської частини web-орієнтованого додатку для продажу одягу розглядається web-додаток «Crinj Shop», в якому користувач може знайти переглянути наявний одяг ручного виробництва від індивідуального дизайнера та передивитися структурований каталог товарів з функціями сортування товарів за такими категоріями як новина, наявність, тип одягу та розмір. Інтернет-магазин розрахований на роздрібну торгівлю, він дає змогу користувачу переглянути повну інформацію про товар, переглянути зображення товару, зареєструватися та авторизуватися на сайті та зробити замовлення на купівлю.

Інтернет-магазин розробляється таким чином, щоб у результаті отримати самостійну інформаційну систему, завдяки якій новий власник сайту (магазину) мав би змогу виконувати керування магазином через наявні у додатку інструменти, навіть не розбираючись у програмуванні. Для комфортного адміністрування магазину розробляється адміністративна панель, у якій адміністратор може дивитися та редагувати замовлення, створювати нові позначки для категорій та редагувати контент магазину: додавати нові товари або оновлювати існуючі.

Розроблений додаток призначений для:

– надання власнику магазину надійного та зручного інструменту для ознайомлення відвідувачів з каталогом інтернет-магазину, який би дозволив надати відвідувачу інформацію про кожен існуючий товар;

– забезпечення відвідувачам сайту простоти і комфортності доступу до каталогу товарів та створення замовлення;

– підвищення продаж магазину за рахунок швидкості роботи інтернет-магазину та швидкості обробки замовлень у адміністративній панелі;

– надання власнику магазину змоги роботи з товарами магазину через панель адміністратора, прибираючи потребу у внесенні змін до коду програми або прямому зверненні до бази даних магазину.

1.3. Підстава для 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» №268-с від 18 травня 2022 р.;

– завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованого додатку з продажу одягу».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є проектування та розробка адміністраторської та клієнтської частини web-орієнтованого додатку для продажу одягу засобами мов програмування HTML, CSS, Javascript і PHP та бази даних MySQL з використанням шаблону проектування та розробки MVC.

Програмне забезпечення призначене для

– надання відвідувачеві змоги переглянути товари, додати їх до кошика, провести реєстрацію та авторизацію у системі, оформити замовлення та переглянути історію замовлень у особистому кабінеті;

– надання власнику магазину універсального інструменту для відображення контенту та адміністрування даного інтернет-магазину (повний спектр функцій для роботи з товарами та замовленнями).

Програма повинна реалізовувати наступні функції:

– легке та доступне адміністрування інтернет магазину, яке полягає у керуванні базою даних товарів та замовлень, використовуючи засоби та інструменти, доступні на сайті;

– формування та заповнення web-сторінок на основі сторінок-шаблонів з використанням інформації, отриманої з бази даних;

– обробка клієнтських запитів, що здійснюються на стороні клієнта;

– перегляд наявних товарів у інтернет-магазині;

– контактування за адміністраторами даного магазину;

– реєстрація та авторизація користувачів на сайті;

– оформлення замовлень.

Для досягнення поставленої мети необхідно:

– вивчити предметну галузь розв'язуваної задачі;

– створити алгоритм для реалізації поставленого завдання;

– створити класи-моделі, що будуть реалізовувати бізнес-логіку, виконувати отримання, обробку та повернення даних для відображення користувачу;

– створити класи-контролери, що будуть реагувати на запити користувача, та відповідно до запиту викликати необхідну модель для обробки, після чого відображати результат обробки користувачу у вигляді представлення;

– створити сторінки-шаблони, що будуть використовуватися для подальшої генерації відображення та повертатися користувачу;

– створити базу даних для збереження інформації про товари, замовлення та користувачів інтернет-магазину.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставлених цілей програмне забезпечення, що розробляється, повинно надавати користувачу змогу переглянути наявні товари,

зареєструватися авторизуватися та зробити замовлення на сайті, а підтримувати зручне та надійне внесення, редагування та зберігання інформації в системі.

Система, що розробляється в кваліфікаційній роботі повинна забезпечувати виконання таких функції як:

– оформлення замовлення клієнтом на сайті;

– підтримка в актуальному стані нормативно-довідкової інформації (НДІ) системи (опис категорій і номерів, прейскурант цін на проживання);

– перегляд, додання та редагування вже введених даних.

Для досягнення поставлених цілей програмне забезпечення, що розробляється, повинно підтримувати виконання наступних дій:

– надання доступу до застосунку через web-браузер на комп'ютері користувача;

– перегляд вже існуючих у базі даних товарів;

– забезпечення зручного додавання нових товарів до бази даних магазину;

– забезпечення зручного редагування інформації про товари, що вже містяться у базі даних магазину;

– підтримка та надання користувачу актуальних товарів (інформацію про товари, що містить назву, категорію, розмір, вартість, детальний опис та наявність товару на складі);

– перегляд існуючих у системі користувачів;

– перегляд оформлених користувачами замовлень;

– надійне та зручне зберігання даних магазину в реляційній базі даних.

1.5.2. Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

– семантичний та синтаксичний контроль вхідних даних;

– обробку виняткових ситуацій;

– виведення повідомлень про помилки;

– можливість повторного введення даних;

– можливість безперервної роботи протягом не менше 120 годин (5 діб);

– платформну незалежність.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для нормального функціонування даного застосунку необхідно, щоб персональний комп’ютер, на якому, буде функціонувати система інтернет магазину, відповідала наступним вимогам:

– процесор класу Intel Pentium з тактовою частотою не менш 2.4 ГГц та двома ядрами;

– доступ до мережі Інтернет;

– не менше 2 GB оперативної пам'яті;

– 500 GB вільного місця на жорсткому диску;

– клавіатура;

– маніпулятор "миша".

Наведені вище технічні характеристики ПК є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний застосунок буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення персонального комп’ютера, на якому буде функціонувати web-орієнтована система, відповідало наступним вимогам:

– операційна система Windows (7+), Linux, MacOs;

– серверне програмне забезпечення з підтримкою PHP-інтерпретатора;

– web-браузер Firefox / Google Chrome / Opera / Microsoft Edge / Safari.

Для побудови зовнішнього вигляду web-сторінок (клієнтської частини) розроблюваного web-додатку мають бути використані такі мови верстки як HTML та CSS, також допускається використання JavaScript для покращення

взаємодії користувача з графічним інтерфейсом web-додатку. Бізнес-логіка web-додатку має бути реалізована на мові програмування PHP. Проект має бути реалізовано за допомогою об’єктно-орієнтованого підходу на базі архітектури клієнт-сервер та відповідно до тришарового шаблону проектування та розробки MVC (Model-View-Controller). У якості сховища даних робочої інформації про товари магазину, користувачів та замовлення має бути використана система керування базами даних MySQL.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

За завданням кваліфікаційної роботи було реалізовано проектування та розробку клієнтської та адміністраторської частини системи відображення і адміністрування інтернет-магазину одягу.

Призначення розробленої системи:

– систематизувати дані компанії;

– забезпечити просте адміністрування магазину;

– надати можливість обмінюватися даними та запитами між користувачем та адміністратором.

Розроблена програма реалізує наступні функції:

– легке та доступне адміністрування інтернет магазину, яке включає в себе; додавання нових та керування наявними товарами, перегляд та редагування наявних замовлень;

– формування web-сторінок на основі шаблону з використанням контенту, отриманого від серверу;

– формування web-сторінок на основі шаблону з використанням контенту, вилученого з відповідної бази даних;

– контактування з адміністратором даного магазину;

– користувальницькі: реєстрація нових та авторизація існуючих користувачів;

– надання користувачам змоги переглянути список їх замовлень у особистому кабінеті;

– автоматизація звернень до бази даних, отримання, обробка та відображення результату запиту кінцевому користувачу;

– робота з кошиком: додання, перегляд або видалення обраних товарів;

– оформлення замовлень на сайті магазину.

Для досягнення поставленої задачі розроблене програмне забезпечення підтримує виконання таких операцій:

– надання віддаленого доступу до застосунку через web-браузер на комп'ютері або іншому девайсі користувача;

– зчитування вхідних даних з хмарних сервісів та бази даних;

– формування замовлення та відправка на адміністративну частину;

– додання нового та корегування існуючого наповнення магазину через адміністративну частину.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці системи відображення та управління контенту інтернет-магазину математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Клієнтська частина розроблюваного web-додатку повинна бути виконана з використанням мов верстки HTML та CSS, також допускається використання JavaScript для покращення взаємодії користувача з графічним інтерфейсом web-додатком. Бізнес-логіка web-додатку має бути реалізована на мові програмування PHP. Робота створена за допомогою об’єктно-орієнтованого підходу на базі архітектури клієнт-сервер та відповідно до тришарового шаблону проектування та розробки MVC (Model-View-Controller). У якості сховища даних має бути використана система керування базами даних MySQL.

2.3.1. Огляд використаної архітектури

Архітектура «клієнт-сервер»

Розроблений web-додаток відноситься до типу програмних засобів, побудованих за архітектурою «клієнт-сервер» (рис. 2.1.).

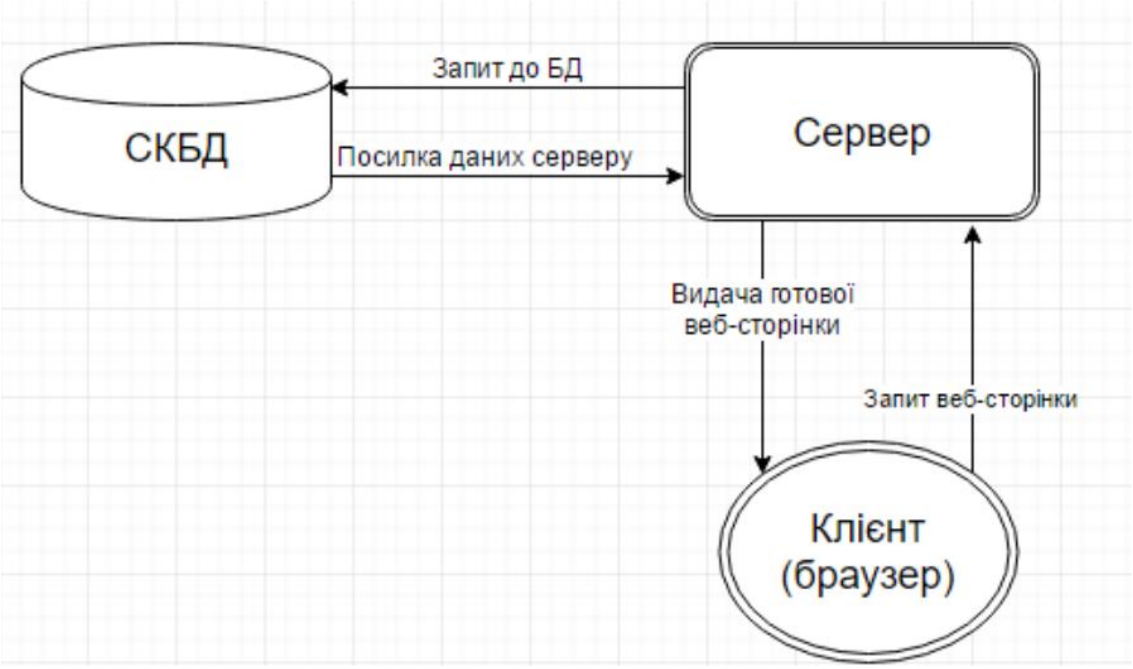

Рис. 2.1. Схема архітектури типу «клієнт-сервер»

Архітектура типу «клієнт-сервер» є однією з найбільш популярних та добре відомих способів організації роботи інформаційних систем та мережевих додатків. Дана архітектура починає свою роботу у той момент коли клієнтський пристрій надсилає до сервера запит, який має бути оброблений на стороні сервера, сформувати результат та відповідь, після чого бути доставленим клієнту та відображеним на екрані його браузера. Даний тип архітектури передбачує наявність наступних компонентів:

– сервер або набор серверів, які являють собою керуючий модуль (manager) або систему таких модулів, що можуть бути об’єднані та розподіляти між собою ресурси для здійснення різноманітної обробки даних та запитів зі сторони клієнта: сервер містить у собі правила або інструкції, написані серверними мовами програмування, які описують алгоритм та процес виконання та надання наявних функцій та можливостей, таких як зберігання, надання доступу та керування усіма можливими ресурсами та послугами, що можуть бути відтворені та надані клієнту;

– від одного до безлічі активних клієнтів (у даному контексті клієнтом позначається будь-який клієнтський пристрій (комп’ютер, смартфон тощо), під’єднаний до центрального сервера у мережі;

– мережа, яка забезпечує взаємодію між клієнтами та серверами. Найбільш часто зустрічається у вигляді звичайного з’єднання через мережу Інтернет або у формі локальних, корпоративних або виробничих мереж організацій, які потребують наявність централізованого пункту зберігання, обробки та надання даних для здійснення своєї діяльності [7].

Сервер може оброблювати запити від кількох клієнтів одночасно, так само як і клієнт може бути одночасно підключений до багатьох серверів, кожен з яких може надавати свій набір послуг. Як сервери, так і клієнти є незалежними один від одного та можуть функціонувати паралельно, жорсткої прив’язки клієнтів до сервері немає. Найбільш часто на практиці можна зустріти ситуацію коли один сервер одночасно обробляє певну множину запитів від різних клієнтів, клієнт при цьому також може вільно звертатися до будь якого з наявних серверів, у загальному вигляді сторона клієнта «знає» про сервери до яких вона може звертатися, але «не знає» нічого про інших клієнтів.

Архітектурний шаблон MVC

MVC (Model–View–Controller) – є одним з найбільш вживаних та популярних способів організації процесу розробки програмних засобів, що використовується на етапах проектування та розробки. Використання якого передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Використовується даний шаблон для надання розроблюваному програмному забезпеченню гнучкого дизайну, який міг би спросити спосіб подальшу взаємодію розробника з програмним продуктом, будь то зміна або розширення наявного функціоналу програми, а також спрощення повторного використання компонентів програми для усунення надлишковості та спрощення сприйняття коду. Крім того, використання цього шаблону у великих системах

сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності [8].

При використанні даного архітектурного шаблону програмний застосунок поділяється на три окремих, але взаємопов'язаних компонента, між якими згруповуються та розподіляються усі можливі функції програмного застосунку.

У загальному вигляді такий компонент як модель являє собою клас або набір класів, що містять у собі усі прояви бізнес-логіки програмного додатку, наприклад, методи, у яких описано яким чином у програмному застосунку виконується зберігання, обробка та структуризація даних, тобто модель є основою, що утворює та описує наявний функціонал та інструментарій кінцевого продукту. Слід зазначити, що модель не починає свою роботу самостійно, а її робота завжди викликається компонентом-контролером, який відповідно до отриманих запитів та дій користувача здійснює виклик, отримання та передачу результатів обробки/роботи моделі до компоненту-вигляду. У моделі описується керування даними, які передаються між компонентами View і Controller, а сама модель не залежить від процесу вводу чи виводу даних та є лише інструментом, що викликається для вирішення оперативних задач [9].

Вигляд (View) відповідає за представлення отриманих або оброблених даних користувачеві, тобто являє собою графічний інтерфейс програмного забезпечення, у рамках web-розробки - html-сторінку, яка обирається контролером з набору заздалегідь створених та визначених під конкретні задачі шаблонів, після чого відповідно до запиту сторінка заповнюється необхідним контентом та повертається користувачу як відповідь на його запит. Також слід зазначити, що компонент-вигляд не має у собі ніяких проявів логіки, він не отримує, не вводить та не виводить дані, а лише є інструментом візуалізації.

Останнім є компонент-контролер (Controller), який здійснює керування існуючими компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель. Контролер діє відповідно до запиту або дій користувача, викликаючи заздалегідь існуючі і визначених алгоритмів та інструкцій, що

описані у компоненті-моделі, після перетворює отримує результат їх роботи та передає на компонент-вигляд. Контролери діють як інтерфейс між компонентами Model і View для обробки всієї бізнес-логіки та вхідних запитів: маніпулювання даними за допомогою компонента Model і взаємодії з Views для відтворення кінцевого результату. Зазвичай контролер являє собою клас, що містить інструкції виклику методів для обробки запитів користувача моделями конкретної сутності бізнес-логіки. Наприклад, контролер клієнта оброблятиме всі взаємодії та введення з перегляду клієнта та буде працювати з областю бази даних, що відповідає за зберігання даних про клієнта, або пов’язаних з клієнтом, за допомогою моделі клієнта. У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас [10]. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля. Зареєстровані події перетворюються у різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися (рис. 2.2.).

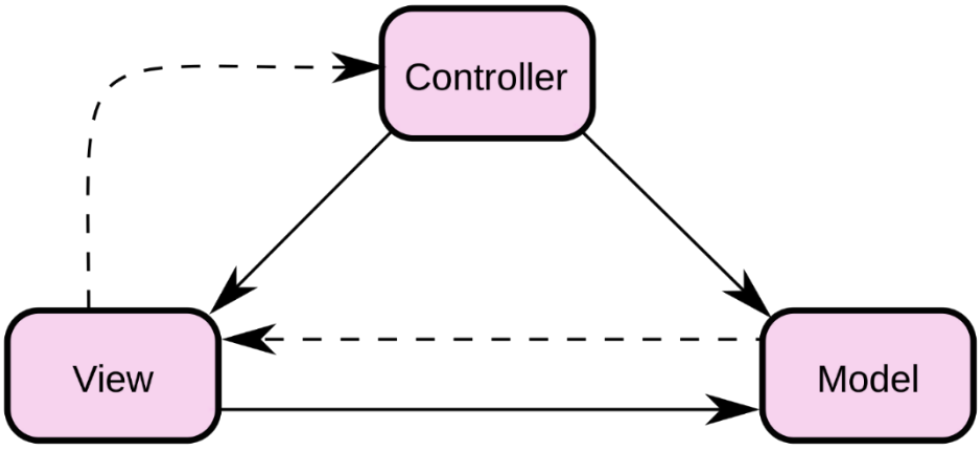

Рис. 2.2. Приклад роботи шаблону MVC

2.3.2. Огляд використаних мов програмування

Мова програмування PHP

PHP це сценарна (скриптова) мова програмування, яка була створена та використовується для генерації кінцевих HTML-сторінок на стороні web-сервера. Хоч популярність PHP і поступово падає, проте буде несправедливо не зауважити, що близько 78,9% усіх сайтів, написаних з використанням серверних мов програмування, написані саме на PHP, тобто можна сказати, що вісім з десяти сторінок, які користувач бачить в інтернеті, написані та працюють саме завдяки PHP, що певним чином свідчить про актуальність мови. Окрім того, мова продовжує отримувати оновлення від розробників, та останні версії, починаючи з версії 7.0 та далі показують значні показники у швидкодії, порівняно з попередніми версіями PHP або іншими серверними мовами програмування [11]. Отже, беручи до уваги усе вищесказане, можна сказати що PHP все ще залишається однією з найпоширеніших мов програмування, що застосовуються у сфері web-розробок, а використання мови PHP на сайті підтримується переважною більшістю хостинг-провайдерів, що є критичним аспектом для коректної роботи web-додатку, функціонал якого написано на PHP.

Оскільки PHP є серверною мовою, то слід зазначити, що, наприклад, на відміну від додатку, написаному з використанням скриптової мови JavaScript, робота якого виконується на стороні клієнта, тобто безпосередньо браузером користувача, робота web-додатку, написаного на мові PHP виконується наступним чином: браузер клієнта формує запит до сервера, після чого останній отримує його, оброблює наявними засобами та інструментам, написаними мовою PHP, після чого інтерпретує результат виконання у HTML-код, який повертається клієнту у вигляді готової web-сторінки з результатом обробки. Це є певною перевагою, адже виконання логіки на стороні сервера дозволяє WEB-додатку бути цілком кросплатформним та не залежати від типу або версії оперативної системи чи прикладних програм, що використовуються на стороні клієнта, від браузера вимагається тільки змога відображати HTML-документи та

виконувати запити до сервера, що і так є обов'язковим та мінімальним функціоналом браузера.

На відміну від іншої скриптової мови JavaScript, користувач не має змоги переглянути PHP-код, що розташований та виконується на сторінці, бо браузер клієнта отримує від сервера лише готову сторінку для відображення у вигляді html-коду. Це надає певні переваги з точки зору безпеки, оскільки виконавчий код скрито від потенційного зловмисника, але погіршує інтерактивність сторінок, роблячи їх роботу цілком залежною від сервера. Проте є варіант використання PHP для генерування JavaScript-кодів, які будуть виконуватися на стороні клієнта, прикладом такої реалізації є Ajax, яку можна за необхідністю використовувати у парі з PHP для того, щоб виконувати часткове оновлення елементів чи блоків web-сторінки без повного її перезавантаження [12].

Мова PHP має певні переваги, такі як наприклад, у можливость вбудовування PHP-код безпосередньо у HTML-код сторінок, після чого сервер обробить таку сторінку з використанням PHP-інтерпретатора та поверне результат відповідно до задачі, що вирішувалася. Але у реальних проектах такий підхід використовується дуже обережно, виносячи усю логіку додатку у окремі класи, та потім за необхідністю використовуючи їх методи, щоб не ускладнювати сприйняття коду сторінки та запобігти повторенню ділянок коду там, де цього можна уникнути. Сюди ж можна віднести значну кількість наявних функцій, інтерфейсів для роботи з іншими сервісами, наприклад, базами даних, та значну кількість сучасних бібліотек та фреймворків, вдале обрання яких, відповідно до поставлених задач, значно спрощує процес розробки.

Ще приємними аспектами роботи з PHP є традиційність коду, а також його відкритість та безкоштовність, тобто значна кількість програмістів, що мають опит роботи з такими мовами як, наприклад, C або Perl, зможуть швидко звикнути та пристосуватися до роботи з мовою PHP через її універсальний та зрозумілий синтаксис. Використання ж стретегії Open Source при випуску мови PHP призвело до вдалого розповсюдження мови та мало благоспритяний на вплив на значну кількість програмістів та проектів, що використовували дану

мову. Окрім цього, завдяки стратегії Open Source, мова PHP і на сьогоднішній день має значну кількість користувачів, які у певному сенсі є свого роду колективною службою підтримки, завдяки чому вирішення типових задач та оперативних питань виконується доволі швидко, оскільки відповіді можна знайти на більшості розробнико-орієнтованих ресурсів, таких як конференції та форуми[13].

Мови верстки HTML та CSS

Можна зазначити, що незважаючи на те, що саме відображається на web-сторінці, від простого тексту до різноманітних складних строкатих зображень та відео, якщо поглянути на код більшості web-сторінок, можна побачити, що усі вони схожі між собою за своєю структурою, нагадуючи дерева з певною ієрархією, це пов’язано з тим, що web-сторінки будуються відповідно до певних правил. В основу синтаксису мови HTML ліг стандарт ISO 8879: 1 986" Information processing. Text and office systems. Standard Generalized Markup Language (SGML). Правда, існує велика відмінність між стандартом офіційному і стандартом фактичним. Мова HTML постійно розвивається, доповнюється новими елементами.

Структура HTML-документа складається з елементів, які у коді подаються у вигляді тегів – спеціальних команд, які керують браузеру який саме елемент та де на web-сторінці потрібно відобразити. Теги бувають парними та одинарними. Парні теги складаються з відкриваючого тегу типу «<ім’я елемента>» та закриваючого тегу «</ім’я елемента>», який дублює правопис відкриваючого, але має «/» на початку. Між відкриваючим та закриваючим тегами знаходиться зміст елемента, що повинен бути відображений – контент, також усі інші елементи, що будуть розташовані між парними тегами називаються «дочірніми» та є ієрархічно вкладеними відносно елемента у якому вони знаходяться, сам же це елемент називається «батьківським» відносно вкладених у нього елементів. Одинарні ж теги мають синтаксис, подібний до відкриваючих тегів, та не потребують закриття, також вони не можуть мати вкладених елементів, а можуть лише самі виступати у даній ролі. Деякі теги можуть містити у собі атрибути, які,

наприклад, вказують на розмір та колір елемента на екрані [14]. Проте найчастіше для визначення зовнішнього вигляду сторінки використовується такий повноцінний інструмент дизайну як CSS.

CSS (Cascading Style Sheets) - каскадні таблиці стилів - спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів. Специфікації мов та версії CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Кожен наступний рівень CSS створюється на основі попередніх, додаючи нові або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі - сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці web-сторінок. Головна перевага блочної верстки - розділення змісту сторінки (даних) та їхньої візуальної презентації.

Завдяки CSS дизайнери та розробники web-сторінок легко можуть задавати для сторінки кольори, шрифти, розташування та багато інших аспектів візуального представлення HTML-документу, написаного на CSS. Таке розділення підвищує доступність документу, його гнучкість та легкість редагування візуальних елементів сторінки, а також спростити документ та зменшити повторюваність в його структурі. Завдяки CSS можна представити одну і ту ж сторінку в багатьох стилях або методах відображення, таких як вивід на екран, друк, голосове відтворення завдяки спеціальним програмам [15].

Стандарт CSS визначає пріоритети, у порядку яких застосовуються правила стилів, якщо для якогось елементу підходять деякі правила одночасно. Це називають «каскадом», в якому для правил розраховуються пріоритети або «ваги», що робить результати передбаченими. Таблиця стилів складається з

набору правил. Кожне правило, у свою чергу, складається з одного або декількох селекторів, розділених комами і блоками визначень. Кожен елемент web-сторінки, який генерується завдяки використанню мови HTML, може мати атрибути, які визначають усі можливі правила поведінки елемента web-сторінки на екрані браузера користувача, такі як колір, розмір, положення та інші [16]. Кожному елементу можна надати уніфіковані атрибути всередині, прописавши їх всередині відкриваючого тегу, або надаючи елементу такі значення як class, та id. Кожен елемент може відноситися до декількох класів, проте містити лише одне значення ідентифікатора id. Множинні значення class записуються через пробіл, . Значення class і id повинні складатися лише з літер, цифр, дефісів та нижніх підкреслень і повинні починатися лише з літер чи цифр.

При завантаженні web-сторінки браузер проглядає (інтерпретує) HTML-документ, вибудовує його структуру (DOM) і відображає її відповідно до інструкцій, включених у відповідний CSS файл (таблиці стилів, скрипти). Якщо розмітка web-сторінки правильна, то у вікні браузера буде відображена HTML-сторінка, що містить HTML-елементи - заголовки, таблиці, зображення тощо у тому вигляді, яким його було створено розробником сторінки [17].

2.3.3. Огляд системи керування базами даних MySQL

Для зберігання та роботи з даними при виконанні завдання даної кваліфікаційної роботи було обрано систему керування базами даних MySQL. MySQL — це система управління реляційною базою даних, розроблена Oracle, яка базується на мові структурованих запитів (SQL). Перевагу було віддано даній СКБД, адже MySQL – це популярне, перевірене часом, але також сучасне та повнофункціональне програмне забезпечення для керування реляційними базами даних. Компанії всюди використовують його для зберігання та обробки критично важливих корпоративних даних, як «бекенд» (основний код, що описує логіку додатків або систем) для основних програм, призначених для клієнтів, і як частину потужних, встановлених стеків веб-програм [18].

Буде справедливо зазначити, що MySQL є однією з найбільш впізнаваних технологій в сучасній екосистемі великих даних, дана СКБД є однією з найпопулярніших баз даних і наразі широко та ефективно використовується у проектах будь-якої галузі та направленості. Можна сказати, що кожен, хто займається корпоративними даними або задіяний у загальних інформаційних технологіях, повинен мати досвід роботи або хоча б базові навички роботи з даною системою керування базами даних. За допомогою MySQL навіть ті, хто вперше працює з реляційними системи, можуть відразу створити швидкі, потужні та безпечні системи зберігання даних. Програмний синтаксис та інтерфейси MySQL також є прекрасним мостом у широкий світ інших популярних мов запитів і систем сховищ структурованих даних.

Основними перевагами, що виділяють MySQL серед ряду інших популярних та не дуже систем керування базами даних, можна назвати:

– сумісність MySQL з значною частиною існуючих технологій, архітектур та операційних систем, хоч MySQL часто асоціюється з інтернет-додатками або веб-сервісами, проте можливості її застосування значно ширші. Архітектура клієнт-сервер MySQL означає, що вона може підтримувати різноманітні серверні програми, а також різні інтерфейси програмування. Завдяки архітектурній та мовній подібності дані можна переносити безпосередньо з MySQL до його підсистем (наприклад, MariaDB), а також до більшості інших СКБД. Встановлені інструменти міграції Oracle і сторонніх розробників дозволяють MySQL переміщувати дані до та з великого набору загальних систем зберігання, незалежно від того, чи розроблені вони як локальні або хмарні. Широка сумісність MySQL з іншими системами та програмним забезпеченням робить MySQL особливо практичним вибором СКБД у більшості ситуацій;

– релятивність - MySQL заснована на реляційній моделі даних. Основний фактор, що відрізняє реляційні бази даних від інших цифрових сховищ, полягає в тому, як дані організовані на високому рівні. Бази даних, такі як MySQL, містять записи в кількох, окремих і чітко кодифікованих таблицях, на відміну від єдиного всеохоплюючого сховища або колекцій напів- чи неструктурованих

документів. Це дозволяє СУБД краще оптимізувати дії, такі як отримання даних, оновлення інформації або більш складні дії, такі як агрегації. Логічна модель визначається для всього вмісту бази даних, описуючи, наприклад, значення, дозволені в окремих стовпцях, характеристики таблиць і представлень, або як пов’язані індекси з двох таблиць;

– відкритий код (проект є Open Source), що означає, що будь-яка особа або підприємство може вільно використовувати, змінювати, публікувати та розширювати базу коду MySQL Oracle з відкритим вихідним кодом. Програмне забезпечення випускається під Загальною публічною ліцензією GNU (GPL). Якщо код MySQL необхідно інтегрувати або включити в комерційну програму (або якщо програмне забезпечення з відкритим кодом не є пріоритетом), підприємства можуть придбати комерційну ліцензійну версію в Oracle. Знову ж таки, ці параметри надають організаціям додаткову гнучкість, якщо вони вирішують працювати з MySQL. Загальнодоступний характер версій та оновлень MySQL із відкритим кодом збагачує документацію та культуру онлайн-підтримки, а також гарантує, що постійні або нещодавно розроблені можливості ніколи не відходять занадто далеко від поточних потреб користувачів;

– легкість використання - хоча реляційна природа MySQL і пов’язані з цим жорсткі структури зберігання можуть здатися обмежуючими, таблична парадигма є, мабуть, найбільш інтуїтивно зрозумілою і в кінцевому підсумку забезпечує більшу зручність використання. Насправді, MySQL робить багато для підтримки найширшого можливого розмаїття структур даних та великого вбудованого набору функцій, екосистема MySQL також включає в себе різноманітні інструменти, які спрощують усе від керування сервером до звітності та аналізу даних. Незалежно від загальної архітектури СКБД, користувачі завжди можуть знайти функцію MySQL, яка дозволяє їм моделювати та кодувати дані, як вони хочуть. MySQL залишається однією з найбільш простих технологій баз даних для вивчення та використання [19].

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Структура і логіка системи

Структура інтернет-магазину складається з таких частин:

- головна сторінка з каталогом товарів;
- сторінка реєстрації та авторизації користувачів;
- сторінка з товарами, що позначені як «Нові»;
- каталог товарів за категоріями;
- сторінка з детальною інформацією про товар;
- сторінка профіля з історією замовлень;
- кошик з обраними товарами;
- контакти з формою зворотного зв'язку.

Адміністративна панель складається з таких вкладок:

- перегляд списку існуючих у базі товарів,
- редагування існуючих у базі товарів;
- додавання нових товарів;
- видалення існуючих товарів;
- інформація про користувачів;
- список замовлень;
- редагування активних замовлень;
- перегляд повної інформації стосовно замовлень.

Так як проект розроблено та реалізований за шаблоном MVC, то виконавчі файли згруповані по репозиторіях, кожна з яких містить один з трьох основних компонентів: моделі, контролери та вигляд. Розроблений сайт є динамічним, тобто для коректної та зручної взаємодії користувача з сайтом потрібно було розробити інструмент, який надавав би змогу автоматизувати та спростити виклик необхідного функціоналу відповідно до оперативних задач, таких як, наприклад, відображення певних сторінок, виклик методів, створення та надсилання запитів тощо. Для вирішення цієї задачі було розроблено та

застосовано програмний компонент, що виконував би функції маршрутизації на сайті. Суть полягає у написанні класу, робота якого починалася би у момент завантаження сторінки та полягала у аналізі та подальшій обробці вхідної адреси URL відповідно до заздалегідь визначених правил вигляду «url request / params => class Controller / method of class Controller / params for method», що дозволяє структурувати та спростити системи адресації на сайті, а також допомагає розробнику краще орієнтуватися у коді, адже кожному запиту залежно від предметної області або сутності до якої він належить відповідає набір класів-моделей та класів-контролерів, що призначені для роботи з даною сутністю. Дана тема пов’язана із поняттям людино-зрозумілих URL адрес, що припускає виключення з адреси сторінки складних параметрів, а також приховати дані, що передаються методами відправки.

Розглянемо процес маршрутизації, тобто обробки запиту у адресному рядку на прикладі з даної кваліфікаційної роботи. У проекті є клас Router у якому описано спосіб отримання поточної адреси сторінки, її розбиття на складники та виклик необхідного компонента-контролера, який у свою чергу залежно від поставлених задач виконує їх, будь то виклик та відображення сторінки або виклик компонента-моделі для здійснення операцій пов’язаних з бізнес-логікою, наприклад отримання або надсилання даних. Алгоритм даного класу полягає виглядає наступним чином: Клас отримує перелік правил «маршрутизації» з заздалегідь визначеного та сформованого масиву routes (який винесено у окремий файл для спрощеного подальшого редагування), у якому зберігається співвідношення вхідної адреси до контролера та метода що має бути викликаний для обробки запиту. Після цього виконується отримання поточної адреси web-сторінки на якій знаходиться користувач, її аналіз, розбиття адреси на параметри відповідно до заздалегідь визначеного шаблону вигляду «Запит» - «Назва контролера / Метод контролера, що має бути викликаний / Параметри, що мають бути передані моделі для виконання запиту». Код класу Router має вигляд:

```
<?php
```

```
    class Router {
        private $routes;
        public function __construct() {
            $routesArray          =          ProjectBase          .
'/configurations/routes.php';
            $this->routes = include ( $routesArray );
        }
        private function splitURI() {
            if ( !empty( $_SERVER['REQUEST_URI'] ) ) { return trim(
$_SERVER['REQUEST_URI'], '/' );
            }
        }

        public function run() {
            $uriLink = $this->splitURI();
            foreach ( $this->routes as $urisArray => $uriPath ) {
                if ( preg_match( "~$urisArray~", $uriLink) ) {
                    //selecting of the Controller and Method (
action ) which should process the request
                    $innerRoute  =  preg_replace(  "~$urisArray~",
$uriPath, $uriLink);
                    $uriComponents = explode( '/', $innerRoute );
                    $ctrlName  =  array_shift(  $uriComponents  )  .
'Controller';
                    $ctrlName = ucfirst( $ctrlName );
                    $actnName  =  'action'  .  ucfirst(  array_shift(
$uriComponents ) );
                    $parameters = $uriComponents;
                    $searchController                              =
ProjectBase.'/controllers/'.$ctrlName.'.php';
                    if ( file_exists( $searchController ) ) {
                        include_once( $searchController );
                    }
                    $routerObject = new $ctrlName;
                    $result    =    call_user_func_array(    array(
$routerObject, $actnName ), $parameters );
```

```
                if ( $result != null ) {
                    break;
                }
            }
        }
    }
}
?>
```

Написання та посилання на класи та методи, що викликаються, спрощено таким чином, щоб прибрати у коді потребу у ручному написанні слів Controller та action – у рамках даного проекту елементи, назва яких починається з action є методами класів-контролерів. Для прикладу розглянемо те яким чином реалізується отримання повного каталогу товарів. Щоб отримати сторінку з повним списком товарів потрібно у адресному рядку браузера прописати шлях «/items» відносно доменного ім’я. Якщо переглянути даний запис у файлі, який описує правила маршрутизації, routes, то ми побачимо, що запиту вигляду «/items» відповідає наступний запис 'items' => 'product/getAllItems', це означає, що для обробки даного запиту потрібно викликати контролер ProductController та його метод actionGetAllItems, який у свою чергу виконує виклик моделі Product, у якій описані усі можливі операції з даними про товари. У даному випадку код методу матиме вигляд:

```
public function actionGetAllItems()    {
   public  $productsList = array();
    $productsList = Product::getAllItems();
    require_once (ProjectBase . '/views/allProducts.php');
    return true;
}
```

Як ми бачимо, викликається метод getAllItems класу Product. Даний метод виконує з’єднання з базою даних, формування запиту на отримання списку товарів за певними параметрами, його конвертацію на масив та повернення результату.

```
public static function getAllItems() {
```

```
            $dataBase = DataBaseConnect::connectToDataBase();
            $productsList = array();
            $queryResult = $dataBase->query( 'SELECT items.id,
           items.name,   items.size,   items.type,   items.price,
items.preview
     FROM   items where status = 1 ORDER BY availability DESC' );
            $i = 0;
            while ( $row = $queryResult->fetch() ) {
                $productsList[$i]['id'] = $row['id'];
                $productsList[$i]['name'] = $row['name'];
                $productsList[$i]['size'] = $row['size'];
                $productsList[$i]['type'] = $row['type'];
                $productsList[$i]['price'] = $row['price'];
                $productsList[$i]['preview'] = $row['preview'];
                $i++;
            }
            return $productsList;
        }
```

Клас Product є компонентом-моделлю, яка реалізує логіку усіх запитів направлених на роботу з даними, що стосуються такої сутності як товари. Після виконання даного методу, результат його виконання має бути переданий відповідному контролеру ProductController у масив productsList, після чого виконується виклик та завантаження сторінки-шаблону, на якій повинні бути відображені отримані товари. Отже, результатом запиту вигляду «/items» є сторінка, на якій буде відображено усі наявні товари (рис. 2.3.).

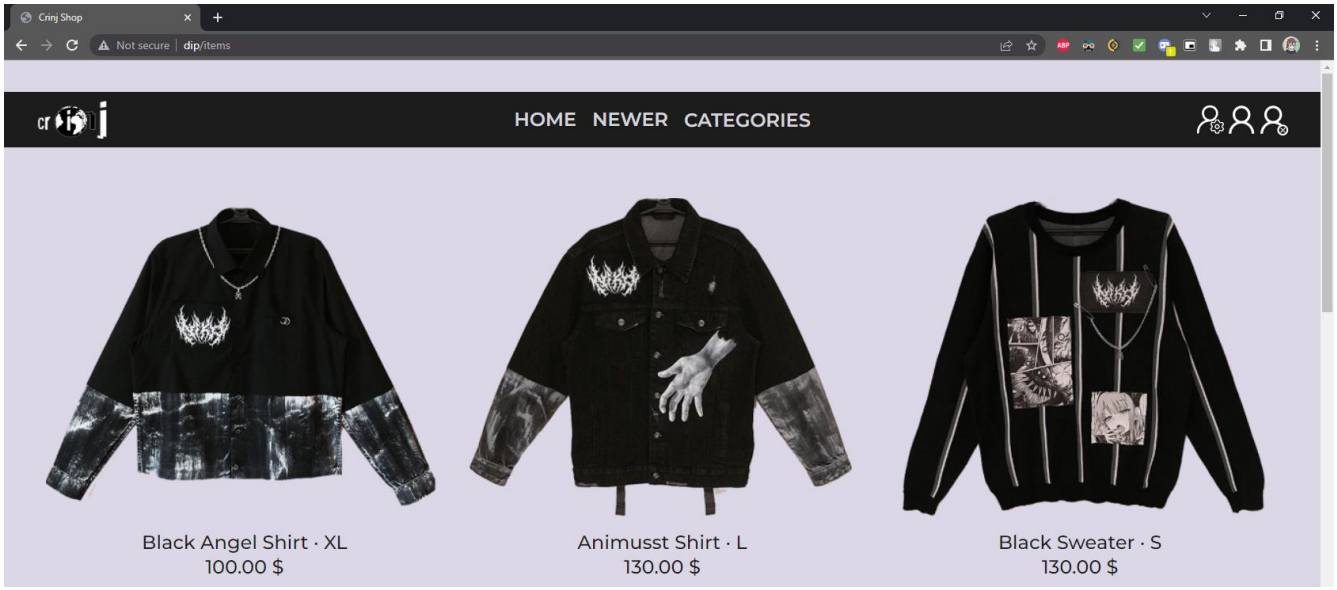

Рис. 2.3. Відображення списку товарів

Схожим чином виглядає і ситуація з більш складними запитами, які допускають наявність більшої кількості параметрів, такі як наприклад отримання інформації про окремий товар. Для цього потрібно ввести запит вигляду «items/([0-9]+)' . Якщо ми знов звернемося до файлу з правила маршрутизації, то побачимо наступний рядок: 'items/([0-9]+)' => 'product/getItemById/$1', це означає, що для обробки запиту, який складається зі слова «items», косої лінії – слешу «/» та хоча б однієї цифри буде викликано клас ProductController, та його метод actionGetItemById, якому у якості параметра буде передано значення, що знаходиться у адресі після слешу. У даному випадку число, що передається відповідає ідентифікатору товару, інформацію про який потрібно відобразити користувачеві. Переглянемо код методу actionGetItemById контролера:

```
public function actionGetItemById($id)
    {
        public $productItem = array();
        $productItem = Product::getItemById($id);
        require_once (ProjectBase . '/views/itemShow.php');
        return true;
    }
```

Як ми бачимо, він не дуже відрізняється від методу, який було використано для отримання та відображення повного списку товарів, окрім того, що цього разу ми передаємо у якості параметра функції значення ідентифікатора за яким

потрібно здійснити пошук. Тепер переглянемо як виглядає метод getItemById($id) для обробки даного запиту у класі-контролері Product:

```
    public static function getItemById( $id ) {
            $id = intval( $id );
            $_SESSION['idView'] = $id;
            if ( $id ) {
                $dataBase = DataBaseConnect::connectToDataBase();
                $queryResult = $dataBase->query( 'SELECT items.id,
          items.name,       items.size,       items.type,.price,
items.details,  items.availability,  items.status,  items.preview,
pictures.pic1,   pictures.pic2,   pictures.pic3,   pictures.pic4,
pictures.pic5, pictures.pic6, pictures.pic7, pictures.pic8
    FROM   items
    INNER JOIN pictures
                  ON items.id = pictures.id_item
    where items.id =' . $id );
                   $queryResult->setFetchMode(   PDO::FETCH_ASSOC
);
                   $productItem = $queryResult->fetch();
                return $productItem;
            }
        }
```

Даний метод виконує запит до бази даних, отримує повну інформацію про товар та перелік посилань на зображення товару, які потім будуть використані для заповнення сторінки товару. Результат повертається у вигляді асоційованого масиву, де ключем є назва колонки з таблиці у базі даних. Переглянемо результат обробки запиту вигляду «items/1», результатом якого має бути web-сторінка, на якій має бути інформація про товар та його зображення (рис. 2.4.).

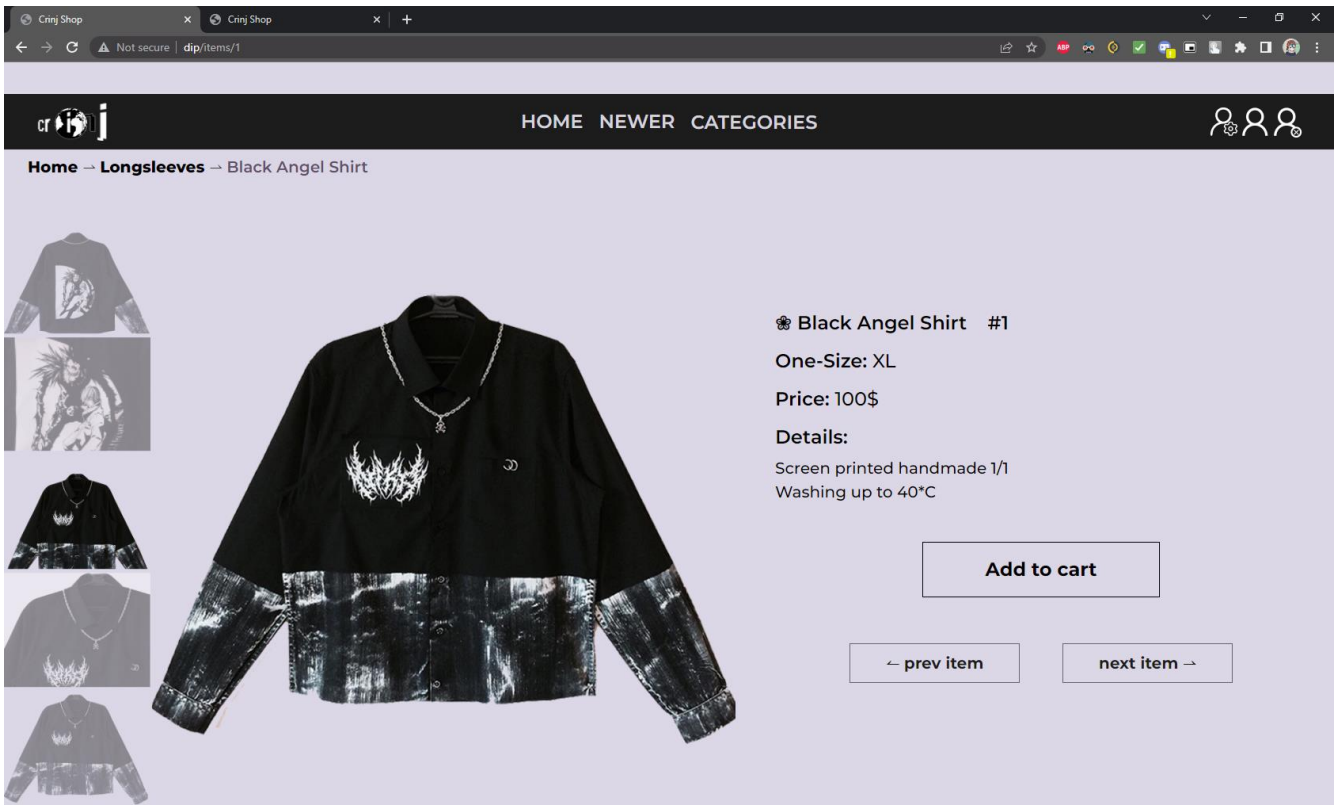

Рис. 2.4. - Відображення обраного товару

Оскільки адреса у адресному рядку завжди використовується у якості поточного запиту, то аналогічний підхід до виклику необхідних засобів обробки можна виконувати і задавши гіперпосиланням у якості адрес (параметр href) необхідні запити. Так, щоб відкрити сторінку обраного товару зі сторінки із усіма товарами достатньо тільки додати елементу, що відповідає за відображення даного товару, посилання, яке б містило у собі запит на отримання сторінки товару. У коді це реалізується наступним чином: навколо кода елемента, що відображає товар відкривається тег для позначення гіперпосилань <a>, та всередині у атрибуті href прописується адреса вигляду «items/», після чого необхідно за допомогою мови PHP додати текстове значення ідентифікатора даного товару. Виглядає це наступним чином: <a href="/items/<?php echo $productItem['id'] ?>"> /Код елемента з товаром/ . Тепер ми маємо сторінку з переліком товарів, кожен з яких автоматично отримав посилання на запит, що відобразить інформацію про нього (рис. 2.5.).

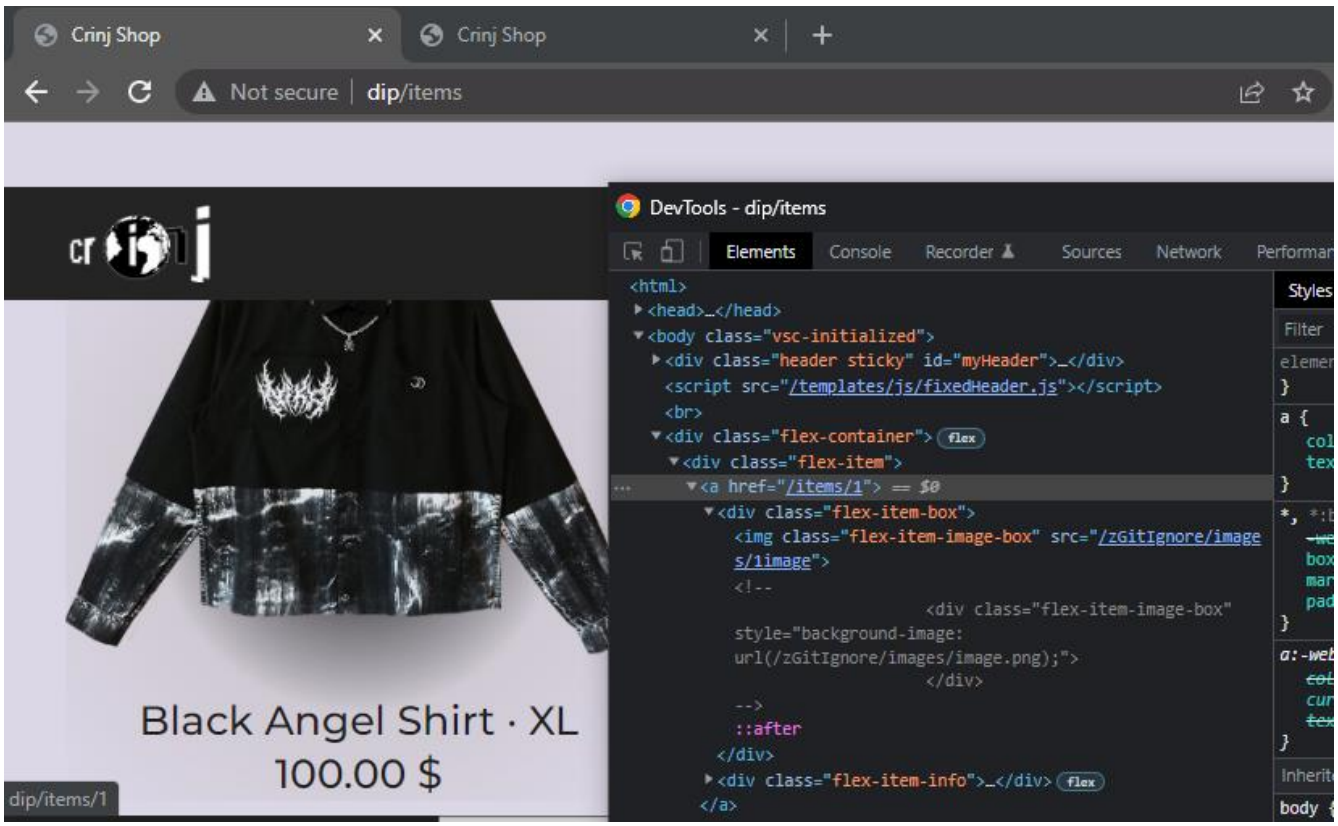

Рис. 2.5. - Привласнення елементу товару посилання-запиту

Так було розглянуто базові принципи взаємодії між собою компонентів моделі, контролера та відображення для виконання запитів користувача. Інший функціонал сайту має схожий вигляд, відрізняючись тільки варіативністю та складністю задач що виконуються.

Структурне розділення та згрупування усіх робочих файлів за їх типами та цільовими задачами у вигляді дерева значно спрощує процес розробки та пошуку необхідних елементів проекту та ділянок коду. Отже, для кожної сутності, що може бути представлена у системі було створено окремі класи, що своєю назвою відображали б суть сутності на роботу з якою вони спрямовані (рис. 2.6.).

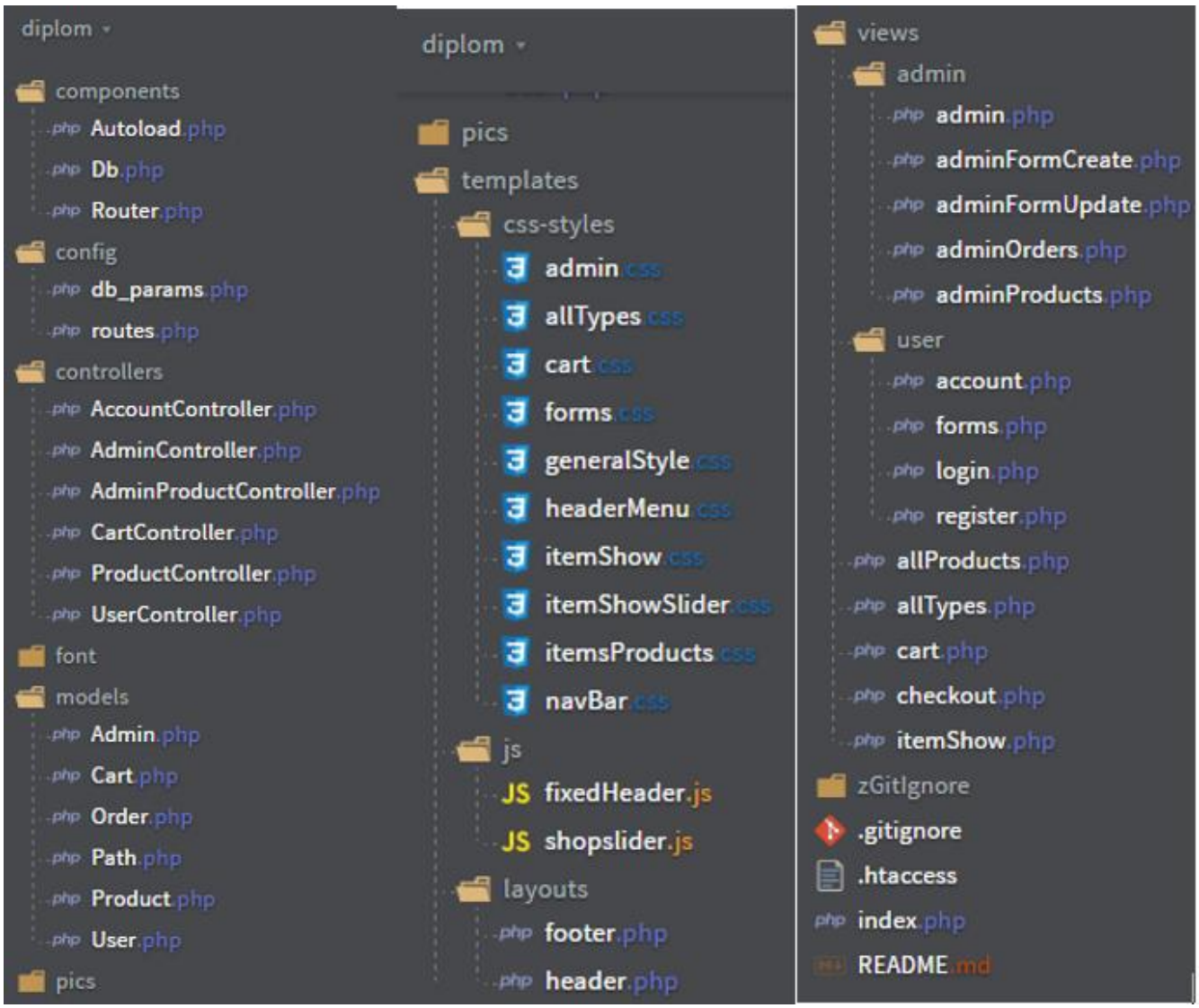

Рис. 2.6. Файлова структура проекту

Файли сайту інтернет-магазину поділені на компоненти, що використовуються більше одного разу у додатку та сторінки, що містять окремі папки з власними компонентами.

Папка components містить загальні класи, що не відтворюють собою сутності системи, а лише є інструментами для зручної роботи для спрощення процесу повторного використання функцій та методів, що часто використовуються, такі як наприклад клас Router, що здійснює перекладання адреси та перетворює її на інструкцію до виклику та виконання класів-контролерів та їх методів, у клас Db винесено усі методи, спрямовані на під’єднання до бази даних, а Autoload містить функцію для автоматичного завантаження усіх існуючих у проекті класів відповідно до заданих репозиторіїв,

це значно спрощує процес розробки, адже прибирає необхідність у ручному підключенні необхідних класів кожного разу.

У папку config винесено усі файли, що не є виконавчими, проте зберігають у собі інструкції, що часто використовуються іншими компонентами та можуть потребувати оперативних змін, так, наприклад у файлі db_params зберігаються параметри підключення до бази даних, такі як інформація про користувача та базу даних до якої треба виконати підключення. Файл routes вже згадувався раніше, у ньому зберігається масив, за яким виконується співвідношення отриманого запиту до класу-контролеру та методу яким він має бути оброблений.

Папка controllers містить усі класи-контролери, що використовуються у системі. Якщо подивитися на їх назву, то стає зрозуміло за роботу з якою сутністю систему кожен клас відповідає.

У папці models зберігаються усі класи-моделі, якими реалізується логіка роботи додатку. Якщо подивитися на їх назву, то стає зрозуміло за роботу з якою сутністю систему кожен клас відповідає.

Папка templates містить повторювані, проте не функціональні елементи, такі як стилі та дизайнерське оформлення інтернет-магазину, що розташовані у під папці css-styles; JavaScript-скрипти, що зберігаються у папці js виконують косметичну функцію на сайті, та не реалізують бізнес-логіку; папка layouts зберігає header та footer елементи сайту, у яких наведено спільні елементи оформлення та підключення файлів для усіх сторінок.

Папка views містить усі сторінки-шаблони, що згруповані за їх метою використання, такі як наприклад, user та admin, що зберігають компоненти-відображення для сторінок, що розраховані на роботу з клієнтами та адміністраторською панеллю. Інші файли виконують відображення сторінок, розрахованих на роботу з товарами інтернет-магазину.

Також для більш зручного та стабільного процесу розробки використовувалася система контролю версій GitLab (рис. 2.7.), яка дозволяє систематизувати процес розробки, зберігати версії додатку відповідно до

обраних розробником контрольних точок, таким чином забезпечуючи можливість завжди повернутися до попередніх у випадку помилки або просто тримати нову та актуальну версію додатку у відкритому доступі. Даним інструментом користуються майже усі більш-менш просунуті розробники, починаючи від фрілансерів та ентузіастів, та до крупних компаній.

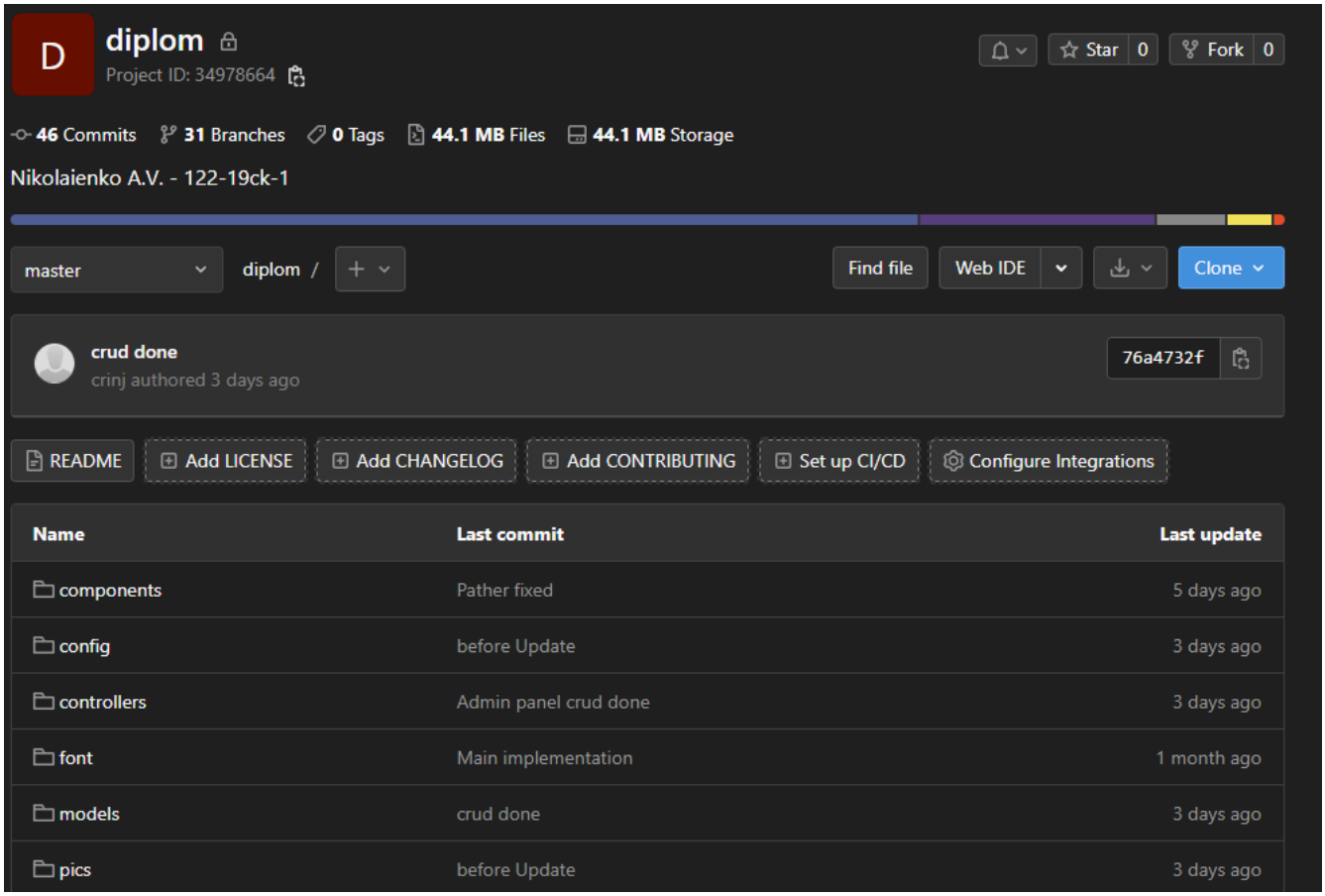

Рис. 2.7. Сторінка проекту на ресурсі GitLab

2.4.2. Структура бази даних

Використана при виконанні проекту кваліфікаційної роботи база даних є реляційною, тому потрібно було заздалегідь визначитися з сутностями, що будуть використані у системі та як вони та їх відносини будуть реалізовані у вигляді таблиць бази даних. У системі є три основні сутності, які можна назвати ядром інформаційної екосистеми та навколо роботи з якими будується увесь подальший функціонал. Дані сутності являють собою товари (Items), замовлення (Product_order) та користувачів системи (Users), усі інші таблиці (pictures, category та order_customer_info) є допоміжними та використовується для

зберігання менш значної інформації для підтримки цілісності основних сутностей (рис. 2.8.).

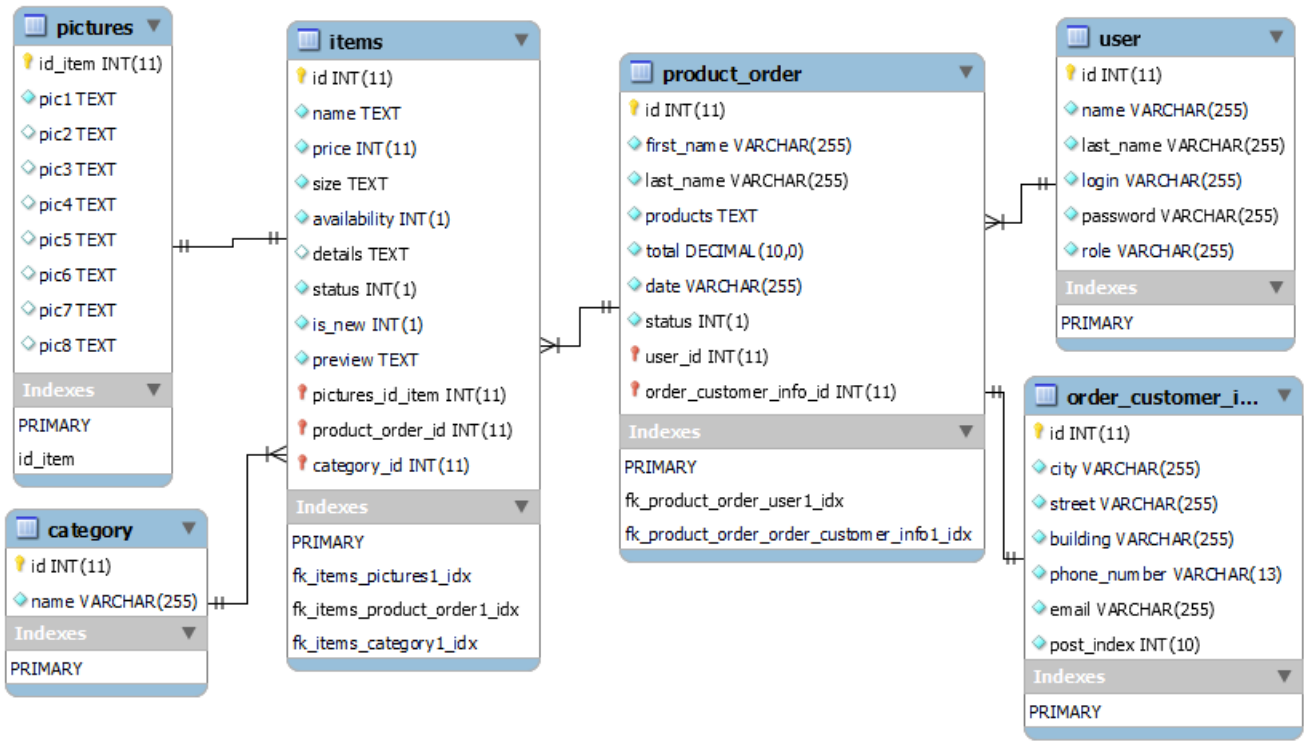

Рис. 2.8. ER-діаграма зі структурою бази даних системи

Розроблена ER-діаграма та база даних, відповідно, відповідають покладеним на систему задачам; а для того, щоб краще їх зрозуміти, потрібно спочатку зазначити основний принцип роботи даного магазину: магазин одягу містить товари ручної роботи, що існують у єдиному екземплярі, тобто робимо логічний висновок, що кожен товар є унікальним та може бути проданим лише один раз, тобто кожен проданий товар може бути присутнім тільки у одному замовленні та може належати лише одному клієнту, після чого стає недоступним для купівлі. Тепер можна розглянути логіку побудови сутностей та відносин бази даних на прикладі наведеної (рис. 2.8.) ER-діаграми.

Кожному товару (items) відповідає його особистий набір зображень з таблиці (pictures), тому сутності items та pictures мають відносини один до одного (one to one relationship) та поєднуються зовнішнім ключем, що зберігається у сутності items та посилається на первинний ключ pictures.

Кожен товар містить атрибут «категорія», що зберігається у таблиці category, кожен товар може мати лише одну категорію, проте до однієї категорії

може відноситися безліч товарів, тому сутності category та items мають відносини один до багатьох (one to many relationship) та поєднуються зовнішнім ключем, що зберігається у сутності items та посилається на первинний ключ category.

Кожен товар може бути продано лише один раз, тобто унікальний товар належить до певного замовлення, проте одне замовлення може містити безліч унікальних товарів, що були придбані у рамках одного замовлення, отже сутності items та product_order мають відносини один до багатьох (one to many relationship) та поєднуються зовнішнім ключем, що зберігається у сутності items та посилається на первинний ключ product_order.

Замовлення здійснюється користувачем, кожен користувач може мати безліч унікальних замовлень, проте кожне унікальне замовлення може належати лише одному користувачу, отже сутності user та product_order мають відносини один до багатьох (one to many relationship) та поєднуються зовнішнім ключем, що зберігається у сутності product_order та посилається на первинний ключ user.

Кожному замовленню з product_order відповідає унікальний кортеж з таблиці order_customer_info, що містить контакти клієнта, що використані при даному замовленні, у загальному випадку кожному унікальному замовленню відповідає унікальна інформація про контакти клієнта, отже сутності product_order та order_customer_info мають відносини один до одного (one to one relationship) та поєднуються зовнішнім ключем, що зберігається у сутності product_order та посилається на первинний ключ order_customer_info.

Перерахуємо усі наявні у базі даних таблиці, їх атрибути (характеристики або властивості опису, які визначають усі елементи, що належать до певної категорії) та їх призначення у системі:

- items – зберігає основну інформацію про товари, таку як:
  - id – первинний ключ, ідентифікатор даного товару;
  - name – назва даного товару;
  - price – вартість даного товару;
  - size – розмір даного товару;

– availability – наявність даного товару;

– details – опис даного товару;

– status – статус, тобто чи потрібно відображувати даний товар у каталозі магазину;

– is_new – атрибут, що використовується для позначення товару як нового та для відображення при формуванні каталогу нових товарів;

– preview – містить зображення, що використовується для попереднього перегляду вигляду товару, що використовується у каталозі;

– picrutes_id_item – зовнішній ключ, що містить посилання на кортеж з набором зображень, що відповідають даному товару;

– product_order_id – зовнішній ключ, що містить посилання на кортеж з замовленням якому належить даний товар;

– category_id – зовнішній ключ, що містить посилання на кортеж з категорією, до якої належить даний товар;

– pictures - містить зображення для товарів:

– id – первинний ключ, ідентифікатор даного набору зображень;

– pic1..pic8 – містять зображення конкретного товару;

– category – таблиця, що містить категорії товарів:

– id – первинний ключ, ідентифікатор даної категорії;

– name – назва даної категорії;

– product_order – таблиця, що містить повну інформацію про замовлення:

– id – первинний ключ, ідентифікатор даного замовлення;

– first_name – ім’я клієнта;

– last_name – прізвище клієнта;

– products – перелік товарів, що були замовлені, містить одним записом ідентифікатор, назву та розмір товарів, що було замовлено;

– total – повна вартість замовлення;

– date – дата створення замовлення;

– status – статус даного замовлення, відображає стан обробки замовлення: 1 - замовлення прийнято, 2 - замовлення виконується, 3 - замовлення виконано та закрито;

– user_id – зовнішній ключ, містить посилання з ідентифікатором користувача, що здійснив замовлення; якщо відвідувач не авторизований, то даний атрибут дорівнює нулю;

– order_customer_info_id – зовнішній ключ, містить посилання на кортеж з описом деталей про дане замовлення;

– user – таблиця, що містить інформацію про користувача, що заповнюється після реєстрації та потрібна при авторизації:

– id – первинний ключ, даного користувача;

– first_name – ім'я користувача;

– last_name – прізвище користувача;

– login – логін для авторизації;

– password – пароль для авторизації;

– role – атрибут-роль, використовується для розділу доступу до частей сайту (за замовчуванням = user, такі користувачі не мають доступу до адміністративної панелі; користувачі з ролями admin мають доступ до адмінстративної панелі та можуть здійснювати звідти керування контентом сайту);

– order_customer_info – таблиця, що зберігає більш детальну інформацію про замовлення, таку як адресу та контакти клієнта:

– id – первинний ключ, ідентифікатор даного запису;

– city – місто куди очікується доставка замовлення клієнта;

– street – вулиця куди очікується доставка замовлення клієнта;

– building – будівля куди очікується доставка замовлення клієнта;

– post_index – поштовий індекс;

– email – електронна адреса клієнта;

– phone_number – мобільний номер телефона клієнта.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані шляхом завантаження інформації із бази даних та сховищ даних, та через передачу значень від інших підсистем через глобальні змінні додатку інтернет-магазину.

Вхідні дані:

- інформація про товари магазину;
- таблиця з зображеннями товарів;
- таблиця типів або категорій товару;
- списки користувачів;
- списки замовлень.

Вихідні дані:

- web-сторінки інтернет-магазину одягу;
- список замовлень користувача;
- кошик користувача;
- загальний список замовлень для адміністратора;
- список товарів для адміністратора;

Дані програми організовані в локальні сховища даних, для кожної сторінки – компонента, що виступає основним контейнером. Локальні сховища організують між собою одне велике сховище даних, до якого підключені усі основні компоненти програмного додатку. Компоненти отримують дані з глобального сховища, та реагують на кожну зміну даних. Дані зі сховища потрапляють у програмний додаток у відповідь на дії та запити користувача з використанням компонентів-моделей, що відповідають за отримання, обробку та повернення даних. Залежно від конкретного запиту буде отримано та відображено дані у необхідному вигляді.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для серверних технічних засобів рекомендована конфігурація, що забезпечує цілодобову роботу програми з резервуванням даних:

– процесор класу AMD Ryzen 5 2600 3.4(3.9)GHz 16MB sAM4 Box (YD2600BBAFBOX);

– материнська плата Gigabyte GA-A320M-H (sAM4, AMD A320);

– модулі пам’яті G.Skill DDR4 8GB 3000Mhz Aegis (F4-3000C16S-8GISB);

– система охолодження Deepcool GAMMA ARCHER;

– SSD диск Transcend MTS820S TLC 120GB M.2 (2280 SATA) (TS120GMTS820S);

– рідкокристалічний монітор з діагоналлю не менше 17";

– доступ до мережі Internet;

– маніпулятор "миша" та клавіатура.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником. Також на сервері обов’язково має бути встановлено транслятор та інтерпретатор мови PHP та систему керування базами даних MySQL.

2.6.2. Використані програмні засоби

Для комфортного та практичного проектування та розробки сайту було використано відповідне програмне забезпечення:

– Brackets - Release 2.0 build 2.0.1-17920;

– Wampserver x64;

– Adobe Photoshop CS6;

– Figma.

Середовище розробки Brackets

Проект реалізовано на мовах програмування PHP, HTML, CSS, JavaScript та SQL. Для написання коду та перевірки його роботи було використано редактор Brackets - вільний текстовий редактор для web-розробників. Не дивлячись на те, що Brackets позиціонує себе як текстовий редактор, по факту він все більше нагадує повноцінну IDE(Integrated Development Environment – Інтегроване середовище розробки - комплексне програмне рішення для розробки програмного забезпечення. Brackets орієнтований на роботу з більшістю мов програмування та особливо web-програмування, таких як HTML, CSS, PHP і JavaScript (рис. 2.9.). Ці ж технології лежать в основі самого редактора, що забезпечує його кросплатформенність, тобто сумісність з операційними системами Mac, Windows і Linux. Brackets створений і розвивається Adobe Systems під ліцензією MIT License та підтримується на GitHub.

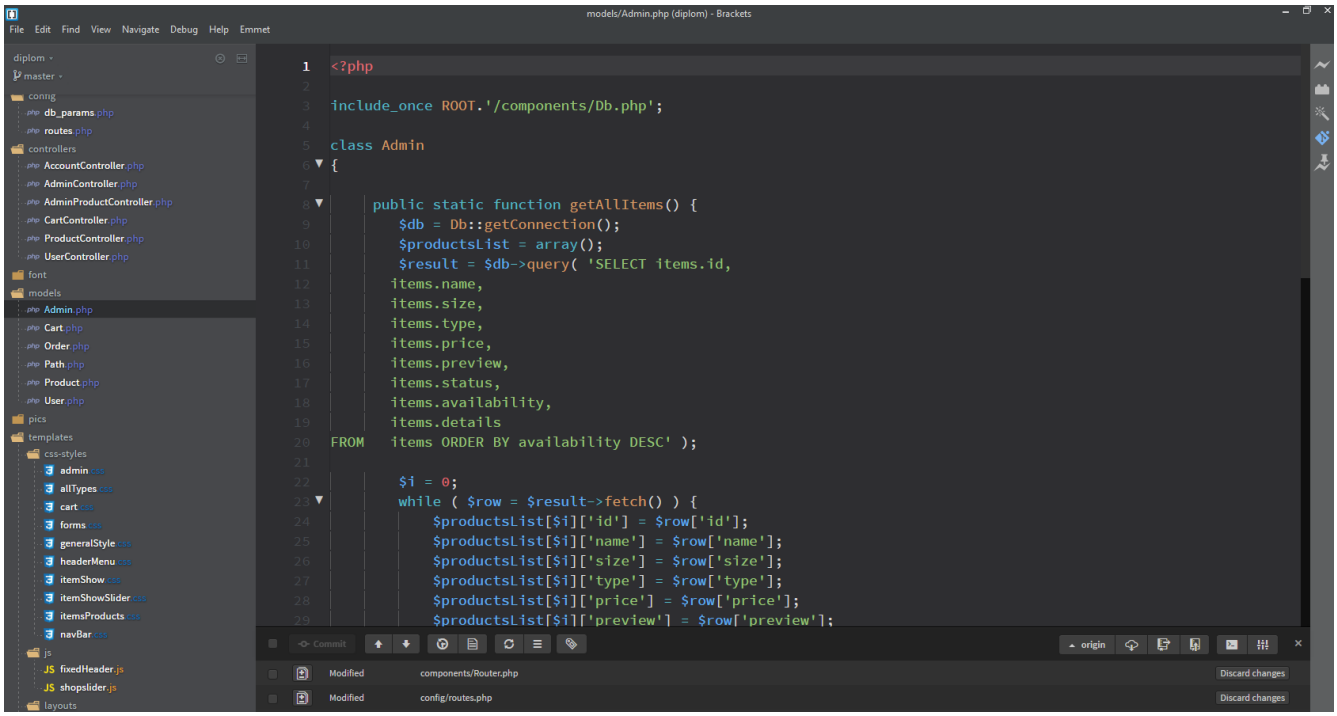

Рис. 2.9. – Вікно редактора Brackets

На сьогоднішній день співтовариством користувачів Brackets створено безліч розширень, що надають розширені інструментів для роботи над кодом та прискорення процесу розробки, такі як, наприклад, система контролю версій Git, перегляд HTML-коду в браузері в реальному часі (Live Preview), синхронізація з

FTP (Git-FTP), Beautify/Beautifer для автоматичного візуального покращення коду (вирівнювання усіх відступів з урахуванням вкладеності коду у батьківський блок), та багато інших. Взяти участь в розробці і підтримці розширень може будь-хто.

Цей редактор було обрано низку значних переваг, адже у ньому значно легше та комфортніше працювати, ніж у багатьох інших програмах типу «Блокнота», більш за все завдяки функції «Підсвічування синтаксису» - у допомогу для написання коду пропонуються відступи та кольорове маркування сегментів для більш легкого розпізнавання та упорядкування всіх елементів проекту в робочій області, де ви можете все розмістити по категоріям у вигляді дерева. Це значно облегшує пошук необхідної частини документу, оскільки код майже завжди виходить достатньо громіздким і розібратися в ньому без кольорової селекції доволі важко. Також корисною є функція автоматизації написання коду, тобто при роботі з HTML тегами редактор автоматично рекомендує можливі варіанти тегів та автоматично виставляє закриваючі теги. При роботі з іншими мовами програмування також з’являються рекомендації, що допомагають прискорити написання коду [20].

WEB-сервер Wampserver

Wampserver - збірка web-сервера, що містить Apache, MySQL, інтерпретатор скриптів PHP, phpMyAdmin і інші доповнення, призначена для web-розробки під Windows. Значною перевагою даної збірки є те, що вона при встановленні автоматично проводить налаштування та встановлення важливих для web-розробки компонентів, а саме : Apache, MySQL, інтерпретатор скриптів PHP та phpMyAdmin. Також Wampserver може створювати віртуальні хости. Без цього працювати над проектом буде неможливо. Wampserver дозволяє обрати версію та налаштувати параметри, модулі та сервіси Apache, MySQL, PHP та phpMyAdmin в залежності від ваших вподобань та потреб проекту [21].

Керування роботою Wampserver виконується через браузер, для цього потрібно ввести у строку пошуку localhost, тоді відкриється web-сторінка, що складається з двох блоків (рис. 2.10.): у верхньому опис поточної конфігурації

сервера та завантажені додатками, у нижньому перелік доступних Інструментів (Tools) (таких як інформація про поточну версію PHP, доступ до панелі phpmyadmin для зручного керування базою даних MySQL, та можливість створення віртуального хосту), Ваші проекти (Your Projects) (яким відповідають спеціальні папки, розташовані у каталозі самого серверу), та Ваші віртуальні хости (Your VirtualHost) (директива в файлі конфігурації web-сервера Apache, призначена для зіставлення доступних на сервері IP-адрес, доменів і директорій на сервері, а також управління доступними на сервері сайтами).

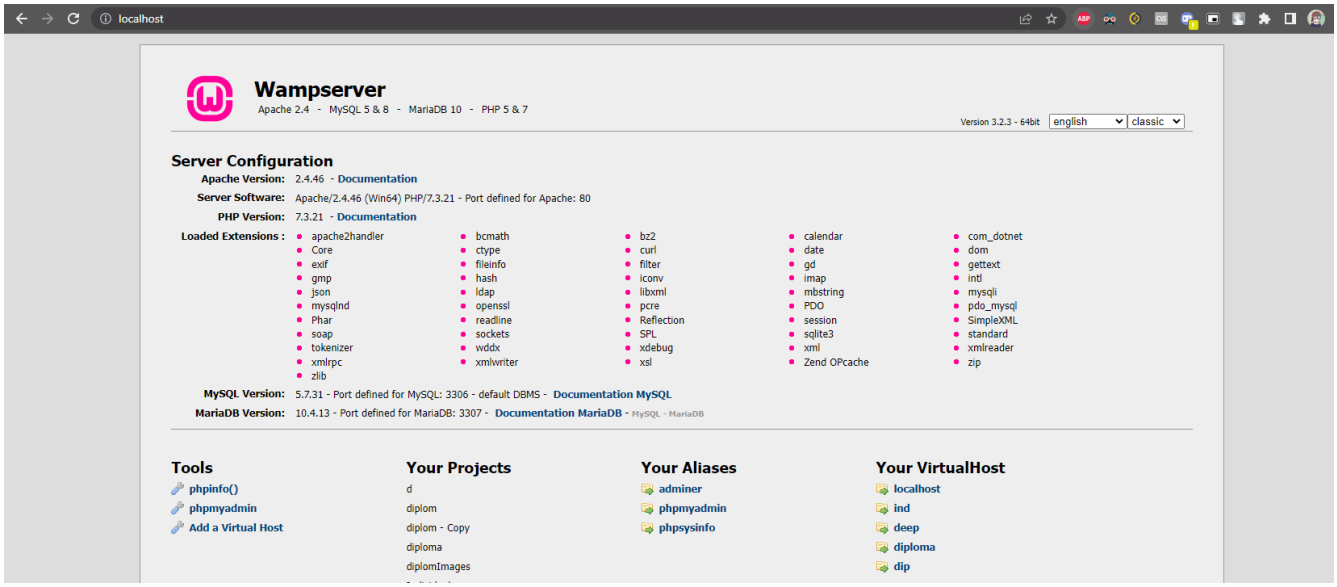

Рис. 2.10. Робоче вікно Wampserver у браузері

Для подальшої роботи над проектом потрібно створити віртуальний хост (VirtualHost). При розробці мого проекту я використовував створений віртуальний хост dip. Йому відповідає спеціальна папка у середовищі каталогу файлів Wampserver. При запиті у пошуковому рядку браузеру dip, відкривається наш віртуальний хост на локальному сервері (рис. 2.11.).

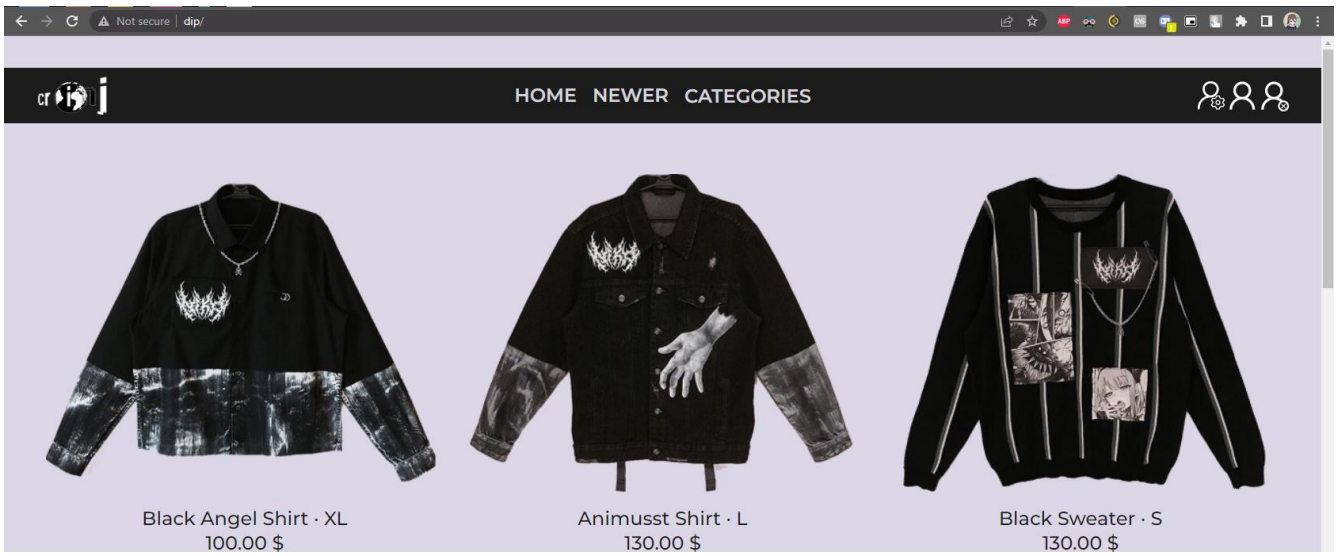

Рис. 2.11. Перегляд хосту dip у браузері

При запиті автоматично відкриваються файли проекту з назвою index та одним з розширень .html, .htm або .php, їх пріоритет також можна налаштувати. Також тут знаходяться усі файлі проекту, такі як PHP-файли, зображення, сторінки-шаблони та інше (рис. 2.12.).

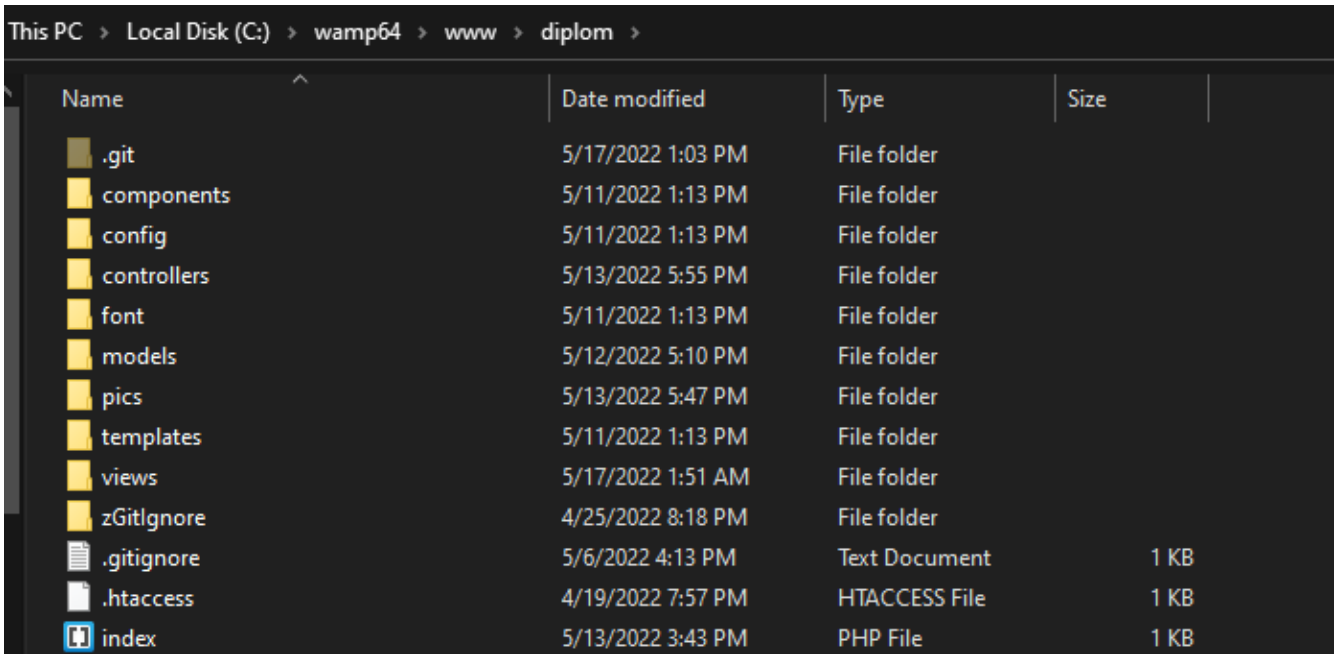

Рис. 2.12. Зміст каталогу diplom

Графічний редактор Adobe Photoshop

Для підготовки та обробки зображень, що мають бути використані на сайті використовувався графічний редактор Photoshop. Photoshop це графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в області комерційних засобів редагування растрових зображень,

і найбільш відомим продуктом фірми Adobe. Часто цю програму називають просто Photoshop (Фотошоп). В даний час Photoshop доступний на платформах Mac OS X/Mac OS і Microsoft Windows.

Незважаючи на те, що спочатку програма була розроблена як редактор зображень для поліграфії, в даний час вона широко використовується і у web-дизайні. Абревіатура CS означає те, що програма належить до Creative Suite, тобто програмного пакету від Adobe, який тісно інтегрований для творчої діяльності.

Програмне забезпечення Adobe Photoshop CS відкриває нові перспективи роботи з цифровими зображеннями, поєднуючи в собі потужні інструменти для роботи з фотографіями, чудові можливості виділення і розфарбовування зображень, а також функцію інтелектуального ретушування.

Сервіс проектування графічних інтерфейсів Figma

Figma це сервіс для проектування графічних інтерфейсів додатків будь-якої спрямованості, від мобільних до комп'ютерних, що дозволяє побудувати прототип графічного дизайну розроблюваного додатку та визначитися з тим як саме має виглядати додаток на екрані користувача, обравши версію, розмір, формат екрану та створити макет, що буде реагувати на поведінку та дії користувача над додатком, дозволяючи анімувати макет та надати приблизне уявлення того яким саме має бути кінцевий програмний продукт.

Сервіс має як браузерну версію, так і десктопну версії (рис. 2.13.), що мають аналогічний функціонал, хоча десктопна і має значні переваги у швидкодії. Також можна зазначити, що сервіс Figma дозволяє поширити свій проект, надавши іншим користувачам змогу переглянути або редагувати його, тим самим роблячи можливим команду розробку та проектування прототипу проекту.

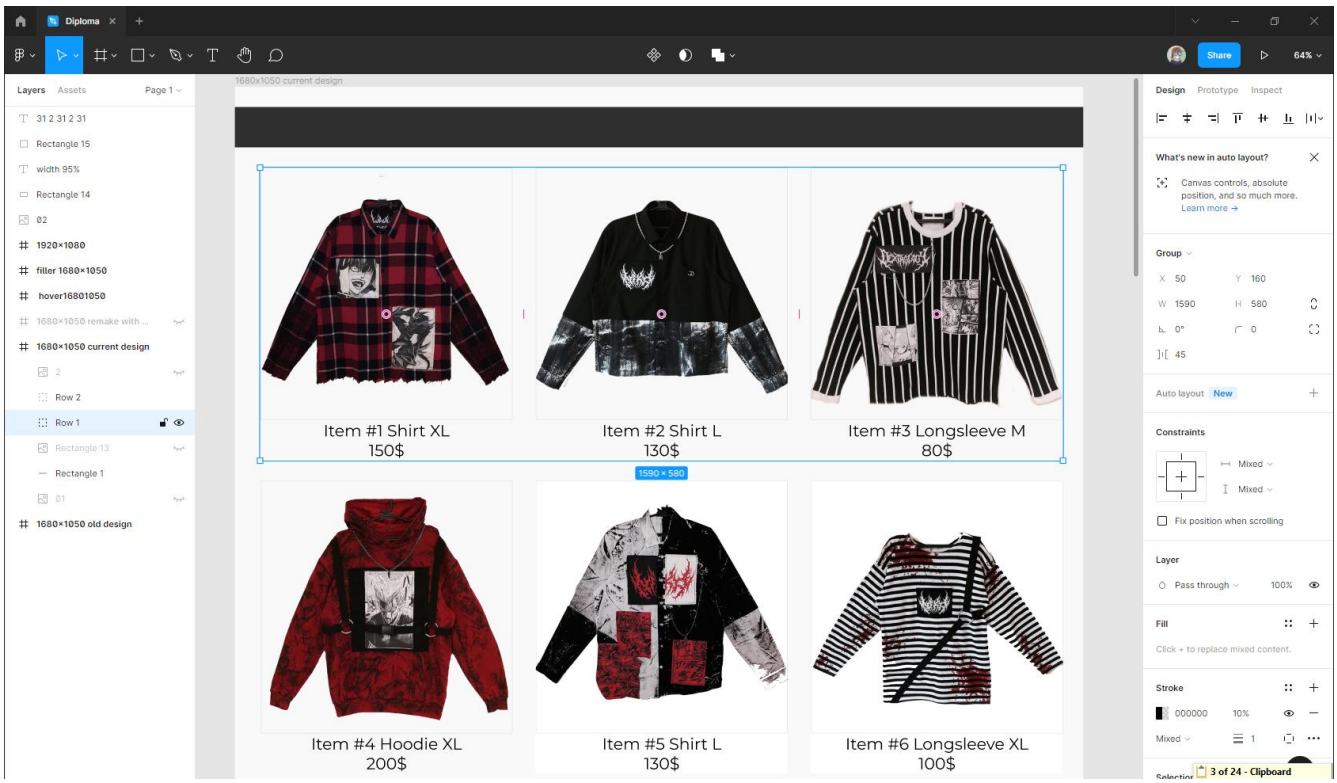

Рис. 2.13. Вікно програми Figma

2.6.3. Виклик та завантаження програми

Для виклику та завантаження необхідно виконати наступні дії:

– необхідно орендувати або придбати серверне обладнання та доменне ім’я для звернення до сайту;

– встановити на серверне обладнання серверне забезпечення Apache та PHP-інтерпретатор;

– налаштувати серверне програмне забезпечення;

– за допомогою FTP клієнта приєднатися до сервера та перемістити у папку src усі скомпільовані файли програми;

– підключити доменне ім’я та сертифікати до серверу;

– увімкнути сервер.

Але розробка та демонстрація проекту велася на локальному сервері, тому далі будемо розглядати саме на цьому прикладі.

2.6.4. Опис інтерфейсу користувача

При відкритті сайту відвідувач потрапляє на головну сторінку, на якій розміщено усі наявні товари та меню сайту (рис. 2.14.).

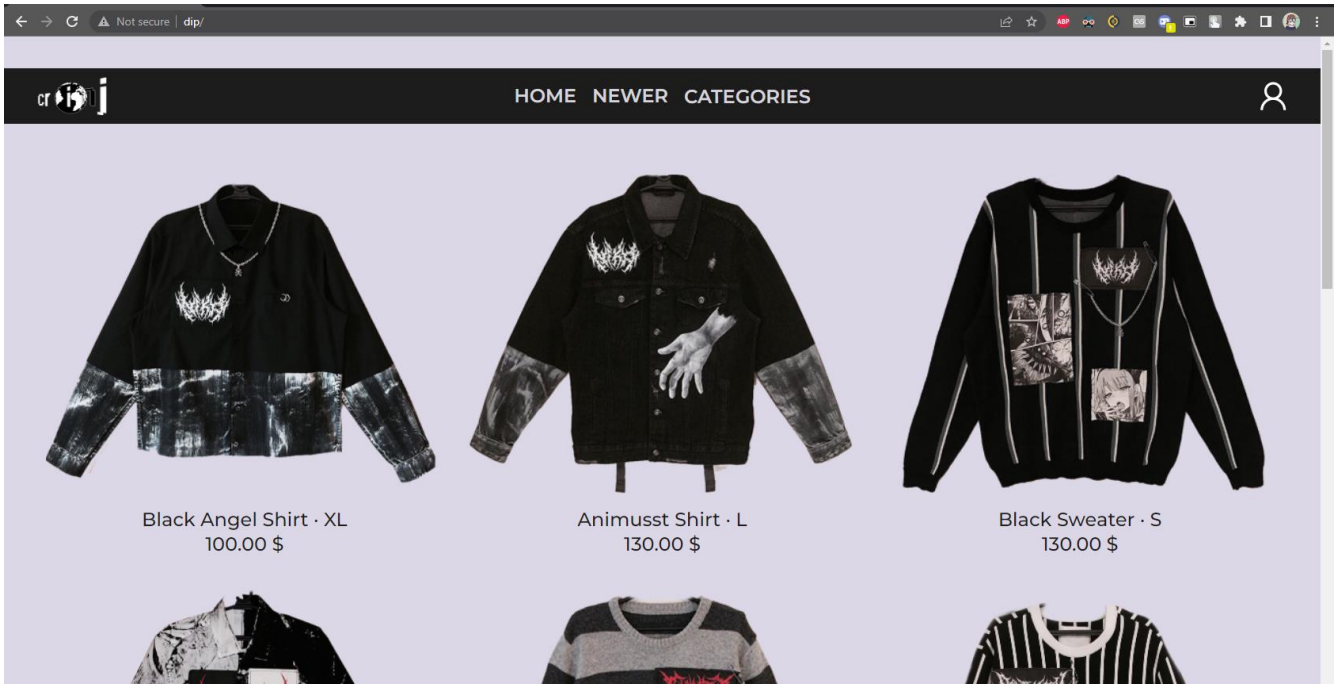

Рис. 2.14. Сторінка з товарами сайту

Відвідувач має змогу продовжити користування сайтом як гість, або виконати реєстрація чи авторизацію на сайті. Якщо відвідувач не авторизований у системі, то у правій частині меню буде тільки одна кнопка, що веде на сторінку з формами реєстрації та авторизації (рис. 2.15.).

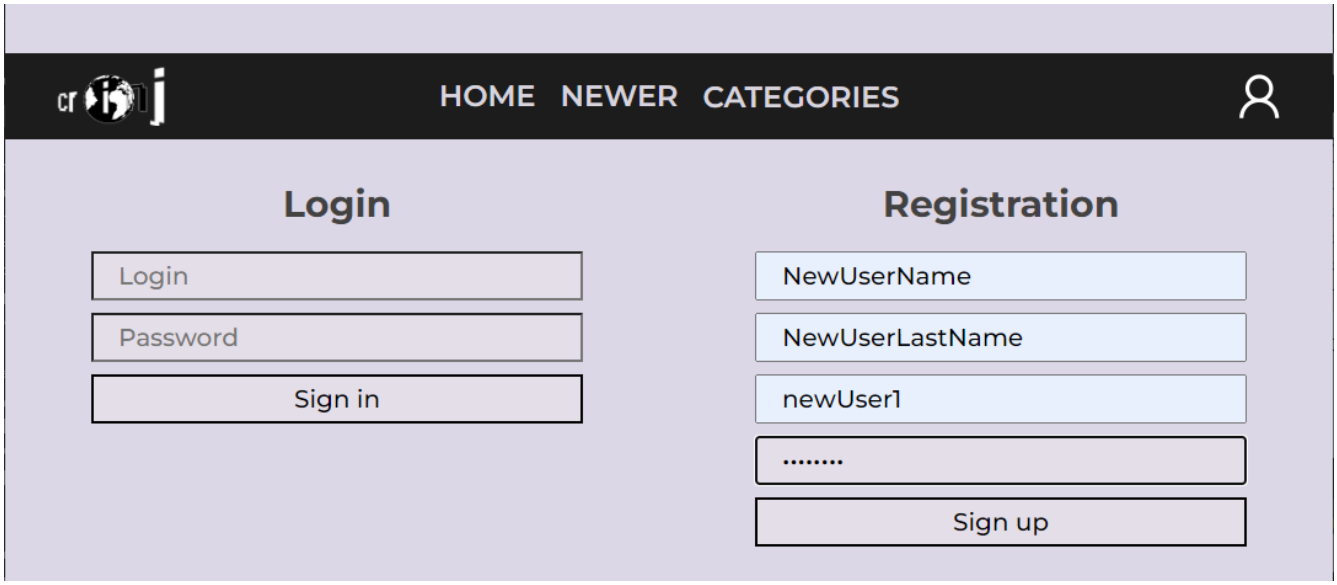

Рис. 2.15. Сторінка з формами реєстрація та авторизації

У випадку вдалої реєстрації відвідувач отримає відповідне повідомлення, та дані будуть автоматично підставлені у форму авторизації (рис. 2.16.). У випадку неправильного введення користувачем даних йому буде відображена помилка, значення у якому полі потрібно виправити.

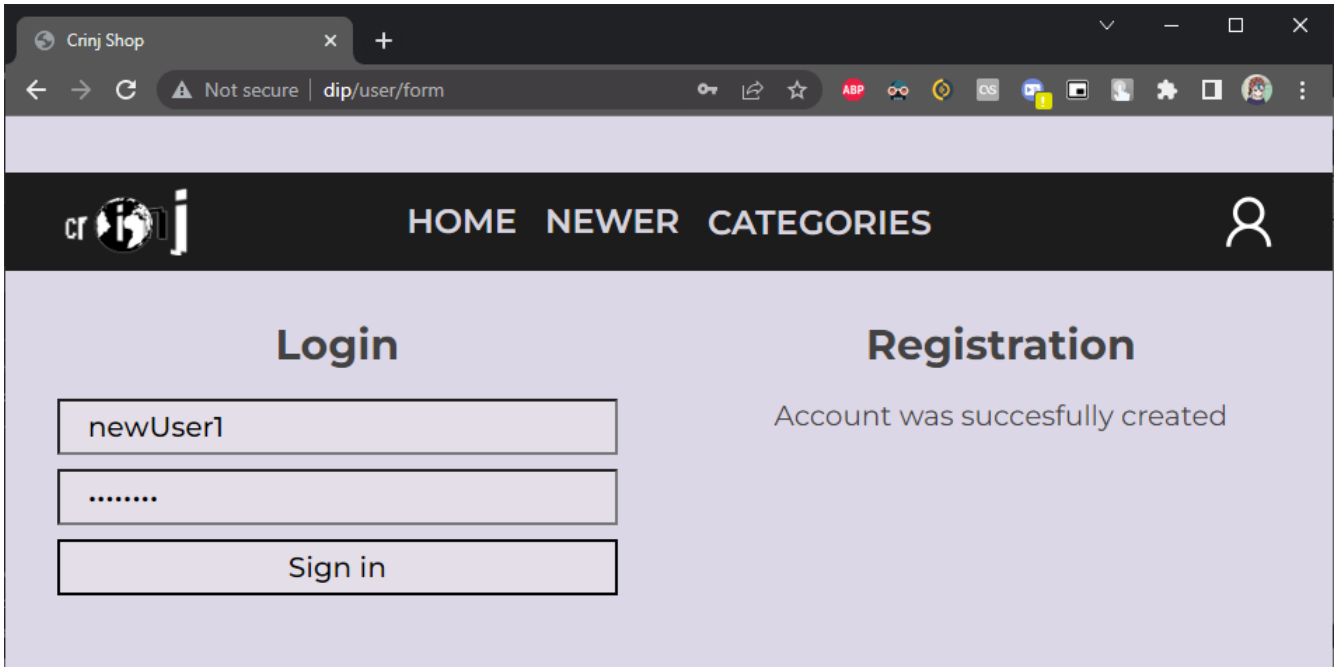

Рис. 2.16. Реєстрацію нового користувача здійснено

При введенні неправильного або неіснуючого логіну або паролю на екрані з’являться помилки у якому саме полі помилки. У даному випадку пароль не відповідає мінімальній необхідній довжині, тому з’являється відповідне сповіщення про невідповідність паролю вимогам, також було неможливо виконати авторизацію, про що свідчить наступне повідомлення (рис. 2.17.).

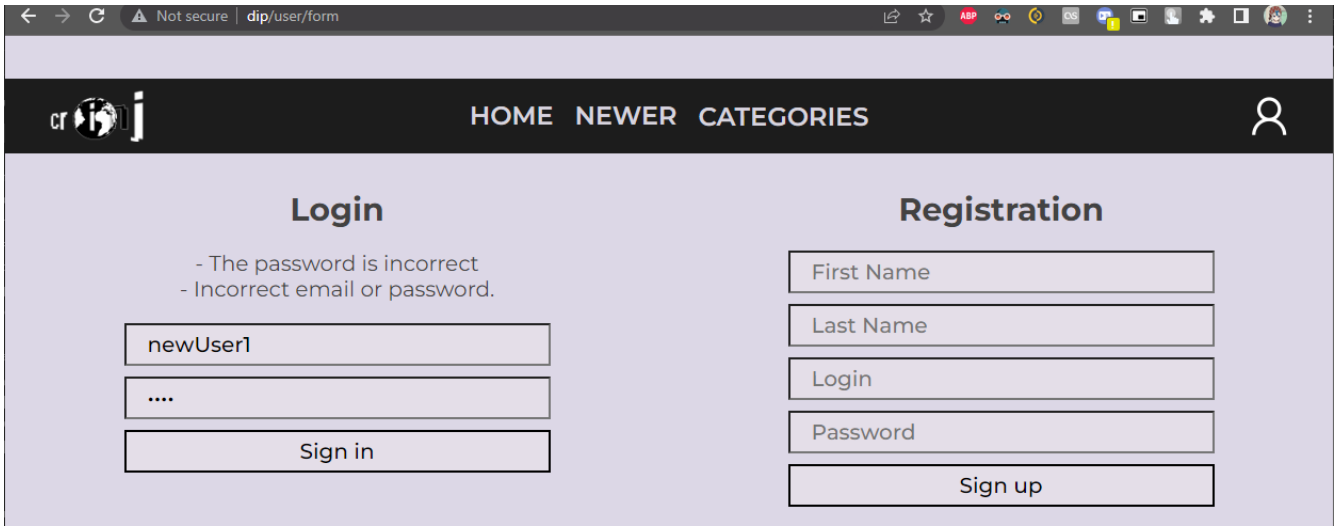

Рис. 2.17. Відображення помилок при авторизації

Після вдалої авторизації права частина меню змінюється таким чином, що тепер там є кнопки, що ведуть на сторінку користувача з замовленнями (особистий кабінет) та кнопка для виконання виходу користувача із системи, що припиняє сесію (logout) (рис. 2.18.).

Рис. 2.18. Зміна кнопок меню користувача

Тепер можна переглянути інші сторінки, такі як товари, згруповані за категоріями. Кожен блок містить у собі вітрину товарів кожної існуючої категорії (до трьох товарів з категорії) (рис. 2.19.) та посилання на повний список товарів з обраної категорії (рис. 2.20.).

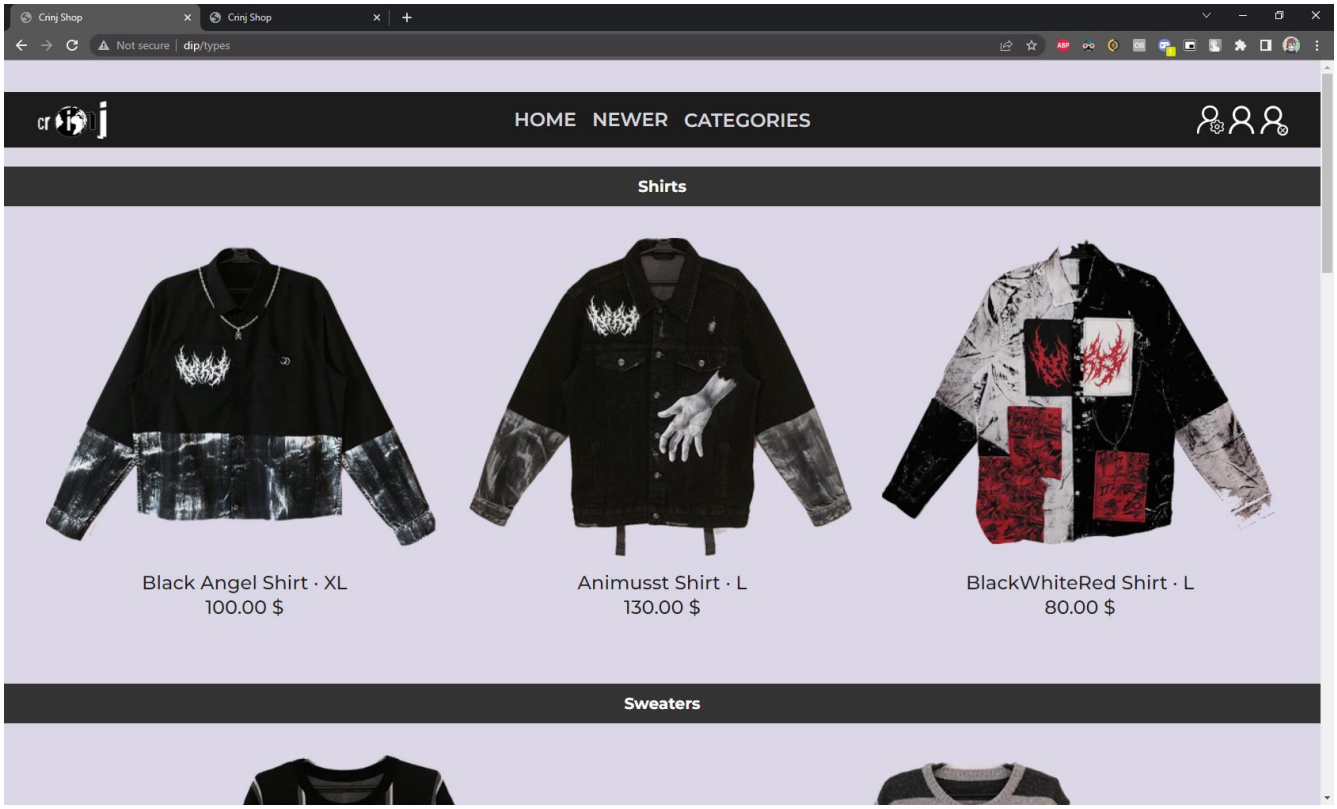

Рис. 2.19. Групування вітрин товарів за категоріями

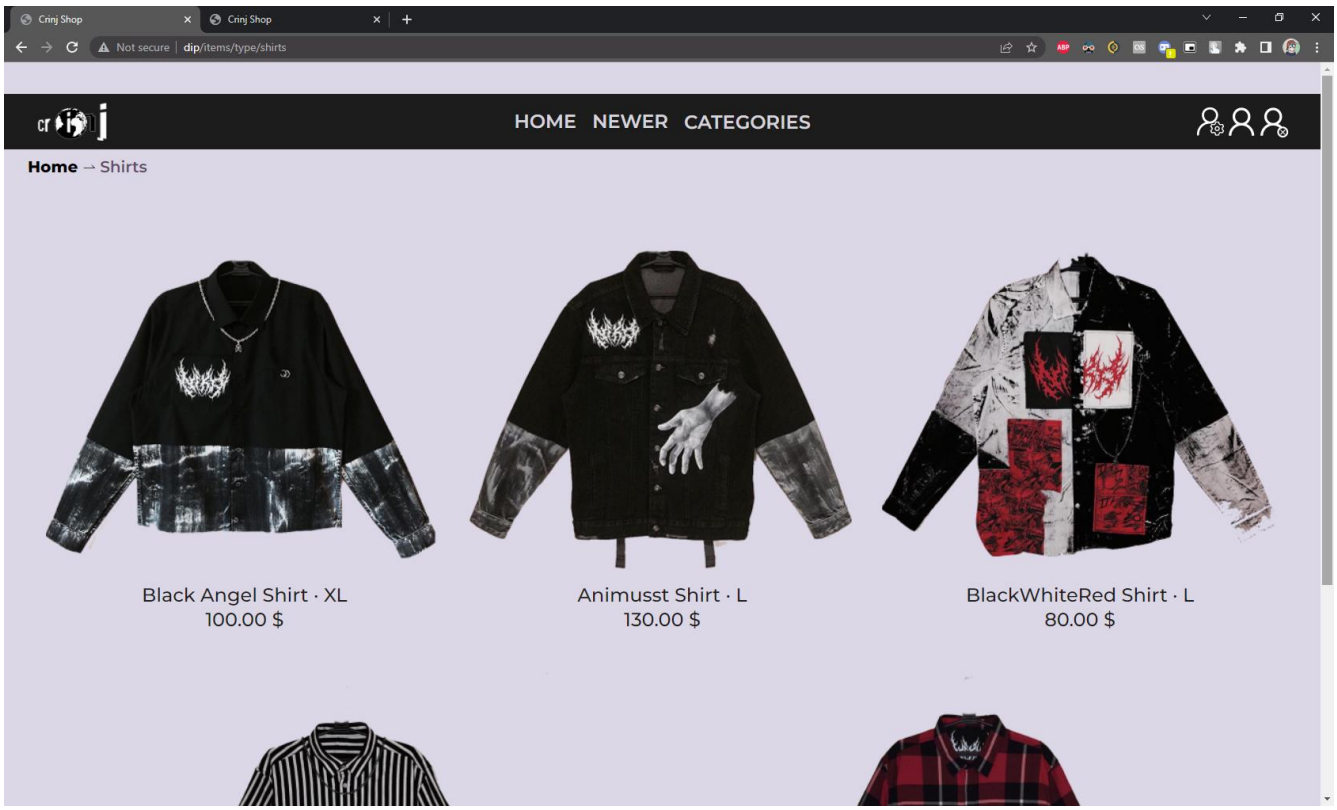

Рис. 2.20. Перегляд товарів за обраною категорією

Для перегляду інформації та зображень обраного товару потрібно натиснути на елемент з відображенням товару, після чого буде згенерована сторінка для перегляду повної інформації про товар (рис. 2.21.).

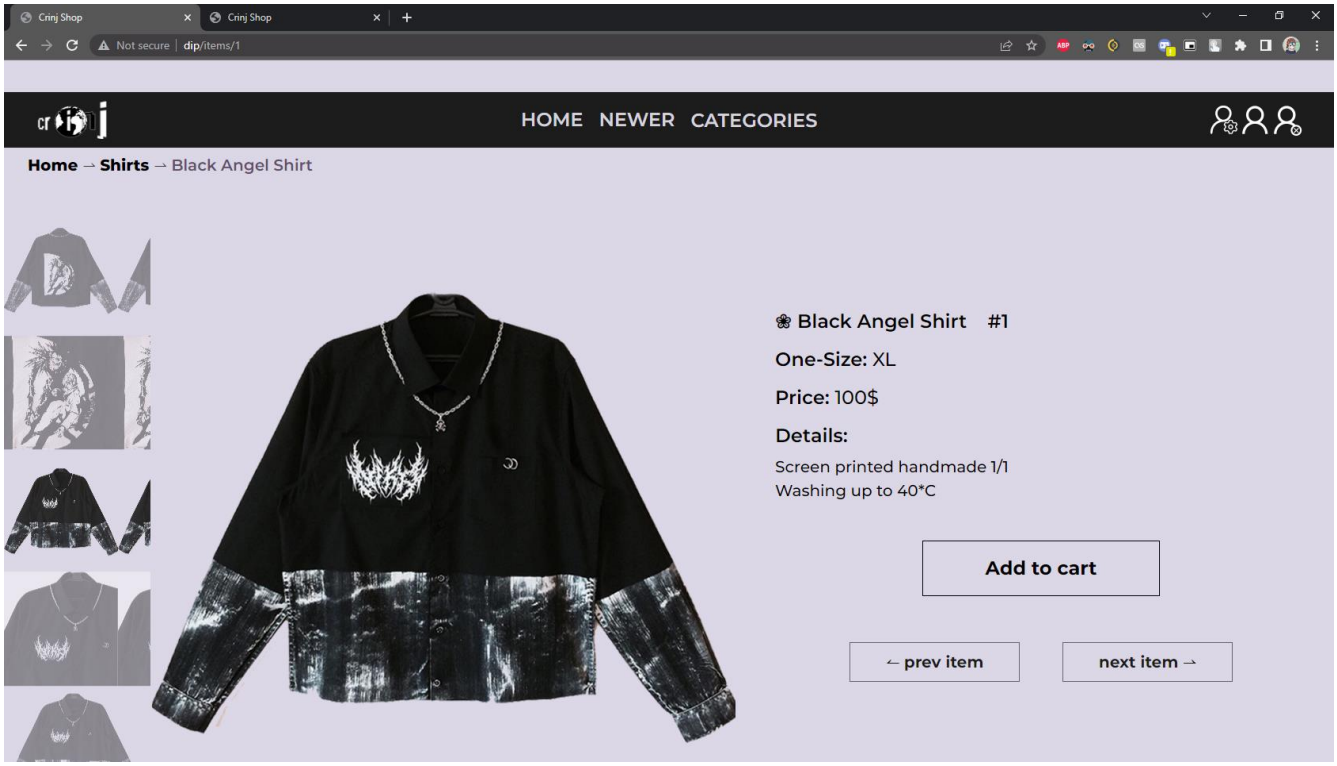

Рис. 2.21. Сторінка перегляду обраного товару

Сторінки для перегляду товарів та категорій також містять посилання, що дозволяють перейти до перегляду усіх товарів, що відносяться до категорії обраного товару або повернутися на головну сторінку (2.22.).

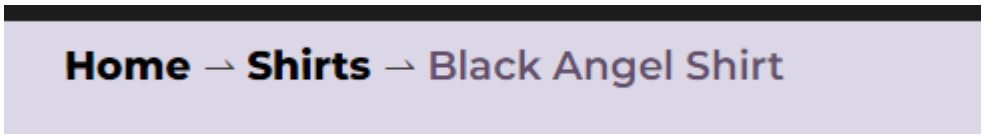

Рис. 2.22. Навігація за товарами та категоріями

При натисканні кнопки «Categories» у навігаційній панелі відвідувач перейде на сторінку з вітринами товарів за категоріями, а якщо просто наведеться на дану кнопку, то з’явиться список наявних категорій за якими можна переглянути товари, при цьому обрана на даний момент категорія буде позначена у списку та відображена у панелі навігації (рис. 2.23.).

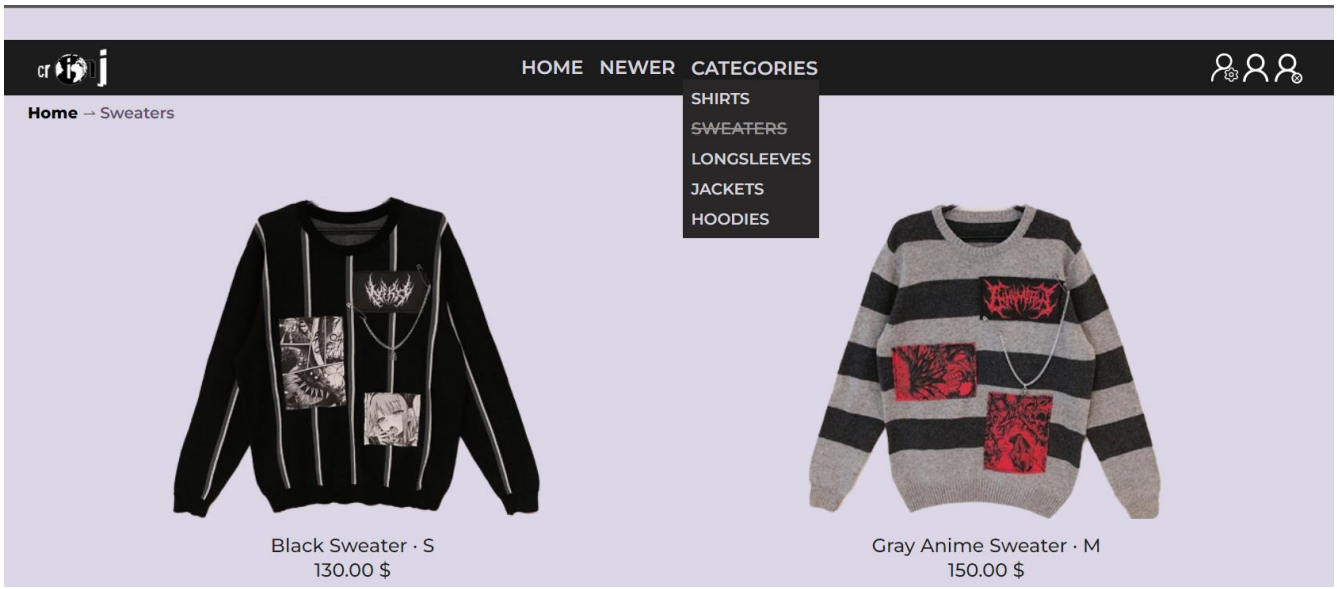

Рис. 2.23. Відображення списку категорій

Для того, щоб додати товар до кошика потрібно відкрити сторінку бажаного товару та натиснути кнопку «Add to cart» (рис. 2.24.). Після цього товар буде додано до кошика, що можна буде побачити у блоці користувача у меню (рис. 2.25.), а текст кнопки додавання товару зміниться (рис. 2.26.).

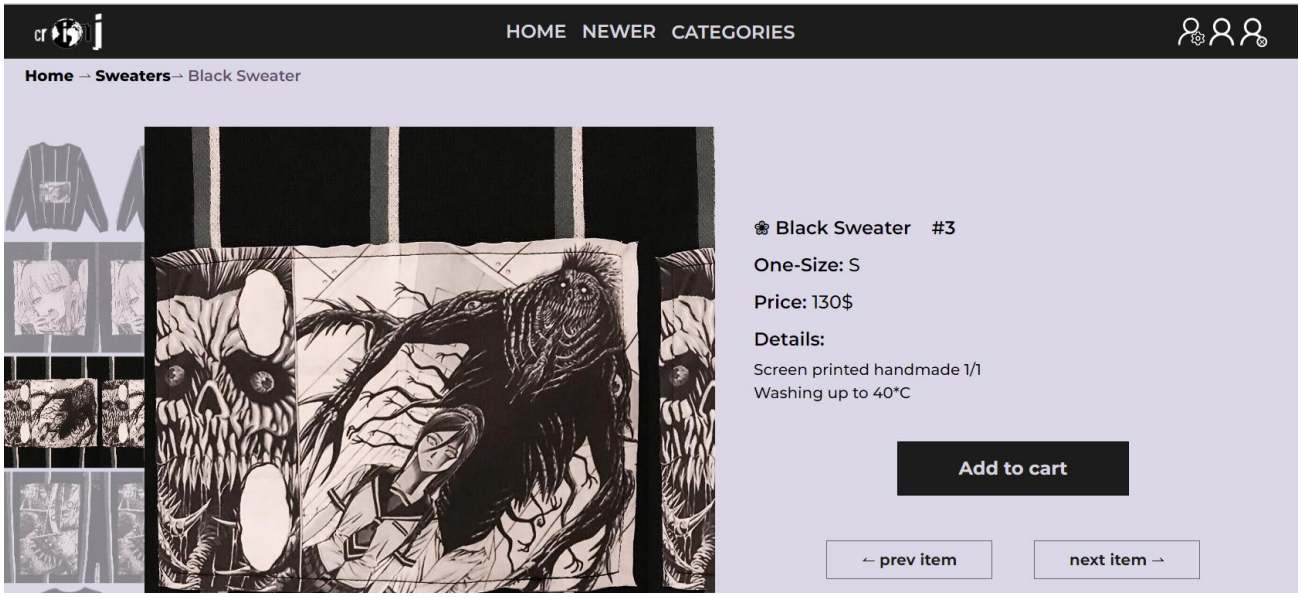

Рис. 2.24. Натискання кнопки «Add to cart»

Рис. 2.25. Поява кошика після додавання товару

Рис. 2.26. Зміна тексту кнопки після додання товару

Для перегляду кошика потрібно натиснути по іконці кошика, після чого вас переведе на сторінку перегляду кошика, де можна переглянути, видалити товари що вам не потрібні та перейти на сторінку оформлення замовлення (рис. 2.27.).

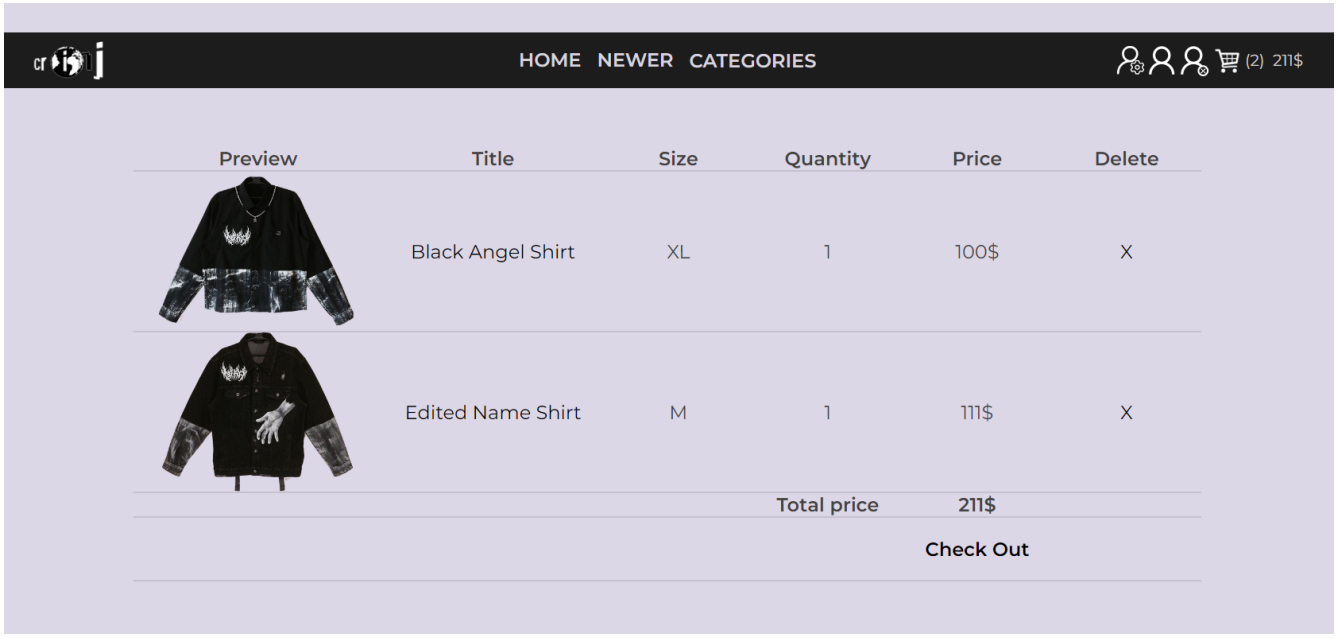

Рис. 2.27. Перегляд кошика

Сторінка оформлення замовлення містить форму для заповнення, на основі якої буде сформовано замовлення (рис. 2.28.).

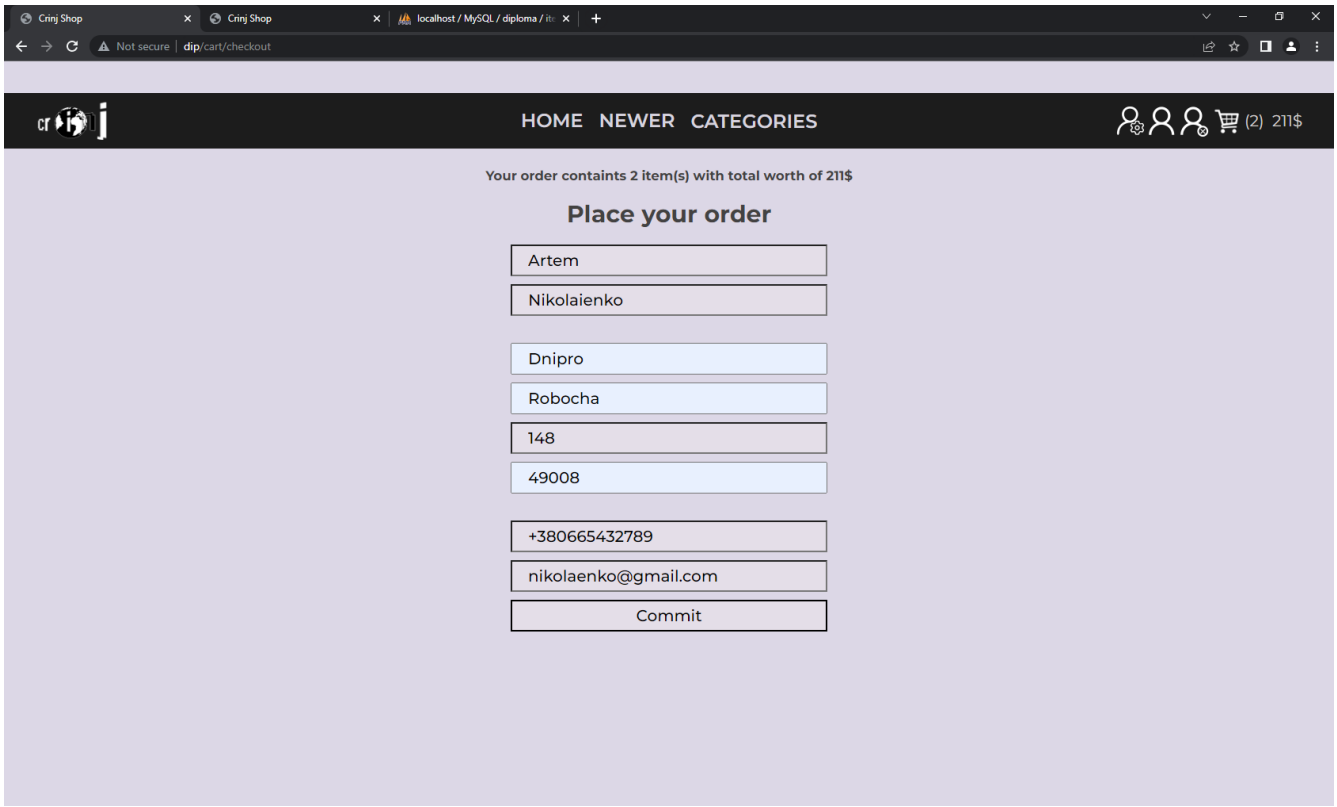

Рис. 2.28. Форма оформлення замовлення

Після оформлення замовлення воно з’явиться у адміністратора, а користувач потрапить у особистий кабінет, де може переглянути історію своїх замовлень, що включає товари, вартість, дату та статус замовлення (2.29).

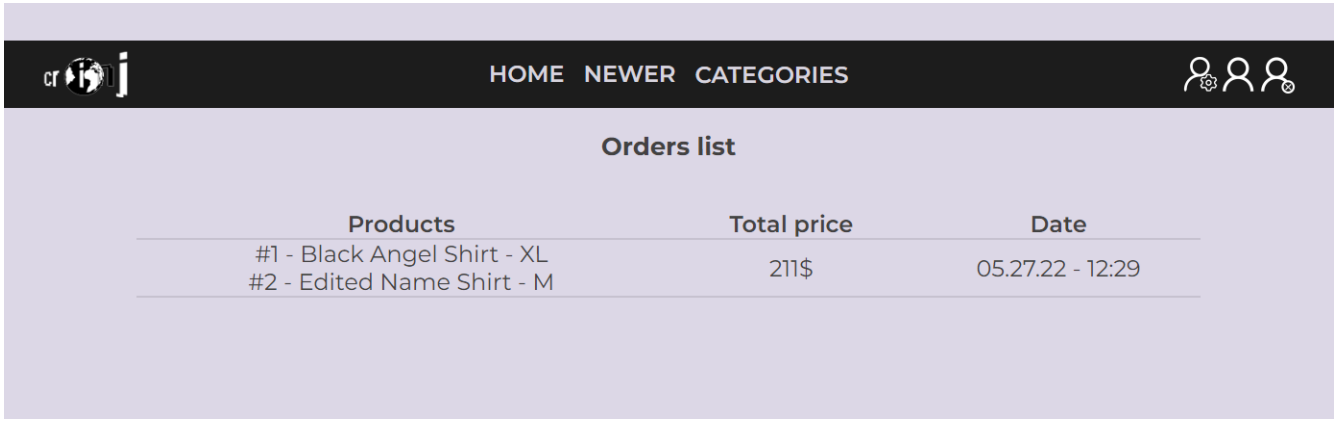

Рис. 2.29. Історія замовлень користувача у особистому кабінеті

Після того як товар було продано його статус у магазині зміниться, прибравши ціну товару та змогу додати його до кошика (рис. 2.30).

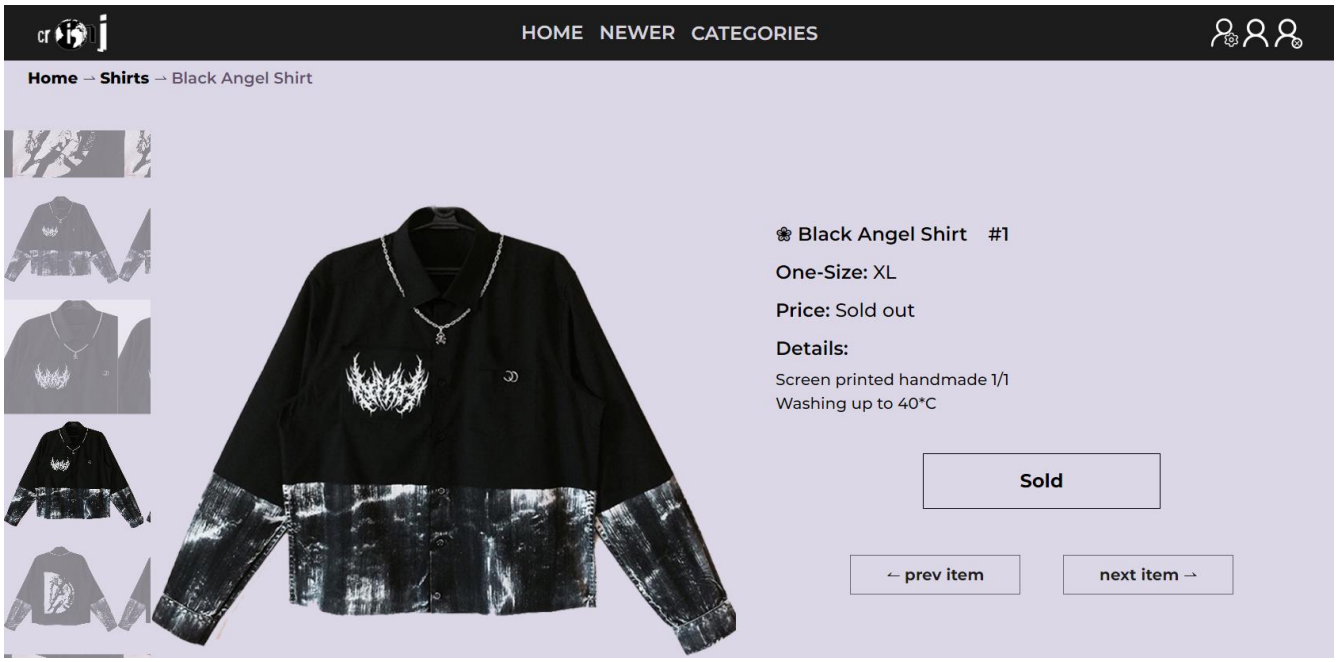

Рис. 2.30. Відображення товару, що було продано

Система передбачає наявність користувачів з ролями «user» та «admin». Користувач з роллю «admin» має доступ до розділу сайту у якому можна працювати з товарами сайту, наприклад додавати, редагувати або видаляти їх (рис. 2.31.).

Рис. 2.31. Доступ до панелі адміністратора для користувача з роллю «admin»

Панель адміністратора дозволяє працювати з такими сутностями системи як Products (товари магазину), Categories (категорії товарів) та Orders (замовлення клієнтів) (рис. 2.32.).

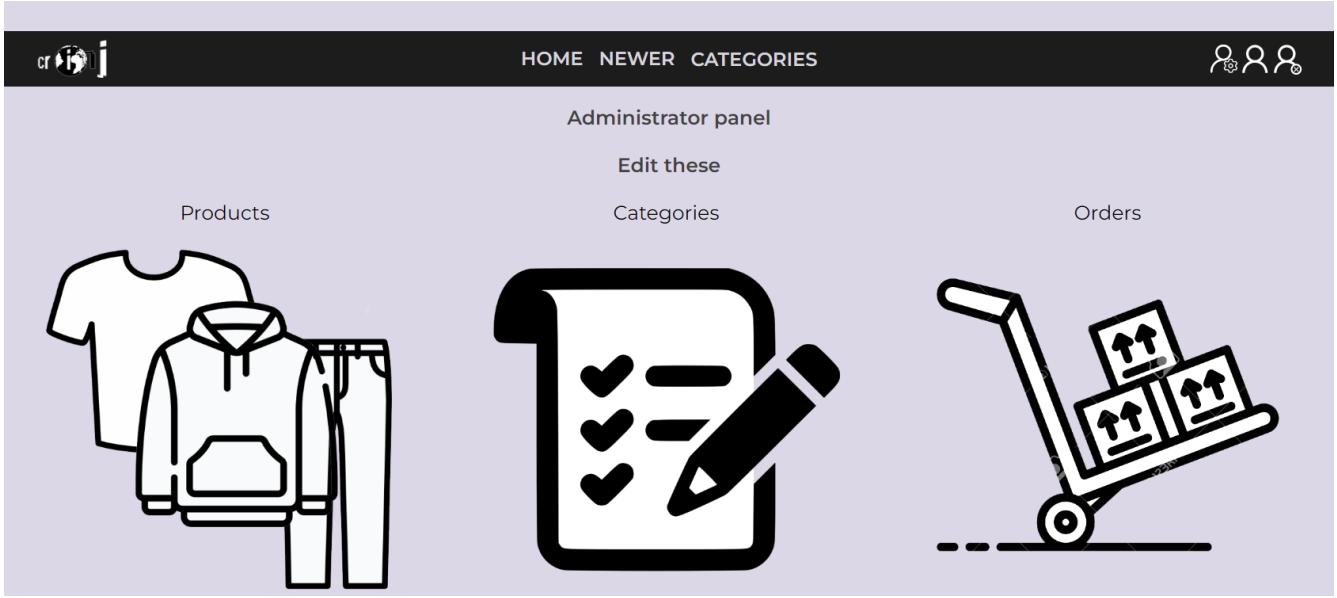

Рис. 2.32. Панель адміністратора

У вкладці «Products» панелі адміністратора здійснюється уся взаємодія з товарами магазину, а саме перегляд (рис. 2.33.), додавання нових, редагування та видалення існуючих товарів та їх зображень.

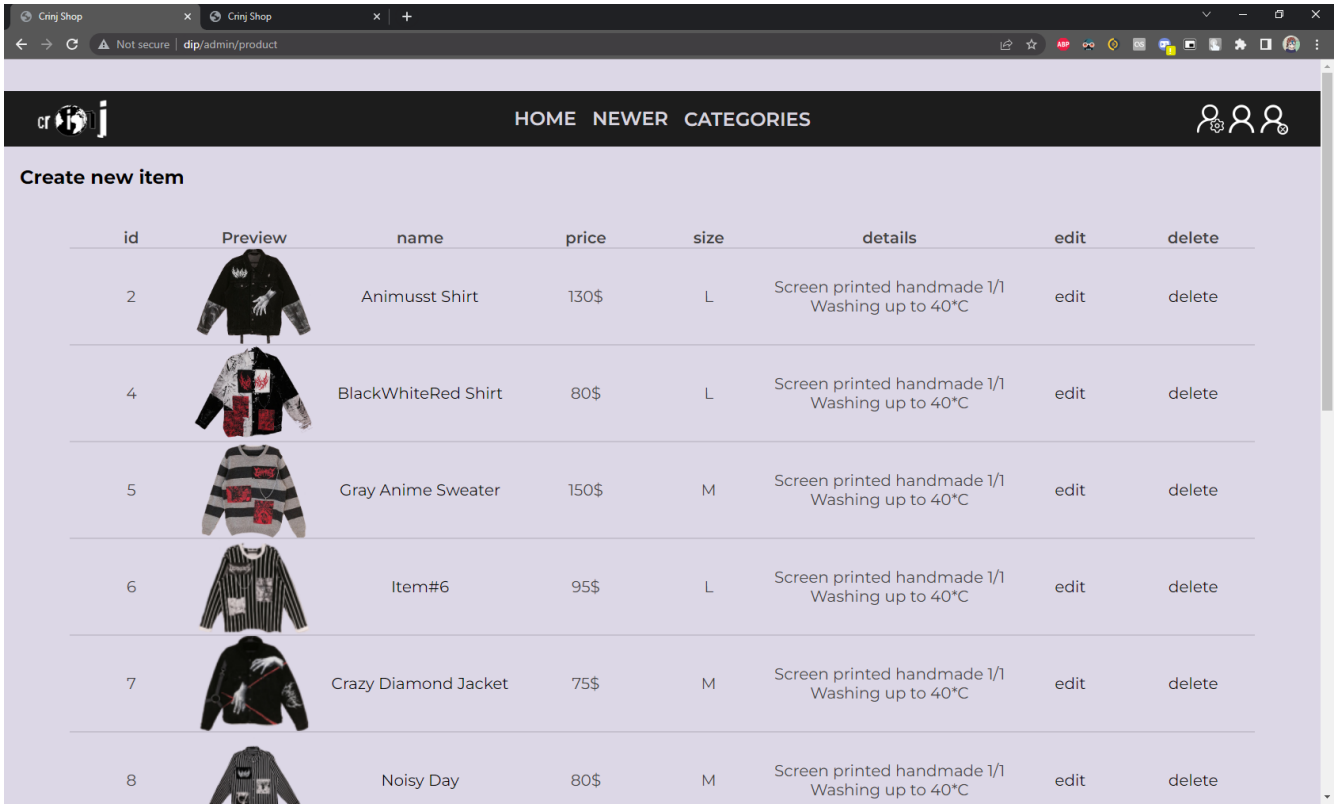

Рис. 2.33. Перегляд товарів через панель адміністратора

Звідси адміністратор має змогу відредагувати існуючі товари, натиснувши кнопку «edit» у рядку товару. При цьому буд2е відкрито нову сторінку з формою у якій знаходяться дані про товар що можна змінити (рис. 2.34., 2.35.).

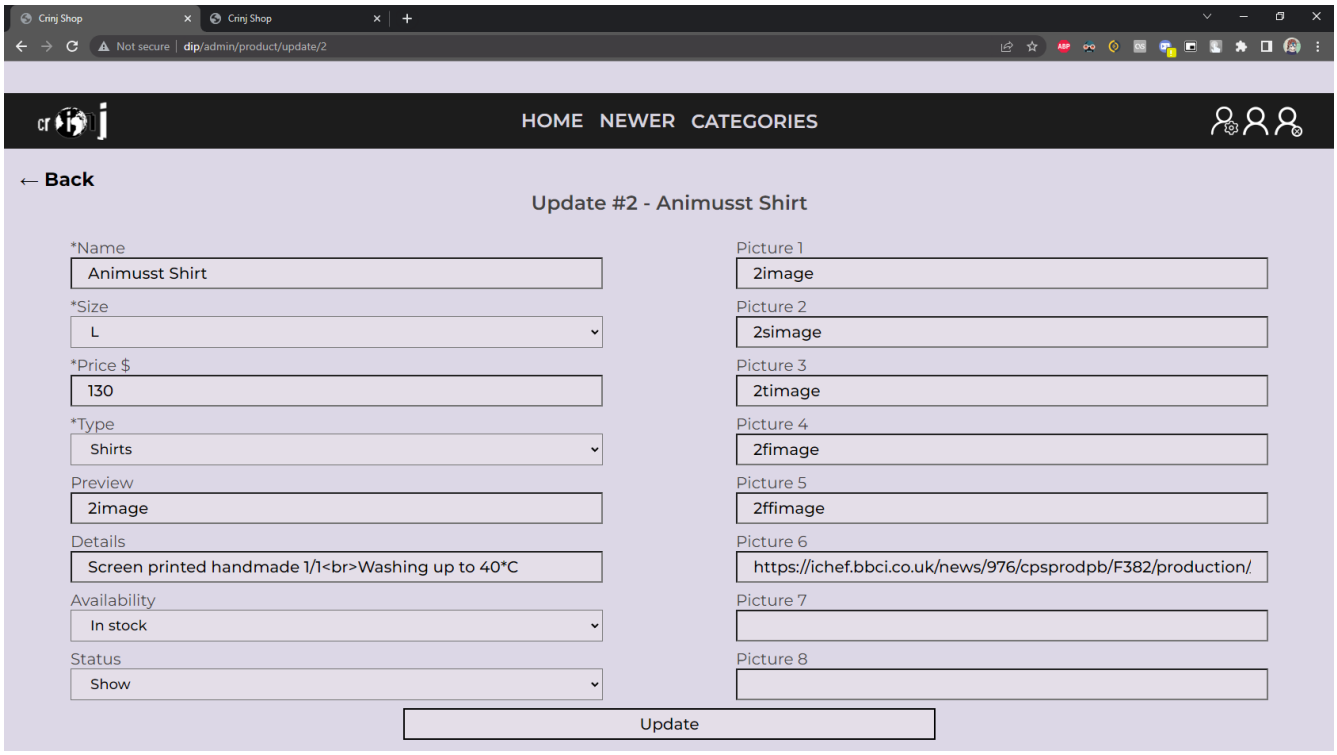

Рис. 2.34. Сторінка редагування інформації про обраний товар

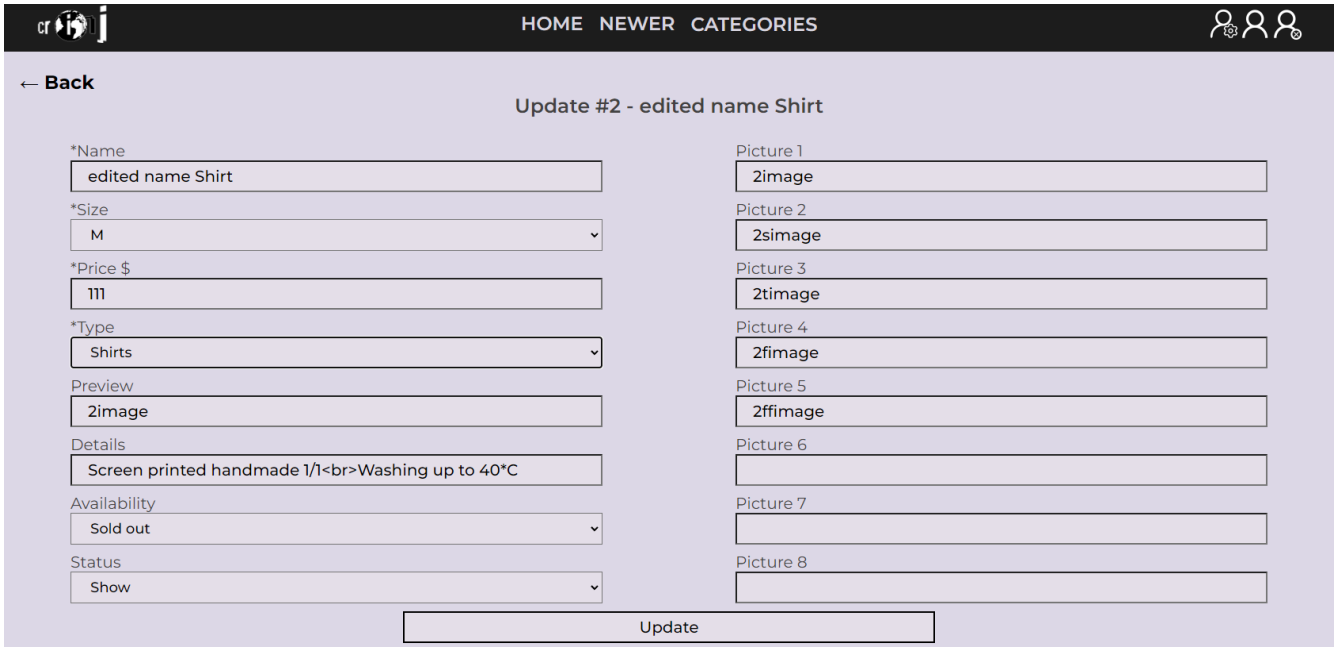

Рис. 2.35. Редагування інформації про товар

Для додавання нового товару потрібно натиснути кнопку «Create new item», після чого буде відкрито сторінку з формою для додання нового товару (рис. 2.36.).

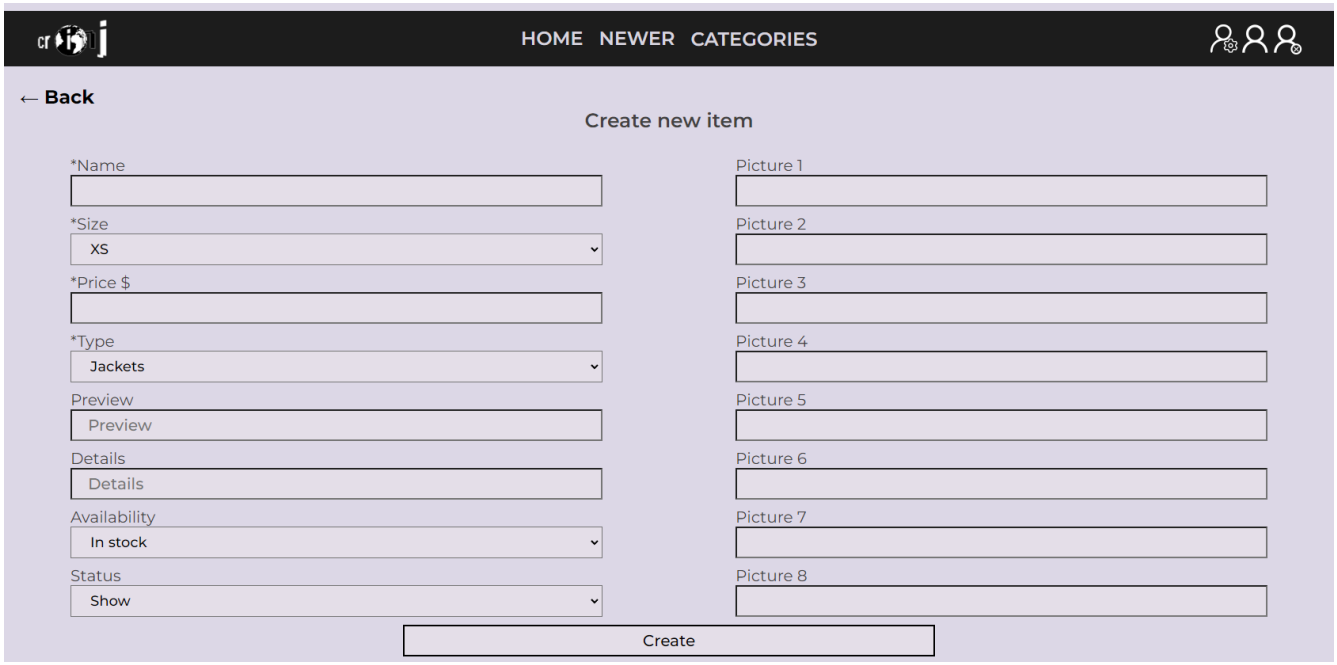

Рис. 2.36. Форма для додання нового товару

Сторінка «Category» дозволяє працювати з категоріями товарів на сайті (додавати нові, редагувати та видаляти існуючі), які потім використовуються при доданні нових товарів. Для цього потрібно з панелі адміністратора перейти на сторінку «Categories», де можна здійснювати роботу з даною сутністю (рис. 2.37.).

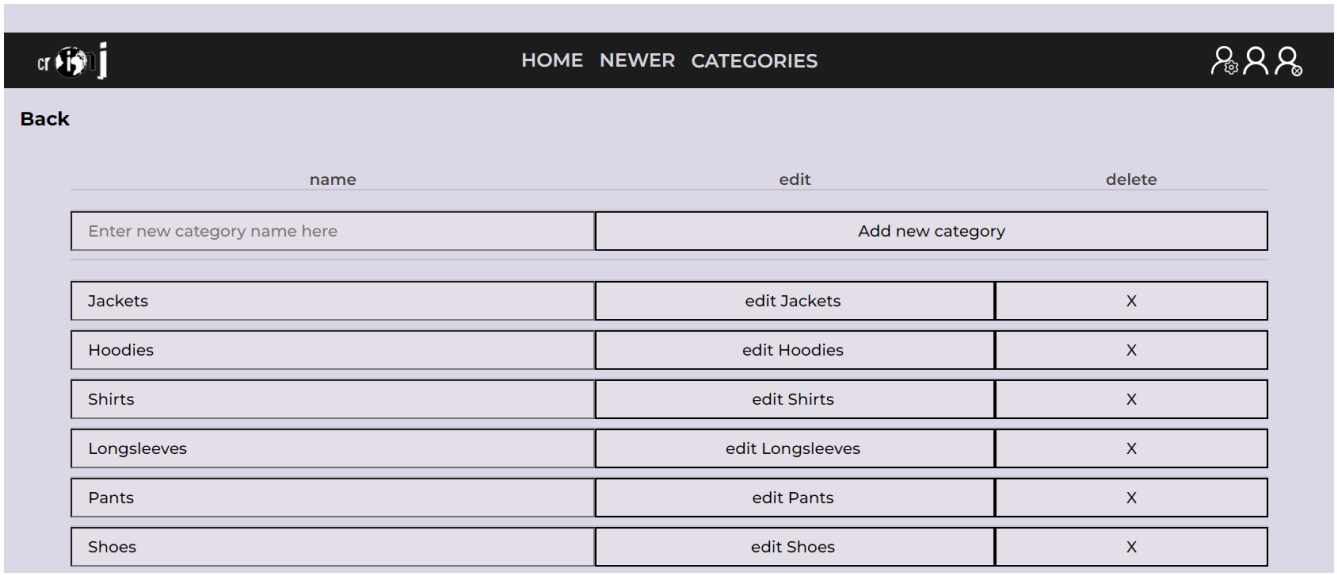

Рис. 2.37. Сторінка «Categories»

Для додавання нової категорії потрібно ввести назву бажаної категорії у поле «Enter new category name here» у першому рядку, після чого натиснути

кнопку «Add new category» (рис. 2.38.). Після цього нова категорія з’явиться у відповідній таблиці бази даних (рис. 2.39.) та на даній сторінці (рис. 2.40.).

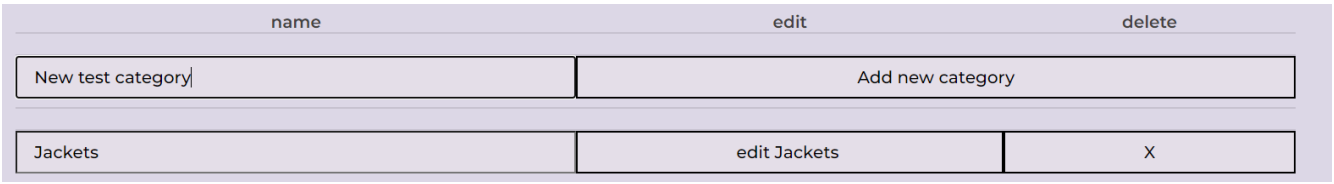

Рис. 2.38. Додавання нової категорії

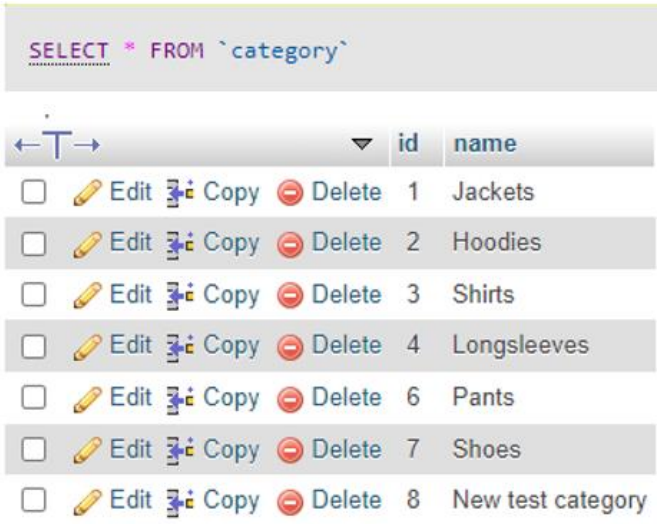

Рис. 2.39. Перегляд списку категорій у базі даних

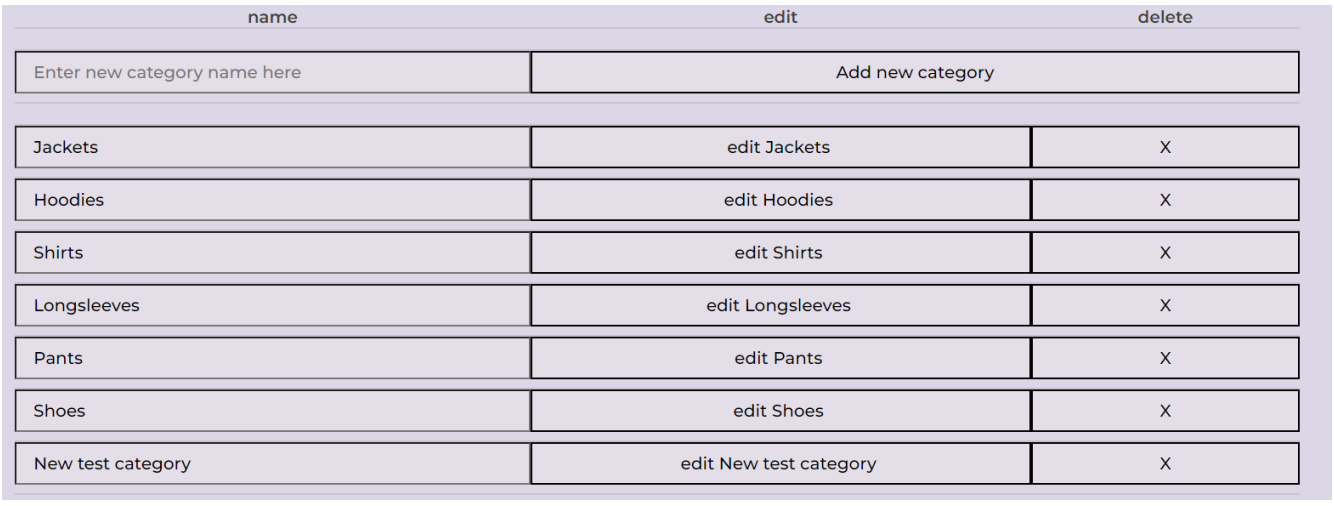

Рис. 2.40. Перегляд списку категорій через панель адміністратора

Для редагування існуючої категорій потрібно у її полі ввести нову назву та натиснути кнопку «Edit» у рядку категорії (рис. 2.41).

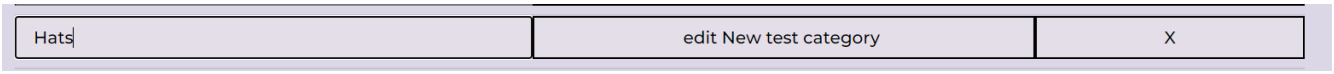

Рис. 2.41. Редагування існуючої категорії

Перегляд списку категорій при додаванні нових товарів (рис. 2.42.).

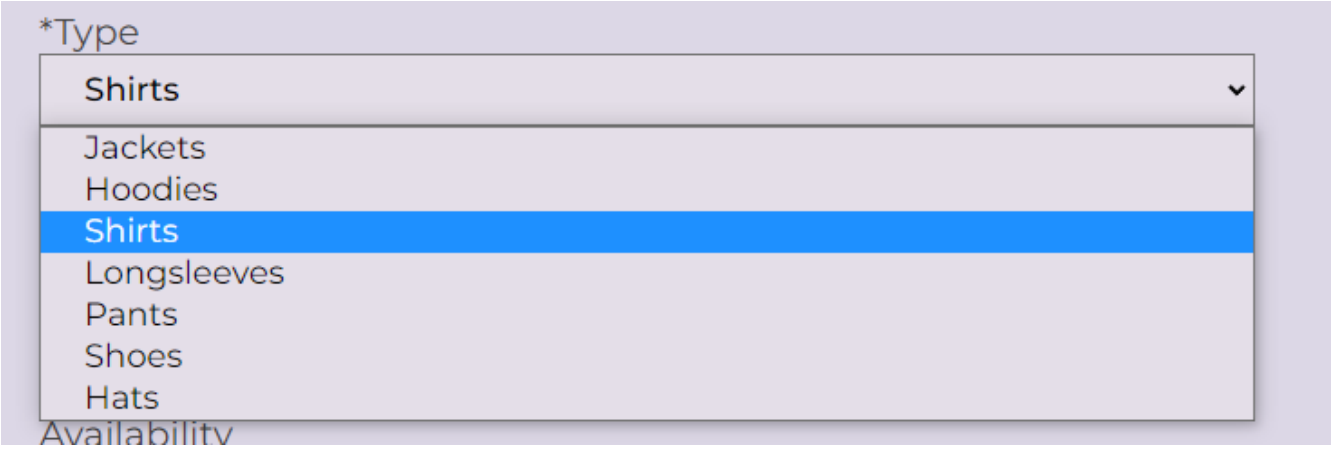

Рис. 2.42. Перегляд списку категорій при додаванні нових товарів

Через панель адміністратора можна переглянути перелік відкритих замовлень (рис. 2.43.), редагувати їх статус (1 - замовлення було прийнято, 2 – замовлення у обробці та 3 – замовлення закрито) (рис. 2.44.).

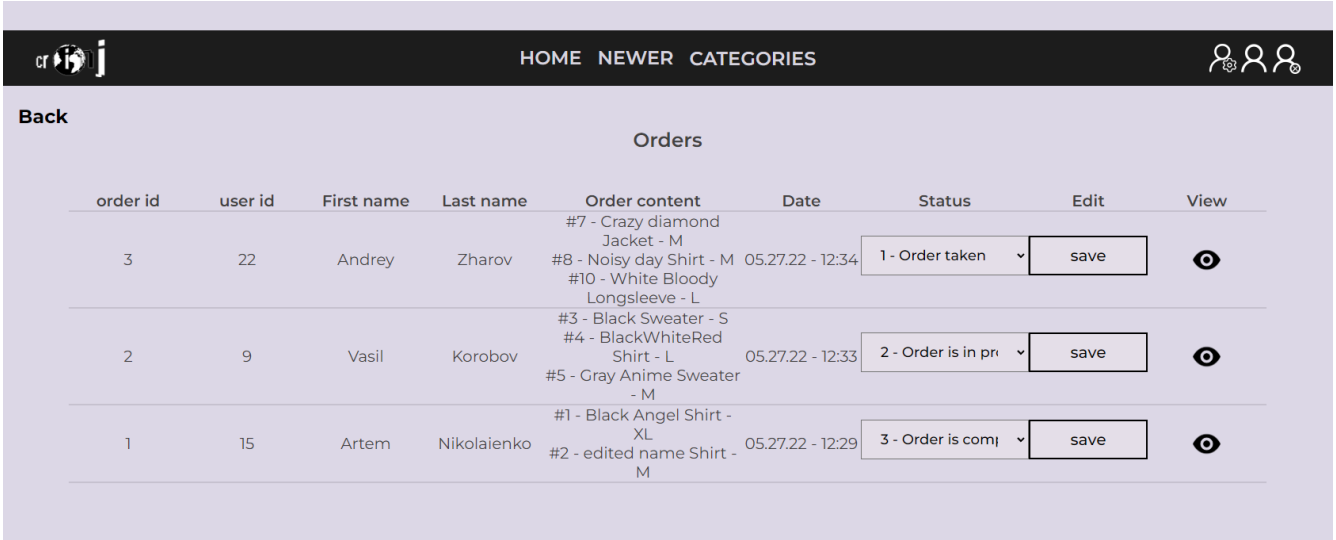

Рис. 2.43. Перегляд списку замовлень

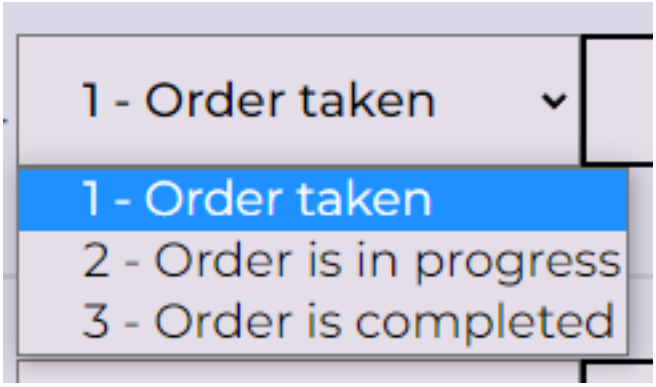

Рис. 2.44. Редагування статусу замовлення

Також з розділу Замовлень панелі адміністратора можна переглянути повну інформацію про замовлення, що містить окрім загальної інфомрації

наведної на (рис. 2.43.) контактну інформацію клієнта, таку як місто, вулиця, будівля, поштовий індекс, номер телефону та адресу електронної пошти. При відображення сторінки з повною інформацією по конкретному замовленню потрібно натиснути у рядку замовлення, що вас цікавить іконку з оком (рис. 2.45.), після чого ви потрапите на відповідну сторінку (рис. 2.46.). Звідси ж, аналогічно сторінці з загальною інформацією можна змінити статус замовлення.

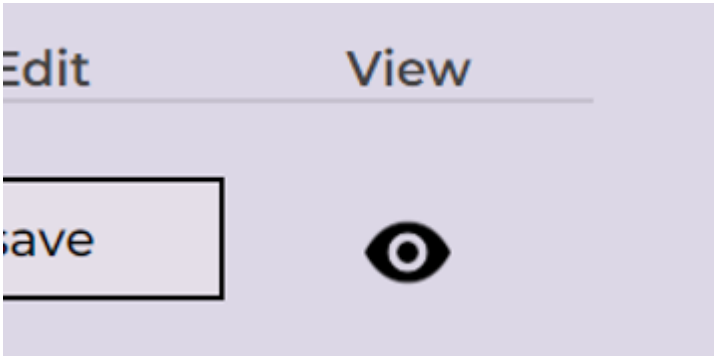

Рис. 2.45. Кнопка для перегляду детальної інформації по замовленню

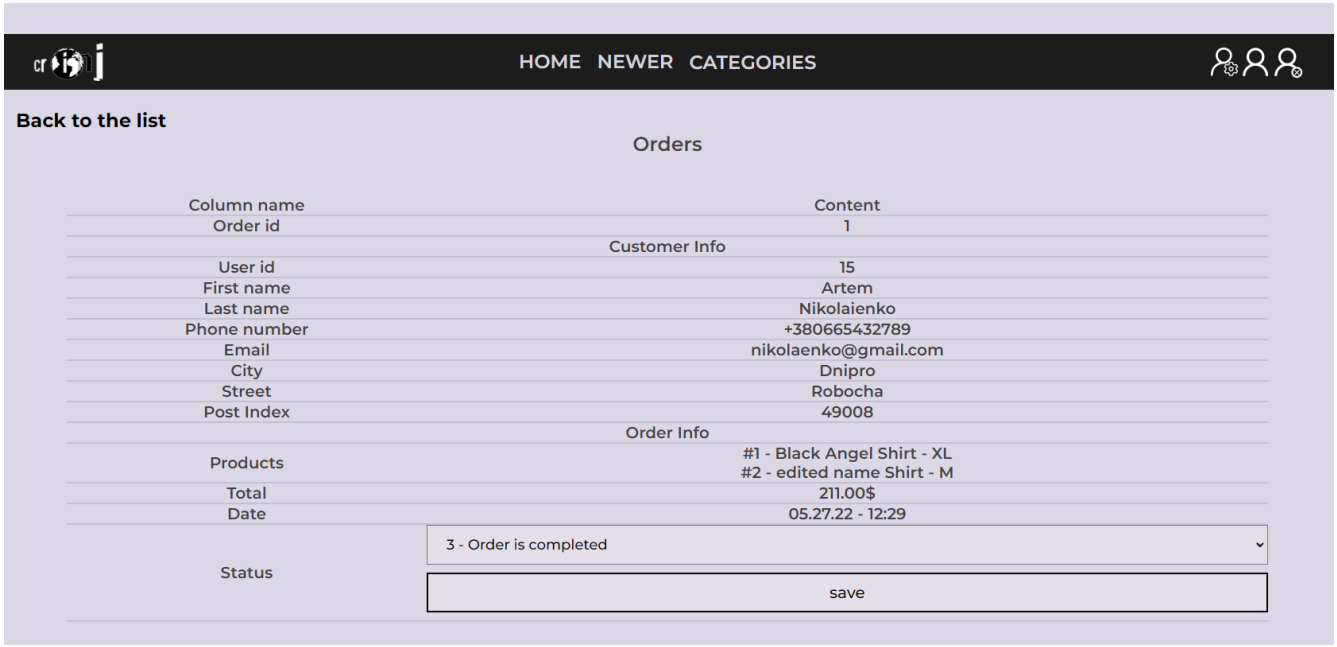

Рис. 2.46. Перегляд детальної інформації про замовлення

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

– передбачуване число операторів – 3000;

– коефіцієнт корекції програми в ході її розробки – 0,05;

– коефіцієнт складності програми – 1,4;

– годинна заробітна плата програміста– 145,4 грн/год;

Середня годинна зарплата Junior PHP Software Developer в Україні була вирахувати виходячи з даних «Української спільноти програмістів (DOU)» [22]. Станом на 2022 рік зарплата Junior PHP розробника простягається від 600$ до 1200$. Вирахувавши середню заробітну плату програміста маємо плату 900$ у місяць. При курсі валют НБУ на початок червня 2022 року один американський долар дорівнює 29,5 грн, тому середня зарплата в гривнях дорівнює 26 596 грн. При стандартному графіку (176 годин/місяць) зарплата за годину буде становити близько 145,4 грн.

– коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,3;

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

– вартість машино-години ЕОМ – 22,7 грн/год;

– кількість розробників – 1.

Оскільки для цього проекту потрібна значна потужність ПК для роботи серверу та підтримки зберігання значної кількість даних на сервері баз даних, гарним рішенням буде оренда ноутбуку на час розробки додатку. Вартість оренди ноутбука місяць становить 4000 грн. При стандартному графіку (176 годин/місяць) вартість машино-години ЕОМ за годину роботи буде становити

22,7 грн. В цю вартість входить ремонт за гарантією та базовий комплект гарнітури (клавіатура та миша) [23].

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

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \tag{3.1}$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-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 програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q - передбачуване число операторів (3000);

C - коефіцієнт складності програми (1,4);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси за формулою (3.2) умовне число операторів в програмі:

$$Q = 3000 * 1,4 * (1+0,05) = 4410 \text{ людино-годин}, \tag{3.3}$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k} \tag{3.4}$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 2 до 3 років він складає 1.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (В = 1,3). З урахуванням коефіцієнта кваліфікації k = 1, отримуємо витрати праці на вивчення опису завдання за формулою (3.4):

$$t_u = (4410 * 1{,}3) / (85 * 1) = 67{,}4 \text{ людино-годин.} \tag{3.5}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k} \tag{3.6}$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.6), отримаємо:

$$t_a = 4410 / (25 \cdot 1) = 176{,}4 \text{ людино-годин,} \tag{3.7}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25) \cdot k} \tag{3.8}$$

За формулою (3.8) та значенням параметру Q, обчислюємо витрати праці на програмування по готовій блок-схемі:

$$t_n = 4410 / (25 \cdot 1) = 176{,}4 \text{ людино-годин,} \tag{3.9}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k} \tag{3.10}$$

Витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання за формулою (3.10):

$$t_{отл} = \ 4410 / (5 \cdot 1) = 882 \text{ людино-годин}, \tag{3.11}$$

– за умови комплексного налагодження завдання:

$$t^{k}_{отл} = 1{,}5 \cdot t_{отл} \tag{3.12}$$

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження завдання за формулою (3.12):

$$t^{k}_{отл} = 1{,}5 \cdot 882 = 1323 \text{ людино-годин}, \tag{3.13}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \tag{3.14}$$

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \tag{3.15}$$

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

$$t_{до} = 0{,}75 \cdot t_{др} \tag{3.16}$$

За формулами (3.15), (3.16) та (3.14) обчислюємо витрати праці на документацію:

$$t_{др} = 4410 / (20 \cdot 1) = 220{,}5 \text{ людино-годин}, \quad (3.17)$$

$$t_{до} = 0{,}75 \cdot 220{,}5 = 165{,}4 \text{ людино-годин}, \quad (3.18)$$

$$t_{д} = 220{,}5 + 165{,}4 = 385{,}9 \text{ людино-годин}, \quad (3.19)$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 67{,}4 + 176{,}4 + 176{,}4 + 882 + 385{,}9 = 1738{,}1 \text{ людино-годин}. \quad (3.20)$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ} \quad (3.21)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР} \quad (3.22)$$

де: t - загальна трудомісткість, людино-годин;

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

З урахуванням того, що середня годинна зарплата програміста становить 145,4 грн / год, за формулою (3.22) отримуємо:

$$З_{ЗП} = 1738{,}1 \cdot 145{,}4 = 252\,722 \text{ грн}, \quad (3.23)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч} \quad (3.24)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (22,7 грн/год).

Підставивши в формулу (3.24) відповідні значення, обчислюємо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 882 \cdot 22,7 = 20\ 021 \text{ грн}, \quad (3.25)$$

Звідси, за формулою (3.21), витрати на створення програмного продукту:

$$К_{ПО} = 252\ 722 + 20\ 021 = 272\ 743 \text{ грн}, \quad (2.26)$$

Очікуваний період створення програмного застосунку:

$$T = \frac{t}{B_k \cdot F_p}, \quad (2.27)$$

де B_k - число виконавців (дорівнює 1);

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

Звідси, за формулою (2.27), витрати на створення програмного продукту:

$$T = 1738 / (1 \cdot 176) \approx 9,88 \text{ місяців} \quad (2.28)$$

Висновки: програмне забезпечення розроблено для забезпечення доступу користувачів до каталогу товарів магазину продажу одягу, ефективної взаємодії між потенційними споживачами та компанією та зручної адміністрації сайту, такої як робота з товарами та замовленнями. Вартість даного програмного забезпечення становить 272 743 грн і не вимагає додаткових витрат як при

розробці програми. Очікуваний час розробки становить 1738 годин, тобто 9,88 місяці. Цей термін пов'язаний зі значним числом операторів та незначним коефіцієнтом кваліфікації програміста, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було поставлено завдання розробити web-орієнтовану систему для відображення та адміністрування інтернет-магазину одягу.

Актуальність теми пов’язана із розширенням ринку електронної комерції та обоюдною потребою, як потенційних покупців, так і продавців, у інструменті для зручної та ефективної взаємодії та здійснення торгових відносин. Проект, виконаний у рамках кваліфікаційної роботи має зручний та мінімалістичний інтуїтивно зрозумілий інтерфейс, що дозволяє середньому користувачу, що має базові навички володіння комп’ютером та інтернетом ефективно взаємодіяти з ресурсами та можливостями розробленого веб-додатку.

Практичне призначення даної системи полягає в забезпеченні відвідувачам сайту простого і комфортного доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту, що зробить процес покупки швидшим і зручнішим, і підвищить ефективність роботи інтернет-магазину. Розроблена система може бути використана будь ким для роботи у подібній сфері діяльності та зі схожими вимогами. Створена система дозволяє оптимізувати та спростити дії по веденню операцій онлайн-продажів: скоротити час на оформлення продажу; підвищити ефективність діяльності компанії за рахунок автоматизації його діяльності в сфері електронної комерції та можливості моніторингу стану системи і аналізу наявних даних (за рахунок наявності адміністративної частини проекту). Розроблене програмне забезпечення є універсальним інструментом, що дозволяє відвідувачам переглянути каталог товарів магазину, провести реєстрацію та авторизацію в системі та оформити замовлення, а також ефективно здійснювати керування контентом інтернет-магазину, таким як усі можливі операції (додавання нових, перегляд, редагування та видалення існуючих) над товарами та замовленнями користувачів.

Під час виконання даного проекту були виконані наступні задачі:

– вивчено предметну галузь розв'язуваної задачі;

– створено алгоритм для реалізації поставленого завдання;

– розроблено макет та графічний дизайн додатку;

– створено базу даних для зберігання виробничої інформації;

– створено web-орієнтований додаток, що реалізує усі необхідні функції інтернет-магазину.

Розроблене програмне забезпечення дозволяє:

– зручне адміністрування товарів та замовлень інтернет-магазину;

– формування web-сторінок на основі шаблону з використанням контенту, отриманого від серверу;

– формування web-сторінок на основі шаблону з використанням контенту, отриманого з бази даних;

– реєстрацію та авторизацію користувачів;

– оформлення замовлень у даному магазині.

Бізнес-логіка web-додатку реалізована на мові PHP, користувальницька частина написана мовами розмітки HTML5, CSS3 та сценарною мовою JavaScript, у якості бази даних використано СКБД MySQL, проект написано відповідно до архітектурного шаблону проектування та розробки MVC.

Також у кваліфікаційній роботі було визначено трудомісткість розробленої системи, на базі середньої зарплати розробника проведений підрахунок вартості роботи по створенню програми, який складає 272 743 грн та розраховано час на створення інтернет-магазину – 1738 людино-годин, тобто 9,88 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Історія та можливості Інтернету. Історія створення Інтернету / Режим доступу: https://sites.google.com/site/istoriatamozlivostiinternetu/home/istoria-stvorenna -internetu. дата звернення: 12.09.2022.
2. TutorialsMate – 20+ Uses of Internet in Daily Life: Education, Business, Banking and more / Режим доступу: https://www.tutorialsmate.com/2022/01/uses-of-internet.html. дата звернення: 12.09.2022.
3. Kernighan B.A. Understanding the Digital World: What You Need to Know about Computers, the Internet, Privacy, and Security: The basics of how computer hardware, software, and systems work and the risks they create for our privacy and security. 2017. Vol. 1, No 1. P. 41–47.
4. Technopedia – Explaining Internet, Web Browsers, Protocols and Web Development / Режим доступу: https://www.techopedia.com/definition/288/web-browser. дата звернення: 14.09.2022.
5. TechTarget - How do web servers work? Explanation with examples / Режим доступу: https://www.techtarget.com/whatis/definition/Web-server. дата звернення: 14.09.2022.
6. Oracle Cloud Infrastructure (OCI) – Database Defined / Режим доступу: https://www.oracle.com/database/what-is-database/. дата звернення: 16.09.2022.
7. Техопедія – Визначення архітектури клієнт-сервер та масштабування систем / Режим доступу: https://uk.theastrologypage.com/client-server-architecture/. дата звернення: 17.09.2022.
8. TutorialsPoint – Web Application Frameworks - MVC Framework: architectural pattern / Режим доступу: https://www.tutorialspoint.com/ mvc_framework /mvc_framework_introduction.htm. дата звернення: 20.09.2022.
9. Freeman A.J. Pro ASP.NET Core MVC: cloud optimized and mobile-ready applications for the .NET platform. 2017. Vol. 6, No 1. P. 88–112.12.
10. TutorialsTeacher – Controllers in ASP.NET MVC / Режим доступу: https://www.tutorialsteacher.com/mvc/mvc-controller. дата звернення: 20.09.2022.

11. Kinsta - Is PHP Dead? Look at the statistics / Режим доступу: https://kinsta.com/blog/is-php-dead/. дата звернення: 21.09.2022.

12. Bierer D.A. PHP 7 Programming Cookbook Kindle Edition: new features of version 7.x, best practices for server-side programming, and MVC frameworks. 2016. Vol. 1, No 3. P. 25–42.

13. Biere D.C., Patala J.A., Helmich M.Y., Pietroluongo N. O. PHP 7 Simplified: PHP 7 coding from the grass ROOTs level and create your very own real-world applications. 2017. Vol. 1, No 2. P. 41-55.

14. WWW Organization - The global structure of an HTML document: introduction to the structure of an HTML document / Режим доступу: https://www.w3.org/TR/html401/struct/global.html. дата звернення: 25.09.2022.

15. James O.A. HTML & CSS IS HARD. But it doesn't have to be: web development tutorial for complete beginners. 2017. Vol. 1, No 1. P. 22-31.

16. Mozilla Developer Network Web Docs - CSS / Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS. дата звернення: 26.09.2022.

17. Booth J.D. W3.CSS: using of features, containers and helper classes. 2018. Vol. 1, No 1. P. 22-31.

18. Tickoo S.P. Introducing PHP 7/MySQL: explaining and learning web development and various features. 2018. Vol. 1, No 3. P. 42-57.

19. Ullman L.S. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide: practical introductions to both PHP 7 and MySQL. 2017. P. 71-93.

20. Brackets API – Documentation and description / Режим доступу: https://brackets.io/docs/current/modules/brackets.html. дата звернення: 28.09.2022.

21. Wampserver – Apache, PHP, MySQL soul Windows / Режим доступу: https://www.wampserver.com/en/. дата звернення: 2.10.2022.

22. Середня заробітна плата Junior PHP Software Developer в Україні станом на грудень 2021 року при досвіді до трьох років / Режим доступу: https://jobs.dou.ua/salaries/?period=2021-12&position=Junior%20SE&technology =PHP&experience=0-3& education=3. дата звернення: 05.10.2022.

23. Вартість оренди одного ноутбуку на місяць [Електронний ресурс] - Режим доступу: https://notebooksbu.com/prokat_noutbuka/. дата звернення: 05.10.2022.

ДОДАТОК А

КОД ПРОГРАМИ

Модель Product.php

```
<?php
include_once ProjectBase . '/components/Db.php';

class Product {

    public static function getPicsForItem( $id ) {
        $id = intval( $id );
        if ( $id ) {
            $dataBase = DataBaseConnect::connectToDataBase();
            $picsList = array();
            $list = array();

            $queryResult = $dataBase->query( 'SELECT
       pictures.pic1,
       pictures.pic2,
       pictures.pic3,
       pictures.pic4,
       pictures.pic5,
       pictures.pic6,
       pictures.pic7,
       pictures.pic8

FROM  pictures where
               pictures.id_item =' . $id );

            $queryResult->setFetchMode( PDO::FETCH_ASSOC );
            $list = $queryResult->fetch();

            foreach ( $list as $pic ) {
                if ( strval( $pic ) ) {
                    array_push( $picsList, $pic );
                }
            }
            return $picsList;
        }
    }

    public static function getItemsByType( $category ) {
        $category = strval( $category );
        $dataBase = DataBaseConnect::connectToDataBase();
        $productsListByType = array();

        $queryResult = $dataBase->query( 'SELECT items.id,
       items.name,
       items.size,
       items.type,
       items.price,
       items.preview
```

```
FROM   items
       where items.type = "' . $category . '"' );

        $i = 0;
        while ( $row = $queryResult->fetch() ) {
            $productsListByType[$i]['id'] = $row['id'];
            $productsListByType[$i]['name'] = $row['name'];
            $productsListByType[$i]['size'] = $row['size'];
            $productsListByType[$i]['type'] = $row['type'];
            $productsListByType[$i]['price'] = $row['price'];
            $productsListByType[$i]['preview'] = $row['preview'];

            $i++;
        }
        $_SESSION['type'] = ucfirst( $category );
        //            $_SESSION['type'] = $category;

        return $productsListByType;
    }

    public static function checkAvailability( $id ) {
        $id = intval( $id );
        $dataBase = DataBaseConnect::connectToDataBase();
        $priceTag = null;
        $list = array();
        $queryResult = $dataBase->query( 'SELECT items.id,
       items.price,
       items.availability
FROM   items
       where items.id ='.$id );

        $queryResult->setFetchMode( PDO::FETCH_ASSOC );

        $list = $queryResult->fetch();

        if ( $list['availability'] == 1 ) {
            return true;
        }
        if ( $list['availability'] == 0 ) {
            return false;
        }

    }

    public static function getItemById( $id ) {
        $id = intval( $id );
        $_SESSION['idView'] = $id;

        if ( $id ) {
            $dataBase = DataBaseConnect::connectToDataBase();

            //    $productsList = array();
            $queryResult = $dataBase->query( 'SELECT items.id,
```

```
        items.name,
        items.size,
        items.type,
        items.price,
        items.details,
        items.availability,
        items.status,
        items.preview,
        pictures.pic1,
        pictures.pic2,
        pictures.pic3,
        pictures.pic4,
        pictures.pic5,
        pictures.pic6,
        pictures.pic7,
        pictures.pic8
FROM   items
INNER JOIN pictures
                ON items.id = pictures.id_item
where items.id =' . $id );
                 $queryResult->setFetchMode( PDO::FETCH_ASSOC );
                 $productItem = $queryResult->fetch();
             return $productItem;
        }
    }

    public static function getItemsCount() {
        $dataBase = DataBaseConnect::connectToDataBase();
        $queryResult  =  $dataBase->query(  'SELECT  COUNT(items.id)
FROM items' );
//         $queryResult = $dataBase->query( 'SELECT COUNT(items.id)
FROM items' );
        $queryResult->setFetchMode( PDO::FETCH_ASSOC );
        $c = $queryResult->fetch();
        $count = implode( $c );
        return $count;
    }

     public static function getPicturesCount() {
        $dataBase = DataBaseConnect::connectToDataBase();
        $queryResult = $dataBase->query( 'SELECT max(id) FROM items'
);
//          $queryResult = $dataBase->query( 'SELECT COUNT(id_item)
FROM pictures' );
        $queryResult->setFetchMode( PDO::FETCH_ASSOC );
        $c = $queryResult->fetch();
        $count = implode( $c );
        $count = intval($count);

         $count = $count; //+1

        return $count;
    }
```

```
    public static function getAllItems() {
        $dataBase = DataBaseConnect::connectToDataBase();
        $productsList = array();
        $queryResult = $dataBase->query( 'SELECT items.id,
       items.name,
       items.size,
       items.type,
       items.price,
       items.preview
FROM   items where status = 1 ORDER BY availability DESC' );
        $i = 0;
        while ( $row = $queryResult->fetch() ) {
            $productsList[$i]['id'] = $row['id'];
            $productsList[$i]['name'] = $row['name'];
            $productsList[$i]['size'] = $row['size'];
            $productsList[$i]['type'] = $row['type'];
            $productsList[$i]['price'] = $row['price'];
            $productsList[$i]['preview'] = $row['preview'];
            $i++;
        }
        return $productsList;
    }

    public static function getNewItems() {

        $dataBase = DataBaseConnect::connectToDataBase();
        $productsList = array();
        $queryResult = $dataBase->query( 'SELECT items.id,
       items.name,
       items.size,
       items.type,
       items.price,
       items.preview
FROM   items
       ORDER BY items.id DESC' );

        $i = 0;
        while ( $row = $queryResult->fetch() ) {
            $productsList[$i]['id'] = $row['id'];
            $productsList[$i]['name'] = $row['name'];
            $productsList[$i]['size'] = $row['size'];
            $productsList[$i]['type'] = $row['type'];
            $productsList[$i]['price'] = $row['price'];
            $productsList[$i]['preview'] = $row['preview'];

            $i++;
        }
        return $productsList;
    }

    public static function getCategoriesList() {
        $dataBase = DataBaseConnect::connectToDataBase();
```

```
        $categoriesList = array();

        $queryResult = $dataBase->query( 'SELECT name FROM category'
);

        $i = 0;
        while ( $row = $queryResult->fetch() ) {

            $categoriesList[$i]['name'] = $row['name'];

            $i++;
        }
        return $categoriesList;
    }

    //get list of all catagories
    public static function getAllTypes() {
        $dataBase = DataBaseConnect::connectToDataBase();
        $typesList = array();

        $queryResult = $dataBase->query( 'SELECT DISTINCT items.type
FROM items' );

        $i = 0;
        while ( $row = $queryResult->fetch() ) {

            $typesList[$i]['type'] = $row['type'];

            $i++;
        }
        return $typesList;

    }
    ///
    public static function getSelectedItemsByTypes( $category ) {

        $category = strval( $category );

        $dataBase = DataBaseConnect::connectToDataBase();
        $pertypesList = array();

        $queryResult = $dataBase->query( 'SELECT items.id,
       items.name,
       items.size,
       items.type,
       items.price,
       items.preview
FROM   items
       where items.type = "' . $category . '" limit 3' );

        $i = 0;
        while ( $row = $queryResult->fetch() ) {
            $pertypesList[$i]['id'] = $row['id'];
```

```
            $pertypesList[$i]['name'] = $row['name'];
            $pertypesList[$i]['size'] = $row['size'];
            $pertypesList[$i]['type'] = $row['type'];
            $pertypesList[$i]['price'] = $row['price'];
            $pertypesList[$i]['preview'] = $row['preview'];

            $i++;
        }
        return $pertypesList;
    }

    public static function getItemsByIds( $idsArray ) {
        $products = array();
        $dataBase = DataBaseConnect::connectToDataBase();
        $idsString = implode( ',', $idsArray );

        //      echo $idsString;

        $query = "select * from items where id in ($idsString)";

        $queryResult = $dataBase->query( $query );
        $queryResult->setFetchMode( PDO::FETCH_ASSOC );

        $i = 0;
        while ( $row = $queryResult->fetch() ) {
            $products[$i]['id'] = $row['id'];
            $products[$i]['name'] = $row['name'];
            $products[$i]['size'] = $row['size'];
            $products[$i]['price'] = $row['price'];
            $products[$i]['preview'] = $row['preview'];
            $i++;
        }
        return $products;
    }

    public static function createProduct($options){

        $dataBase = DataBaseConnect::connectToDataBase();
        $query = ('
        INSERT INTO items (name , price , size , type ,
availability , details , status , preview ) VALUES ( :name ,
:price , :size , :type , :availability , :details , :status ,
:preview)
        ');

        $queryResult=$dataBase->prepare($query);
        $queryResult-
>bindParam(':name',$options['name'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':price',$options['price'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':size',$options['size'],PDO::PARAM_STR);
```

```
        $queryResult-
>bindParam(':type',$options['type'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':availability',$options['availability'],PDO::PARAM_STR
);
        $queryResult-
>bindParam(':details',$options['details'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':status',$options['status'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':preview',$options['preview'],PDO::PARAM_STR);

        if ($queryResult->execute()){
            return $dataBase->lastInsertId();
        }
        return 0;
    }



    public static function updateProductById($id,$options){
        $dataBase = DataBaseConnect::connectToDataBase();
        $product = Product::getItemById($id);
        if (isset($_POST['submitUpdate'])){

        $query = ('update items set name = :name , price = :price ,
size = :size ,  type = :type , availability = :availability ,  details
= :details , status = :status , preview = :preview where id = :id');

        $queryResult=$dataBase->prepare($query);
        $queryResult->bindParam(':id',$id,PDO::PARAM_STR);
        $queryResult-
>bindParam(':name',$options['name'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':price',$options['price'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':size',$options['size'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':type',$options['type'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':availability',$options['availability'],PDO::PARAM_STR
);
        $queryResult-
>bindParam(':details',$options['details'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':status',$options['status'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':preview',$options['preview'],PDO::PARAM_STR);

        return $queryResult->execute();
        header("Location: /admin/product");
        }
```

```
        require_once                    (ProjectBase                    .
'/views/admin/adminFormUpdate.php');
        return true;
    }


     public static function insertPictures($options){

        $dataBase = DataBaseConnect::connectToDataBase();
        $id = self::getPicturesCount();
        $query = ('
        INSERT INTO pictures (id_item, pic1,  pic2 ,  pic3,  pic4,
pic5 , pic6 , pic7 , pic8 ) VALUES ( :id, :pic1 ,  :pic2 ,  :pic3 ,
:pic4,  :pic5 ,  :pic6 ,  :pic7 ,  :pic8 )
        ');

        $queryResult=$dataBase->prepare($query);
        $queryResult->bindParam(':id',$id,PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic1',$options['pic1'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic2',$options['pic2'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic3',$options['pic3'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic4',$options['pic4'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic5',$options['pic5'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic6',$options['pic6'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic7',$options['pic7'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic8',$options['pic8'],PDO::PARAM_STR);


        if ($queryResult->execute()){
            return $dataBase->lastInsertId();
        }
        return 0;
    }

     public static function updatePicturesById($id,$options){
        $dataBase = DataBaseConnect::connectToDataBase();
        $product = Product::getItemById($id);
        if (isset($_POST['submitUpdate'])){

        $query = ('update pictures set pic1 = :pic1, pic2 = :pic2,
pic3 = :pic3, pic4 = :pic4, pic5 = :pic5, pic6 = :pic6, pic7 = :pic7,
pic8 = :pic8 where id_item = :id');

        $queryResult=$dataBase->prepare($query);
        $queryResult->bindParam(':id',$id,PDO::PARAM_STR);
```

```
        $queryResult-
>bindParam(':pic1',$options['pic1'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic2',$options['pic2'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic3',$options['pic3'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic4',$options['pic4'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic5',$options['pic5'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic6',$options['pic6'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic7',$options['pic7'],PDO::PARAM_STR);
        $queryResult-
>bindParam(':pic8',$options['pic8'],PDO::PARAM_STR);
        return $queryResult->execute();        }
        return true;
    }

    public static function createPictures(){
        $dataBase = DataBaseConnect::connectToDataBase();
        $id = self::getPicturesCount();
        $query = ('
        INSERT INTO pictures (id_item, pic1, pic2, pic3, pic4, pic5,
pic6,          pic7,          pic8)          VALUES          (:id,
NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL)  ');

        $queryResult=$dataBase->prepare($query);
        $queryResult->bindParam(':id',$id,PDO::PARAM_STR);
        if ($queryResult->execute()){
            return $dataBase->lastInsertId();
        }
        return 0;
    }
}
?>
```

Модель Admin.php

```
<?php
include_once ProjectBase.'/components/Db.php';
class Admin
{
     public static function getAllItems() {
        $dataBase = DataBaseConnect::connectToDataBase();
        $productsList = array();
        $queryResult = $dataBase->query( 'SELECT items.id,
       items.name,
       items.size,
       items.type,
       items.price,
       items.preview,
       items.status,
       items.availability,
```

```
        items.details
FROM    items ORDER BY availability DESC' );

        $i = 0;
        while ( $row = $queryResult->fetch() ) {
            $productsList[$i]['id'] = $row['id'];
            $productsList[$i]['name'] = $row['name'];
            $productsList[$i]['size'] = $row['size'];
            $productsList[$i]['type'] = $row['type'];
            $productsList[$i]['price'] = $row['price'];
            $productsList[$i]['preview'] = $row['preview'];
            $productsList[$i]['status'] = $row['status'];
            $productsList[$i]['availability']                    =
$row['availability'];
            $productsList[$i]['details'] = $row['details'];

            $i++;
        }
        //        $_SESSION['path'] = "category";

        return $productsList;
    }

    public static function deleteProductById($id){
        $dataBase = DataBaseConnect::connectToDataBase();
        $query = 'delete from items where id = :id';
        $queryResult= $dataBase->prepare($query);
        $queryResult->bindParam(':id',$id,PDO::PARAM_INT);
        return $queryResult->execute();
    }

    public static function deletePicturesById($id){
        $dataBase = DataBaseConnect::connectToDataBase();
        $query = 'delete from pictures where id_item = :id';
        $queryResult= $dataBase->prepare($query);
        $queryResult->bindParam(':id',$id,PDO::PARAM_INT);
        return $queryResult->execute();
    }
}
?>
```

Модель User.php

```
<?php

include_once ProjectBase.'/components/Db.php';

class User {
    public static function register( $name, $login, $password ) {
        $dataBase = DataBaseConnect::connectToDataBase();

        $query = 'insert into user (name, login, password) values
(:name,:login,:password)';
```

```
        $queryResult = $dataBase->prepare( $query );
        $queryResult->bindParam( ':name', $name, PDO::PARAM_STR );
        $queryResult->bindParam( ':login', $login, PDO::PARAM_STR
);
        $queryResult->bindParam( ':password', $password,
PDO::PARAM_STR );

        return $queryResult->execute();
    }



    public static function auth( $userId ) {

        $_SESSION['user'] = $userId;
    }

    public static function getUserById( $id ) {

        $dataBase = DataBaseConnect::connectToDataBase();
        $query = 'SELECT * FROM user WHERE id = :id';
        $queryResult = $dataBase->prepare( $query );
        $queryResult->bindParam( ':id', $id, PDO::PARAM_INT );
        $queryResult->setFetchMode( PDO::FETCH_ASSOC );
        $queryResult->execute();

        return $queryResult->fetch();

    }

    public static function getUserOrdersById() {
        $orderList = array();
        $dataBase = DataBaseConnect::connectToDataBase();
        //        $idsString = implode( ',', $idsArray );

        //      echo $idsString;

        $query = ( "select * from product_order where userId =
".User::checkLogged()." order by id desc" );

        $queryResult = $dataBase->query( $query );
        $queryResult->setFetchMode( PDO::FETCH_ASSOC );

        $i = 0;
        while ( $row = $queryResult->fetch() ) {
            $orderList[$i]['userId'] = $row['userId'];
            $orderList[$i]['products'] = $row['products'];
            $orderList[$i]['total'] = $row['total'];
            $orderList[$i]['date'] = $row['date'];

            $i++;
        }
```

```
        return $orderList;
    }


}

?>
```

Контролер ProductController.php

```
<?php

include_once ProjectBase.'/components/Db.php';

class User {
    public static function register( $name, $login, $password ) {
        $dataBase = DataBaseConnect::connectToDataBase();

        $query = 'insert into user (name, login, password) values
(:name,:login,:password)';

        $queryResult = $dataBase->prepare( $query );
        $queryResult->bindParam( ':name', $name, PDO::PARAM_STR );
        $queryResult->bindParam( ':login', $login, PDO::PARAM_STR
);
        $queryResult->bindParam( ':password', $password,
PDO::PARAM_STR );

        return $queryResult->execute();
    }

        $query = 'SELECT COUNT(*) FROM user WHERE login = :login';

        $queryResult = $dataBase->prepare( $query );
        $queryResult->bindParam( ':login', $login, PDO::PARAM_STR
);
        $queryResult->execute();

        if ( $queryResult->fetchColumn() )
        return true;
        return false;
    }

    public static function getUserById( $id ) {

        $dataBase = DataBaseConnect::connectToDataBase();
        $query = 'SELECT * FROM user WHERE id = :id';
        $queryResult = $dataBase->prepare( $query );
        $queryResult->bindParam( ':id', $id, PDO::PARAM_INT );
        $queryResult->setFetchMode( PDO::FETCH_ASSOC );
        $queryResult->execute();

        return $queryResult->fetch();

    }
```

```
    public static function getUserOrdersById() {
        $orderList = array();
        $dataBase = DataBaseConnect::connectToDataBase();
        //        $idsString = implode( ',', $idsArray );

        //      echo $idsString;

        $query = ( "select * from product_order where userId =
".User::checkLogged()." order by id desc" );

        $queryResult = $dataBase->query( $query );
        $queryResult->setFetchMode( PDO::FETCH_ASSOC );

        $i = 0;
        while ( $row = $queryResult->fetch() ) {
            $orderList[$i]['userId'] = $row['userId'];
            $orderList[$i]['products'] = $row['products'];
            $orderList[$i]['total'] = $row['total'];
            $orderList[$i]['date'] = $row['date'];

            $i++;
        }

        return $orderList;
    }

    public static function checkAdmin() {
        if ( User::checkLogged() ) {
            $id = User::checkLogged();
            $dataBase = DataBaseConnect::connectToDataBase();
            $queryResult = $dataBase->query( 'SELECT role FROM user
where id='.$id );
            $queryResult->setFetchMode( PDO::FETCH_ASSOC );
            $c = $queryResult->fetch();
            $role = implode( $c );
            if ( $role == 'admin' ) {
                return true;
            }
        }
        return false;
    }

}

?>
```

Контролер AdminProductController.php

```
<?php

Class AdminProductController {

    public function actionIndex() {
```

```
        $productsList = Admin::getAllItems();
        require_once           (           ProjectBase           .
'/views/admin/adminProducts.php' );
        return true;
    }

    public function actionDelete( $id ) {
        Admin::deleteProductById( $id );
        Admin::deletePicturesById( $id );
        header( "Location: /admin/product" );
    }

    public function actionCreate() {
//        $errors = false;
        $categoryList = Product::getCategoriesList();
        //        $name = '';

        if ( isset( $_POST['submitCreate'] ) ) {
            $options['name'] = $_POST['name'];
            $options['size'] = $_POST['size'];
            $options['price'] = $_POST['price'];
            $options['type'] = $_POST['type'];
            $options['preview'] = $_POST['preview'];
            $options['details'] = $_POST['details'];
            $options['status'] = $_POST['status'];
            $options['availability'] = $_POST['availability'];

            $options['pic1'] = $_POST['pic1'];
            $options['pic2'] = $_POST['pic2'];
            $options['pic3'] = $_POST['pic3'];
            $options['pic4'] = $_POST['pic4'];
            $options['pic5'] = $_POST['pic5'];
            $options['pic6'] = $_POST['pic6'];
            $options['pic7'] = $_POST['pic7'];
            $options['pic8'] = $_POST['pic8'];
//

            $errors = false;

            if     (     !isset(     $options['name']     )     ||
strlen($options['name']) < 4  ) {
                $errors[] = 'Wrong name';
            }


            if ( $errors == false ) {
                $id = Product::createProduct( $options );
                Product::createPictures();
//                Product::insertPictures($options);
            } else {
                header( "Location: /admin/product/create" );
            }
```

```
            //              header( "Location: /admin/product" );
        }
        require_once           (           ProjectBase           .
'/views/admin/adminFormCreate.php' );

        return true;
    }

    public function actionUpdate($id) {
//        $errors = false;
        $categoryList = Product::getCategoriesList();
        $product = Product::getItemById($id);

        if ( isset( $_POST['submitUpdate'] ) ) {
            $options['name'] = $_POST['name'];
            $options['size'] = $_POST['size'];
            $options['price'] = $_POST['price'];
            $options['type'] = $_POST['type'];
            $options['preview'] = $_POST['preview'];
            $options['details'] = $_POST['details'];
            $options['status'] = $_POST['status'];
            $options['availability'] = $_POST['availability'];

            $options['pic1'] = $_POST['pic1'];
            $options['pic2'] = $_POST['pic2'];
            $options['pic3'] = $_POST['pic3'];
            $options['pic4'] = $_POST['pic4'];
            $options['pic5'] = $_POST['pic5'];
            $options['pic6'] = $_POST['pic6'];
            $options['pic7'] = $_POST['pic7'];
            $options['pic8'] = $_POST['pic8'];

            $errors = false;

            if     (     !isset(     $options['name']     )     ||
strlen($options['name']) < 4  ) {
                $errors[] = 'Wrong name';
            }

            if ( $errors == false ) {
                Product::updateProductById($id, $options );
                Product::updatePicturesById($id, $options );
            } else {

            }
        }
        require_once           (           ProjectBase           .
'/views/admin/adminFormUpdate.php' );
        return true;
    }
}
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка веб-орієнтованого додатку з продажу одягу»
студента групи 122-19ск-1 Ніколаєнка Артема Віталійовича

Керівник економічного розділу
доцент каф. ПЕП та ПУ, к.е.н **Л. В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом Ніколаєнко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом Ніколаєнко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
diplom.zip	Архів. Містить коди програми.
Презентація	
Презентація Ніколаєнко.ppt	Презентація кваліфікаційної роботи.