МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система інтернет-магазину з продажу продуктів харчування»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Назаренко Л.Д.**

Студента групи ІНдн-71о — **Дяченка О.В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Центр заочної, дистанційної і вечірньої форм навчання

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІНдн-71о спеціальності "Комп'ютерні науки" дистанційної форми навчання Дяченка Олексія Віталійовича.

Тема: " Інформаційна система інтернет-магазину з продажу продуктів харчування "

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) Аналітичний огляд задачі; 2) Проектування інформаційної системи; 3) Практична реалізація інформаційної системи.

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____ Назаренко Л.Д.

Завдання прийняв до виконання _____ Дяченко О.В.

РЕФЕРАТ

Записка: 46 стр., 20 рис., 8 табл., 2 додатки, 9 джерел.

Об’єкт дослідження — процес проектування інформаційної системи інтернет-магазину з продажу продуктів харчування.

Мета роботи — розробити проект інформаційної системи інтернет-магазину з продажу продуктів харчування. Програмно реалізувати створений проект з використанням сучасних технологій.

Методи дослідження — аналіз існуючих рішень в області інтернет-торгівлі, методи проектування інформаційних систем, методи створення структури бази даних, методи програмної інженерії, методи функціонального тестування.

Результати — створено функціонуюче програмне забезпечення системи інтернет-магазину з продажу продуктів харчування з використанням сучасних технологій веб-розробки. Серед виконаних підзадач можна виділити наступні: створення та реалізація моделі даних інформаційної системи, розробка користувальницького інтерфейсу, програмна реалізація інформаційної системи, виконане тестування основних модулів системи. Серед технологій, що були використані при реалізації виділяються наступні: HTML, CSS, JavaScript, PHP.

ІНТЕРНЕТ-МАГАЗИН, ТОВАР, ОБЛІК
JAVASCRIPT, HTML, CSS, PHP,
BOOTSTRAP, MYSQL
ВЕБ-САЙТ, ВЕБ-СЕРВЕР, БАЗА ДАНИХ

ЗМІСТ

ВСТУП 5
1 АНАЛІТИЧНИЙ ОГЛЯД ЗАДАЧІ 7
1.1 Загальні відомості щодо інтернет-комерції 7
1.2 Огляд існуючих рішень для реалізації інтернет-магазину 8
1.3 Постановка задачі 10
2 ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ 12
2.1 Принципова схема роботи інтернет-магазину 12
2.2 Потоки даних 13
2.3 Сховища даних 14
2.4 Структура бази даних 16
2.5 Архітектура програмного засобу 18
2.6 Організація інтерфейсу користувача 22
3 ФІЗИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 23
3.1 Вибір засобів розробки 23
3.2 Опис програмної реалізації 25
3.3 Інструкція користувача 29
3.4 Інструкція адміністратора 39
ВИСНОВКИ 45
БІБЛІОГРАФІЯ 46
ДОДАТОК А 47
ДОДАТОК Б 58

ВСТУП

Наразі сучасні технології проходять розвиток у надзвичайному темпі. Серед усього різноманіття технологічних засобів, створених людством найбільш наочними є цифрові технології, адже зараз ми з ними стикаємося кожен день. Ті системи, які впродовж відомої історії існували у фізичному втіленні наразі переходять у цифровий вигляд, зокрема це стосується торгівлі. Так звані процеси дігіталіації змінили наше уявлення про те, як слід продавати або покупати – в нашій діяльності з'явився клас таких сервісів, як інтернет-магазини.

Інтернет-магазин – це цифрова інформаційна система, яка на базі комп'ютерних технологій дозволяє проводити операції торгівлі[1]. Серед функцій інтернет-магазинів розглядаються наступні: відображення товарів та інформації про них покупцеві; створення замовлень та їхня обробка; проведення фінансових транзакцій при придбанні товарів, а також передача інформації логістичній службі для подальшої доставки придбаного товару.

У порівнянні із фізичними магазинами, інтернет-варіант має ряд особливостей. З одного боку, покупець, звичайно, не може потримати в руках товар, щоб оцінити його властивості. Також, зрозуміло, що товар не буде отриманий у ту ж мить, коли він був оплачений.

З іншого боку, і цифрові технології дають ряд переваг, які складно або, навіть, неможливо отримати при фізичній торгівлі. Це, наприклад, можливість забезпечити покупця максимально повною інформацією про товар. Інша перевага – відсутність необхідності у виставковому залі, а звідси витікає майже необмежена кількість товарних позицій у магазині. Також, слід відзначити, що інтернет-торгівля спрощує покупки і для юридичних осіб. Наразі усі необхідні документи можна подавати в електронному вигляді, без необхідності очікувати тривалі поштові відправлення цих документів.

Дослідження показують, що кількість користувачів інтернету постійно зростає, а із нею, відповідно, і кількість інтернет-покупців. Різні джерела демонструють цифри у 25%-30%[2] покупок, що були здійснені через інтернет.

При цьому тенденція направлена у бік зростання цих показників. Потенційний користувач високо цінує можливість здійснювати покупки у будь-який час доби, не покидаючи домівки. Потенційним же продавцям до вподоби можливість зекономити на приміщеннях та охороні, а також для них є привабливим географічне охоплення цільової аудиторії – значно ширше, ніж у звичайних фізичних магазинів.

Мета цього проекту полягає в тому, щоб створити інтернет-магазин з продажу харчових продуктів. На думку автора, цей напрям є актуальним, адже в умовах існуючої пандемії[2,3] ми можемо спостерігати різкий підйом інтернет-торгівлі, особливо тих товарів, що відносяться до категорії життєво необхідних. В будь-який момент карантинні заходи можуть бути посилені, отже переваги щодо можливостей покупки прямо із домівки будуть вирішальними.

В даній роботі буде вирішений ряд задач. А саме – буде проведений огляд існуючих рішень і на основі нього будуть виведені типові вимоги до інформаційних систем цього класу. Далі, будуть проведені проектні роботи по створенню інформаційної моделі цієї системи. Після цього розглянемо фізичну реалізацію і її особливості. І, нарешті, ми розглянемо демонстраційний приклад. В кінці буде наведено висновок із роботи із описом отриманих результатів.

1 АНАЛІТИЧНИЙ ОГЛЯД ЗАДАЧІ

1.1 Загальні відомості щодо інтернет-комерції

Про процеси, що призводять до формування інформаційної економіки, стали говорити ще в 70-х роках 20 століття. Основа цих процесів – бурхливе зростання інформаційних і комунікаційних технологій, перетворення комп'ютера в домінуючий снаряд праці та поширення мереж передачі даних. Спостережуваний сьогодні підйом економіки розвинених країн не може бути обумовлений тільки кількісним зростанням використовуваних комп'ютерів - ключове значення набула організація їх в мережу.

Сьогодні електронна комерція надає найширші можливості як постачальникам, так і клієнтам. Серед цих можливостей:

- самостійна реєстрація покупця на сайті постачальника. Це створює додаткові зручності в обслуговуванні для клієнтів, а постачальникам дозволяє здійснювати адресну рекламу своїх товарів і послуг і маркетингові дослідження в процесі продажів;
- оформлення замовлень через інтернет за допомогою електронних каталогів і прайс-листів. Дана можливість забезпечує значну економію часу при пошуку необхідного товару або послуги і порівнянні цін різних постачальників;
- електронна обробка замовлення, включно з перевіркою наявності товару на складі, розрахунок можливих термінів поставки. Дана можливість є невід'ємною функціональною частиною логістичної системи підприємства;
- прийом оплати за покупку через інтернет. Оплата при цьому може здійснюватися за допомогою банківських карт через захищені платіжні термінали, а також за допомогою локальних або міжнародних платіжних інтернет систем.

Електронна комерція має всі можливості для подальшого розвитку. По-перше, економія на витратах інтернет-магазинів дозволяє їм знижувати ціни, і

купувати в інтернет-магазинах сьогодні часом набагато вигідніше, ніж в звичайних магазинах. Цей факт забезпечує і постійний приплив покупців, і поява нових гравців на ринку. Електронний бізнес, і зокрема інтернет-магазин як його складова, стає більш конкурентним, що в свою чергу позитивно позначається на рівні якості надаваних послуг і товарів, що пропонуються [1].

1.2 Огляд існуючих рішень для реалізації інтернет-магазину

На практиці існує два способи створити інтернет-магазин. Перший – розробити систему «під ключ». Тобто, виходячи із потреб конкретного сервісу, розробник збирає дані та вимоги, а далі – проектує та реалізує систему.

Інший підхід – використання існуючих систем, так званих CMS (Content Management System). Такі системи дозволяють користувачам без навичок програмування встановити веб-сервіс з обслуговування клієнтів. Задачею таких користувачів є наповнення веб-ресурсу контентом і організація правильного доступу до нього різними користувачами.

Обидва підходи мають свої як переваги, так і недоліки. Серед переваг індивідуальної розробки можна виділити те, що власник подібної системи буде мати над нею повний контроль в рамках архітектури та технологій для реалізації. Подібні системи менш вразливі до кібератак, адже їхній програмний код не буде широко доступним. Недоліком таких систем є те, що по мірі росту кількості користувач систему доведеться горизонтально масштабувати і ускладнювати її архітектуру. А це, зазвичай, призведе до коштовності подальшої розробки.

Що стосується систем CMS – їхні недоліки та переваги є протилежними. З одного боку, це готові рішення, якими можна відразу ж користуватися. З іншого боку, вони не реалізують задачі конкретного власника інтернет-магазину, а натомість реалізують цілий клас функцій, типових для даного виду веб-сервісів.

В даній роботі проект був реалізований за першим шляхом, але далі буде проведено огляд систем CMS. Серед найбільш розповсюджених систем CMS можна виділити такі: Joomla, WordPress, 1С-Бітрікс. Розглянемо їх детальніше.

Joomla – це безкоштовна CMS, що створена для публікації контенту і інтернеті. Серед усіх її можливостей, нас найбільше цікавить можливість створити на її основі інтернет-магазин. Насправді, Joomla найчастіше для цього і використовується.

Найпримітніші характеристики цієї CMS:

- Реалізована на PHP з використанням СУБД MySQL;
- Інтегровані модулі безпеки для аутентифікації та авторизації користувачів системи;
- Легкість керування контентом, зокрема товарами та новинами;
- Широкі можливості налаштування зовнішнього інтерфейсу – блоків або меню;
- Ряд соціальних функцій, такі як опитування, чати та форуми;
- Можливість роботи під усіма найпоширенішими операційними системами.

На поточний час відомо про більше, ніж 6000 розширень для Joomla.

На сьогодні можна знайти більше 3000 модулів і компонентів, які задовольнять вимоги будь-якого розробника сайту. Дану CMS можна використовувати для інтернет-магазинів будь-якого розміру.

Ще одною популярною CMS є **WordPress**. Це також безкоштовна система, яка спершу з’явилась в якості рушія для створення блогів, але згодом вона отримала додаткову функціональність. Наразі WordPress є однією із найпопулярніших у світі систем для інтернет-комерції.

Найпримітніші характеристики цієї CMS:

- Реалізована на PHP з використанням СУБД MySQL;
- Можливість публікації контенту за допомогою стороннього ПЗ;
- Легкість встановлення та налаштування системи;
- Унікальна система розширень (т.з. плагінів);

- Легкість керування інтерфейсом користувача;
- Велика колекція додаткових модулів до системи;
- Система адрес браузеру, зрозумілих для людини;
- Наявність українського перекладу [4].

Bitrix – продукт, призначений перш за все для корпоративного використання, хоча існують відносно недорогі базові версії для створення простих сайтів. Спочатку компанія Bitrix розробила цю CMS, але потім вона була поглинена компанією 1С, внаслідок чого нова назва продукту - 1С-Bitrix.

Специфіка Bitrix досить оригінальна. Ця CMS розроблялася як продукт з максимально простим і зручним інтерфейсом, при цьому дозволяє повністю налаштовувати будь-які параметри без застосування навичок програмування.

І якщо реалізувати зручний інтерфейс їм, можна сказати, вдалося, то принципи адміністрування сайтом перетворилися в власну мову програмування, освоїти який без курсу навчання практично неможливо. Причому з'явилася сертифікація не тільки для адміністраторів Bitrix, але і для простих користувачів-менеджерів - що багато про що говорить.

Але 1С-Bitrix - це комерційний проект, як і будь-який продукт компанії 1С, тому використовувати його для створення невеликого інтернет-магазину вкрай дорого і невигідно [5].

1.3 Постановка задачі

Розглянувши існуючі рішення, ми можемо сформулювати постановку задачі. Треба створити програмне забезпечення, що дозволяє вести облік та продаж товарів (продуктів харчування). Метою проекту є розробка інформаційної системи для обліку та продажу харчових продуктів, а також для виконання консультативних та довідкових функцій з цього напрямку. Серед етапів виконання проекту виділимо наступне:

- Розробка інформаційної структури системи;
- Побудування архітектурного рішення для майбутньої системи;

- Вибір програмних засобів для фізичної реалізації;
- Власне, фізична реалізація;
- Тестування отриманого програмного засобу та опис рішення.

Результатом виконання поставленої задачі є web-сайт у вигляді електронного каталогу товарів. Користувач повинен мати можливість переглянути інформацію про товар із обраної категорії, а також здійснювати пошук інформації на web-сайті за одним параметром (наприклад, назвою товару).

Інформація про категорії товарів, товари та користувачів web-додатку має зберігатися у базі даних MySQL. Дизайн сайту має бути виконаний з використанням таблиць стилів CSS.

Перегляд каталогу товарів має бути доступним лише для зареєстрованих користувачів. Для усіх інших відвідувачів сайту має бути доступна лише головна сторінка сайту із загальною інформацією та перехід на сторінку з реєстрацією-авторизацією. Необхідно передбачити перевірку валідності даних, що вводяться користувачем у поля форм web-додатку.

Також необхідно реалізувати модуль адміністратора сайту для редагування інформації, що міститься у базі даних.

2 ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Принципова схема роботи інтернет-магазину

Маючи поставлену задачу, опишемо типовий сценарій роботи системи інтернет-магазину.

Система працює на віддаленому сервері і очікує на запити від користувачів. З іншого боку, система є свого роду обгорткою над базою даних. В базі даних наявне наступне наповнення:

- Список товарів;
- Список категорій, до яких відносяться товари;
- Список замовлень товарів користувачами;
- Список зареєстрованих користувачів.

Користувач завантажує стартову сторінку і на ній отримує ряд посилань, через перехід на які можна виконати ряд дій:

- Переглянути список усіх товарів;
- Переглянути список товарів, відповідно до категорії;
- Переглянути інформацію про конкретний товар;
- Знайти товар за ключовим словом;
- Зареєструватися в системі;
- Провести авторизацію користувача в системі;
- Вибрати товар для покупки;
- Замовити вибрані товари;
- Переглянути контактну інформацію про магазин.

Коли користувач переходить по відповідному посиланню, його браузер відправляє HTTP-запит на сервер. Сервер, в свою чергу, має коректно опрацювати запит, виділити яку дію необхідно виконати та відправити відповідь клієнтові.

2.2 Потоки даних

Розуміючи принцип роботи інформаційної системи за типовим сценарієм, перелічимо основні сутності, що наявні в ній. Серед них виділяються: товар, категорія товару, користувач, замовлення.

Крім цього, з’ясуємо рівні доступу для різних груп осіб, що приймають участь в інформаційних процесах. Серед них присутні звичайний користувач та інформаційний адміністратор.

Інформаційний адміністратор (далі – просто «адміністратор») може створювати, видаляти, редагувати та переглядати перелічені вище сутності.

Звичайний користувач (далі – просто «користувач») може переглядати відомості про усі товари та категорії, а також створювати, змінювати та переглядати відомості по замовленнях, що належать тільки йому.

Нижче, на рисунку 2.1, наведена DFD-діаграма 1-го де показані відношення, між користувачами, сутностями та інформаційними процесами.

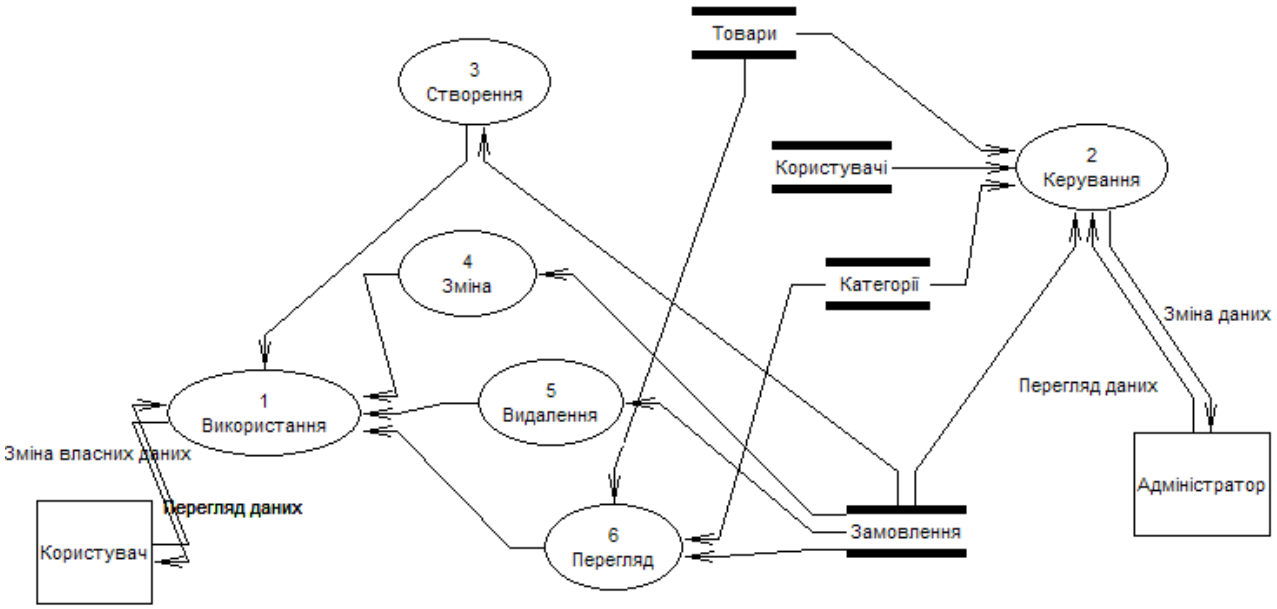


Рисунок 2.1 – DFD-діаграма 1-го рівня

2.3 Сховища даних

До розглянутих інформаційних сутностей наведемо відповідні сховища даних: «Товар», «Категорія», «Замовлення», «Користувач». У таблиці 2.1. наведемо опис атрибутів цих сутностей, що будуть наявні у сховищах.

Атрибут	Тип даних
Користувач	
Ім’я	Персональне імя користувача
Email	Email користувача
Пароль	Пароль для входу в систему
Роль (користувач/адміністратор)	Роль користувача в системі (користувач/адміністратор)
Категорія товару	
Назва	Назва категорії
Порядковий номер	Порядковий номер при відображенні у списках категорії
Стан	Стан відображення категорії (активна/неактивна)
Товар	
Назва	Назва товару
Категорія	До якої категорії належить товар
Артикул	Артикул товару
Вартість	Вартість товару

Наявність	Чи є товар наявним? (Чи можна його замовити?)
Виробник	Назва компанії або країни-виробника
Опис	Текстовий опис товару
Чи є новим (так/ні)	Чи буде товар позначений як новий на сайті
Чи є рекомендованим (так/ні)	Чи буде товар відображатися у списку рекомендованих?
Стан (активний/неактивний)	Неактивні товари існують в системі, але не відображаються у каталогах
Замовлення	
Ім’я користувача	Персональне імя замовника
Телефон користувача	Телефон замовника
Коментарій користувача	Текстовий коментарій до замовлення
Користувач	Id користувача, що здійснив замовлення
Дата оформлення	
Список товарів	Масив типу *Товар*
Стан замовлення	Числовий код одного з наступних варіантів: *нове, в обробці, в доставці, завершене*

Таблиця 2.1 – Перелік сховищ і їхніх атрибутів

Зауваження. В об’єкті «Замовлення» атрибути «Ім’я користувача» та «Телефон користувача» використовуються для незареєстрованих користувачів. При цьому параметр «Користувач» - не заданий. Якщо ж користувач зареєстрований, то ситуація зворотна.

2.4 Структура бази даних

Перенесемо отримані описи сховищ на структуру майбутньої бази даних. Сховищам будуть відповідати таблиці у СУБД, а атрибутам – поля цих таблиць.

Результати наведені далі, у табл. 2.2-2.4.

user					
#	Чи є ключовим	Назва поля	Тип даних	Чи може бути Null	Чи є значення унікальним
1	PK	Id	Integer	-	+
2		Name	Varchar(255)	-	-
3		email	Varchar(255)	-	-
4		Password	Varchar(255)	-	-
5		role	Varchar(50)	+	-

Таблиця 2.2 – Структура таблиці user

category					
#	Чи є ключовим	Назва поля	Тип даних	Чи може бути Null	Чи є значення унікальним
1	PK	Id	Integer	-	+
2		Name	Varchar(255)	-	-
3		Sort_order	Integer	-	-
4		Status	Integer	-	-

Таблиця 2.3 – Структура таблиці category

product					
#	Чи є ключовим	Назва поля	Тип даних	Чи може	Чи є значення унікальним

				бути Null	
1	PK	Id	Integer	-	+
2		Name	Varchar(255)	-	-
3	FK	Category_id	Integer	-	-
4		Code	Integer	+	-
5		price	Varchar(50)	-	-
6		availability	Integer	-	-
7		brand	Varchar(255)	+	-
8		description	text	+	-
9		Is_new	Integer	-	-
10		Is_recommended	Integer	-	-
11		status	Integer	-	-

Таблиця 2.4 – Структура таблиці product

Product_order					
#	Чи є ключовим	Назва поля	Тип даних	Чи може бути Null	Чи є значення унікальним
1	PK	Id	Integer	-	+
2		User_name	Varchar(255)	-	-
3	FK	User_phone	Varchar(255)	-	-
4		User_comment	text	+	-
5		User_id	Integer	+	-
6		Date	Timestamp	-	-
7		Products	Text	-	-
8		status	Integer	+	-

Таблиця 2.4 – Структура таблиці product_order

Наведемо ER-діаграму отриманої структури бази даних.

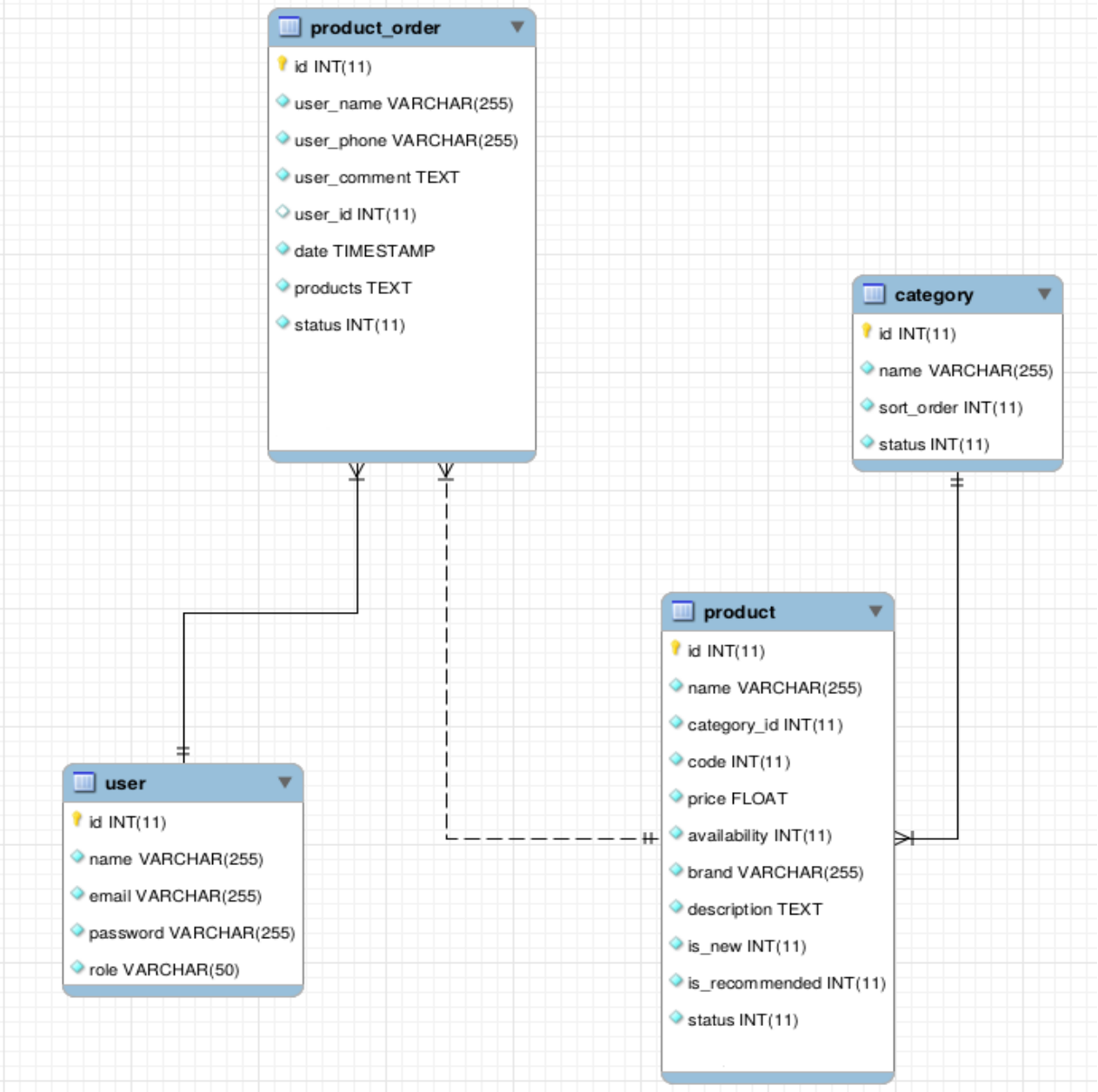


Рисунок 2.2 – ER-діаграма бази даних

2.5 Архітектура програмного засобу

Центральним елементом архітектури майбутньої системи є шаблон MVC (Model-View-Controller). Основне його призначення відокремити друг від друга інтерфейс користувача, логіку роботи з даними та власне дані. Для цього в систему вводиться 3 види програмних компонентів – модель (Model), вигляд(View) та контролер(Controller).

Розглянемо ці компоненти детальніше, враховуючи веб-орієнтованість додатку.

Модель – це набір програмних компонентів, що відповідальні за програмне уособлення даних. З одного боку компоненти моделі реалізують інтерфейс, що висловлює функціональність та властивості об’єктів сховища. З іншого боку, ці компоненти реалізують безпосередньо логіку роботи з самими сховищами (в нашому випадку – із СУБД). Інакше кажучи, модель абстрагує низькорівневі CRUD-операції, надаючи високорівневий функціональний інтерфейс об’єктів. Зазвичай, одному такому об’єкту відповідає одна програмна одиниця (клас, у мові PHP).

Вигляд – це набір компонентів, відповідальних за відображення користувальницького інтерфейсу системи. Зокрема, у веб-додатках ці компоненти формують веб-сторінки, які бачить відвідувач ресурсу. В задачі вигляду входять як отримання команд від користувача на дії із системою, так і відображення результатів цих команд.

Контролер – це зв’язуючий ланцюг між моделлю та виглядом. Компоненти контролеру отримують запити від вигляду, змінюють стан моделі і відображають ці зміни на вигляді (можливо на іншому).

Розглянемо зв'язок між моделлю, контролером та виглядом на рисунку 2.3.

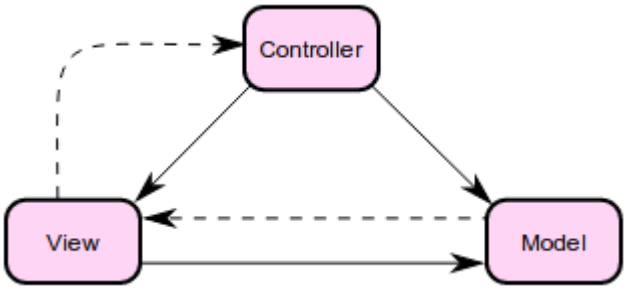


Рисунок 2.3 – Схематичне зображення шаблону MVC

Подібний підхід дозволяє архітектурно структурувати програмний продукт. Наприклад, якщо розробник побажає повністю змінити інтерфейс користувача, він це легко зробить, змінивши компоненту вигляду. При цьому вносити зміни в модель та контролер, скоріш за все, не доведеться, адже логіка залишиться незмінною.

Крім цього в проекті буде використовуватися компонент Маршрутизатор (Router). Його основна задача полягає в тому, щоб надати доступ к веб-сервісу через єдину точку входу. Усі запити від клієнтів поступають веб-серверу, а той, в свою чергу, замість обробки, як це робиться традиційно, віддає їх компоненту веб-сервісу – маршрутизатору. Маршрутизатор же самостійно приймає рішення, як поступити із запитом.

Використання маршрутизатору обумовлене тим, що за замовченням URI із запиту клієнту транслюється безпосередньо у шлях у файловій системі серверу. Так, наприклад, запит http://myshop.com/products/get.php?id=331

буде переведено у подібний шлях:

/var/www/myshop.com/products/get.php

Звідси ми бачимо недоліки. По-перше, потенційний зловмисник таким чином може вивчати файлову структуру додатку. Також він, із структури запиту може дізнатися, на якій мові розроблений додаток. По-друге, такий вид шляху незручний і непривабливий для користувача. По-третє, розробник має у структурі файлів і каталогів мати на увазі логічну вкладеність функціоналу системи.

Маршрутизатор дає змогу подолати ці недоліки за рахунок того, що він перехоплює URI, надісланий клієнтом у запиті, і самостійно його транслює у деякий внутрішній шлях, заданий розробником через шаблони. Далі із цього внутрішнього шляху виділяються контролер (controller), дія (action) та її параметри. Далі маршрутизатор, знаючи, власну структуру додатку, а також яку дію треба виконати, перенаправляє виконання до відповідного обробника дії в заданому контролері.

Таким чином, запит вище з використанням маршрутизатора може прийняти наступний вигляд:

http://myshop.com/product/331

Маршрутизатор переведе цей URI у внутрішній віртуальний шлях, наприклад наступний: /product/view/331. Після розбору цього шляху можна

отримати кінцеву мету запиту: контролер ProductController, дія actionView, productId = 331.

Таким чином, типовий життєвий цикл запиту до веб-ресурсу наступний:

- Користувач надсилає запит, в якому міститься URI бажаного ресурсу.
- Веб-сервер отримує запит і, завдяки власним налаштуванням, відразу ж передає його маршрутизатору.
- Маршрутизатор конвертує зовнішній URI у внутрішній віртуальний, а з нього отримує інформацію про те, що далі повинно обробляти запит.
- Маршрутизатор викликає необхідний action-метод визначеного контролера.
- Action-метод через використання методів моделі працює із базою даних.
- Результати роботи передаються у вигляд (view). Вигляд за необхідністю обробляється і компілюється.
- Оброблений вигляд передається користувачеві у виді веб-сторінки.

Використання маршрутизатору дозволяє зручно розподілити URI ресурсів системи. Для користувача це означатиме наочні та зрозумілі адреси, які він бачитиме у браузері при роботі із веб-додатком. У таблиці далі наведений перелік основних адрес та відповідних дій або ресурсів:

URI (відносна адреса)	Ресурс / дія
/	Головна сторінка
/catalog	Відображення усіх наявних товарів
/category/<id>	Відображеня усіх товарів за категорією <id>
/product/<id>	Відображення відомостей про товар <id>
/search/<text>	Виконання пошуку товару, назва якого мість <text>
/user/register	Реєстрація користувача
/user/login	Вхід в систему користувача

Таблиця 2.5 – Розподіл адрес основних ресурсів

2.6 Організація інтерфейсу користувача

Вище був даний перелік функцій, доступних користувачеві. Нижче наведемо принципову схему організації веб-сторінок.

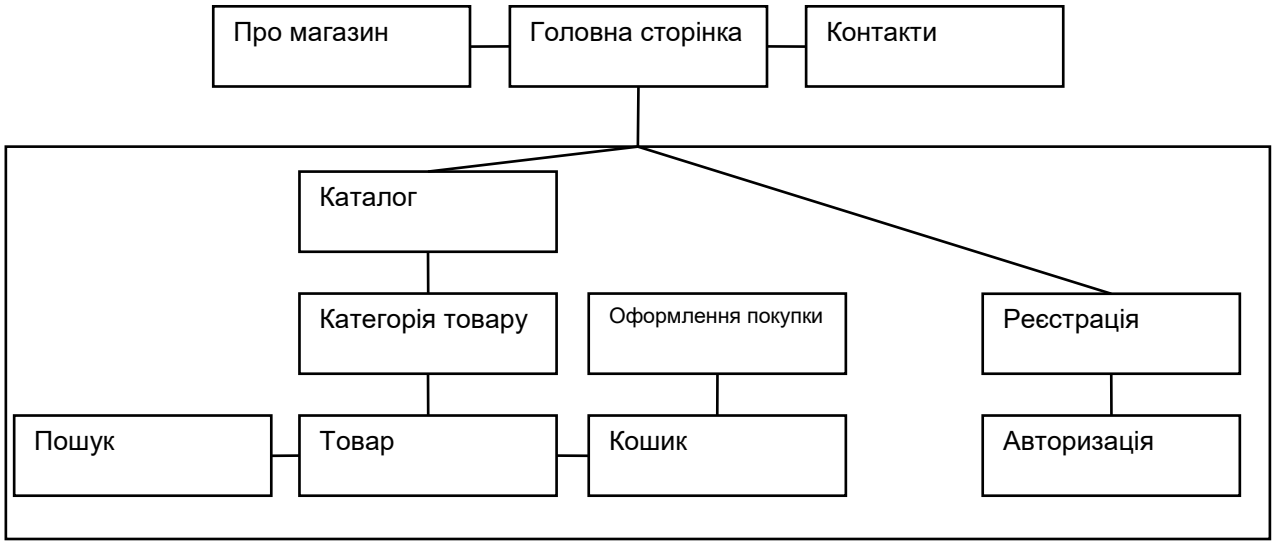


Рисунок 2.4 – Структурна схема інтернет-магазину

3 ФІЗИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір засобів розробки

Для виконання фізичної реалізації проекту треба вибрати відвідні програмні засоби. При виборі ми будемо користуватися наступними положеннями:

- Засоби мають бути безкоштовним або свободним програм забезпеченням;
- Засоби мають розповсюдженими, так щоб можна було легко отримати усю необхідну інформацію та підтримку;
- Засоби мають бути кросплатформенними з можливістю їхнього переносу на інші операційні та апаратні системи;
- Засоби мають бути достатньо легкими в освоєнні.

Враховуючи вказане вище, ми будемо використовувати наступні платформи та технології. Для серверної частині – це Apache2, PHP та MySQL. Для клієнтської частини – це стандартнй набір HTML5+CSS3+JavaScript, а також фреймворк bootstrap.

Apache — свободне програмне забезпечення, що реалізує функціональність веб-серверу для передачі статичного та динамічного веб-контенту[6]. Хоча Apache найчастіше використовується на Linux, наразі він доступний для усіх основних операційних систем.

Серед основних характеристик цього веб-серверу виділяють наступні:

- Підтримка динамічного завантаження модулів;
- Подійно-асинхронна та багатопотока моделі обробки веб-запитів;
- Функціональність у якості зворотнього проксі-серверу;
- Підтримка сучасних шифрувальних протоколів, таких як SSL/TLS разом з SNI та OSCP;
- Підтримка стискування з використанням gzip;

- Можливість динамічного редагування запитів, зокрема їхніх заголовків та URL (це використовується в даному проекті для маршрутизації).

PHP — популярна скриптова мова програмування. Хоча це мова загального призначення, вона створювалася для динамічної генерації веб-контенту сервером[7]. Рушій мови PHP, Zend Engine є відкритим програмним забезпеченням. По відношенню до веб-серверу PHP може працювати або як його модуль, або як окремий системний сервіс. Наразі, в 2021 році, PHP використовується майже на 80% веб-сайтів.

Характерними особливостями мови PHP є наступне:

- Динамічна, слабка типізація з можливістю використання поступової типізації;
- Мультипарадгменність з підтримкою функціонального, об’єктно-орієнтованого та процедурного стилів;
- Зріла кодова база;
- Можливість інтеграції PHP-коду в HTML та інші сторінки на сервері;

MySQL — одна з найпопулярніших реляційним систем керування базами даних[8]. Наразі ця СУБД найчастіше застосовується для розробки веб-орієнтованих інформаційних систем у складі так званого LAMP (Linux, Apache, MySQL, PHP). MySQL – це програмне забезпечення із відкритим кодом.

Бази даних MySQL влаштовані за клієнт-серверною моделлю. Таким чином MySQL являє собою сервер, що очікує підключення користувача, який надсилає запити на мові SQL.

Серед основних характеристик даної СУБД виділяють наступні:

- Майже повна реалізація стандарту ANSI SQL 99 та його розширень;
- Підтримка SSL;
- Підтримка Unicode;

- Підтримка декількох рушіїв сховищ: InnoDB, MyISAM та ін.;
- Кластерізація баз даних за допомогою технології MySQL Cluster;
- Існування програмних інтерфейсів для усіх основних мов програмування.

Bootstrap — відкритий CSS-фреймворк для створення веб-орієнтованого інтерфейсу користувача[8]. Являє собою набір шаблонів, що дозволяють створювати графічні елементу інтерфейсу, такі як форми, кнопки, навігаційні елементи тощо. Основне призначення фреймворку – спрощення фронтенд-розробки.

Характерні особливості Bootstrap[13]:

- Підтримка усіх основних браузерів;
- Підтримка підходу адаптивного верстання;
- Одноманітність CSS-стилів.

3.2 Опис програмної реалізації

В корневому каталозі веб-ресурсу знаходиться файл index.php та ряд підкаталогів, в якому знаходяться модулі, відповідальні за відповідні функції. Крім того, в корені знаходиться файл .htaccess. Він не є програмним кодом, а є лише конфігураційним файлом для веб-серверу Apache2. Тим не менш, цей файл незамінний в цьому проекті, адже завдяки йому може коректно працювати компонент Router. Конкретніше, цей файл примушує веб-сервер перенаправляти усі клієнтські запити на файл index.php.

index.php, в свою чергу, будучі запущеним, ініціалізує необхідні змінні, починає користувальницьку сесію, завантажує необхідні класи та інстанціює об’єкт Router, після чого йому ж і передає керування. Принцип роботи компоненту Router (маршрутизатор) був розглянутий у попередньому розділі.

Розглянемо інші складові проекту. Вони згруповані по каталогах, отже для короткості тут описуються лише призначення каталогів та лише деяких файлів.

Більш детальний опис можна знайти у коментарях кожного з файлів з PHP-кодом.

Каталог	Опис
components	Компоненти програмного додатку. Сюди входять класи, які виконують важливі задачі, але не підпадають під модель MVC. Це, свого роду, аналоги dll-бібліотек. Більш детально про компоненти описано нижче.
config	Конфігураційні файли, які містять змінні, що керують параметрами додатку. Ці файли може і, навіть повинен, редагувати адміністратор системи. Сюда входять db_params.php - параметри підлючення до СУБД та routes.php – масив шаблонів для маршрутизатора.
controllers	Контролери додатку. Ці класи обробляють користувальницький ввід та приймають рішення про його подальшу обробку та вивід результаті. Методи цих класів являють собою дії, які мають привести до конкретного ефекту в інформаційній системі.
models	Моделі додатку. Ці класи реалізують функціональну обгортку над інформаційними об’єктами сховища (тобто, над таблицями бази даних, в нашому випадку). Методи цих класів оперують безпосередньо самими даними.
views	Вигляди додатку. Більшість з них є звичайними HTML-сторінками, але з вкрапленнями PHP-коду. Вигляди призначені для відображення користувачеві результатів роботи відповідних контролерів.
views/layouts	Тут знаходять фрагменти коду, які використовуються одночасно в декількох виглядах.

upload	Каталог для зберігання користувальницьких файлів. В нашому випадку тут розміщуються зображення товарів.
template	Каталог для зберігання веб-ресурсів, таких як css та js-кодів, а також шрифтів.

Таблиця 3.1 – Опис каталогів додатку

Серед компонентів додатку можна виділити наступні:

Компонент	Опис
Autoload.php	Відповідальний за стандартну PHP-функціональність щодо автозавантаження класів.
AdminBase.php	Містить загальну функціональність для контролерів адміністратора. В нашому випадку тут є лише метод перевірки прав доступу.
Cart.php	Функціональність щодо кошику користувача, а саме можливість зберегти список покупок через використання cookies.
Db.php	Компонент для роботи із базою даних.
Pagination.php	Компонент для генерації посилань постраничної навігації.
Router.php	Компонент, що реалізує функціональність маршрутизатора
upload	Каталог для зберігання користувальницьких файлів. В нашому випадку тут розміщуються зображення товарів.
template	Каталог для зберігання веб-ресурсів, таких як css та js-кодів, а також шрифтів.

Таблиця 3.2 – Опис компонентів додатку

Приклади основних складових проекту наведені у додатку А.

Розглянемо типовий життєвий цикл запиту користувача. Наприклад, користувач хоче переглянути товар на головній сторінці, який має відносну адресу /product/77.

Веб-сервер отримує відповідний get-запит і завдяки налаштуванням у файлі .htaccess перенаправляє запит на файл index.php. Він, в свою чергу,

завантажує основні налаштування і інстанціює роутер. Роутер завантажує відповідний файл налаштувань із маршрутами і шукає там шаблон, якому відповідає запит - /product/77.

В нашому випадку цьому запиту буде відповідати шаблон:

'product/([0-9]+)' => 'product/view/$1'

Далі роутер зможе прийняти рішення про подальші дії. Тут буде завантажений контролер ProductController.php, а в ньому буде викличений метод actionView за параметром - $1.

Метод actionView() звертається до моделі Product.php і в неї викликає інший метод - Product::getProductById(). Той, в свою чергу, звертаєся до бази даних і заздалегідь сформованим запитом отримує список продуктів за відомим id. Далі вибірка із бази даних буде переведена у масив, який буде повернуто назад до методу actionView().

Далі actionView() підключає відповідний вигляд - /views/product/view.php. Вигляд отримує масив із товарами, заповнює HTML-шаблон і генерує відповідь клієнту.

Нарешті, клієнт отримує сформовану веб-сторінку із переглядом товару.

3.3 Інструкція користувача

Сайт запускається за посиланням http://localhost (для інших варіантів треба відповідним чином налаштувати віддалений веб-сервер). З'являється головне вікно, де ми побачимо шапку сайту меню із категоріями зліва, «підвал» сайту знизу та список найновіших товарів в основній області.

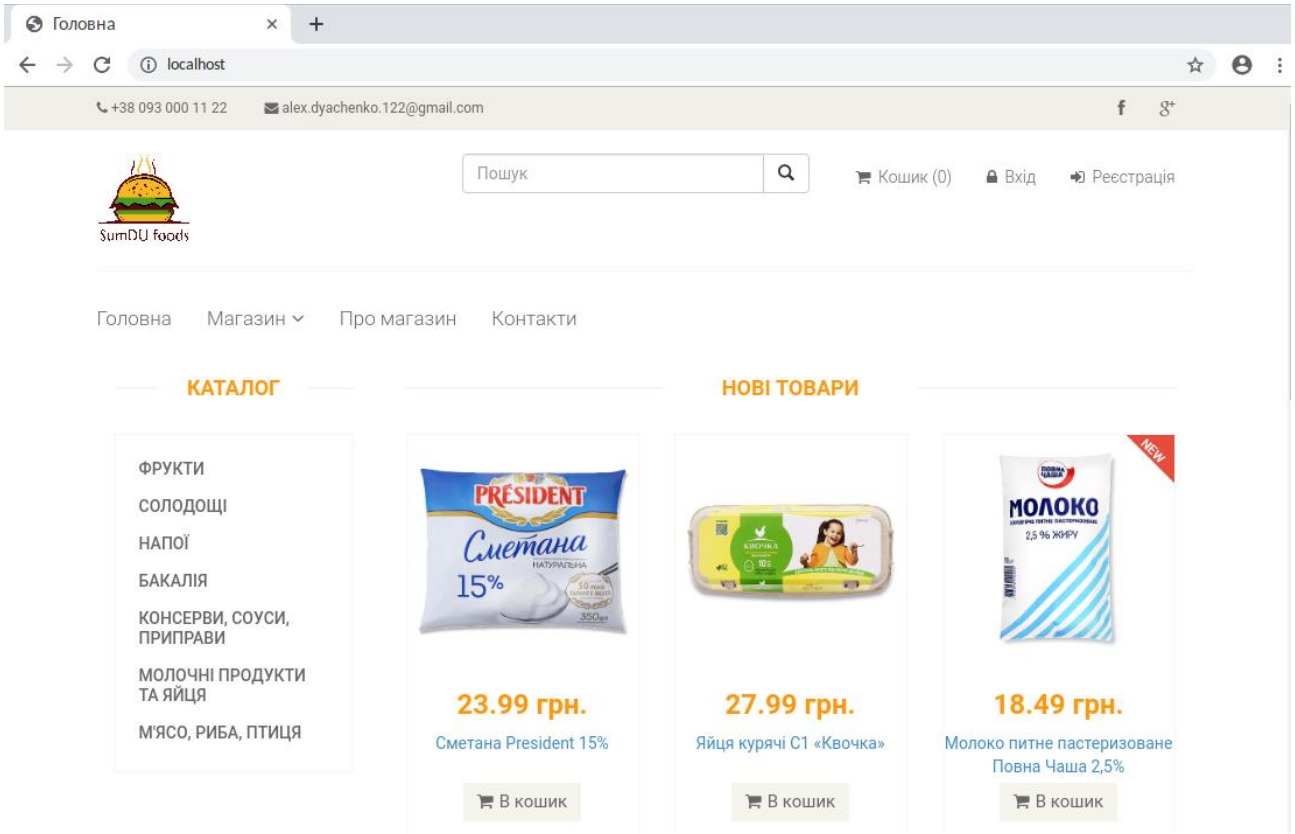


Рисунок 3.1 – Вигляд головного вікна додатку

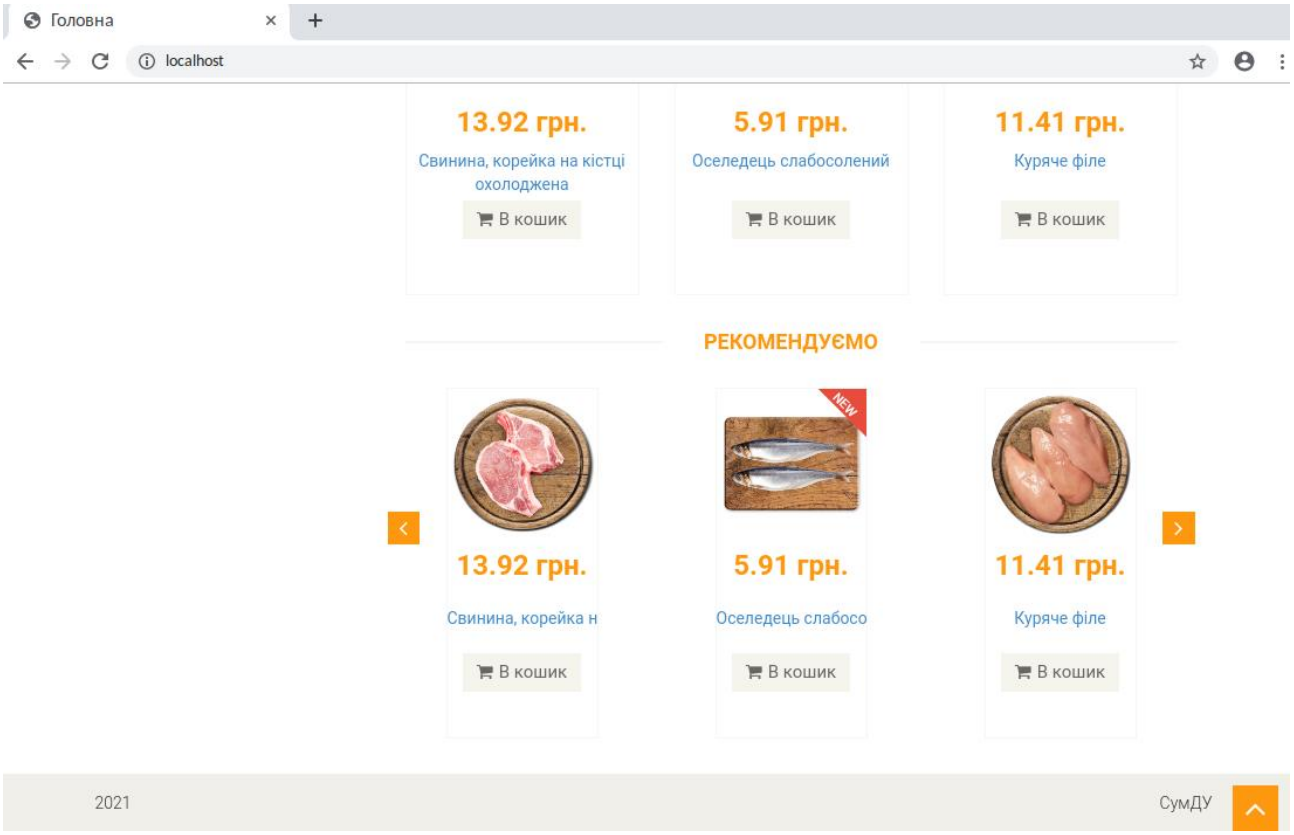


Рисунок 3.2 – Вигляд головного вікна додатку (знизу)

З головної сторінки користувач може перейти на наступні функціональні сторінки сайту:

- Перегляд каталогу усіх товарів та по категоріях;
- Перегляд конкретного товару;
- Перегляд контактної інформації, та загальної інформації про магазин;
- Кошик з переглядом обраних товарів;
- Пошук товару за деяким ключовим словом;
- Реєстрація в системі;
- Вхід зареєстрованого користувача.

Для перегляду списку усіх товарів користувач має перейти по посиланню в меню «Магазин->Каталог товарів».

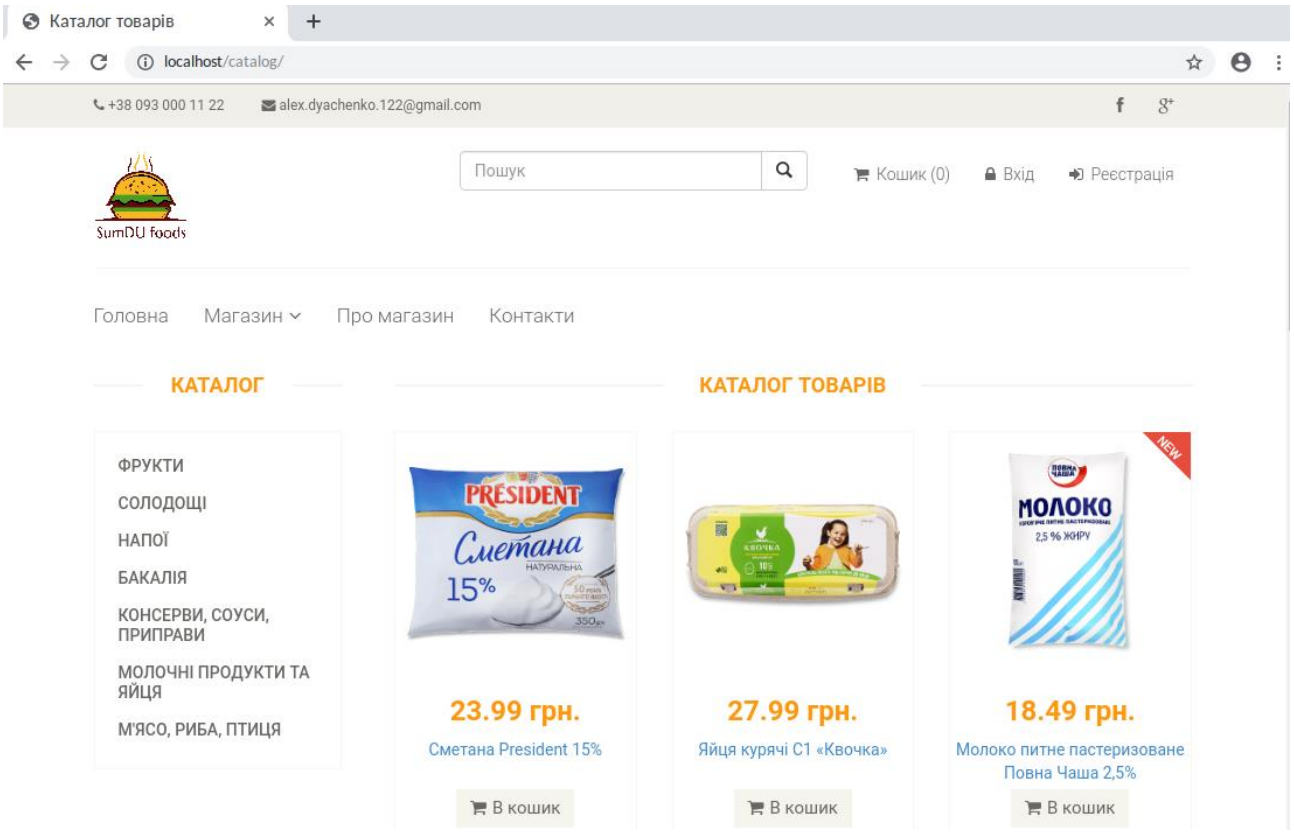


Рисунок 3.3 – Каталог товарів

Для перегляду товарів за деякою категорією користувач має перейти по відповідному посиланню з лівого стовпця сайту. Наприклад, розглянемо список товарів у категорії «Бакалія»:

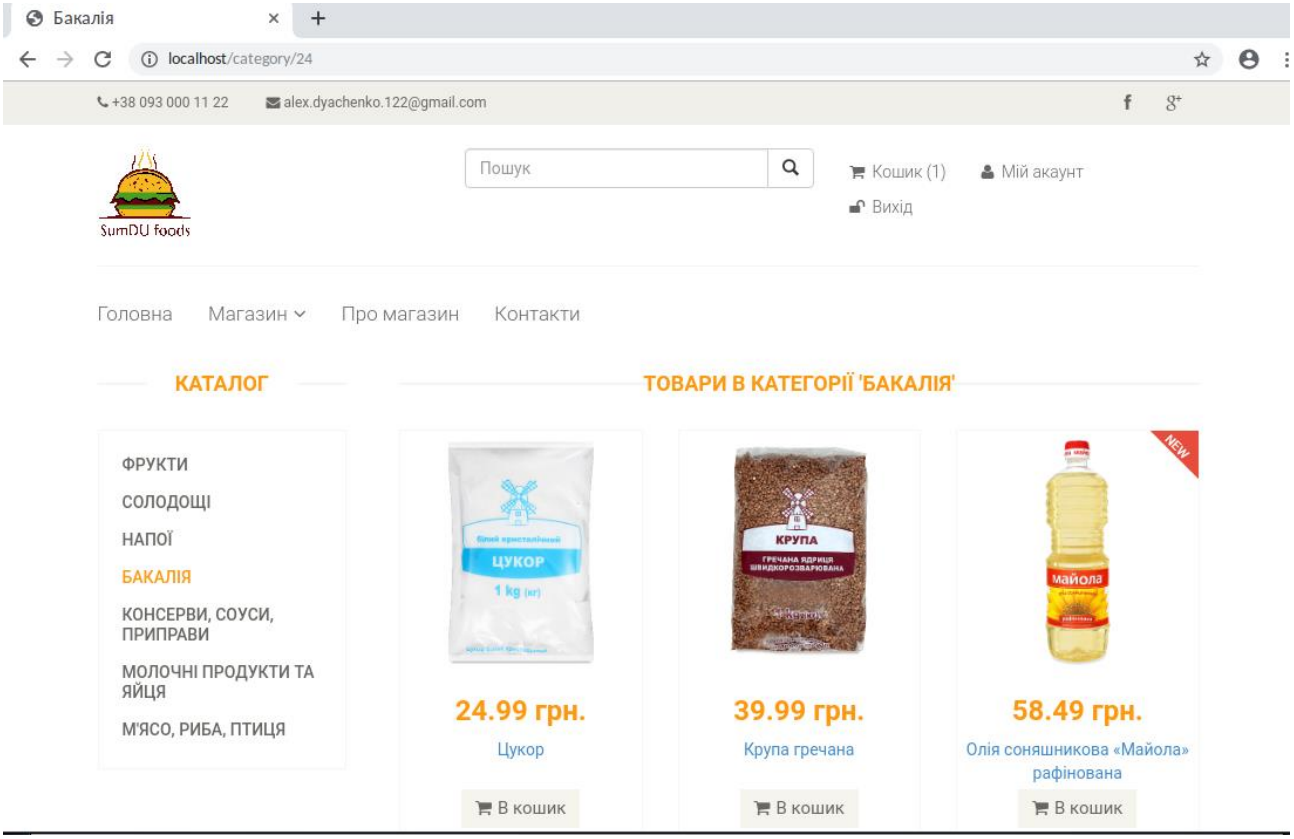


Рисунок 3.4 – Каталог товарів у окремій категорії

Користувач може написати адміністраторові побажання або деяку скаргу. Для цього із шапки сайту потрібно перейти по посиланню «Контакти». У формі, що зявиться слід ввести свій email та текст повідомлення. Після цього треба натиснути кнопку «Відправити». Адміністратор системи отримає повідомлення на свій email.

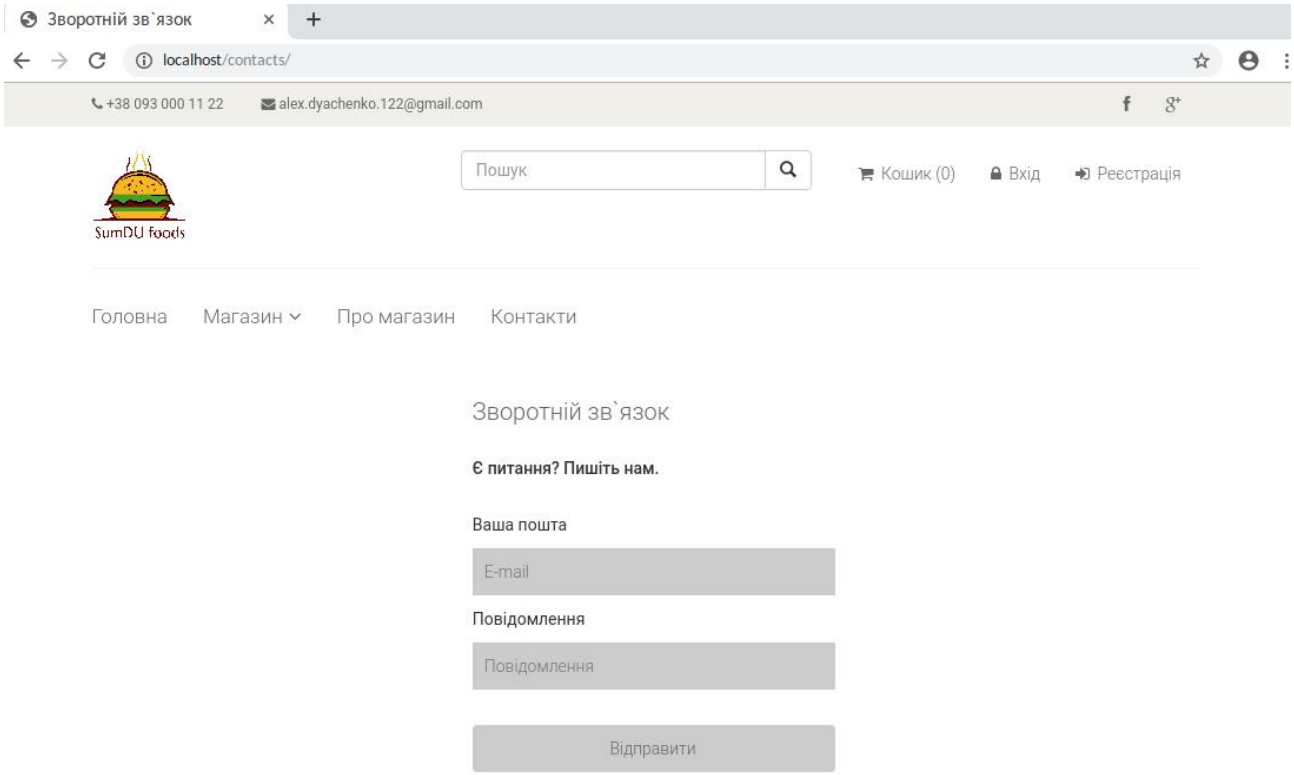


Рисунок 3.5 – Форма відправлення повідомлення

Для перегляду товару треба натиснути на відповідне посилання при перегляді будь-якого із наведених вище списків товарів. Наприклад, розглянемо товар «Сметана President 15%». В цьому режимі користувач може побачити більш масштабне зображення товару, його ціну, детальну інформацію. Також він може додати цей товар у кошик.

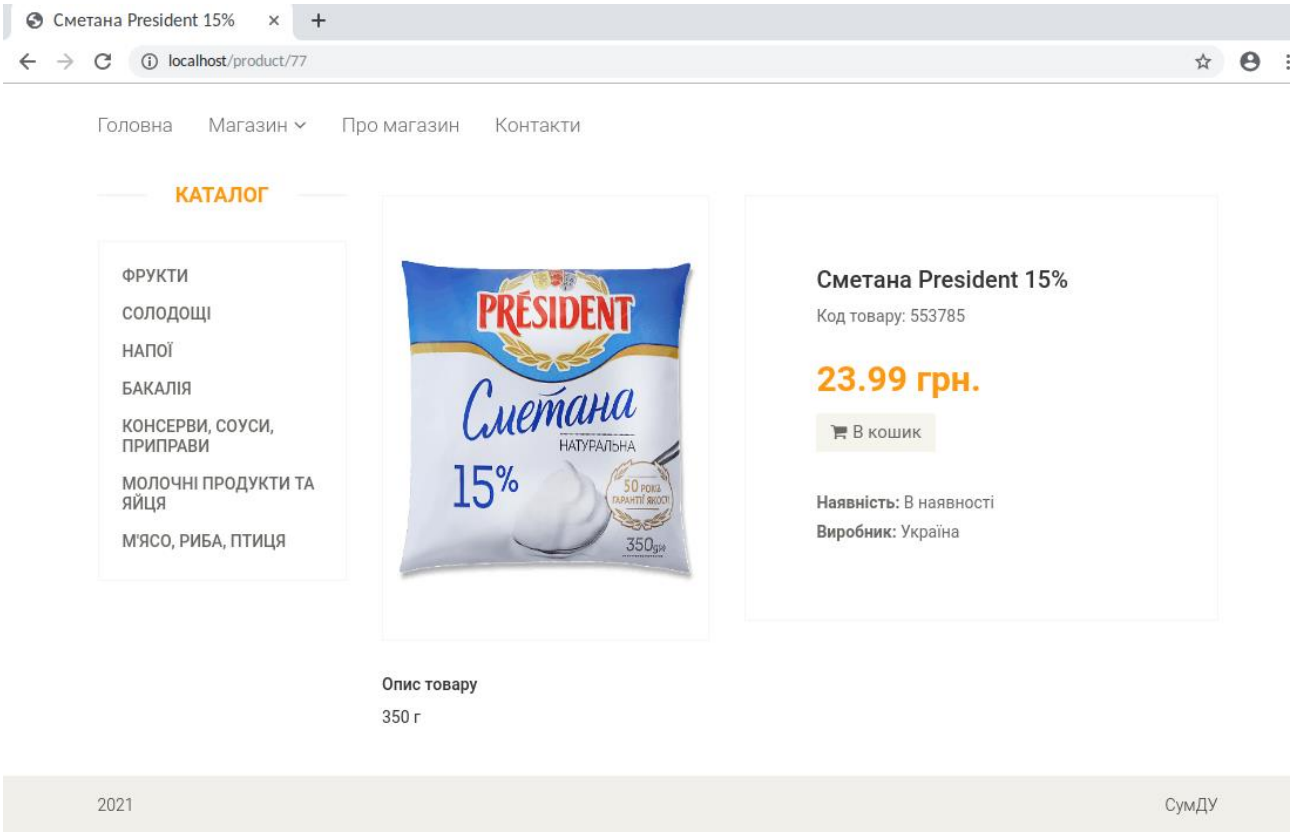


Рисунок 3.6 – Форма відправлення повідомлення

При додаванні товару в кошик, на відповідній кнопці у шапці сайту збільшиться значення лічильнику.

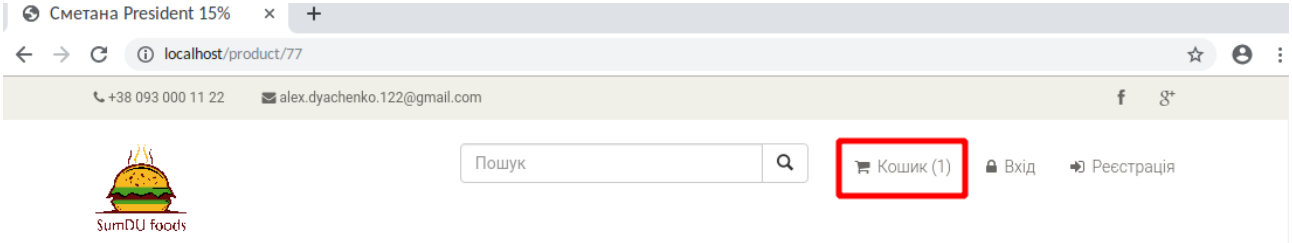


Рисунок 3.7 – Зміна стана кошику додаванні товару

Якщо натиснути на відповідну кнопку, користувач перейде до сторінки перегляду вмісту кошику. На цій сторінки можна побачити список товарних позицій, їхні кількості та загальну вартість замовлення. Також на цій сторінці можна оформити замовлення. Для цього потрібно натиснути відповідну кнопку і, далі, заповнити форму.

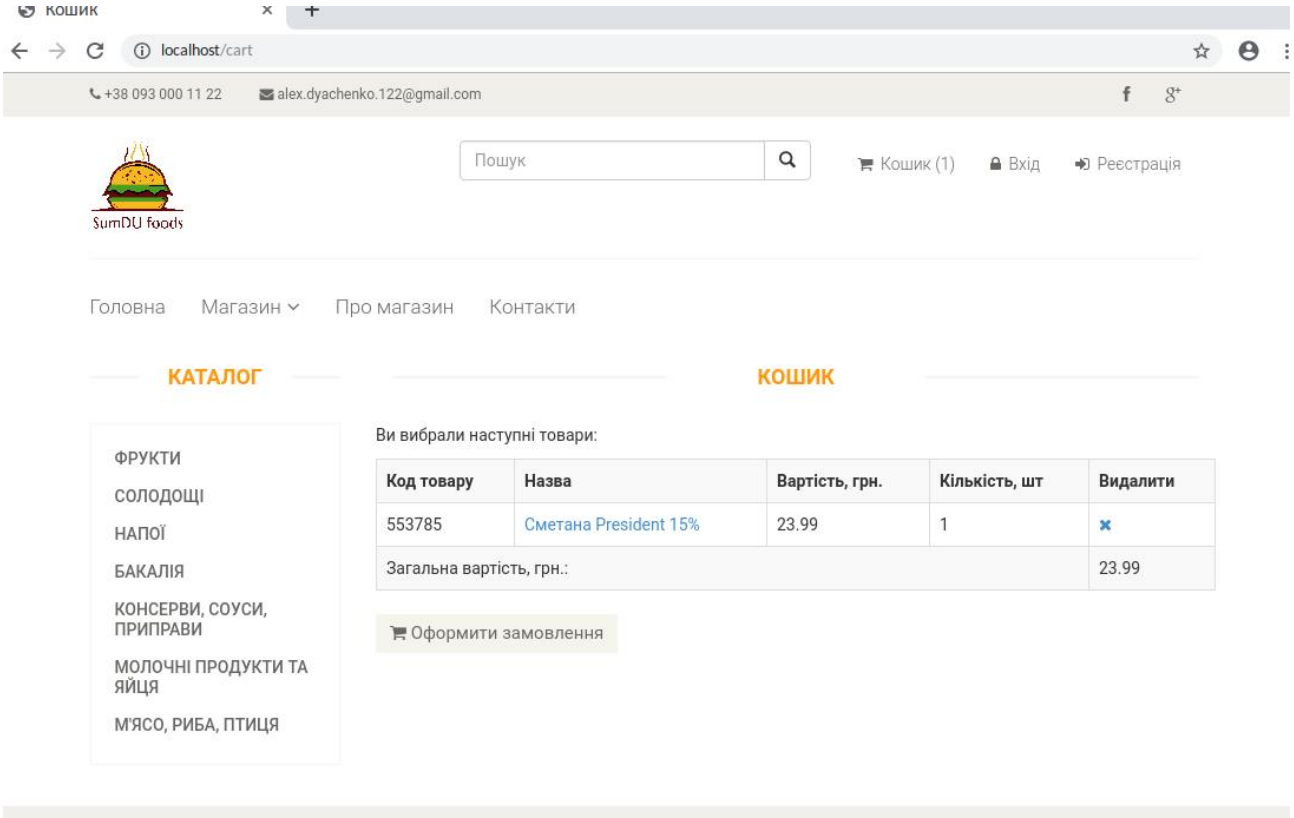


Рисунок 3.8 – Перегляд вмісту кошику

Користувач може шукати товарні позиції за ключовим словом. Для цього у шапці сайту, в пошуковому рядку треба ввести це слово і натиснути на піктограму пошуку, або на клавішу Enter. Наприклад, розглянемо результати пошуку за словом «сметана».

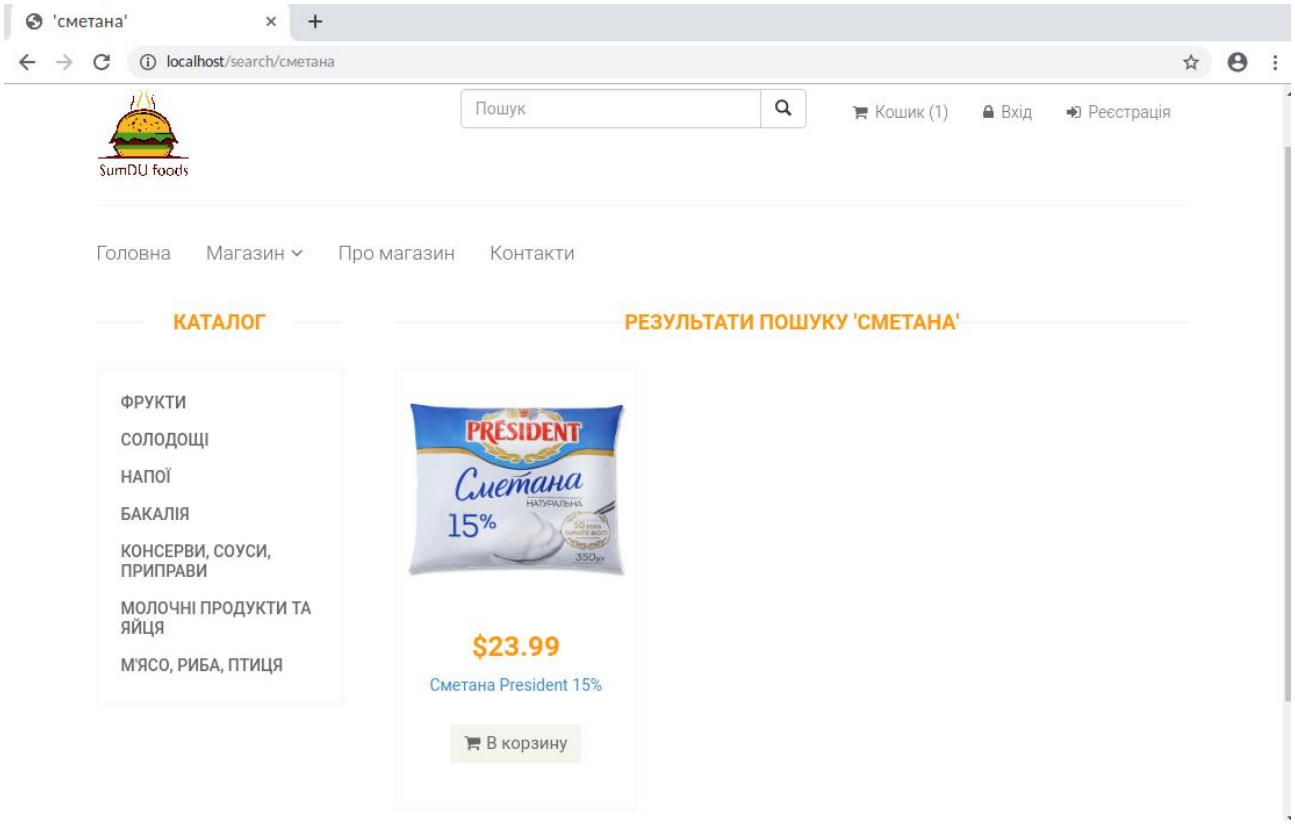


Рисунок 3.9 – Результати пошуку за ключовим словом

Для спрощення процедури оформлення замовлення користувач може зареєструватися в системі. Для цього потрібно перейду по відповідному посиланню у шапці сайту. Після цього користувач має заповнити форму реєстрації і натиснути кнопку «Реєстрація».

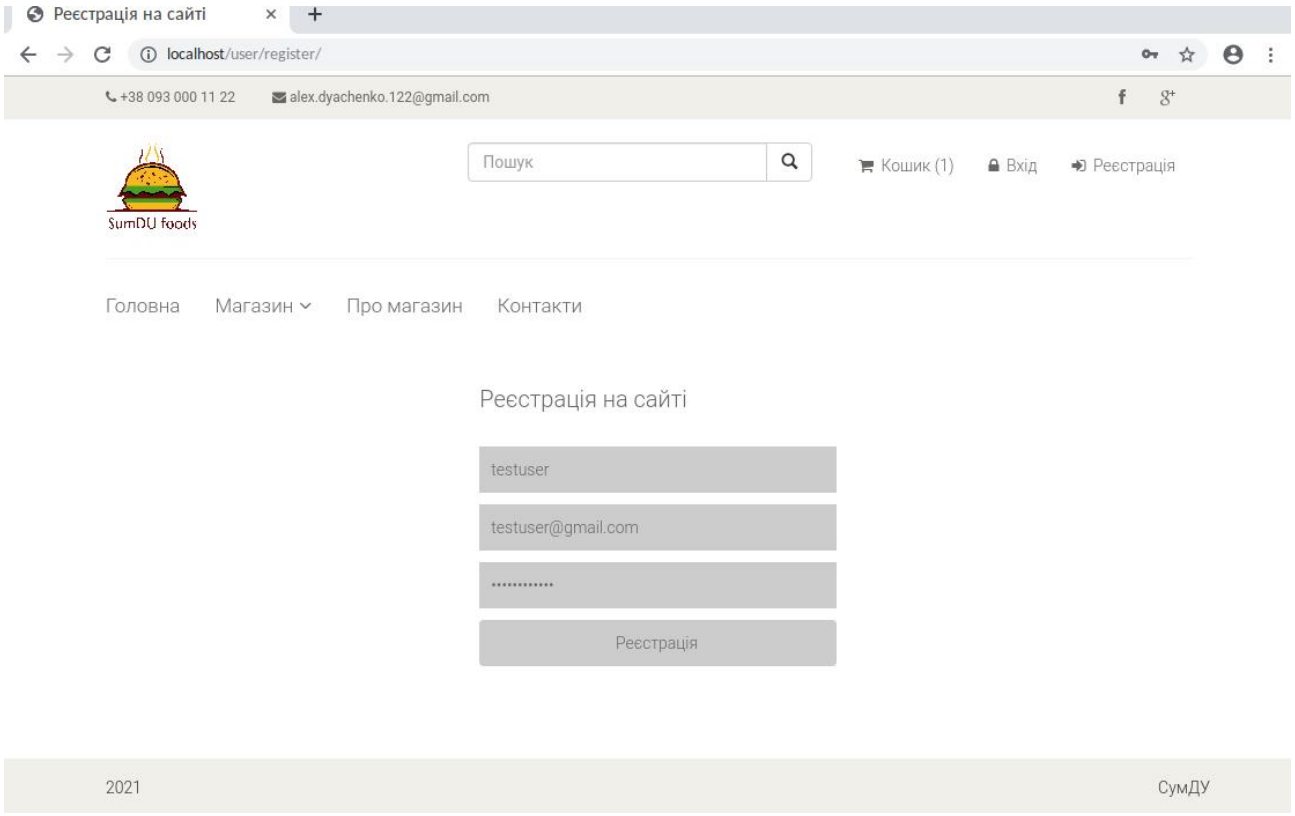


Рисунок 3.10 – Реєстрація нового користувача

Далі зареєстрований користувач зможе зайти в систему. Для цього із шапки сайту потрібно перейти по посиланню «Вхід» і ввести дані у форму.

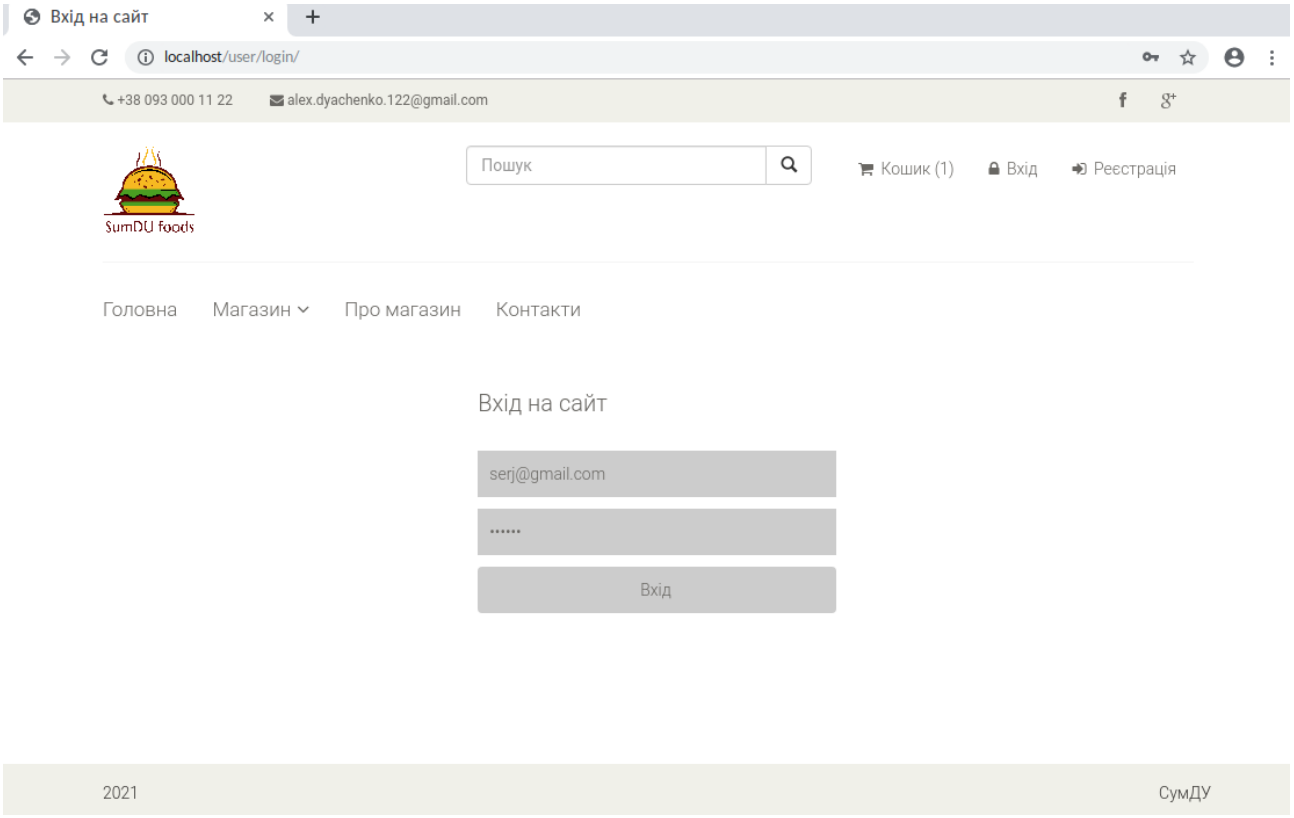


Рисунок 3.11 – Вхід в систему для зареєстрованого користувача

Зареєстрований користувач може відредагувати власні дані. Для цього у шапці потрібно перейти по посиланню «Мій акаунт».

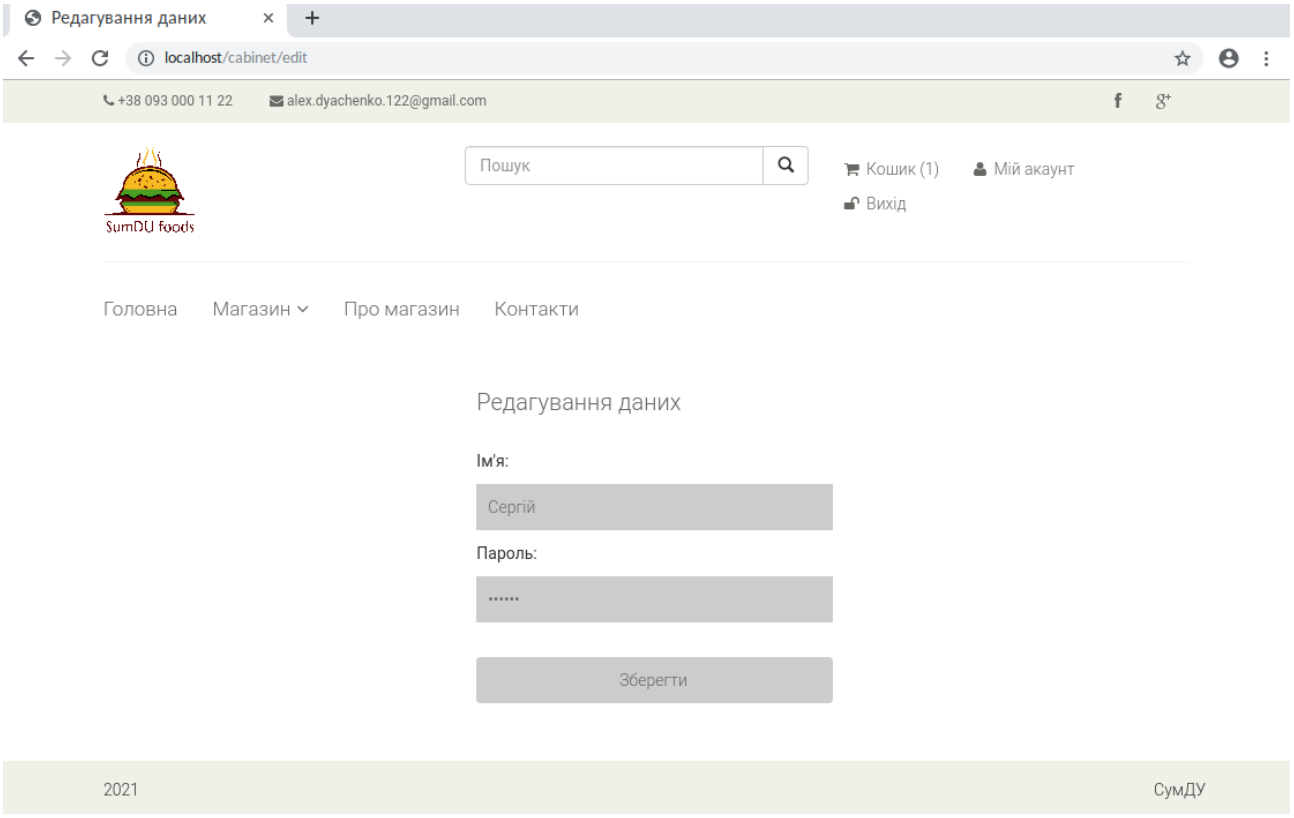


Рисунок 3.12 – Редагування власних даних користувачем

3.4 Інструкція адміністратора

Веб-додаток розроблений із використанням наступного програмного забезпечення:

- Linux з ядром >= 5.8, зокрема Ubuntu Linux >=20.04
- Веб-сервер Apache2
- PHP >= 7
- MySQL >= 14

Припускається, що мається в наявності налаштований та робочій LAMP-сервер (або подібний). Для коректної роботи додатку, в Apache2 має бути підключений модуль mod_rewrite.

Адміністратор має розмістити пакет вихідних кодів у корені сайту і завантажити встановлювальний скрипт у СУБД MySQL. У файлі config/db_params.php необхідно налаштувати параметри зєднання із СУБД:

- host – назва хосту, де встановлена СУБД;
- dbname – назва бази даних;
- user – ім'я користувача;
- password – пароль користувача.

Після цих операцій веб-додаток є працездатним, але в системі не буде користувача із ролью адміністратора. Стандартна реєстрація таких користувачів не передбачена заради безпеки, отже адміністратор має власноручно створити такого користувача через маніпуляції у СУБД. Найпростіший спосіб – зареєструвати звичайного користувача, а далі виконати наступний SQL-запит у контексті бази даних:

UPDATE user WHERE email='test@gmail.com' SET role='admin';

Звичайно, замість test@gmail.com слід використати email, що був вказаний при первинній реєстрації користувача.

Для управління наповненням веб-ресурсу існує спеціальна сторінка адміністратора. Для того, щоб дістатися її, залогіненому користувачеві із роллю адміністратор треба вручну зайти за адресою /admin. Тоді користувач побачить панель адміністратора.

З цієї панелі адміністратор може керувати категоріями товарів, самими товарами та замовленнями, що були зроблені покупцями.

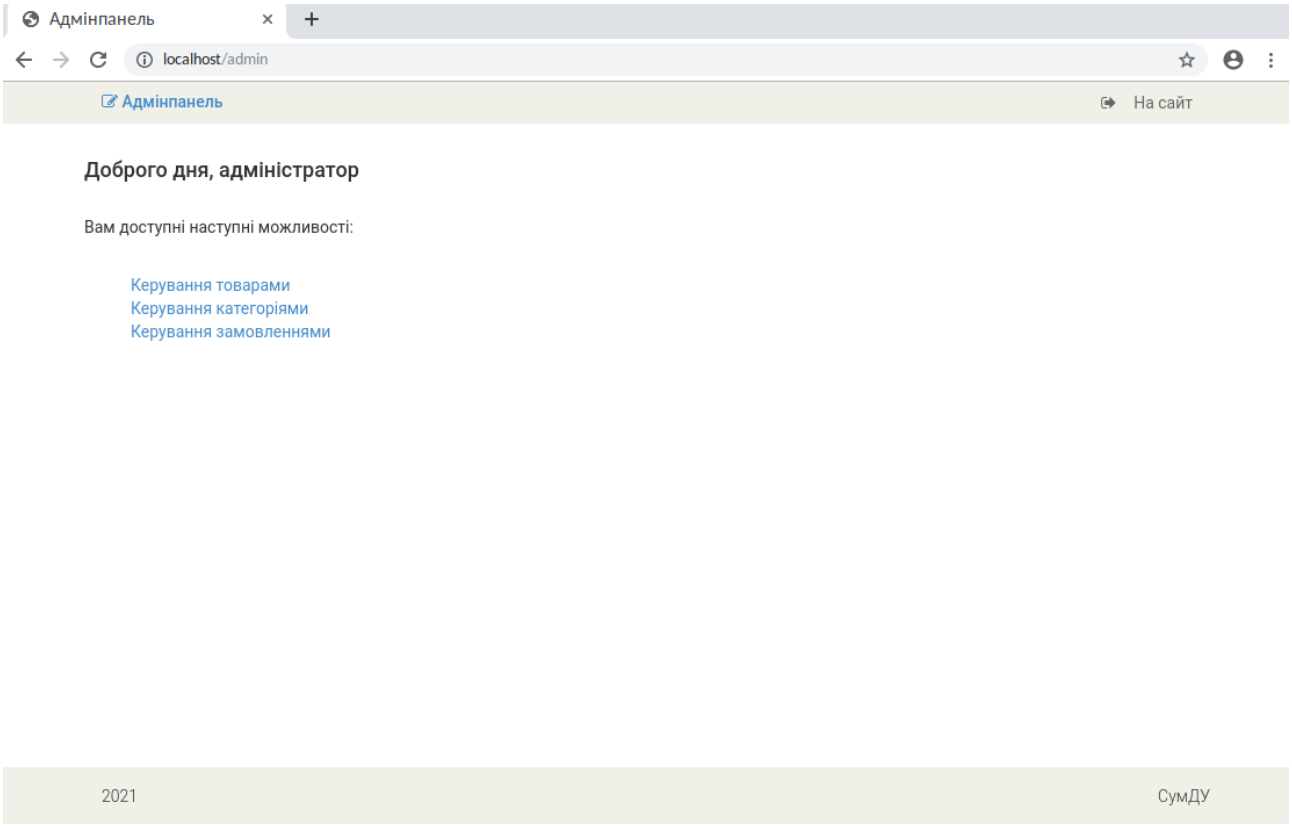


Рисунок 3.13 – Панель адміністратора

Для керування категоріями треба перейти по відповідному посиланню із панелі адміністратора. Тут адміністратор може створювати нові, а також видаляти та редагувати існуючі категорії товарів.

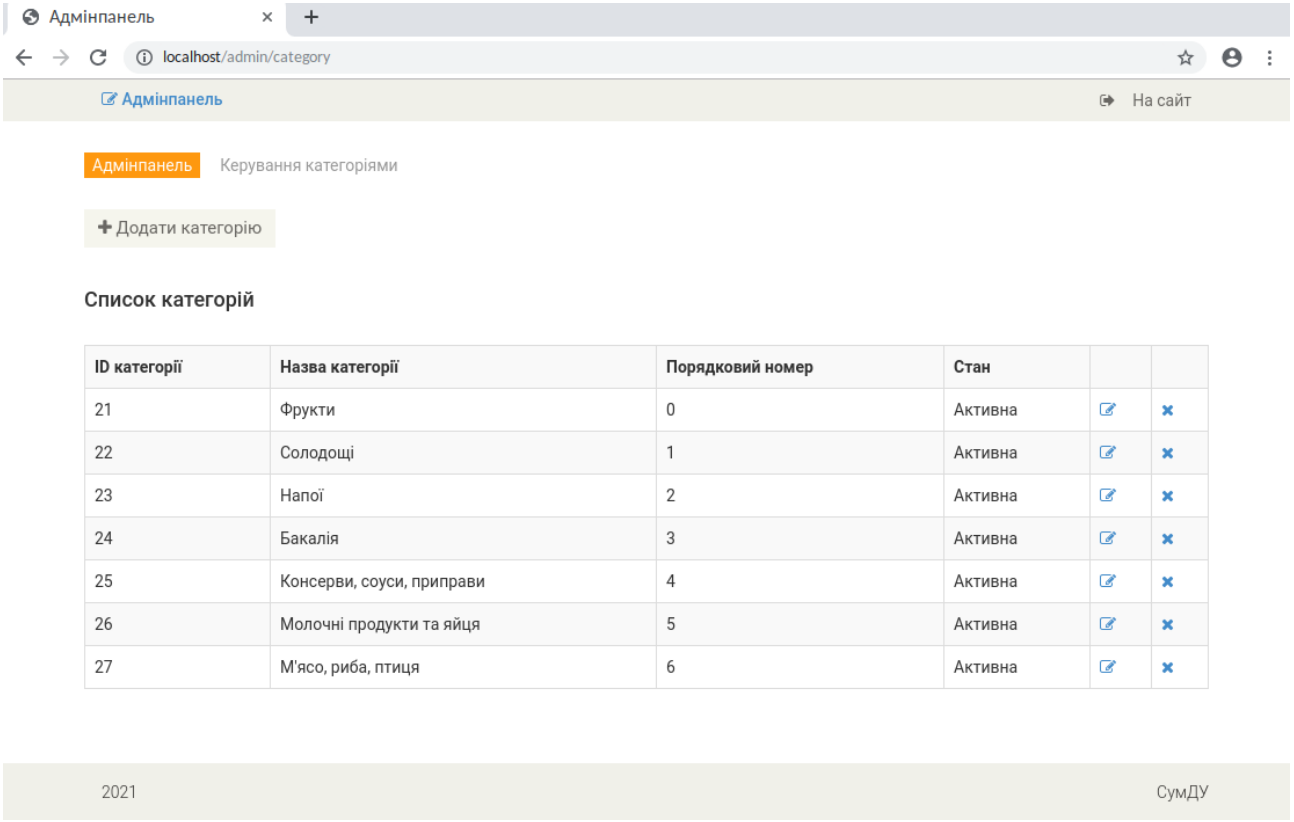
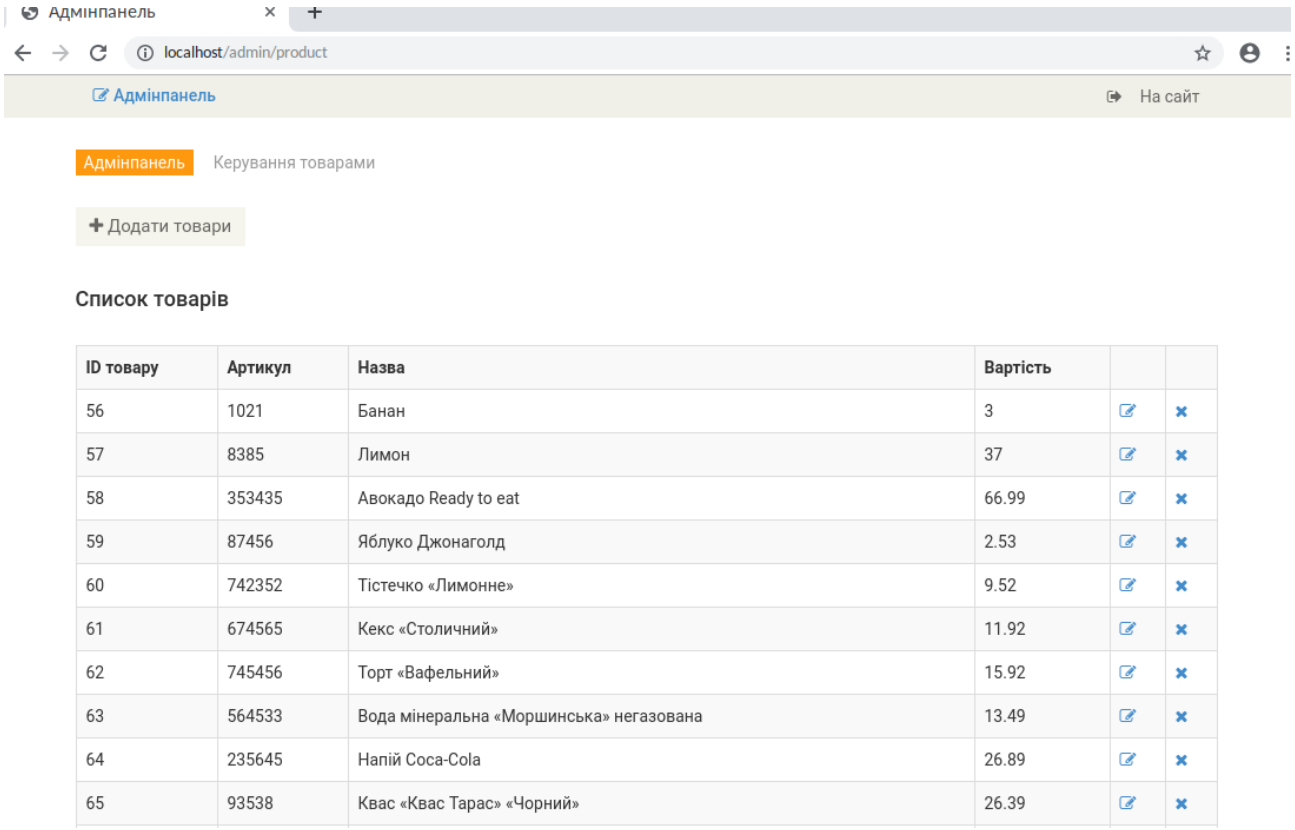


Рисунок 3.14 – Керування категоріями

Таким само чином адміністратор може керувати товарами.



ID товару	Артикул	Назва	Вартість		
56	1021	Банан	3		
57	8385	Лимон	37		
58	353435	Авокадо Ready to eat	66.99		
59	87456	Яблуко Джонаголд	2.53		
60	742352	Тістечко «Лимонне»	9.52		
61	674565	Кекс «Столичний»	11.92		
62	745456	Торт «Вафельний»	15.92		
63	564533	Вода мінеральна «Моршинська» негазована	13.49		
64	235645	Напій Coca-Cola	26.89		
65	93538	Квас «Квас Тарас» «Чорний»	26.39		

Рисунок 3.15 – Керування товарами

При керуванні замовленнями адміністратор може переглянути інформацію по це замовлення і прийняти рішення про його подальший стан.

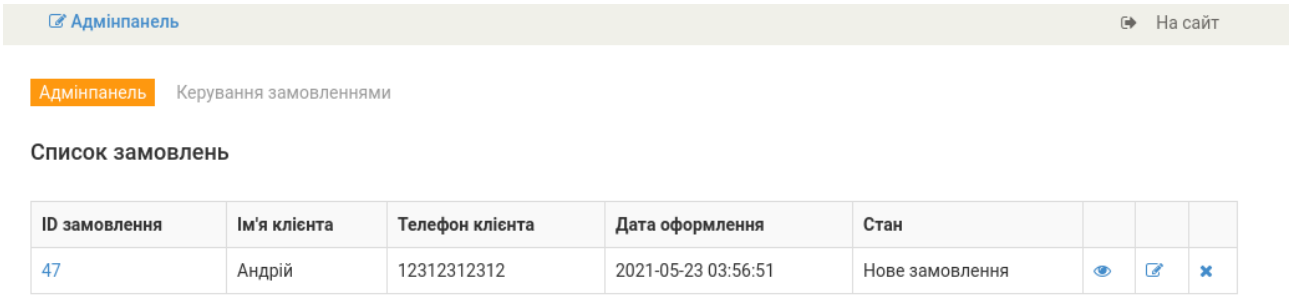


Рисунок 3.16 – Керування замовленнями

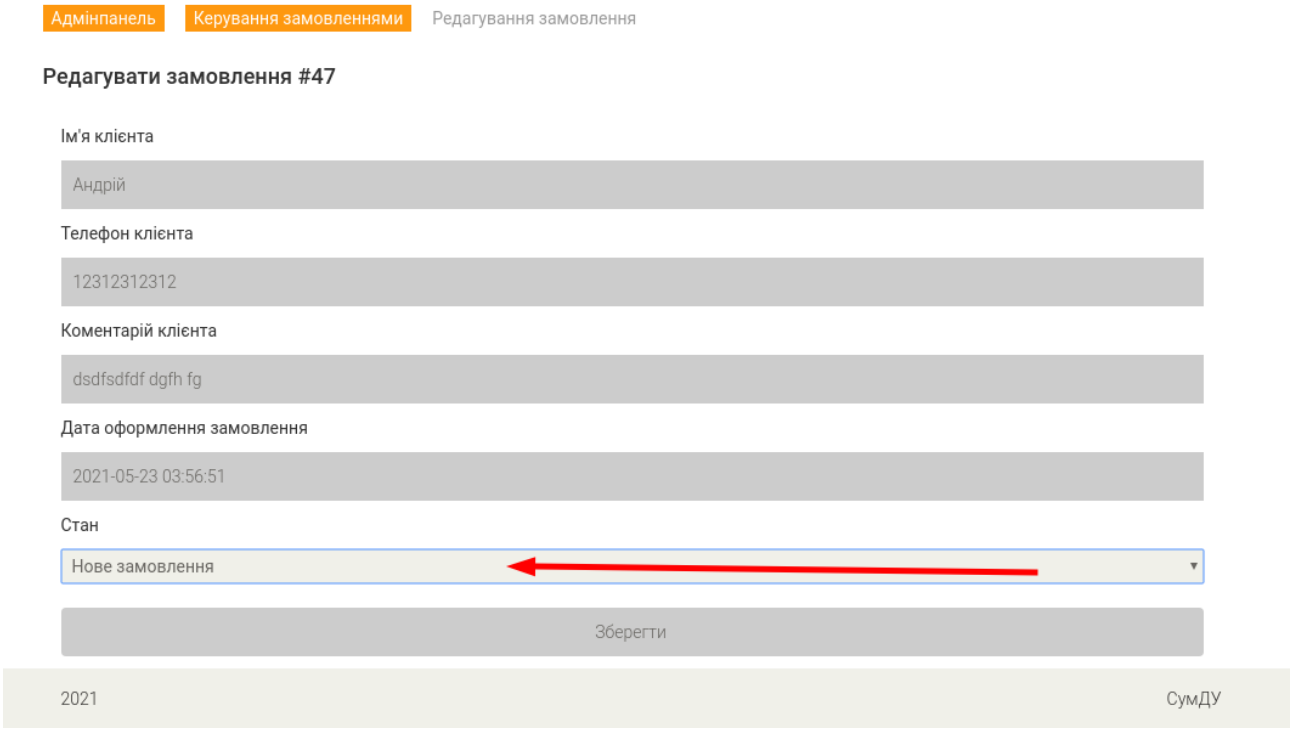


Рисунок 3.17 – Керування станом замовлення

Інші можливості адміністратора аналогічні таковим у звичайного користувача.

ВИСНОВКИ

Інтернет-магазин – це веб-сервіс, який дозволяє торгувати через інтернет. В даній роботі була інформаційно-програмна система інтернет-магазину на прикладі продажу продуктів харчування. В ній був використаний ряд сучасних технологій, а саме: Apache, PHP та Bootstrap.

Спочатку, для отримання уявлення про розглянутий клас систем, ми розглянули ряд існуючих CMS, таких як Joomla, WordPress та 1C Bitrix. Отримавши базові відомості, ми змогли описати вимоги до майбутньої системи та поставити задачу.

Далі, в результаті проектування, ми змогли створити інформаційну модель системи та вибрати відповідну архітектуру. Після цього ми фізично реалізували як базу даних, так і програмне забезпечення інтернет-магазину, користуючись сучасними технологіями.

Виділимо переваги розробленої системи:

- Зручний інтерфейс користувача;
- Можливість переглядати товар без реєстрації;
- Невелике навантаження на системні ресурси

Однак, ми можемо виділити і недоліки цієї розробки:

- Відсутність просунутих фільтрів для товарів;
- Відсутність просунутого пошуку, наприклад з використанням elastic search.

Враховуючи сказане вище, ми можемо прийти до висновку, що розроблене забезпечення можна використовувати для створення невеликого інтернет-магазину. Звісно, з ростом кількості клієнтів доведеться в подібну розробку вносити суттєві зміни. Разом з тим поставлена у проекті задача була виконана.

БІБЛІОГРАФІЯ

1. Юрасов А.В. Электронная коммерция : учебное пособие/ Юрасов А.В. — М.: Дело, 2003. – 482 с.
2. Дослідження: Як змінювалися звички українських інтернет-покупців під час пандемії [інфографіка] // ITCua. URL: https://itc.ua/news/doslidzhennya-yak-zminyuyutsya-zvichki-ukra%D1%97nskih-internet-pokupcziv-pid-chas-pandemi%D1%97/ (дата звернення: 01.06.2021)
3. Пандемия ускорила переход к цифровому миру – исследование ЮНКТАД // Новости ООН. URL: https://news.un.org/ru/story/2020/10/1387842 (дата звернення: 01.06.2021)
4. Описание CMS WordPress // GoldSerfer. URL: http://goldserfer.ru/cms-wordpress/description-of-the-cms-wordpress.html (дата звернення: 02.06.2021)
5. Общая информация и основные особенности CMS Bitrix // CMS Magazine. URL: http://bitrix.cmsmagazine.ru/ (дата звернення: 02.06.2021)
6. Apache HTTP Server // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Apache_HTTP_Server (дата звернення: 03.06.2021)
7. PHP // Вікіпедія. URL: https://uk.wikipedia.org/wiki/PHP (дата звернення: 03.06.2021)
8. MySQL // Вікіпедія. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення: 03.06.2021)
9. Bootstrap // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Bootstrap (дата звернення: 03.06.2021)

ДОДАТОК А

Приклад моделі (лістинг файлу Product.php)

```
<?php

/**
 * Клас Product - модель для роботи з товарами
 */
class Product
{

    // Кількість товарів, що відображаються за замовченням
    const ITEMS_NUMBER = 6;

    // Рядки запитів до бази даних

    const GET_PRODUCT = 'SELECT * FROM product WHERE id = :id';
    const GET_LATEST_PRODUCTS = 'SELECT id, name, price, is_new FROM product WHERE status = "1" ORDER BY id DESC LIMIT :number';
    const GET_PRODUCTS_BY_CATEGORY = 'SELECT id, name, price, is_new FROM product WHERE status = 1 AND category_id = :cat_id ORDER BY id ASC LIMIT :max_pages OFFSET :offset';
    const SEARCH_PRODUCTS = 'SELECT id, name, price, is_new FROM product WHERE status = 1 AND name LIKE CONCAT("%",:search_query,"%") ORDER BY id ASC LIMIT :max_pages OFFSET :page_num_offset';

    const GET_TOTAL_PRODUCTS = 'SELECT count(id) AS count FROM product WHERE status="1" AND category_id = :cat_id';
    const GET_TOTAL_PRODUCTS_BY_QUERY = 'SELECT count(id) AS count FROM product WHERE status="1" AND name LIKE CONCAT("%",:search_query,"%")';
    const GET_PRODUCTS_BY_IDS = "SELECT * FROM product WHERE status='1' AND id IN ($id_row)";
    const GET_RECOMMENDED = 'SELECT id, name, price, is_new FROM product WHERE status = "1" AND is_recommended = "1" ORDER BY id DESC';
    const GET_PRODUCTS = 'SELECT id, name, price, code FROM product ORDER BY id ASC';
    const DELETE_PRODUCT = 'DELETE FROM product WHERE id = :id';

    const UPDATE_PRODUCT = "UPDATE product SET
            name = :name,
            code = :code,
            price = :price,
            category_id = :cat_id,
            brand = :brand,
            availability = :availability,
            description = :description,
            is_new = :is_new,
            is_recommended = :is_recommended,
            status = :status
            WHERE id = :id";
```

```
    const CREATE_PRODUCT = 'INSERT INTO product (name, code, price, category_id, brand,
availability, description, is_new, is_recommended, status) VALUES (:name, :code, :price, :cat_id,
:brand, :availability, :description, :is_new, :is_recommended, :status)';



    /**
     * Повертає масив останніх товарів
     * @param type $number [optional] <p>Кількість</p>
     * @param type $page_num [optional] <p>Номер поточної сторінки</p>
     * @return array <p>Масив товарів</p>
     */
    public static function getLatestProducts($number = self::ITEMS_NUMBER)
    {
        // З'єднання з базою даних
        $conn = Db::getConnection();

        // Використовуємо підготовлений запит
        $res = $conn->prepare(self::GET_LATEST_PRODUCTS);
        $res->bindParam(':number', $number, PDO::PARAM_INT);
        $res->setFetchMode(PDO::FETCH_ASSOC);
        $res->execute();

        // Отримання та повернення результатів
        return fetchRows($res, array('id', 'name', 'price', 'is_new'));
    }

    /**
     * Повертає список товарів у вказаній категорії
     * @param type $cat_id <p>id категорії</p>
     * @param type $page_num [optional] <p>Номер сторінки</p>
     * @return type <p>Масив з товарів</p>
     */
    public static function getProductsListByCategoryId($cat_id, $page_num = 1)
    {
        $max_pages = Product::ITEMS_NUMBER;
        // Зсув сторінок
        $page_num_offset = ($page_num - 1) * self::ITEMS_NUMBER;

        // З'єднання з базою даних
        $conn = Db::getConnection();

        // Використовуємо підготовлений запит
        $res = $conn->prepare(self::GET_PRODUCTS_BY_CATEGORY);
        $res->bindParam(':cat_id', $cat_id, PDO::PARAM_INT);
        $res->bindParam(':max_pages', $max_pages, PDO::PARAM_INT);
        $res->bindParam(':page_num_offset', $page_num_offset, PDO::PARAM_INT);
        $res->execute();

        return fetchRows($res, array('id', 'name', 'price', 'is_new'));
    }
```

```
/**
 * Повертає список товарів, назви яких відповідають критерію пошуку
 * @param type $cat_id <p>id категорії</p>
 * @param type $page_num [optional] <p>Номер сторінки</p>
 * @return type <p>Масив з товарами</p>
 */
public static function searchProductsByQuery($searchQuery, $page_num = 1)
{
    $max_pages = Product::ITEMS_NUMBER;
    // Зсув сторінок
    $page_num_offset = ($page_num - 1) * self::ITEMS_NUMBER;

    // З'єднання з базою даних
    $conn = Db::getConnection();

    // Використовуємо підготовлений запит
    $res = $conn->prepare(self::SEARCH_PRODUCTS);
    $res->bindParam(':search_query', $searchQuery, PDO::PARAM_STR);
    $res->bindParam(':max_pages', $max_pages, PDO::PARAM_INT);
    $res->bindParam(':page_num_offset', $page_num_offset, PDO::PARAM_INT);
    $res->execute();

    return fetchRows($res, array('id', 'name', 'price', 'is_new'));
}

/**
 * Повертає товар із вказаним id
 * @param integer $id <p>id товару</p>
 * @return array <p>Масив з відомостями про товар</p>
 */
public static function getProduct($id)
{
    // З'єднання з базою даних
    $conn = Db::getConnection();

    // Використовуємо підготовлений запит
    $res = $conn->prepare(self::GET_PRODUCT);
    $res->bindParam(':id', $id, PDO::PARAM_INT);
    $res->setFetchMode(PDO::FETCH_ASSOC);

    // Виконання команди
    $res->execute();

    // Отримання та повернення результатів
    return $res->fetch();
}

/**
 * Повертає кількість товарів у вказаній категорії
 * @param integer $cat_id
```

```
     * @return integer
     */
    public static function getTotalProductsInCategoryId($cat_id)
    {
        // З'єднання з базою даних
        $conn = Db::getConnection();

        $res = $conn->prepare(self::GET_TOTAL_PRODUCTS);
        $res->bindParam(':cat_id', $cat_id, PDO::PARAM_INT);

        // Виконання команди
        $res->execute();

        // Повертаємо значення count - кількість
        $row = $res->fetch();
        return $row['count'];
    }

    /**
     * Повертає кількість товарів, що відповідають критерію пошуку
     * @param integer $cat_id
     * @return integer
     */
    public static function getTotalProductsByQuery($searchQuery)
    {
        // З'єднання з базою даних
        $conn = Db::getConnection();

        $res = $conn->prepare(self::GET_TOTAL_PRODUCTS_BY_QUERY);
        $res->bindParam(':search_query', $searchQuery, PDO::PARAM_STR);

        // Виконання команди
        $res->execute();

        // Повертаємо значення count - кількість
        $row = $res->fetch();
        return $row['count'];
    }

    /**
     * Повертає список товарів із вказаними ідентифікаторами
     * @param array $id_list <p>Масив з ідентифікаторами</p>
     * @return array <p>Масив з товарами</p>
     */
    public static function getProducts($id_list)
    {
        // З'єднання з базою даних
        $conn = Db::getConnection();

        // Формуємо рядок із масиву
        $id_row = implode(',', $id_list);
```

```
    $res = $conn->query(self::GET_PRODUCTS_BY_IDS);

    // Вказуємо, що хочемо отримати дані у вигляді масиву
    $res->setFetchMode(PDO::FETCH_ASSOC);

    // Отримання та повернення результатів
    return fetchRows($res, array('id', 'code', 'name', 'price'));
}

/**
 * Повертає список товарів, що рекомендуються
 * @return array <p>Масив з товарами</p>
 */
public static function getRecommended()
{
    // З'єднання з базою даних
    $conn = Db::getConnection();

    // Отримання та повернення результатів
    $res = $conn->query(self::GET_RECOMMENDED);

    return fetchRows($res, array('id', 'name', 'price', 'is_new'));
}

/**
 * Повертає список товарів
 * @return array <p>Масив з товарами</p>
 */
public static function getProductsList()
{
    // З'єднання з базою даних
    $conn = Db::getConnection();

    // Отримання та повернення результатів
    $res = $conn->query(self::GET_PRODUCTS);
    return fetchRows($res, array('id', 'name', 'code', 'price'));
}

/**
 * Видаляє товар із вказаним id
 * @param integer $id <p>id товару</p>
 * @return boolean <p>Результат виконання методу</p>
 */
public static function deleteProduct($id)
{
    // З'єднання з базою даних
    $conn = Db::getConnection();

    // Отримання та повернення результатів. Використовуємо підготовлений запит
    $res = $conn->prepare(self::DELETE_PRODUCT);
```

```
        $res->bindParam(':id', $id, PDO::PARAM_INT);
        return $res->execute();
    }

    /**
     * Редагує товар із заданим id
     * @param integer $id <p>id товару</p>
     * @param array $product_attributes <p>Масив з інформацією про товар</p>
     * @return boolean <p>Результат виконання методу</p>
     */
    public static function updateProduct($id, $product_attributes)
    {
        // З'єднання з базою даних
        $conn = Db::getConnection();

        // Отримання та повернення результатів. Використовуємо підготовлений запит
        $res = $conn->prepare(self::UPDATE_PRODUCT);
        $res->bindParam(':id', $id, PDO::PARAM_INT);
        $res->bindParam(':name', $product_attributes['name'], PDO::PARAM_STR);
        $res->bindParam(':code', $product_attributes['code'], PDO::PARAM_STR);
        $res->bindParam(':price', $product_attributes['price'], PDO::PARAM_STR);
        $res->bindParam(':category_id', $product_attributes['cat_id'], PDO::PARAM_INT);
        $res->bindParam(':brand', $product_attributes['brand'], PDO::PARAM_STR);
        $res->bindParam(':availability', $product_attributes['availability'], PDO::PARAM_INT);
        $res->bindParam(':description', $product_attributes['description'], PDO::PARAM_STR);
        $res->bindParam(':is_new', $product_attributes['is_new'], PDO::PARAM_INT);
        $res->bindParam(':is_recommended', $product_attributes['is_recommended'],
PDO::PARAM_INT);
        $res->bindParam(':status', $product_attributes['status'], PDO::PARAM_INT);
        return $res->execute();
    }

    /**
     * Додає новий товар
     * @param array $product_attributes <p>Масив з інформацією про товар</p>
     * @return integer <p>id доданого в таблицю запису</p>
     */
    public static function createProduct($product_attributes)
    {
        // З'єднання з базою даних
        $conn = Db::getConnection();

        // Отримання та повернення результатів. Використовуємо підготовлений запит
        $res = $conn->prepare(self::CREATE_PRODUCT);
        $res->bindParam(':name', $product_attributes['name'], PDO::PARAM_STR);
        $res->bindParam(':code', $product_attributes['code'], PDO::PARAM_STR);
        $res->bindParam(':price', $product_attributes['price'], PDO::PARAM_STR);
        $res->bindParam(':cat_id', $product_attributes['cat_id'], PDO::PARAM_INT);
        $res->bindParam(':brand', $product_attributes['brand'], PDO::PARAM_STR);
        $res->bindParam(':availability', $product_attributes['availability'], PDO::PARAM_INT);
        $res->bindParam(':description', $product_attributes['description'], PDO::PARAM_STR);
```

```
        $res->bindParam(':is_new', $product_attributes['is_new'], PDO::PARAM_INT);
        $res->bindParam(':is_recommended', $product_attributes['is_recommended'],
PDO::PARAM_INT);
        $res->bindParam(':status', $product_attributes['status'], PDO::PARAM_INT);
        if ($res->execute()) {
            // Якщо запит виконаний успішно, повертаємо id доданого запису
            return $conn->lastInsertId();
        }
        // Інакше повертаємо 0
        return 0;
    }

    /**
     * Повертає текстове пояснення наявності товару:<br/>
     * <i>0 - Немає, 1 - В наявності</i>
     * @param integer $availability <p>Стан</p>
     * @return string <p>Текстове пояснення</p>
     */
    public static function getAvailabilityText($is_available)
    {
        if ($is_available === '1') {
            return 'В наявності';
        } else {
            return 'Немає';
        }
    }

    /**
     * Повертає шлях до зображення
     * @param integer $id
     * @return string <p>Шлях до зображення</p>
     */
    public static function getImageByProductId($id)
    {
        // Назва пустого зображення
        $empty_image = 'no-image.jpg';

        // Шлях до папки із товарами
        $uploadFolderPath = '/upload/images/products/';

        // Шлях до зображення товару
        $pathToImageFile = $uploadFolderPath . $id . '.jpg';

        if (file_exists($_SERVER['DOCUMENT_ROOT'] . $pathToImageFile)) {
            // Якщо зображення існує
            // Повертаємо шлях до зображення товару
            return $pathToImageFile;
        }

        // Інакше повертаємо пусте зображення
        return $uploadFolderPath . $empty_image;
```

```
    }

    private static function fetchRows($res, $fields)
    {
        $result = array();
        $i = 0;
        while ($row = $res->fetch()) {
            foreach ($fields as $field) {
                $result[$i][$field] = $row[$field];
            }
            $i++;
        }
        return $result;
    }

}
```

Приклад контролеру (лістинг файлу Product.php)

```
<?php


/**
 * Дії з товаром
 */


class ProductController
{


    /**
     * Сторінка з відомостями про товар
     * @param integer $productId <p>id товару</p>
     */
    public function actionView($id)
    {
        // Категорії бокового меню
        $cats = Category::getCategories();


        // Інформація про конкретний товар
        $product = Product::getProductById($id);
```

```
        // Заповнюємо поля у шаблоні вигляду
        require_once(ROOT . '/views/product/view.php');

        // Повертаємо керування веб-серверу
        return true;
    }
}
```

Приклад вигляду (лістинг файлу /view/search/Search.php):

```
<?php $searchString = """ . $searchQuery ."""; ?>
<?php $page_title = $searchString; ?>
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
   <div class="container">
      <div class="row">

         <?php include ROOT . '/views/layouts/left_sidebar.php'; ?>

         <div class="col-sm-9 padding-right">
            <div class="features_items">
               <h2 class="title text-center"><?php echo 'Результати пошуку ' . $searchString;
?></h2>

               <?php if ($total > 0): ?>
                  <?php foreach ($foundProducts as $product): ?>
                     <div class="col-sm-4">
                        <div class="product-image-wrapper">
                           <div class="single-products">
                              <div class="productinfo text-center">
                                 <img src="<?php echo Product::getImage($product['id']); ?>"
alt="" />
                                 <h2>$<?php echo $product['price']; ?></h2>
                                 <p>
                                    <a href="/product/<?php echo $product['id']; ?>">
                                       <?php echo $product['name']; ?>
                                    </a>
                                 </p>
                                 <a href="/cart/add/<?php echo $product['id']; ?>" class="btn btn-
default add-to-cart" data-id="<?php echo $product['id']; ?>"><i class="fa fa-shopping-
cart"></i>В корзину</a>
                              </div>
                              <?php if ($product['is_new']): ?>
                                 <img src="/template/images/home/new.png" class="new" alt=""
/>
                              <?php endif; ?>
                           </div>
                        </div>
                     </div>
                  <?php endforeach; ?>
               <?php else: ?>
                  <?php http_response_code(404); ?>
                  <p>Нажаль, за запитом "<?php echo $searchQuery; ?>" нічого не
знайдено.</p>
               <?php endif; ?>
            </div>

            <?php echo $pages->get(); ?>
```

```
                </div>
            </div>
        </div>
    </section>

    <?php include ROOT . '/views/layouts/footer.php'; ?>
```

Приклад компоненту (лістинг файлу Router.php):

```
<?php

/**
 * Клас Router
 * Компонент для роботи із маршрутами
 */
class Router
{
    // Файл маршрутів
    const ROUTE_LIST_PATH = ROOT . '/config/routes.php';

    /**
     * Масив маршрутів
     * @var array
     */
    private $routeList;
    public function __construct()
    {
        // Отримуємо маршрути із файлу
        $this->routeList = include($self::ROUTE_LIST_PATH);
    }

    private function clearURI()
    {
        $URI = $_SERVER['REQUEST_URI'];
        if (!empty($URI)) {
            return trim($URI, '/');
        }
    }

    /**
     * Метод для обробки запиту
     */
    public function run()
    {
        // Отримуємо рядок запиту
        $uri = $this->clearURI();

        // Шукаємо шаблон до заданого URI
        foreach ($this->routeList as $pattern => $path) {

            // Порівнюємо шаблон та URI
```

```
            if (preg_match("~$pattern~", $uri)) {

                // Визначаємо внутрішній шлях
                $internal = preg_replace("~$pattern~", $path, $uri);

                // Визначаємо контролер, дію, параметри

                $params = explode('/', $internal);
                $temp = array_shift($params);
                $controllerClass = ucfirst($temp . 'Controller');

                $action = 'action' . ucfirst($temp);

                $parameters = $params;

                // Завантажуємо контролер
                $controllerFile = ROOT . '/controllers/' . $controllerClass . '.php';


                if (file_exists($controllerFile)) {
                    include_once($controllerFile);
                }

                // Створюємо об'єкт так викликаємо метод-action
                $instance = new $controllerClass;

                /* Викликаємо необхідний метод у визначеного класу із заданими параметрами
                 */
                $res = call_user_func_array(array($instance, $action), $parameters);

                if ($res != null) {
                    break;
                }
            }
        }
    }

}
```

ДОДАТОК Б

Лістинг дампу бази даних:

```
DROP TABLE IF EXISTS `category`;
CREATE TABLE `category` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `sort_order` int(11) NOT NULL DEFAULT '0',
  `status` int(11) NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=28 DEFAULT CHARSET=utf8;

LOCK TABLES `category` WRITE;
INSERT INTO `category` VALUES
(21,'Фрукти',0,1),(22,'Солодощі',1,1),(23,'Напої',2,1),(24,'Бакалія',3,1),(25,'Консерви, соуси,
приправи',4,1),(26,'Молочні продукти та яйця',5,1),(27,'М\'ясо, риба, птиця',6,1);
UNLOCK TABLES;


DROP TABLE IF EXISTS `product`;
CREATE TABLE `product` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `category_id` int(11) NOT NULL,
  `code` int(11) NOT NULL,
  `price` float NOT NULL,
  `availability` int(11) NOT NULL,
  `brand` varchar(255) NOT NULL,
  `description` text NOT NULL,
  `is_new` int(11) NOT NULL DEFAULT '0',
  `is_recommended` int(11) NOT NULL DEFAULT '0',
  `status` int(11) NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=78 DEFAULT CHARSET=utf8;


LOCK TABLES `product` WRITE;
INSERT INTO `product` VALUES (56,'Банан',21,1021,3,1,'Еквадор','100
г',1,1,1),(57,'Лимон',21,8385,37,1,'Туреччина','1 кг',0,1,1),(58,'Авокадо Ready to
eat',21,353435,66.99,1,'Кенія','2шт/уп',1,1,1),(59,'Яблуко
Джонаголд',21,87456,2.53,1,'Україна','100 г',0,1,1),(60,'Тістечко
«Лимонне»',22,742352,9.52,1,'Україна','100 г',0,0,1),(61,'Кекс
«Столичний»',22,674565,11.92,1,'Україна','100 г',1,1,1),(62,'Торт
«Вафельний»',22,745456,15.92,1,'Україна','100 г',0,0,1),(63,'Вода мінеральна «Моршинська»
негазована',23,564533,13.49,1,'Україна','1.5 л',0,1,1),(64,'Напій Coca-
Cola',23,235645,26.89,1,'Україна','2 л',0,1,1),(65,'Квас «Квас Тарас»
«Чорний»',23,93538,26.39,1,'Україна','1.5 л',0,1,1),(66,'Цукор',24,7979345,24.99,1,'Україна','1
кг',0,0,1),(67,'Крупа гречана',24,3007654,39.99,1,'Україна','1 кг',0,1,1),(68,'Олія соняшникова
«Майола» рафінована',24,547456,58.49,1,'Україна','850 мг',1,1,1),(69,'Паста «Чумак» томатна,
```

```
пірамідка',25,685667,5.44,1,'Україна','70 г',0,1,1),(70,'Перець «Мрія» чорний
мелений',25,346456,4.99,1,'Україна','20 г',0,1,1),(71,'Кукурудза «Повна Чаша»® цукрова
консервована',25,95434,19.99,1,'Україна','420 г',1,1,1),(72,'Куряче
філе',27,74456,11.41,1,'Україна','100 г',0,1,1),(73,'Оселедець
слабосолений',27,74565,5.91,1,'Норвегія','100 г',1,1,1),(74,'Свинина, корейка на кістці
охолоджена',27,9433,13.92,1,'Україна','100 г',0,1,1),(75,'Молоко питне пастеризоване Повна
Чаша 2,5%',26,767567,18.49,1,'Україна','900 г',1,1,1),(76,'Яйця курячі С1
«Квочка»',26,777334,27.99,1,'Україна','10шт/уп',0,1,1),(77,'Сметана President
15%',26,553785,23.99,1,'Україна','350 г',0,1,1);
        UNLOCK TABLES;


        DROP TABLE IF EXISTS `product_order`;
        CREATE TABLE `product_order` (
          `id` int(11) NOT NULL AUTO_INCREMENT,
          `user_name` varchar(255) NOT NULL,
          `user_phone` varchar(255) NOT NULL,
          `user_comment` text NOT NULL,
          `user_id` int(11) DEFAULT NULL,
          `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
          `products` text NOT NULL,
          `status` int(11) NOT NULL DEFAULT '1',
          PRIMARY KEY (`id`)
        ) ENGINE=MyISAM AUTO_INCREMENT=48 DEFAULT CHARSET=utf8;

        LOCK TABLES `product_order` WRITE;
        INSERT INTO `product_order` VALUES (47,'Андрій','12312312312','dsdfsdfdf dgfh
fg',2,'2021-05-23 00:56:51','{\"53\":1}',1);
        UNLOCK TABLES;


        DROP TABLE IF EXISTS `user`;
        CREATE TABLE `user` (
          `id` int(11) NOT NULL AUTO_INCREMENT,
          `name` varchar(255) NOT NULL,
          `email` varchar(255) NOT NULL,
          `password` varchar(255) NOT NULL,
          `role` varchar(50) NOT NULL,
          PRIMARY KEY (`id`)
        ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;


        LOCK TABLES `user` WRITE;
        INSERT INTO `user` VALUES
(1,'Сергій','serg@gmail.com','111111',''),(2,'Андрій','admin@gmail.com','111111','admin');
        UNLOCK TABLES;
```