МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-АПТЕКИ

Здобувач освіти гр. ІН – 83 Микола ГАГЕНКО

Науковий керівник,
кандидат фізико-математичних наук,
асистент кафедри комп'ютерних наук Ольга ШУТИЛЄВА

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ
до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-83 спеціальності «122 – Комп'ютерні науки» денної форми навчання Гагенка Миколи Андрійовича.

Тема: «ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-АПТЕКИ»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) аналіз принципів побудови інтернет-магазинів; 2) постановка задачі й формування завдань дослідження; 3) проектування структури та дизайну веб-додатку; 5) розробка веб-додатку; 6) конфігурування та розгортання на сервері.

Дата видачі завдання «_____»_____2022 р.
Керівник роботи _____ Ольга ШУТИЛЄВА
Завдання прийняв до виконання _____ Микола ГАГЕНКО

РЕФЕРАТ

Записка: 71 стор., 49 рис., 2 додатки, 15 джерел.

Об’єкт дослідження – веб-сайти електронної торгівлі.

Предмет дослідження – веб-сайт інтернет аптеки побудований засобами фреймворку Laravel та СУБД MySql.

Мета роботи – розробка веб-сайту інтернет аптеки засобами фреймворку Laravel та СУБД MySql.

Методи дослідження – технології та методи побудови сучасних веб-додатків.

Результати – розроблено веб-сайт інтернет аптеки. Веб-сайт побудовано засобами PHP фреймворку Laravel та СУБД MySql.

ВЕБ-САЙТ, ВЕБ-ДОДАТОК, PHP, LARAVEL, MYSQL, NGINX, UBUNTU, BOOTSTRAP

ЗМІСТ

ВСТУП

Електронна комерція, або онлайн торгівля – це бізнес, який функціонує через електронні канали, такі як Інтернет за допомогою спеціальних програмних додатків. За останні кілька років електронна комерція стала домінуючим способом купівлі та продажу товарів і послуг. На це є декілька причин. По-перше онлайн торгівля економічно ефективна, зручна та швидка. По-друге вона дає можливість швидко охопити велику кількість клієнтів. По-третє реалізація є доволі доступною і надає величезні можливості як для малого, так і для великого бізнесу.

За даними статистики у 2022 році глобальний обсяг продажів на ринку електронної комерції в усьому світі складе 5,5 трильйона доларів, а до 2025 року 7,4 трильйона доларів. Очікується, що ця сума продовжить зростати в найближчі роки, що доводить, що електронна комерція стає все більш прибутковим варіантом бізнесу. Крім того, очікується, що 20,3% загального світового роздрібного продажу в 2022 році буде здійснюватися через мережу Інтернет [1, 2].

В Україні інтернетом користуються близько 70% населення, в порівнянні в 2010 році цей показник дорівнював 30%. Загальна сума фізичних товарів і послуг, які придбали українці в інтернеті у 2020 році, сягнула 107 мільярдів гривень [3].

Багато показників говорять про те що ринок онлайн торгівлі дуже стрімко розвивається і є перспективним напрямком розвитку сучасних IT технологій, а отже ця тема є досить актуальною для дослідження.

Метою роботи є розробка веб-сайту інтернет аптеки. Перед створенням було проведено аналіз існуючих аналогів. Для досягнення цієї мети було проведено аналіз існуючих засобів розробки веб-додатків, розроблено інформаційну систему сайту, проведено розробку UI та UX дизайну сайту. Виконати мануальне тестування веб-додатку, з метою виявлення критичних помилок.

1. ЛІТЕРАТУРНИЙ ОГЛЯД

1.1 Аналіз принципів побудови інтернет-магазинів

Успішність інтернет-магазину залежить від дотриманні деяких основних принципів. В 2021 році близько 73% покупок у світі було проведено за допомогою мобільних пристроїв [4]. Через це сучасний інтернет-магазин повинен бути оптимізованим для мобільних пристроїв, а якщо про це не подбати то буде висока вірогідність втратити велику кількість клієнтів. Просто мати гарний дизайн сайту сьогодні недостатньо. Він повинен бути чутливим до якомога більшої кількості пристроїв різних розмірів і мати гарний UX-дизайн.

Звичайно що працюючий функціонал сайту більш важливий, але в питанні електронної комерції UI-дизайн відіграє не останню роль. Хороший користувальницький інтерфейс важливий у тому сенсі, що він полегшує користувачам чітке уявлення про продукти. Візуальний дизайн покращує декоративну привабливість сайту. При старанному виконанні візуальний дизайн робить сторінку гарною без шкоди для її функцій чи вмісту. Зображення використовуються для надання візуальної інформації про продукт, а також для посилення ідентичності та унікальності бренду або магазину, тому вони повинні бути якісними.

Важливо зробити веб-сайт не тільки привабливим візуально але й з хорошим UX-дизайном. Для цього треба використовувати такі прості методи, як добре складене меню, проста навігація по сайту та приваблива цільова сторінка, це допоможе перетворити звичайних відвідувачів у клієнтів. Хороший UX-дизайн може виділити сайт серед конкурентів. Але досить важливо балансувати між бажанням привнести щось нове та загальними тенденціями. В UX-дизайні дуже важливі очікування користувачів. Визначення досвіду який користувачі мали з подібними веб-сайтами раніше у своєму житті, допомагає встановити їх очікування щодо того, як інтерфейси мають працювати. Якщо не приділяти цьому уваги то можна ненавмисно спроектувати інтерфейсну

взаємодію, яка здається логічною для розробника, але порушує загальноприйняті стандарти. Користувачам не подобається, коли інтерфейс поводиться зовсім по іншому, ніж вони очікували.

Ще однією проблемою є те що не у всіх користувачів є потужні пристрої. При недостатній увазі до цієї проблеми це може призвести до зменшення швидкості роботи веб-додатку на малопотужних пристроях. Повільна швидкість завантаження сторінки є одним з найбільших факторів, що сприяють збільшенню показнику відмов сайту. За даними дослідження Google час завантаження 1-5 секунд збільшує ймовірність відмов на 90% [5]. У онлайн-покупців у наш час дуже мало терпіння та часу і вони залишать сайт, якщо будуть змушені чекати. При розробці сайту треба подбати про оптимізацію його роботи, зменшуючи розміри файлів зображень, використовуючи кешування браузера та зменшуючи кількість перенаправлень. При цьому треба підтримувати баланс між гарною продуктивність та гарним дизайном сайту щоб користувачі отримували якомога більше гарних емоцій від користування і частіше поверталися за новими покупками.

Найважче залучити клієнтів на сайт. Тому дуже важливо не давати їм причин покинути сайт і не завершити покупку. Важливо зробити процес придбання товарів на сайті легким та інтуїтивним. Наприклад, кнопки «Купити» та «Оформити» треба зробити легко видимими. Також корисним є не вимагати від клієнтів обов’язкового створення облікового запису для оформлення замовлення та перерахування вартості товарів у місцевій валюті. 23% користувачів залишать свій кошик для покупок, якщо їм доведеться створити новий обліковий запис користувача [6].

Оплата на сайті має бути легкою. Коли клієнт хоче оформити замовлення, важливо максимально спростити цей процес і не давати йому причин не завершити покупку. Наприклад, треба переконатися, що на сайті передбачено широкий вибір варіантів оплати товару. Також досить корисним є чітке зазначення витрат користувача на доставку та політику повернення. Можна сказати що чим менше від користувача приховано інформації і чим

надійнішими є операції тим більше його довіра до цього сервісу і тим більше вірогідність його повторних покупок [8].

Важливим інструментом зв'язку та спілкування між магазином та клієнтом є використання електронної пошти. За допомогою неї можна легко повідомити користувача про розпродажі, випуск нових продуктів, або про наявність попередньо відсутніх товарів. Електронна пошта є потужним інструментом для відновлення взаємодії з клієнтами, які покинули свої товари у кошику, наприклад, листом про те що на ці товари діє знижка. Важливим аспектом якісного сервісу є швидке надсилання електронною поштою квитанцій про покупки. Наприклад, у США середній дохід від електронної пошти за залишений електронний лист при оплаті товару становить 27,12 доларів [7].

Гарним варіантом розповісти про магазин або сервіс новому користувачу є створення на сайті розділу «Про нас». На цій сторінці можна надати інформацію про компанію або інформацію про переваги інтернет магазину в порівнянні з іншими.

1.2. Інструменти для розробки веб-додатків

Інструменти для розробки веб-додатків призначені для спрощення та оптимізації процесу веб-розробки без шкоди для продуктивності. Існує багато видів інструментів розробки веб-додатків, кожний з яких полегшує або підтримує певний елемент розробки.

В першу чергу потрібен редактор коду. Є багато різних варіантів, наприклад безкоштовні Notepad++ або Sublime Text, але оскільки планується розробка інтернет магазину на основі мови PHP то раціональним рішенням буде обрати більш вузько направлений редактор PhpStorm. PhpStorm це інтелектуальний редактор для PHP, HTML і JavaScript з можливостями аналізу коду в режимі реального часу з просунутою системою миттєвої перевірки коду на помилки і автоматизованими механізмами рефакторингу для PHP та JavaScript. Також PhpStorm має гарну документацію, велику кількість плагінів доступних

для завантаження і в ньому є вбудований SQL-редактор з функціями редагування та перегляду отриманих результатів запитів.

Для налагодження, редагування або просто перегляду параметрів HTML, CSS або JavaScript в режимі реального часу є, досить корисний, вбудований прямо у браузер інструмент Chrome DevTools.

При розробці проекту потрібно налаштувати систему контролю версій, для запобігання негативних непередбачених ситуацій. Є різні системи контролю версій, але найпопулярнішою є Git. Git працює у фоновому режимі і записує зміни в спеціальну базу даних, яка має назву репозиторій. В цій базі даних можна побачити, хто вніс якісь зміни у проект коли і чому. Система Git має дуже обширний функціонал, наприклад, щоб надіслати деякі зміни в репозиторій треба виконати операцію commit. Якщо щось пішло не по плану є можливість скасувати фіксацію, за допомогою reset команди, щоб повернутися до попередньої версії коду. Git – це розподілена система в якій кожен член команди розробки зберігає копію проекту на своїй локальній машині. На відміну від централізованої системи контролю версій, в якій розробники підключені до одного серверу і працюють з останньою копією коду, в розподіленій системі, якщо з сервером щось трапиться і він вийде з ладу, розробники все ще зможуть працювати в автономному режимі.

Для зберігання свого коду не тільки на локальній машині, а ще у хмарному сховищі використовується GitHub. GitHub – це веб-додаток, на якому розміщується Git. Це онлайн репозиторій, де можна досить легко почати зберігати свої проекти, з системою контролю версій. Окрім звичайного зберігання різних версій коду GitHub має дві важливі функції – запит на витяг pull requests і процес рецензування коду code reviews. Pull requests запити дають можливість розпочати процедуру злиття локальної версії та основного коду який зберігається у репозиторії. Злиття коду відбувається тільки після того як буде виконаний процес code reviews під час якого члени команди розробки аналізують та тестують код і визначають чи можна його злити з основним репозиторієм. Ці процедури запобігають непередбаченим і неконтрольованим змінам у коді.

Для того щоб користувачі змогли отримати доступ до веб-додатку його треба розмістити на веб-сервері. Сервер складається з програмного та апаратного забезпечення, яке для відповіді на запити клієнтів використовує протокол HTTP та інші протоколи. Доступ до програмного забезпечення веб-сервера здійснюється через доменні імена веб-сайтів і забезпечує доставку вмісту сайту користувачеві, який запитує. Популярним варіантом програмного забезпечення сервера є веб-сервер є Nginx. Перевагами його серед інших є його доволі простий процес налаштування та гарна продуктивність. Nginx користуються такі великі компанії як Alibaba та Clouflare, і через це його розробка і підтримка є на досить високому рівні.

В якості апаратної частини веб-серверу гарним варіантом є використання одного з сервісів Amazon Web Services (AWS) платформи Amazon EC2. EC2 платформа яка дозволяє користувачам орендувати віртуальні машини, які називаються «екземпляри» для своїх потреб, наприклад, на них можна запустити веб-сервер. AWS є економічно вигідною оскільки оплата нараховується тільки за використовувані обчислювальні потужності і, що не менш важливо, має безкоштовний план користування, який підходить для невеликих проектів або тестування. Сервіс AWS є гнучким і дозволяє обирати операційну систему віртуальної машини, мову програмування, платформу інтернет-програм, бази даних та інші сервіси. EC2 надає можливість контролювати географічне розташування екземплярів, що дозволяє зменшити затримку, та змінювати їх технічні параметри в залежності від потреб. За допомогою вбудованого сервісу Amazon CloudWatch можна здійснювати у реальному часі моніторинг ресурсів екземпляру, таких як навантаження на процесор, диск або мережу [9].

Онлайн магазин повинен десь зберігати базу даних клієнтів та товарів, а для керування базами даних потрібна система управління базами даних (СУБД). MySQL – це безкоштовна реляційна СУБД з відкритим кодом яка є простою у налаштуванні та засвоєнні і є популярною при створенні веб-додатків. MySQL є безпечним, оскільки має надійний рівень безпеки даних для захисту конфіденційних даних від зловмисників, а паролі в MySQL

шифруються. MySQL невибагливий до середовища запуску і сумісний з більшістю популярних операційних систем.

Веб-фреймворки потрібні для автоматизації накладних витрат, пов'язаних із одноманітними діями та рутинними задачами, що виконуються при веброзробці. Laravel – це безкоштовний веб-фреймворк PHP з відкритим вихідним кодом, призначений для створення веб-додатків за архітектурним шаблоном model-view-controller (MVC) і заснований на Symfony. Laravel має модульну систему пакування що дозволяє легко розширювати його функціонал спеціальними пакетами. Також, Laravel використовує Composer як менеджер залежностей для додавання PHP-пакетів, які не залежать від фреймворку, і специфічних тільки для Laravel, доступних із репозиторію Packagist. Вбудований у фреймворк механізм шаблонізації Blade призначений для поєднання шаблонів з моделлю даних для отримання результатів перегляду, роблячи це шляхом транспіляції шаблонів у кешований PHP-код для збільшення продуктивності. Blade також надає набір власних структур управління, таких як умовні оператори та цикли, які внутрішньо зіставляються зі своїми аналогами PHP. Крім того, служби Laravel можна викликати з шаблонів Blade, можливості механізму шаблонування можна збільшити за допомогою користувацьких директив. Міграції забезпечують систему контролю версій для схем бази даних, що надає можливість пов’язувати зміни в кодовій базі програми та необхідні зміни в макеті бази даних. В результаті ця функція спрощує розгортання та оновлення програм на основі Laravel [10].

Composer – це менеджер залежностей на рівні програми для мови програмування PHP, який надає стандартний формат для керування залежностями програмного забезпечення PHP та необхідних бібліотек. Composer запускається з командного рядка та встановлює залежності, наприклад, необхідні бібліотеки для програми. Він надає можливості автоматичного завантаження для бібліотек, які позначені для автозавантаження або автоматичного оновлення, щоб полегшити використання стороннього коду і для уникнення проблем пов’язаних з застарілими версіями пакетів та бібліотек [11].

Для спрощення процесу створення і налаштування користувацької частини веб-сайту можна використати один з багатьох front-end фреймворків, наприклад, Bootstrap. Bootstrap – це безкоштовний фреймворк CSS з відкритим вихідним кодом, спрямований на адаптивну веб-розробку для мобільних пристроїв і комп'ютерів. Він містить багато готових шаблонів дизайну на основі HTML, CSS і JavaScript для форм, таблиць, фігур, кнопок, навігації та інших компонентів інтерфейсу. Основна мета його додавання до веб-проекту – застосувати базові набори кольору, розміру, шрифту та макета Bootstrap до цього проекту. Після підключення до проекту Bootstrap надає базові визначення стилів для всіх елементів HTML. Результатом є особливий вигляд тексту, таблиць і елементів форм у всіх веб-браузерах. Крім того, великою перевагою є можливість тонкого налаштування класів CSS, визначених у Bootstrap, для різних форматів і розмірів екранів. Bootstrap також постачається з кількома компонентами JavaScript у вигляді плагінів jQuery. Вони забезпечують роботу додаткових елементів інтерфейсу користувача, таких як діалогові вікна, підказки та різні базові анімації. Кожен компонент Bootstrap складається з HTML-структури, декларацій CSS і в деяких випадках супровідного коду JavaScript. Вони також розширюють функціональні можливості деяких існуючих елементів інтерфейсу, включаючи, наприклад, функцію автозаповнення для полів введення [12].

1.3 Постановка задачі

Головною метою роботи є розробка веб-додатку онлайн аптеки.

У процесі виконання кваліфікаційної роботи необхідно реалізувати наступні задачі:

1. виконати огляд існуючих інструментів для розробки веб-додатків та обрати найбільш оптимальні;

2. розробити та створити інтуїтивно зрозумілий та адаптивний інтерфейс веб-додатка;

3. виконати проектування та реалізувати архітектури веб-додатка реалізації товарів;

4. розробити алгоритм роботи та діяти за ним;

5. виконати тестування веб-додатку та зробити відповідні висновки по роботі.

Після створення проекту буде можливим слідкувати за існуючими позиціями товарів та контролювати їх зміст, здійснювати контроль над обліковими записами користувачів, формувати замовлення або робити підписку на товар який закінчився.

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Структура веб-додатку інтернет-магазину

Великий веб-сайт призначений для онлайн продаж може мати багато сторінок, і він повинен мати чітку структуру, щоб користувачі могли знайти те, що вони хочуть, за найменш можливу кількість кліків. Веб-сайт призначений для онлайн торгівлі повинен мати ієрархічну структуру.

Головна сторінка повинна знаходитися у верхній частині ієрархії, нижче неї можете бути 2 або 3 рівні, нижньою частиною ієрархії сайту мають бути сторінки продуктів (рис.2.1).

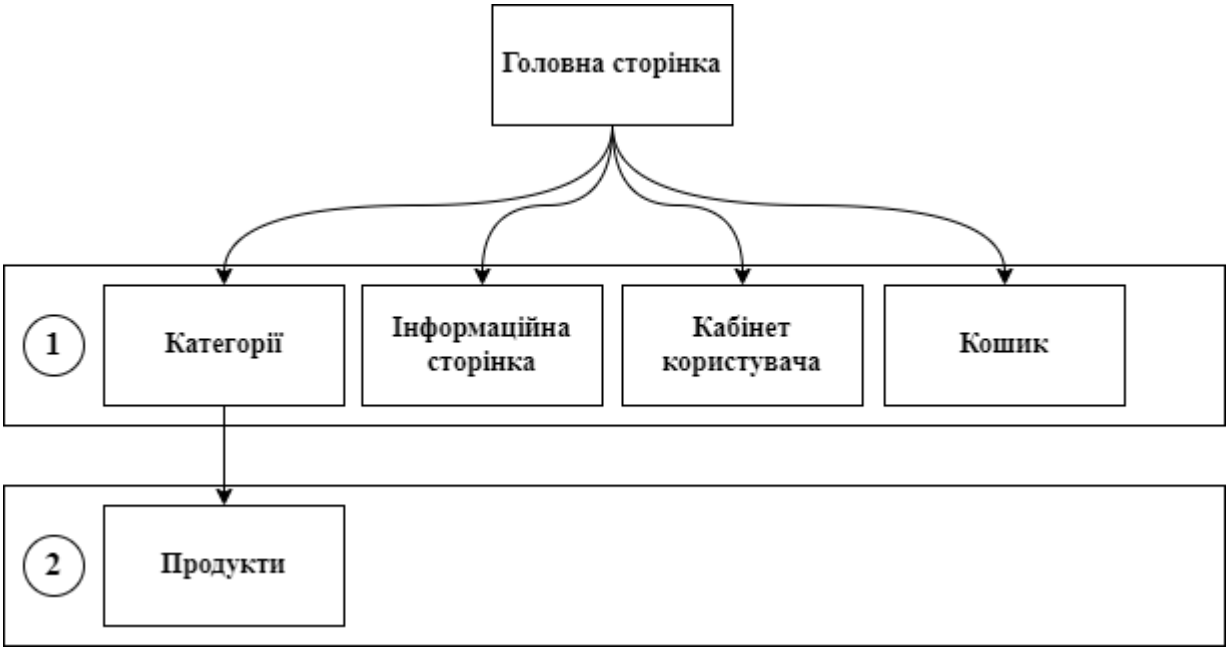

Рисунок 2.1 – Архітектура інтернет-магазину

Домашня сторінка знаходиться у верхній частині ієрархії, за нею йдуть категорії (рівень 1) і далі йдуть сторінки продуктів (рівень 2).

2.2 Розробка дизайну веб-додатку

Процес розробки розпочався з проектування дизайну. Для створення макету було використано онлайн інструмент Figma. Figma це популярний зручний інструмент створений однією з функцій якого є створення дизайну. Програма має зрозумілий та інтуїтивний інтерфейс, в якому можна дуже швидко освоїтися та почати працювати.

Першим кроком було моделювання головної сторінки сайту. Спочатку було зроблено макет меню навігації. Навігаційна панель повинна містити назву сайту, кнопки-посилання «Всі товари», «Категорії», «Кошик», «Про нас», «Увійти» та «Вийти», «Мій кабінет», кнопки зміни мови сайту і кнопку зміни валюти та поле пошуку. На головній сторінці буде міститися каталог товарів. Він повинен містити адаптивну до різних пристроїв кількість карток товару. Макет навігаційної панелі для персонального комп'ютера або ноутбука представлено на рисунку 2.2.

Рисунок 2.2 – Макет навігаційної панелі веб-сайту для ПК

Оскільки сайтом скоріше за все будуть користуватися і з мобільних пристроїв, то слід подбати про зручну навігацію на маленьких екранах. Повторення меню такого як і для персональних комп'ютерів не є хорошим варіантом через обмежений простір на екрані мобільного пристрою. Меню навігації на смартфоні буде мати можливість згортатися та розгортатися для зручності перегляду. Макет розгорнутої навігаційної панелі для смартфону представлено на рисунку 2.3.

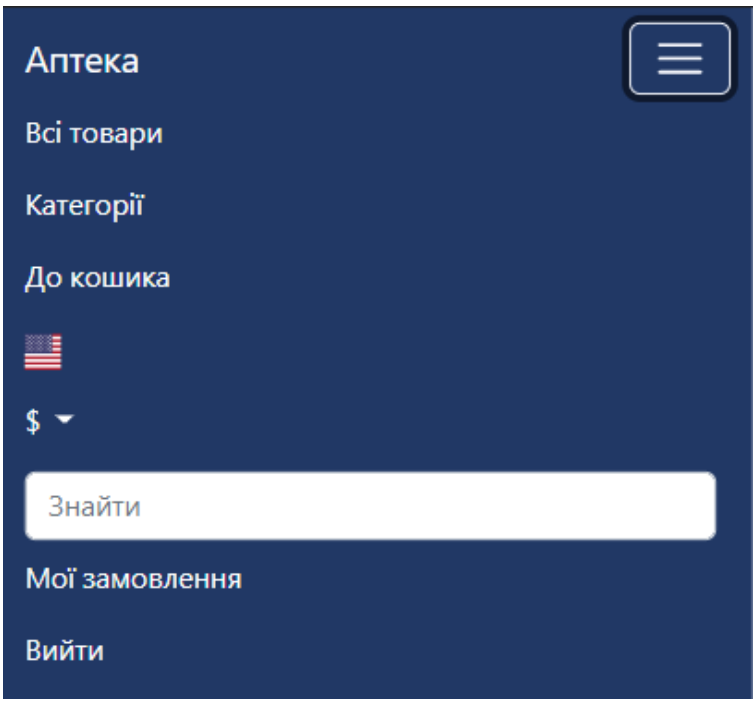

Рисунок 2.3 – Макет навігаційної панелі веб-сайту для смартфону

Також було розроблено дизайн адаптивного меню фільтрів. Меню фільтрів для смартфонів в компактному вигляді показано на рисунку 2.4.

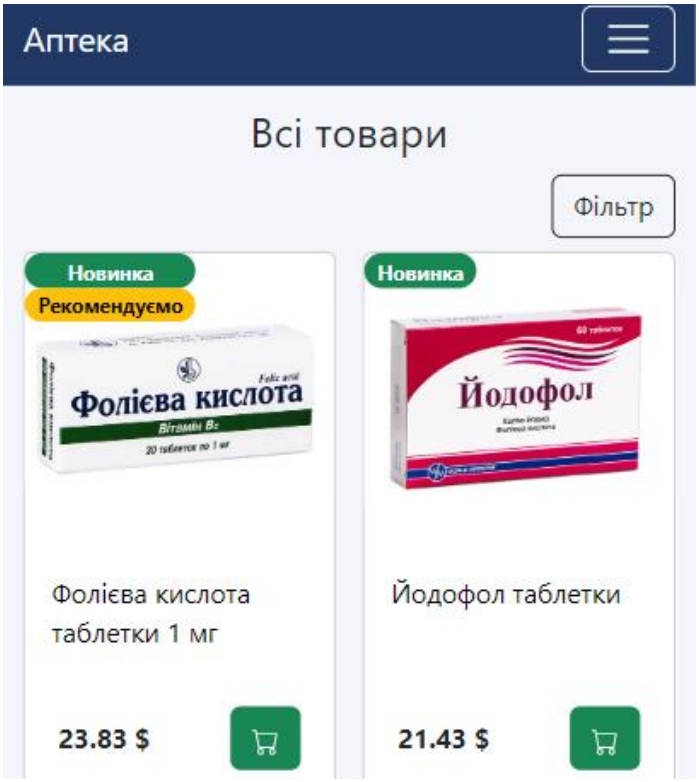

Рисунок 2.4 – Згорнута панель фільтрів на смартфоні

Меню фільтрів в розгорнутому вигляді показано на рисунку 2.5.

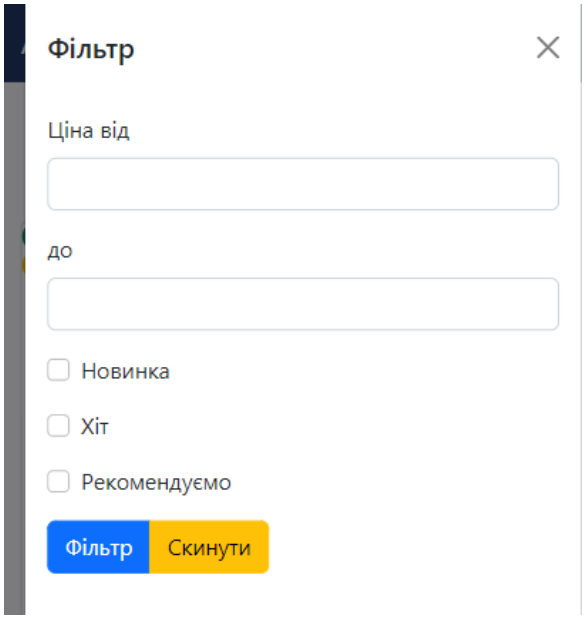

Рисунок 2.5 – Розгорнута панель фільтрів на смартфоні

Сторінка товару має містити: назву товару, ціну, перелік властивостей, опис товару, зображення товару, інструкцію товару та кнопку «До кошика».

Сторінка категорій повинна містити картки категорії товарів з приблизним зображення товару, та назву категорії.

Далі було розроблено дизайн футеру сайту. Він має містити список найпопулярніших товарів та список посилань на категорії сайту («Вітаміни», «Гігієнічні засоби», «Косметика», «Медичні товари», «Медикаменти»). Макет футеру сайту зображено на рисунку 2.6.

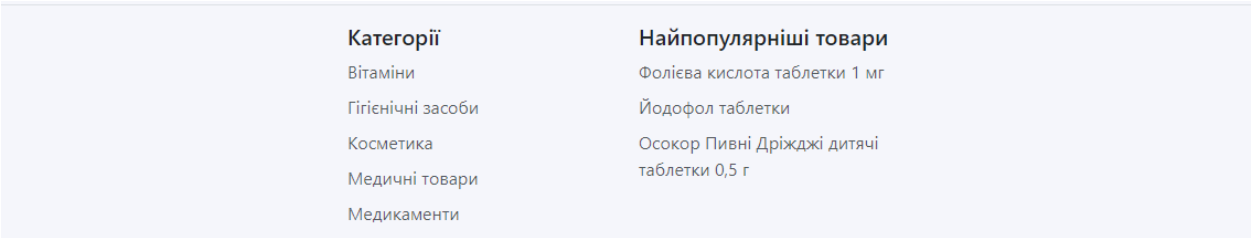

Рисунок 2.6 – Макет навігаційної панелі веб-сайту для смартфону

Футер буде відображатися на всіх сторінках крім кабінету користувача.

2.3 Проектування бази даних

Після основної розробки візуального макету сайту було розпочато роботу над проектуванням і створенням бази даних. Початкове проектування було здійснено в DataGrip. Це крос-платформна IDE від компанії JetBrains для роботи з різними СУБД такими як MySQL, PostgreSQL, Oracle. DataGrip дозволяє працювати з об'єктами бази даних та має ряд корисних можливостей таких як автодоповнення коду, шаблони однотипного коду, фільтр даних та навігація по даним також має вбудовану інтеграцію із системами контролю версій. Для бази даних веб-сайту онлайн аптеки було створено 16 таблиць. Далі наведено перелік головних таблиць:

- категорії;
- валюти;
- замовлення;
- продукти;
- товарні пропозиції;
- властивості продуктів;
- підписки на товар;
- користувачі.

Нижче на зображеннях показані дані таблиці. На рисунку 2.7 зображено таблицю категорій.

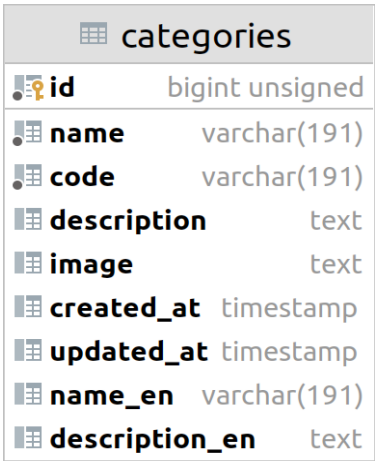

Рисунок 0.7 – Таблиця категорій

На рисунку 2.8 зображено таблицю валют.

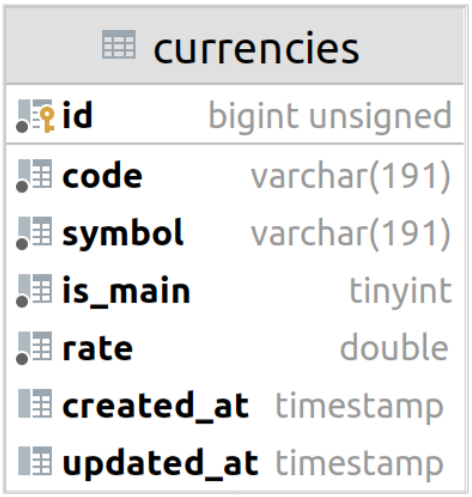

Рисунок 2.8 – Таблиця валют

На рисунку 2.9 зображено таблицю замовлень.

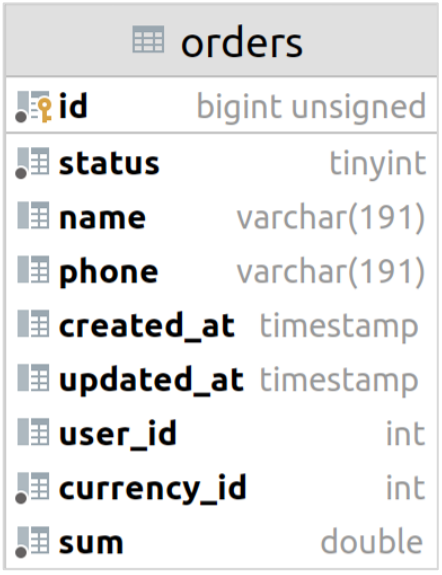

Рисунок 2.9 – Таблиця замовлень

На рисунку 2.10 зображено таблицю продуктів.

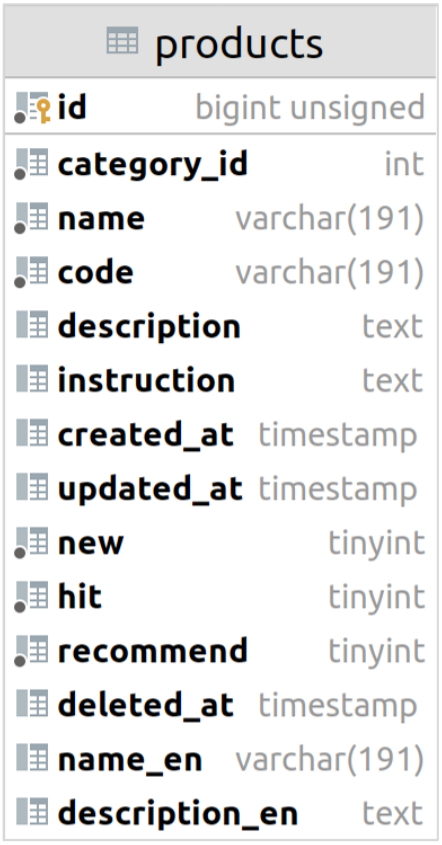

Рисунок 2.10 – Таблиця продуктів

На рисунку 2.11 зображено таблицю товарних пропозицій.

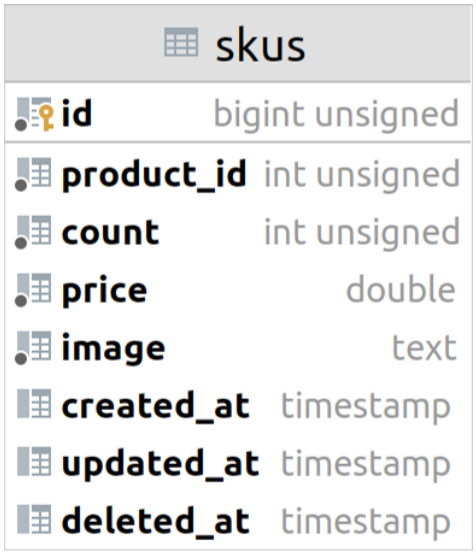

Рисунок 2.11 – Таблиця товарних пропозицій

На рисунку 2.12 зображено таблицю товарів.

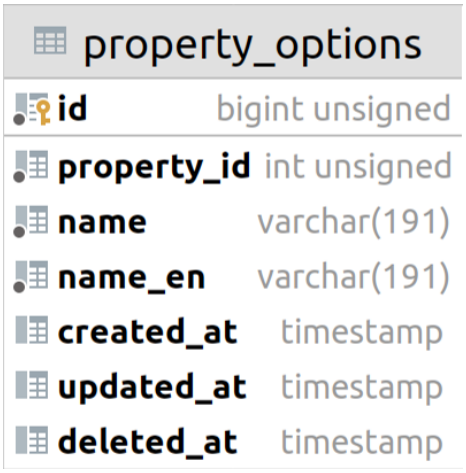

Рисунок 2.12 – Таблиця властивостей товарів

На рисунку 2.13 зображено таблицю підписок на товар.

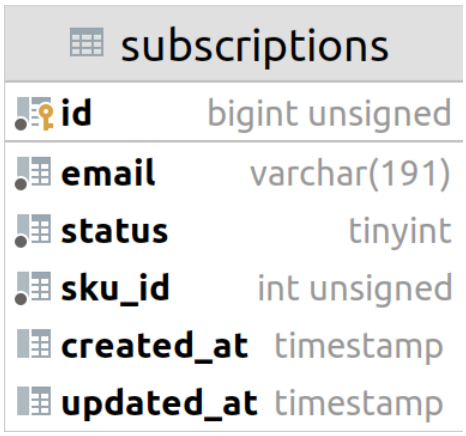

Рисунок 2.13 – Таблиця підписок на товар

На рисунку 2.14 зображено таблицю користувачів.

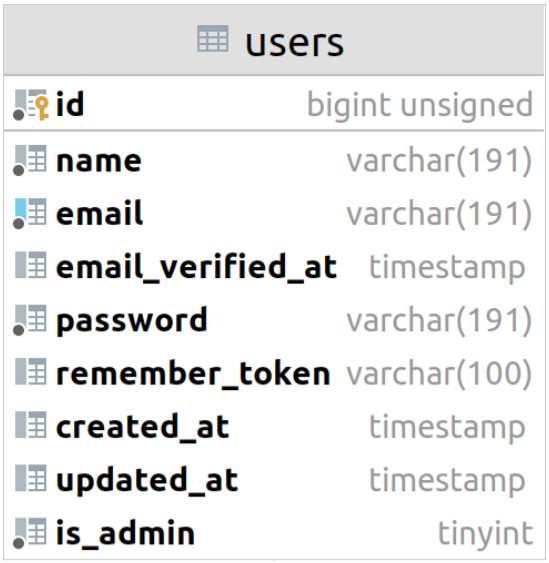

Рисунок 2.14 – Таблиця користувачів

Повна діаграма міжтабличних зв’язків бази даних наведена у Додатку А.

Оскільки іноді один товар може мати декілька варіацій з деякими відмінностями, наприклад, у кольорі або характеристиках було прийнято рішення додати таблицю товарних пропозицій (рис.2.11) яка пов’язана зовнішнім ключем з таблицею продуктів (рис.2.10). Зовнішній стовпець або комбінація зовнішніх стовпців – це стовпець або стовпці які пов’язані з первинним ключем іншої таблиці.

Інші таблиці були створені як допоміжні до основних, в основному для створення зв’язків багато-до-багатьох. Допоміжні таблиці також використовують зовнішні стовпці для поєднання.

Таблиця підписок (рис.2.13) містить в собі інформацію про підписки користувачів на товар який закінчився для того щоб отримати сповіщення на електронну пошту як тільки товар з’явиться.

Для кращого розуміння структури системи було створено діаграму варіантів використання. Діаграма варіантів використання – це графічне зображення різних варіантів використання та різних типів користувачів, яких має система і їх можливі варіанти взаємодії. Ця діаграма допомагає уявити проектовану систему на більш високому рівні і являється своєрідним кресленням

системи. Діаграма варіантів використання веб-додатку онлайн-магазину зображено на рисунку 2.15.

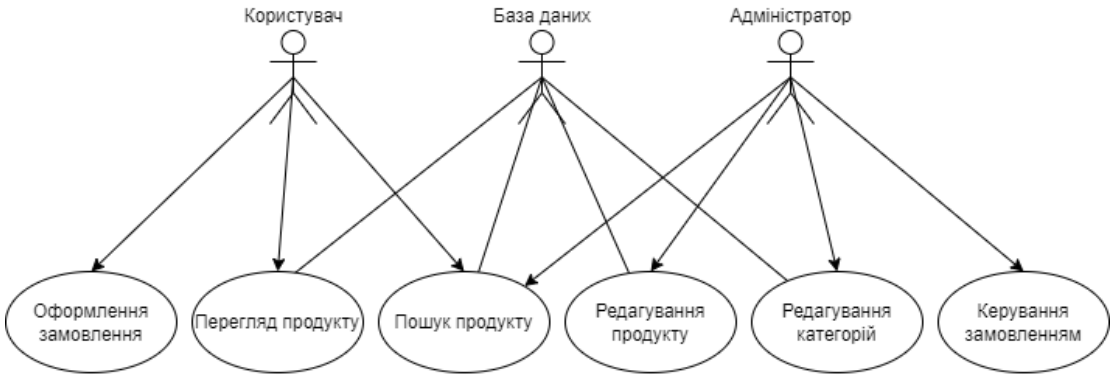

Рисунок 2.15 – Діаграма варіантів використання веб-додатку

Для зручності проектування та візуалізації бази даних веб-додатку було створено функціональну модель IDEF0. IDEF0 є методологією графічного опису систем і процесів та діяльності систем як безлічі взаємопов’язаних функцій. Особливістю функціонального моделювання є акцент на представлення об'єктів в ієрархічній послідовності, що досить сильно спрощує розуміння предметної області. На рисунку 2.16 зображено діаграму IDEF0 для веб-додатку.

Рисунок 2.16 – Діаграма IDEF0 для веб-додатку

На рисунку 2.17 зображено декомпозицію діаграми IDEF0 для веб-додатку.

Рисунок 2.17 – Декомпозиція IDEF0-діаграми для веб-додатку

Декомпозиція – це процес розбиття IDEF0-діаграми на функціональні фрагменти.

Таким чином проектування бази даних додатку було завершено. Повна ER-діаграма бази даних наведена у Додатку А.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Процес розробки веб-додатку

Робочим оточенням для процесу розробки було обрано операційну систему Ubuntu 20.04 LTS, стабільна версія з довгостроковою підтримкою. Такий вибір було зроблено в рахунок деяких переваг:

1. оточення розробки буде близьке або ідентичне продакшену;

2. зручна робота з консоллю з коробки, можливість розширення додатковими пакетами;

3. покращується розуміння процесу роботи серверу, що спрощує його підтримку та конфігурацію.

У якості веб-серверу було обрано вбудований у PHP веб-сервер, він не виконує всіх функцій повноцінного веб-сервера, але його функціоналу достатньо для розробки та тестування веб-додатку. Мова реалізації веб-додатку PHP 7.3, база даних MySQL 8.0.

У створеному робочому середовищі процес розробки базується на фреймворку Laravel. Першим етапом було створено моделі для основних сутностей: product, category, order та user. Для кожної з цих сутностей було згенеровано ресурсні контролери за допомогою інтерфейсу командного рядка Artisan, який входить до складу Laravel. Ресурсні контролери містять у собі базові методи: index, create, store, show, edit, update, destroy, які дозволяють переглядати, створювати, редагувати та видаляти сутності.

У Laravel всі запити відображаються за допомогою маршрутів, для цього необхідні маршрути були прописані в спеціальному файлі web.php. Маршрутизація направляє запити до асоційованих з URI методів контролерів, наприклад при натисненні посилання з URI /users/{user}/orders має відкритися метод orders контролера UserController, який в свою чергу має вивести правильне представлення (англ. view).

Для полегшення інтеграції PHP коду в представлення було використано механізм шаблонізації Blade. Шаблонізатор дозволяє створювати функціональні макети, які можна перевикористовувати іншими view. Для кожного представлення було створено свій шаблон Blade, наприклад product.blade.php, order.blade.php, index.blade.php, category.blade.php.

Для автоматичного створення необхідних таблиць в базі даних було створено класи міграцій. Клас міграції зазвичай містить два методи один з яких виконується щоб внести зміни в базу даних, а інший для відновлення до попереднього стану. Міграції можна запустити з терміналу командою Artisan migrate.

Для тестування, та початкового наповнення баз даних інформацією було створено seed-класи. Клас наслідуваний від Seeder за замовчуванням містить лише один метод: run. Цей метод викликається, коли виконується db:seed команда Artisan. У рамках run методу описуються дані для занесення в базу даних.

Під час розробки виникла необхідність використання функцій відлагодження. Для цього було встановлено Laravel Debugbar (рис. 3.1). Це зручна панель, завдяки якій проводилося налагодження функціоналу, відстеження помилок та контроль часу завантаження сторінок.

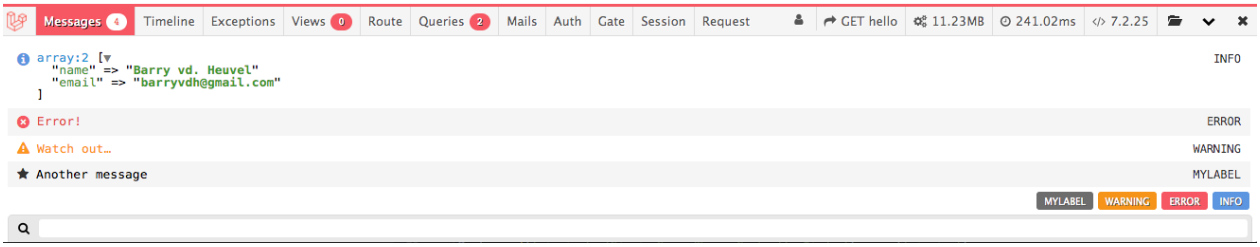

Рисунок 3.1 – Laravel Debugbar

При розробці сайту інтернет магазину було потрібно створити механізм відправки електронних листівок користувачам. В процесі розробки для зручного тестування відправки пошти було використано сервіс Mailtrap (рис. 3.2). Для цього було змінено данні в конфігураційному файлі .env на надані

сервісом Mailtrap. Далі було створено відповідні спеціальні mailable контролери окремо для надсилання листівок при підписці на товар SendSubscriptionMessage та при оформленні замовлення OrderCreated, та створено відповідні маршрути.

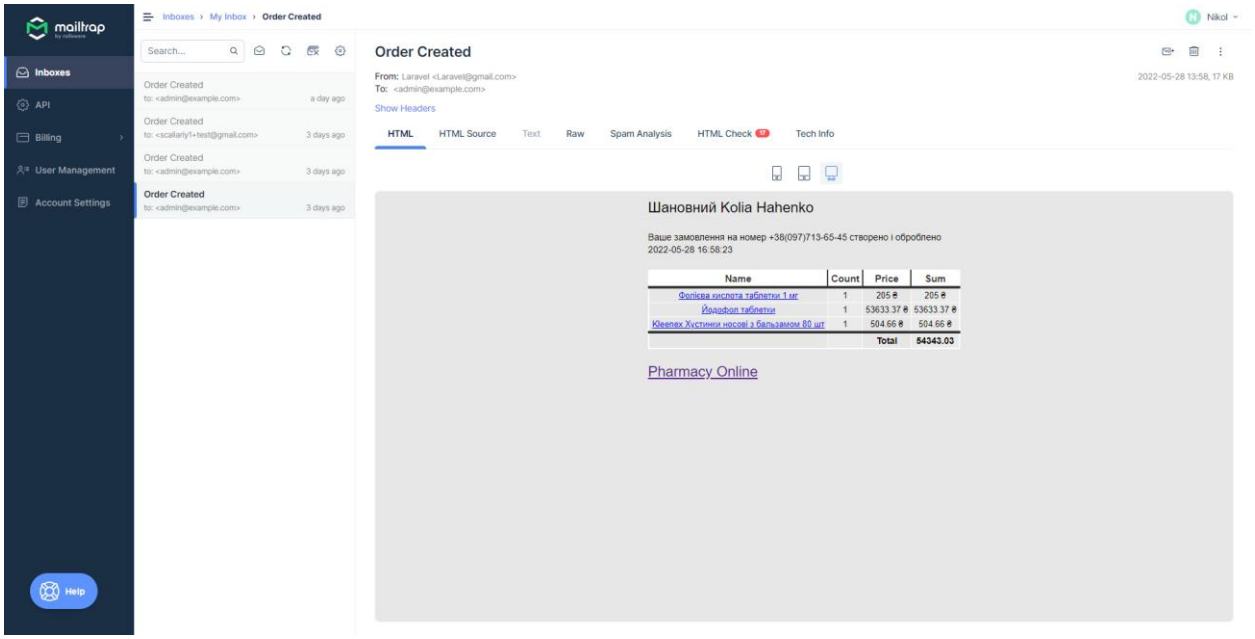

Рисунок 3.2 – Вікно Mailtrap

У процесі подальшого тестування було знайдено проблему в надмірному навантаженні веб-додатку при тимчасовій відправці пошти при виконанні відповідних дій. Було прийнято рішення зробити відправку пошти з застосуванням механізму черги. В Laravel черги забезпечують уніфікований API для різних сервісів черги, таких як Amazon SQS, Redis або реляційна база даних [15]. Для реалізації в проекті було обрано реляційну база даних, як самий простий в реалізації варіант. Для цього було створено відповідні таблиці jobs та failed_jobs для зберігання завдань у черзі та для зберігання невдалих завдань відповідно. Далі було внесено зміни в код контролерів, для відправки листів у чергу замість прямої відправки. Для контролю процесу обробки черг було налаштовано Supervisor, монітор процесів для операційної системи Linux, який автоматично перезапускає процеси обробки черг, якщо виникають якісь проблеми. Приклад заповнення таблиці jobs зображено на рисунку 3.3.

	id	queue	payload	attempts	reserved_at	available_at	created_at
Edit Copy Delete	1534	default	{"displayName":"App\\Jobs\\TaskOrderNotifySmsQueue...	0	NULL	1599991865	1599991855
Edit Copy Delete	1535	default	{"displayName":"App\\Jobs\\TaskOrderNotifySmsQueue...	0	NULL	1599991895	1599991885
Edit Copy Delete	1536	default	{"displayName":"App\\Jobs\\TaskOrderNotifySmsQueue...	0	NULL	1599992070	1599992060
Edit Copy Delete	1537	default	{"displayName":"App\\Jobs\\TaskOrderNotifySmsQueue...	0	NULL	1599996104	1599996104
Edit Copy Delete	1538	default	{"displayName":"App\\Jobs\\TaskOrderNotifySmsQueue...	0	NULL	1599996282	1599996272
Edit Copy Delete	1539	default	{"displayName":"App\\Jobs\\TaskOrderNotifySmsQueue...	0	NULL	1599997290	1599997280

Рисунок 3.3 – Таблиця черги jobs

Для теоретичного розширення клієнтської бази було зроблено локалізацію сайту, та доданий переклад на англійську. Зміна мови сайту відбувається при натисканні відповідної кнопки яка знаходиться в навігаційній панелі сайту. Локалізацію мають всі сторінки торгової частини веб-сайту та кабінет користувача. Панелі адміністратора перекладу не має. Приклад сторінки на англійській зображено на рисунку 3.4. Та сама сторінка але на українській зображена на рисунку 3.5.

Рисунок 3.4 – Локалізована сторінка

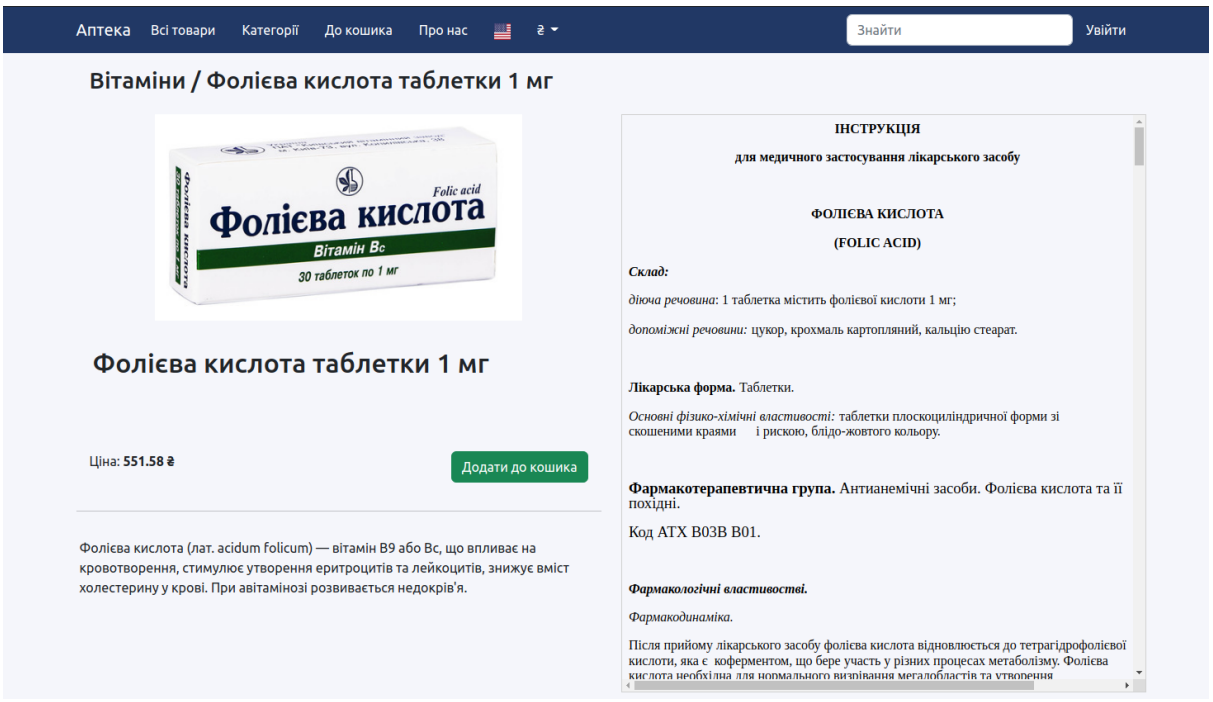

Рисунок 3.5 – Локалізована сторінка

При створенні локалізація було прийнято рішення додати можливість зміни декількох валют на сайті. Для отримання актуальних курсів валют було використано Exchange rates API сервісу exchangerate.host , простий та легкий безкоштовний сервіс для отримання поточних та історичних курсів іноземних валют і курсів криптовалют, також сервіс має можливість самостійного конвертування валют. Було створено таблицю currencies та відповідні класи CurrencyConversion і CurrencyRates для конвертації валют та для безпосереднього отримання курсу відповідно. Отримати потрібну інформацію можна звичайним GET запитом з потрібними параметрами які перераховані в документації до сервісу, наприклад щоб дізнатися курси валют відносно гривні треба до базового запиту додати base=UAH щоб вийшло https://api.exchangerate.host/latest?base=UAH. Було зроблена підтримка трьох валют: гривня, долар та євро. Зміна валюти відбувається шляхом вибору потрібної на панелі навігації. Приклад конвертації ціни товарів з гривні у долар та євро зображено на рисунках 3.6 та 3.7.

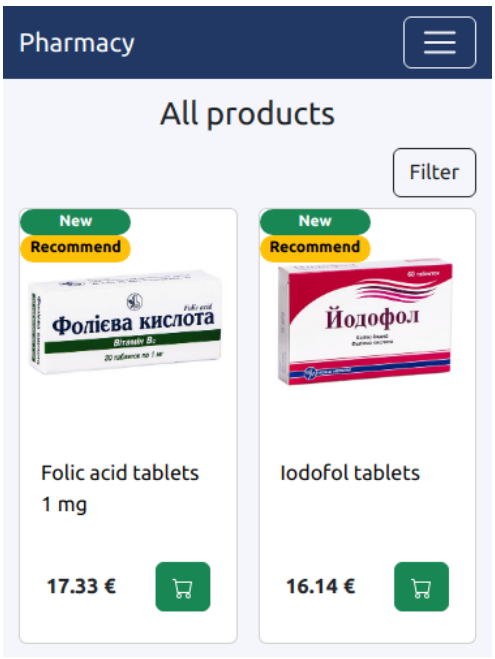

Рисунок 3.6 – Зміна валюти

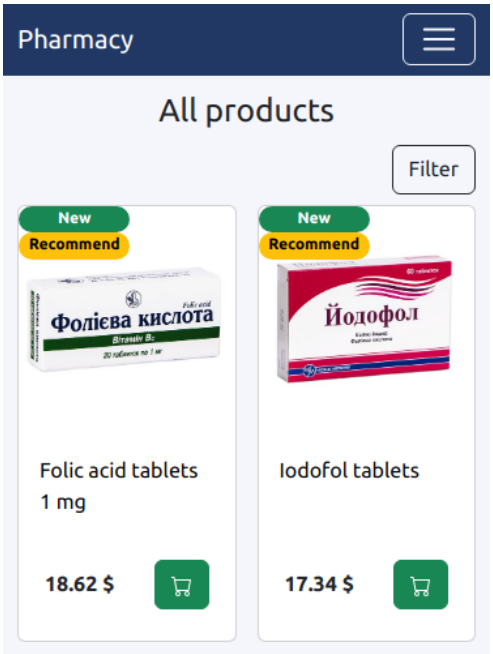

Рисунок 3.7 – Зміна валюти

Під час розробки виникла потреба розділяти товар на товарні пропозиції (англ. sku) через це було через міграції додано таблицю skus та створено модель Sku і контролер SkuController. Було змінено пов’язані з продуктом та товарною пропозицією методи та класи. Після цих змін продукт залишився як

обгортка над товарними пропозиціями які являють собою деякі варіації цього продукту.

Також в процесі розробки було створено можливість додавати до товарів певні властивості. Для цього було через механізм міграцій додано таблиці properties, property_options, sku_property_options, property_product. Також було створено модель Property та контролер PropertyController. Після цього було додано відповідні пункти в панель адміністратора, для контролювання властивостей товарів. Приклад властивостей товару зображено на рисунку 3.8.

Рисунок 3.8 – Властивості товару

3.2 Налаштування інтерфейсу веб-додатку

Оскільки розробка проекту виконувалася з обмеженими часовими та фінансовими ресурсами було прийнято рішення провести мінімально необхідну для повноцінного функціонування розробку frontend дизайну сайту. У зв’язку з цим було проведено огляд існуючих frontend фреймворків та обрано найоптимальніший варіант Bootstrap. Оскільки серед більшості інших безкоштовних фреймворків він має найобширнішу кількість готових макетів та класів

оформлення, також має гарну документацію та досить велику спільноту прихильників. Окрім цього було використано безкоштовну бібліотеку піктограм Bootstrap Icons.

Спочатку було створено головну сторінку сайту (рис. 3.9). Головна сторінка сайту, згідно запланованому дизайну, містить пагінований список всіх товарів та панель фільтрів. Панель фільтрів та картки товарів є адаптивними для різних пристроїв.

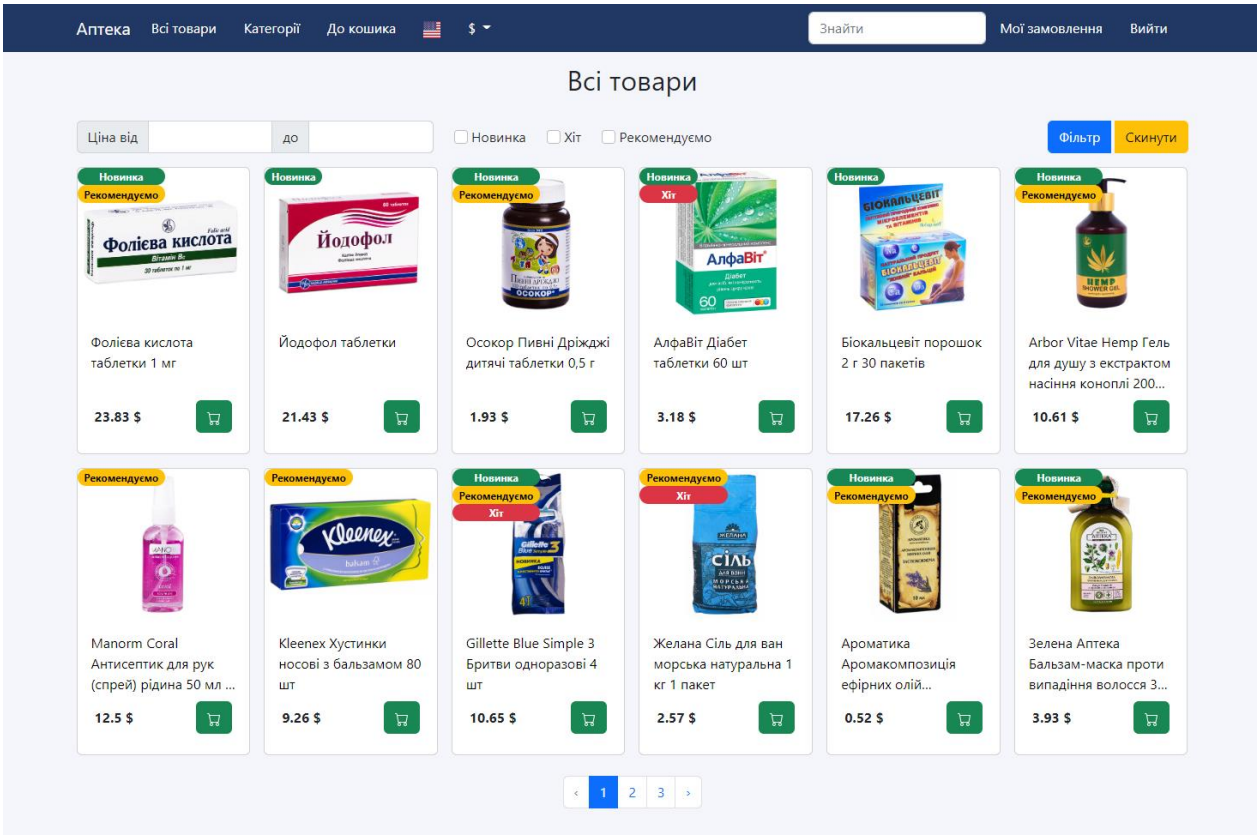

Рисунок 3.9 – Список товарів з панеллю фільтрів

Далі було створено сторінку категорій яка зображена на рисунку 3.10. Сторінка категорій також має адаптивний дизайн та масштабується на різних розмірах екранів.

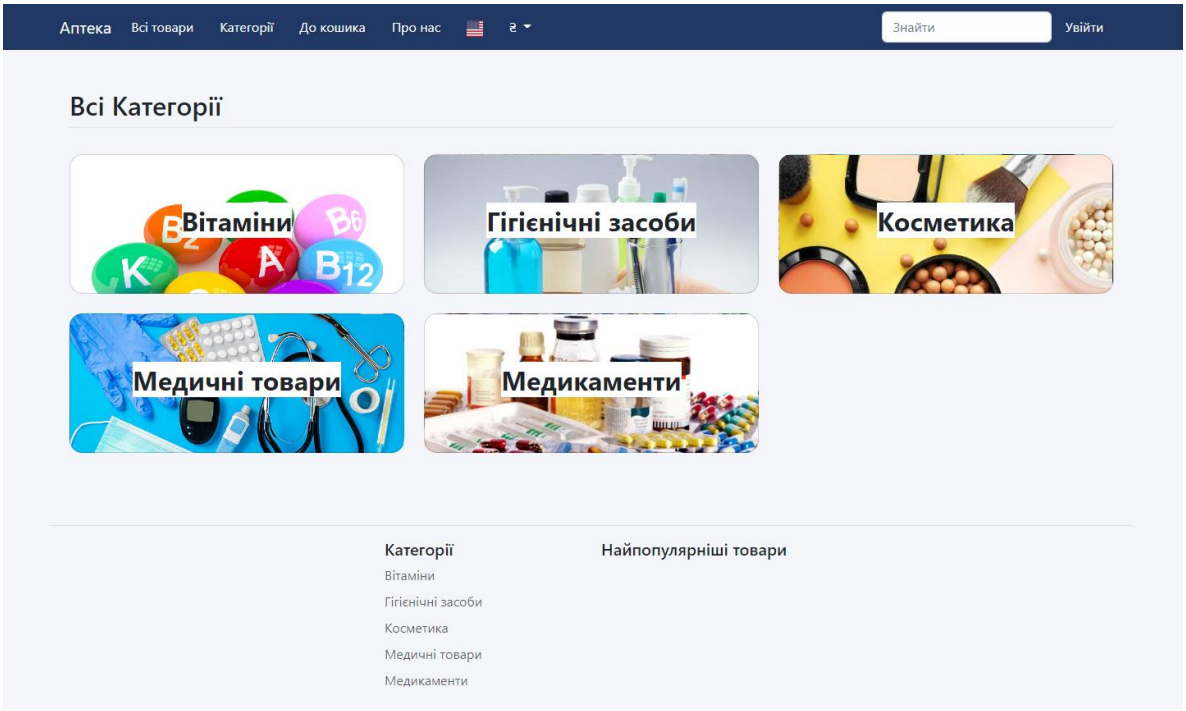

Рисунок 3.10 – Сторінка категорій товарів

При натисканні на категорію відкривається сторінка цієї категорії з переліком усіх товарів цієї категорії (рис. 3.11).

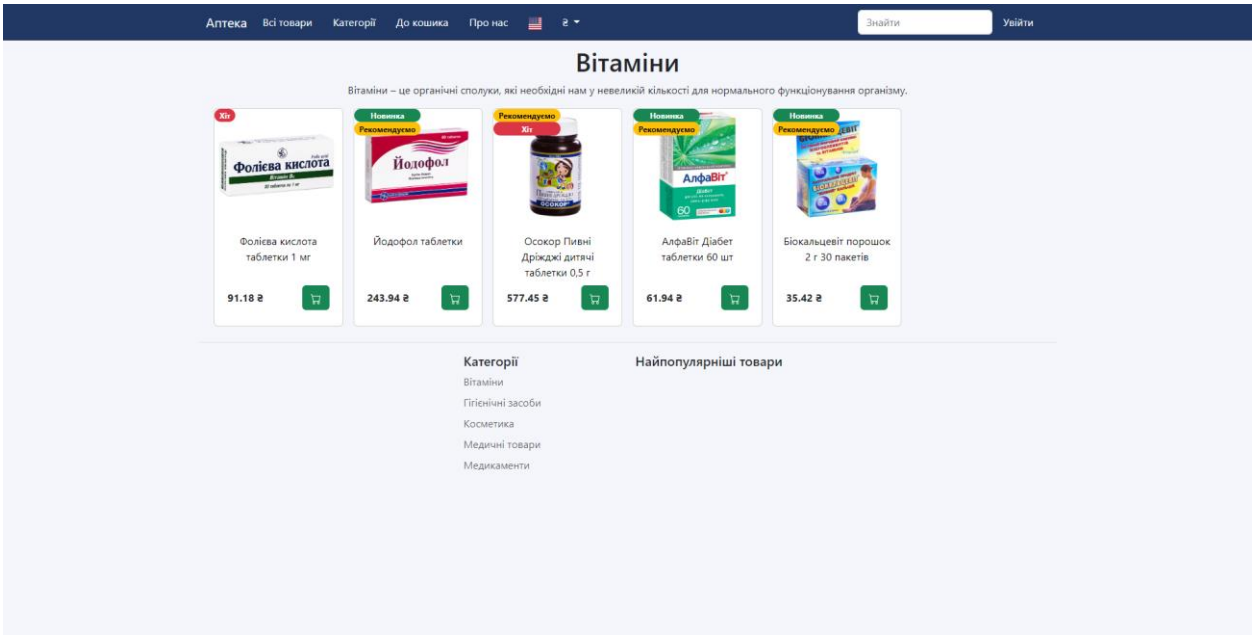

Рисунок 3.11 – Сторінка категорії

Згідно з запланованим дизайном і поставленими вимогами було створено сторінку товару, її зображено на рисунку 3.12.

Рисунок 3.12 – Сторінка товару

На сайті було створено систему показу флеш-повідомлень які сповіщають користувача про якісь проблеми або про успішність виконання операції. Наприклад, якщо за вказаними фільтрами нічого не було знайдено користувач побачить повідомлення (рис. 3.13).

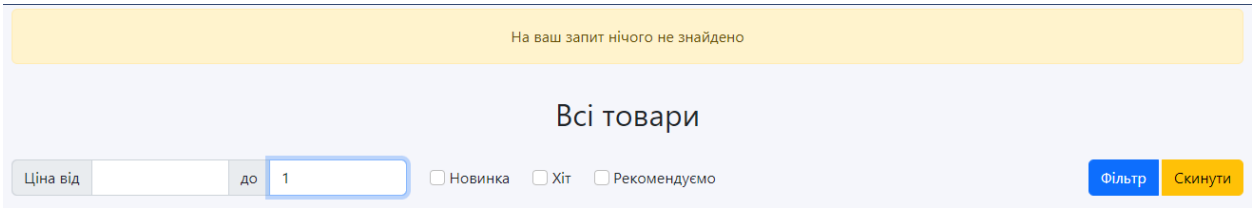

Рисунок 3.13 – Приклад флеш-повідомлення

Завдяки таким флеш повідомленням покращується розуміння виконуваного процесу користувачем.

Далі було створено сторінку кошику для товарів. На рисунку 3.14 зображено сторінку кошику для товарів.

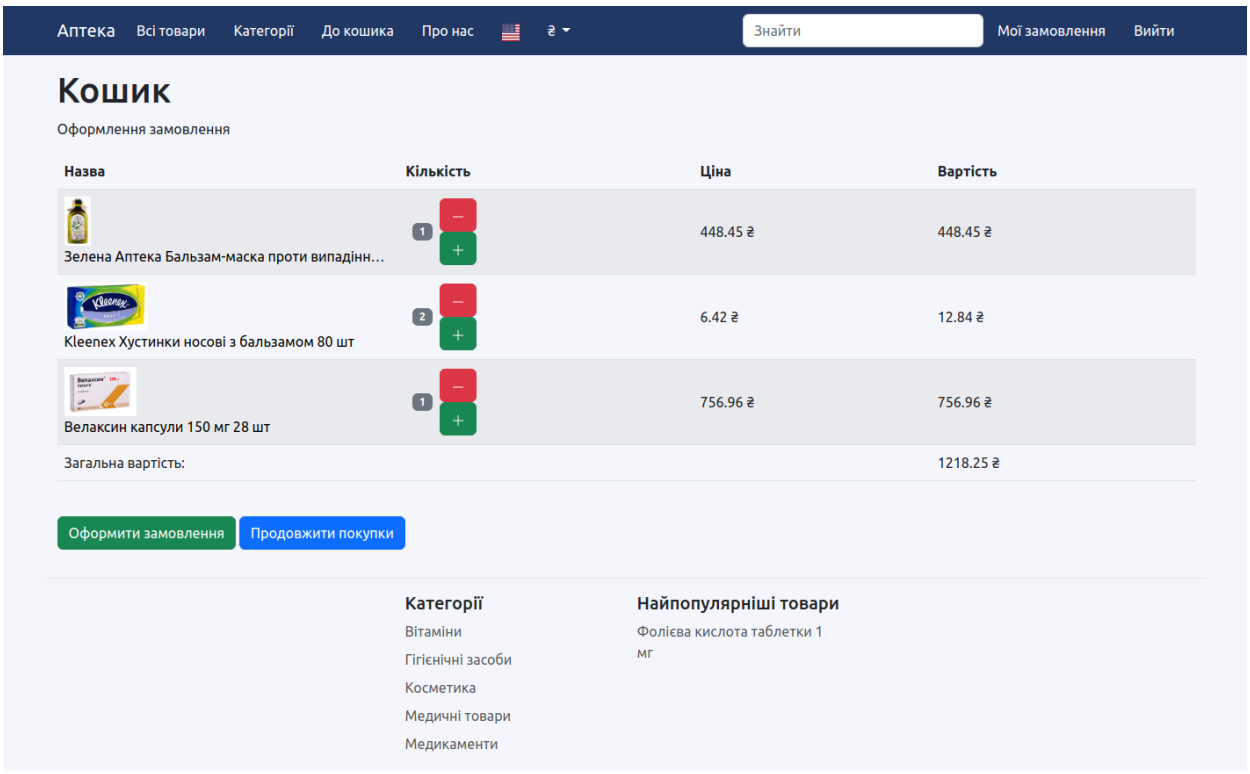

Рисунок 3.14 – Сторінка кошику для товарів

На рисунку 3.15 зображено сторінку оформлення замовлення. Для замовлення користувач має ввести свій номер телефону та ім’я особи на яку оформлюється замовлення. Для незареєстрованих користувачів ще потрібно ввести електронну пошту.

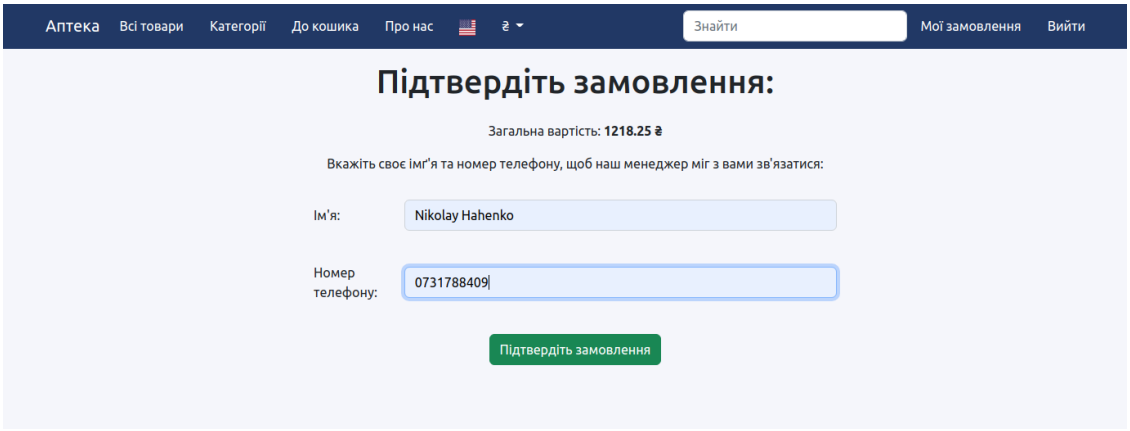

Рисунок 3.15 – Сторінка оформлення замовлення

На рисунку 3.16 зображено сторінка замовлень в кабінеті користувача.

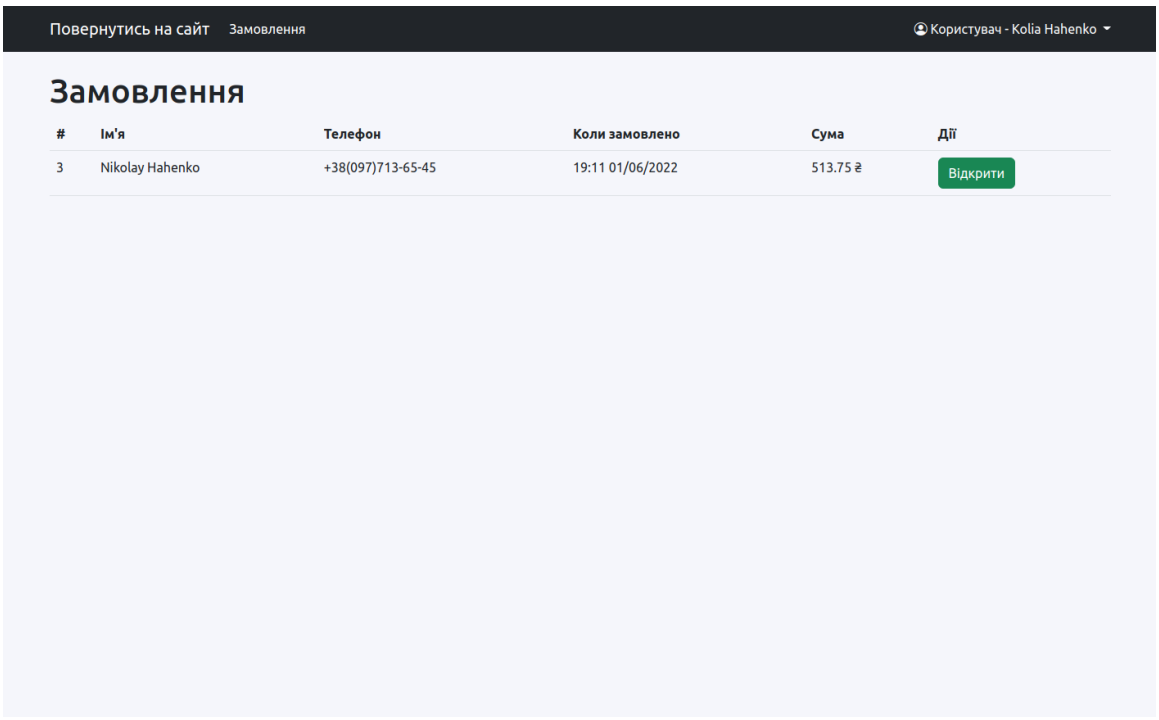

Рисунок 3.16 – Сторінка замовлень користувача

На рисунку 3.17 зображена сторінка замовлення.

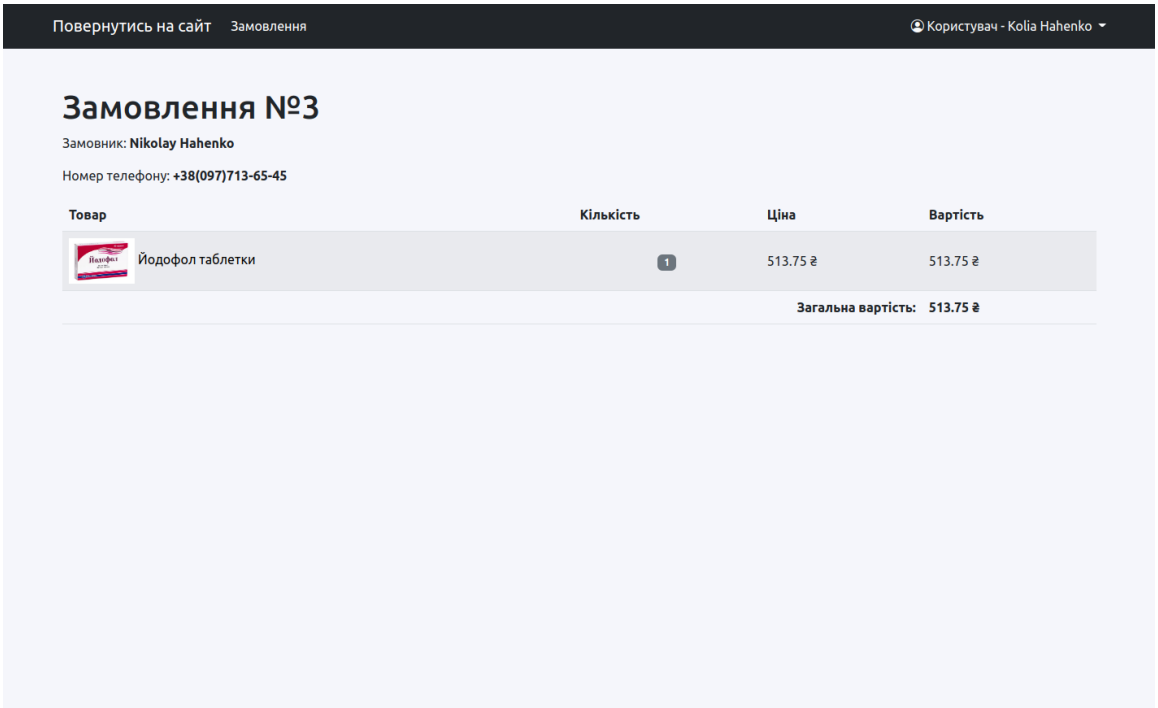

Рисунок 3.17 – Сторінка замовлень

На рисунку 3.18 зображена інформаційна сторінка сайту.

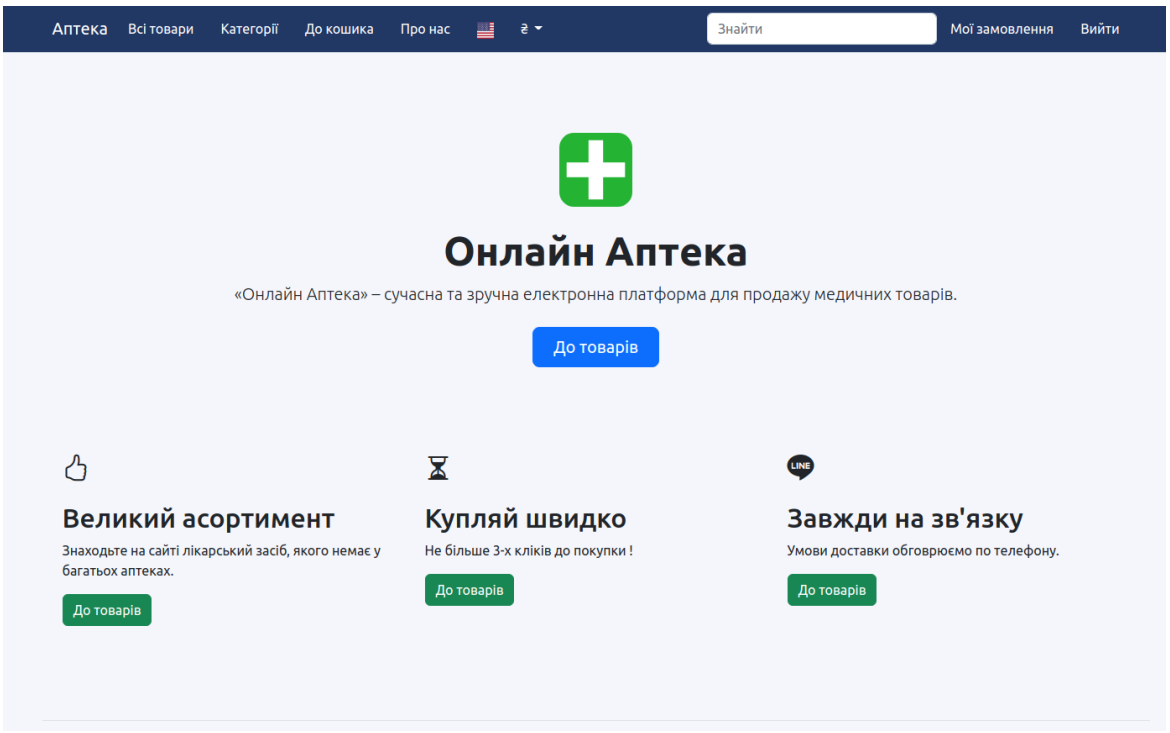

Рисунок 3.18 – Інформаційна сторінка сайту

На рисунку 3.19 зображена сторінка редагування профілю.

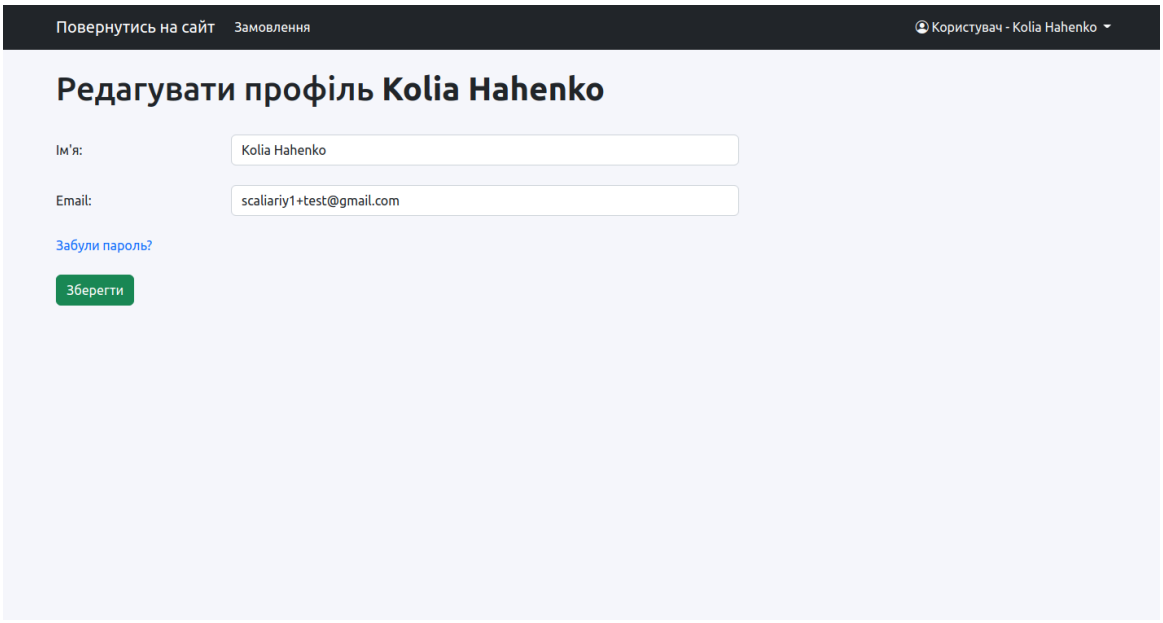

Рисунок 3.19 – Сторінка редагування профілю

Важливою частиною сайту інтернет торгівлі є керування товаром. Керування буде здійснюватися на панелі адміністратора. В панелі навігації адміністратора є пункти «Замовлення», «Категорії», «Товари», «Властивості», та

«Користувачі». При виборі пункту «Замовлення» відкривається сторінка перегляду замовлень, тут є можливість дізнатися всі деталі замовлення (рис. 3.20). При виборі пункту «Категорії» відкривається сторінка керування категоріями товарів, тут можна додавати, видаляти або змінювати існуючі категорії. При натисканні на пункт «Товари» буде відкрито сторінку перегляду всіх товарів сайту, тут можна додати, налаштувати або видалити товар.

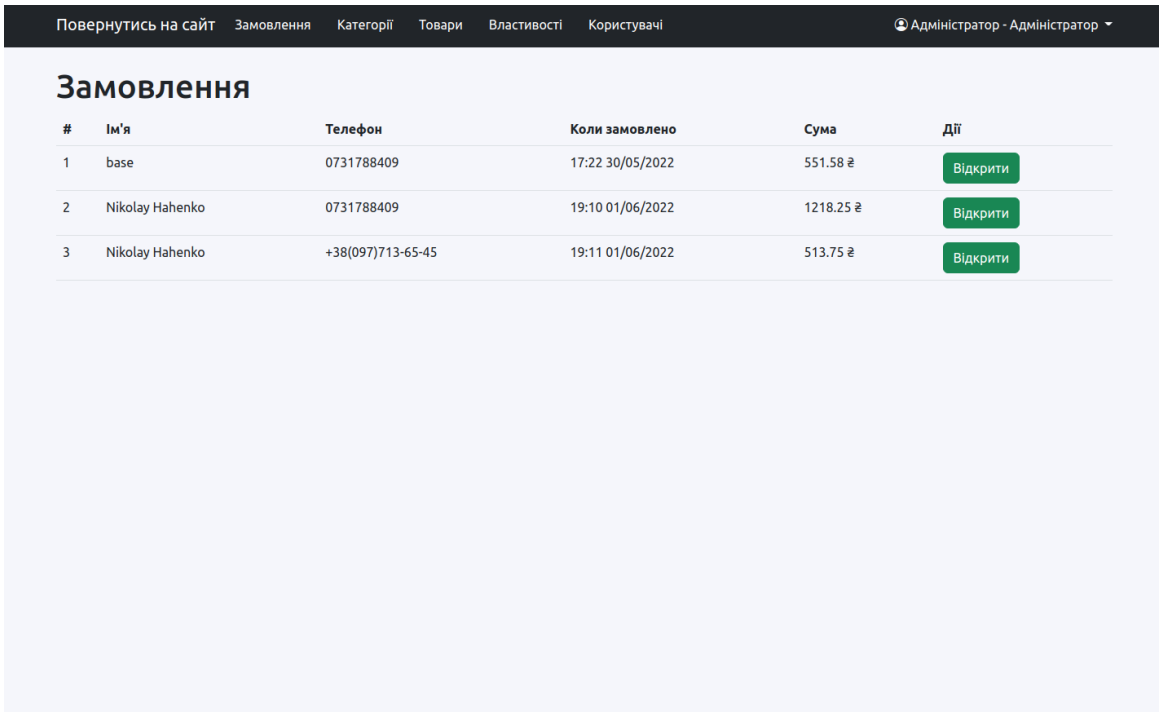

Рисунок 3.20 – Сторінка замовлень панелі адміністратора

Також згідно зі структурою сайту у кожного товару є одна або декілька товарних пропозицій, при виборі пункту «Товарні пропозиції» у випадаючому списку доступних дій (рис. 3.21) над товаром буде здійснено перехід до сторінки їх налаштування (рис. 3.22). На сторінці налаштування параметрів товарної пропозиції можна налаштовувати її ціну кількість та завантажувати зображення.

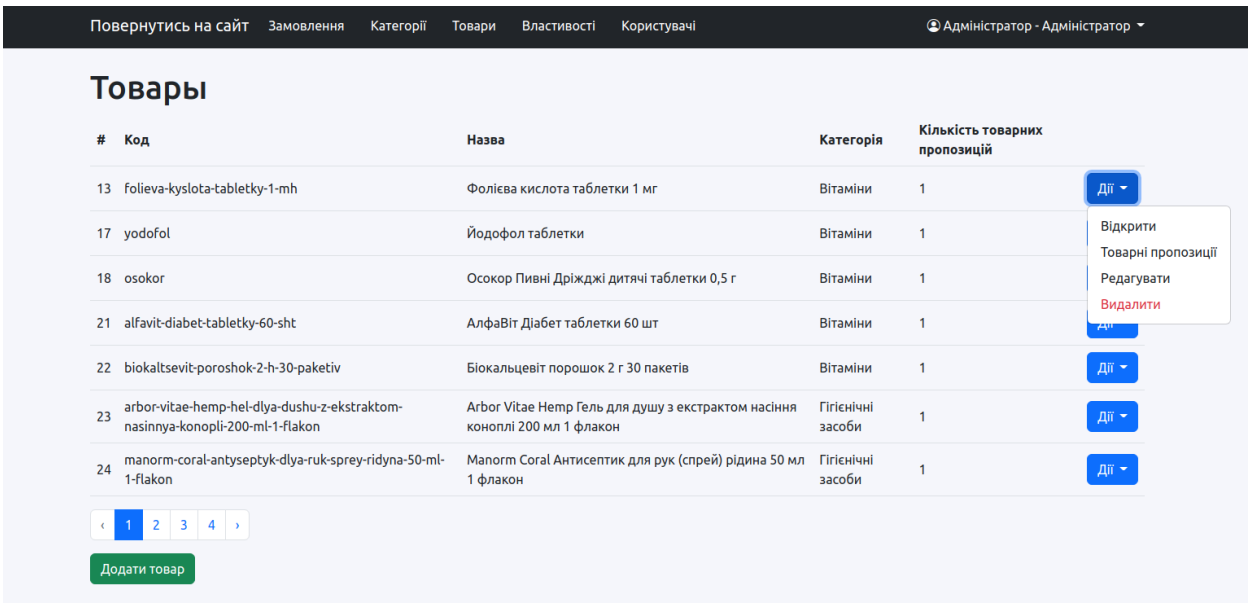

Рисунок 3.21 – Сторінка товарів панелі адміністратора

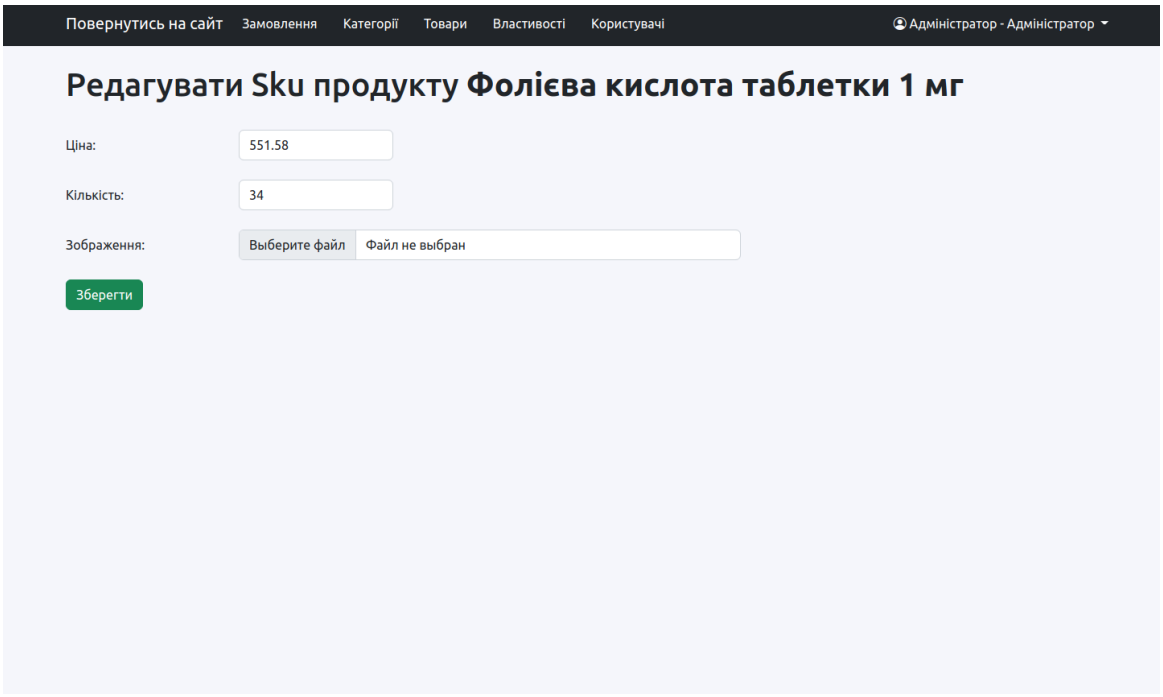

Рисунок 3.22 – Сторінка редагування товарної пропозиції

Товар також має свою сторінку редагування, тут можна змінювати параметри товару та додавати файл інструкції до товару. Сторінка редагування зображена на рисунку 3.23.

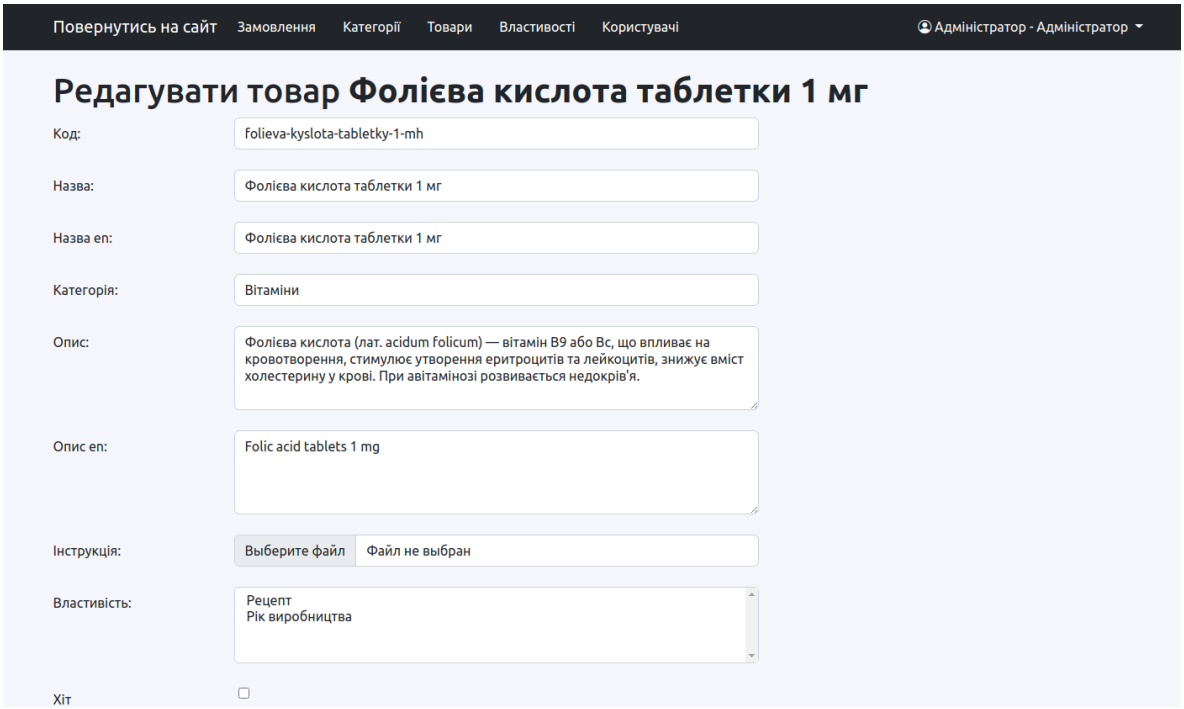

Рисунок 3.23 – Сторінка редагування товару

На рисунку 3.24 зображена сторінка контролю користувачів сайту.

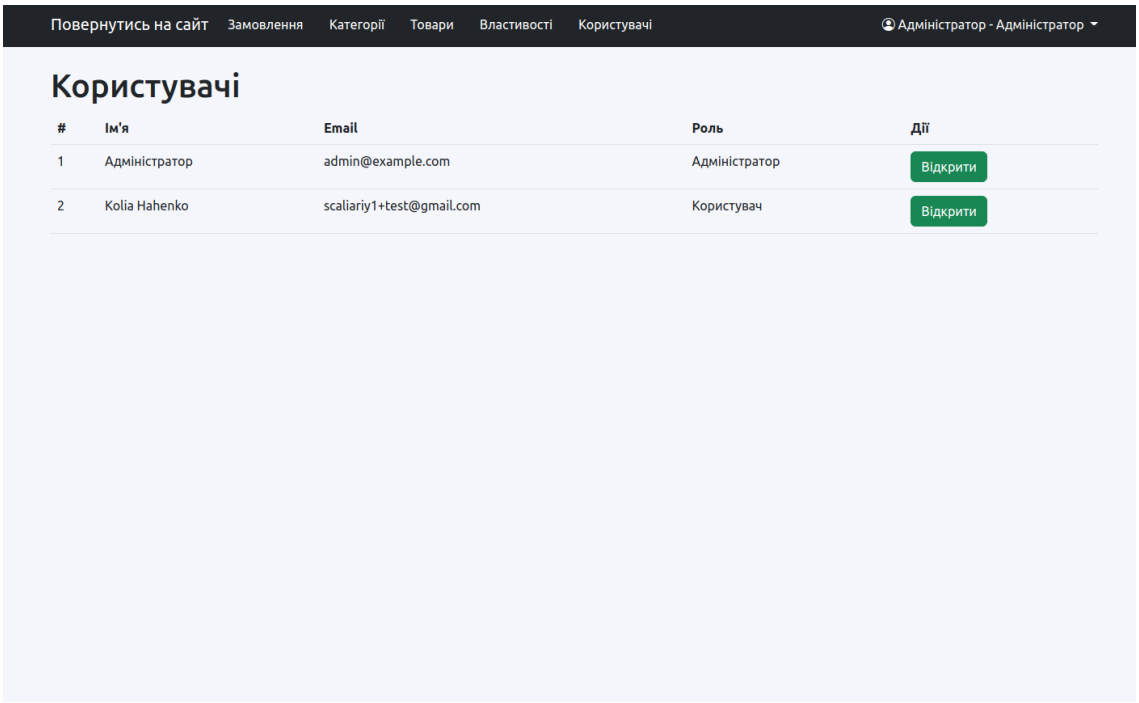

Рисунок 3.24 – Сторінка контролю користувачів

На рисунку 3.25 зображена сторінка перегляду параметрів товарної пропозиції.

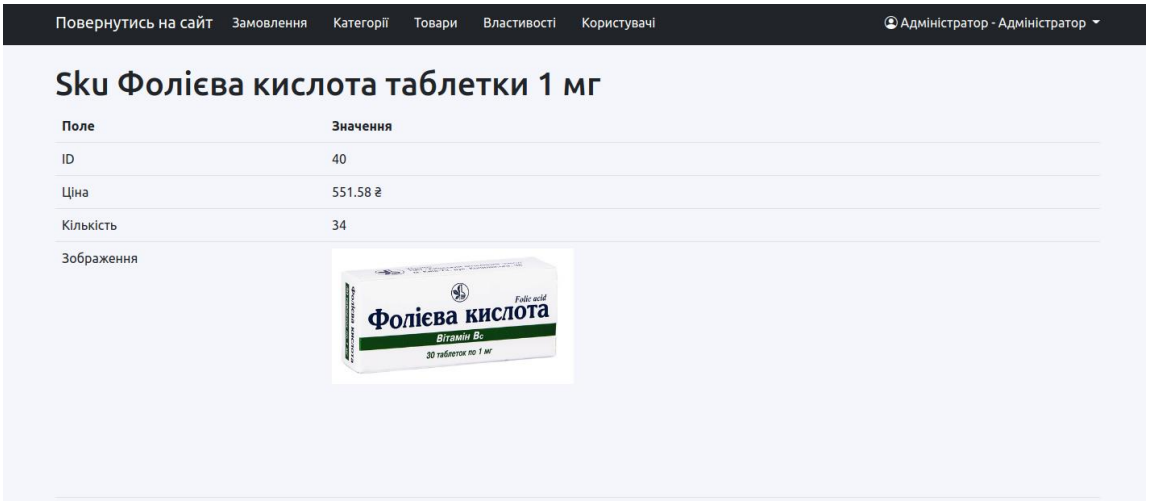

Рисунок 3.25 – Сторінка товарної пропозиції

На сайті розроблено механізм відновлення паролю через пошту, відповідне посилання знаходиться на сторінці авторизації та на сторінці редагування профілю. На рисунку 3.26 зображена форма авторизації.

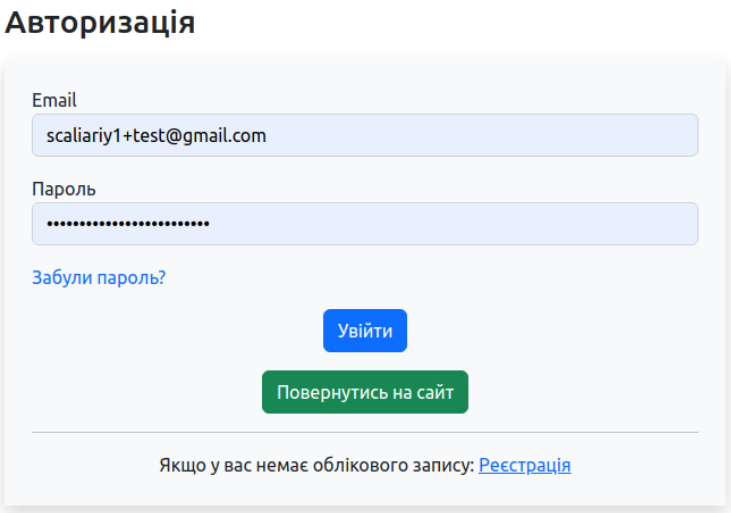

Рисунок 3.26 – Форма авторизації

На рисунку 3.27 зображена форма реєстрації.

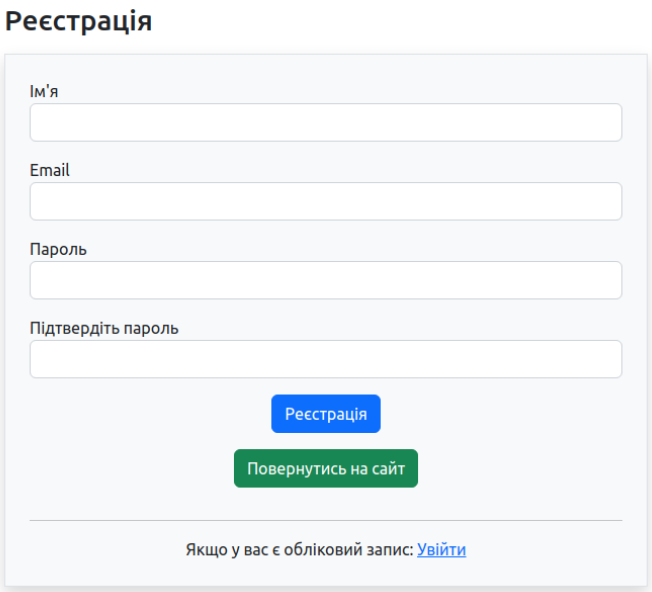

Рисунок 3.27 – Форма реєстрації

3.4 Розгортання веб-додатку на сервері

Після закінчення розробки та тестування веб-сайту. Було прийнято рішення розмістити його на сервері. Для цього було обрано сервіс AWS EC2, цей сервіс дозволяє створити віртуальну машину у хмарі та працювати з нею.

Згідно з безкоштовним планом користування, було створено сервер який має 1 Гб оперативної пам’яті, та 1 ядро процесору, та SSD диск на 8 Гб з можливістю динамічного розширення, такі технічні характеристики цілком задовільнять тестовий запуск веб-сайту але, якщо буде потрібно працювати з реальними більш великими навантаженнями, то потужності серверу можна збільшити за додаткову плату. Сторінка створення екземпляру зображена на рисунку 3.28.

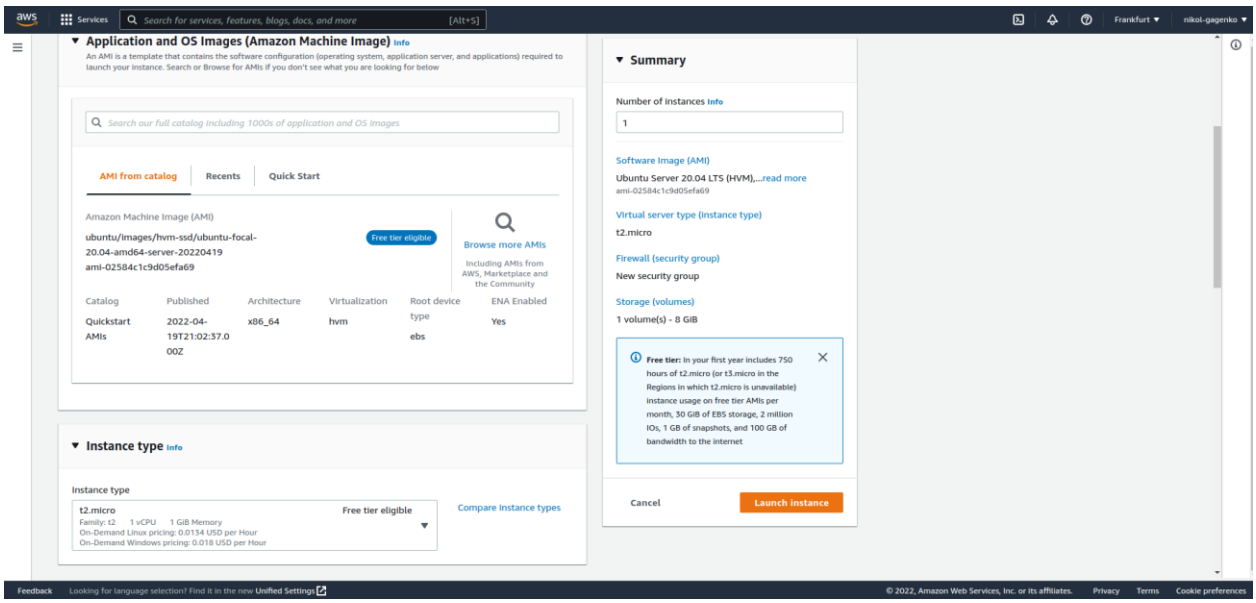

Рисунок 3.28 – Створення екземпляру EC2

Після створення екземпляру ним можна керувати та моніторити на спеціальній панелі (рис. 3.29).

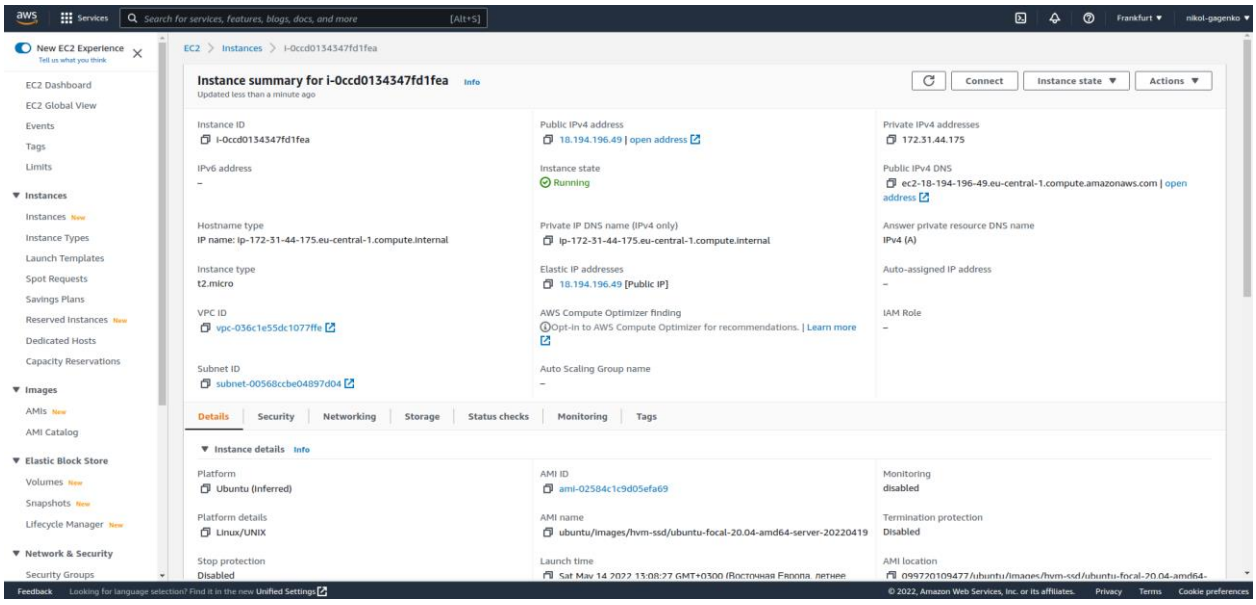

Рисунок 3.29 – Панель керування EC2

Також екземпляру було надано еластичну IP адресу (18.194.196.49), яка відіграє роль статичної адреси яка не змінюється при перезавантаженні екземпляру і доменне ім'я (ec2-18-194-196-49.eu-central-1.compute.amazonaws.com).

Операційною системою було обрано Ubuntu Server 20.04, ця версія не має графічної оболонки, що добре впливає на швидкість роботи серверу. Налаштування та керування сервером здійснювалося через протокол SSH.

EC2 екземпляр вже містить брандмауер тому на самому сервері його налаштовувати не треба, оскільки можуть виникнути конфлікти.

Першим чином треба було встановити програмний стек LEMP, комплекс програмного забезпечення, використовуваний для обслуговування динамічних веб-сторінок та веб-додатків. Абревіатура LEMP обумовлює операційну систему Linux з веб-сервером Nginx. Дані серверної частини містяться в базі даних MySQL, а динамічна обробка виконується PHP [14].

Спочатку було завантажено та налаштовано веб-сервер Nginx. Після завантаження, для перевірки роботи веб-серверу було відкрито браузер та здійснено перехід за адресою серверу. Відкрилася привітальна сторінка веб-серверу, а значить сервер запустився вдало.

Далі було встановлено і налаштовано СУБД MySQL та створено базу даних проекту. Для адміністрування бази даних веб-додатку було створено додаткового користувача, йому було надано повний доступ для бази даних веб-додатку.

Наступним кроком було встановлення PHP. Nginx вимагає окреме встановлення PHP та налаштування для взаємодії між самим інтерпретатором PHP та веб-сервером. Це забезпечує більш високу продуктивність веб-сайтів на базі PHP, але для цього необхідно виконати додаткове налаштування. Було встановлено диспетчер процесів PHP fastCGI. Також було встановлено php-mysql модуль, який дозволяє PHP взаємодіяти з базами даних MySQL. Інші ключові пакети автоматично встановилися в якості залежностей [14].

Далі було створено файл конфігурації Nginx (рис. 3.30).

```
 GNU nano 4.8                                                    /etc/nginx/sites-available/laravel
server {

    server_name ec2-18-194-196-49.eu-central-1.compute.amazonaws.com www.ec2-18-194-196-49.eu-central-1.compute.amazonaws.com;
    server_name onlineapteka.ml www.onlineapteka.ml;

    root /var/www/laravel/public;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-Content-Type-Options "nosniff";

    index index.php;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    error_page 404 /index.php;

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        include fastcgi_params;
     }

    location ~ /\.(?!well-known).* {
        deny all;
    }
```

Рисунок 3.30 – Файл конфігурації Nginx

Після виконання всіх налаштувань, за допомогою команди git pull origin master в кореневий веб-каталог було завантажено файли проекту з репозиторію GitHub.

Далі було внесено необхідні зміни у файл конфігурації проекту Laravel .env (рис. 3.31).

```
 GNU nano 4.8                                                          /var/www/laravel/.env
APP_NAME=Laravel
APP_ENV=prodaction
APP_KEY=base64:xXGkF7cT8IqXYWn+TDb040l0O8LGz39CoZAwKnI
APP_DEBUG=false
APP_URL=http://ec2-18-194-196-49.eu-central-1.compute.amazonaws.com/

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=laravel_admin
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DRIVER=public
QUEUE_CONNECTION=database
```

Рисунок 3.31 – Файл конфігурації Laravel

Далі було запущено команду composer install, щоб встановити всі залежності фреймворку.

Наступним етапом було запущено міграції та сидери, для створення таблиць та заповнення їх початковими даними. На цьому процес розгортання вебдодатку на сервер завершився. Доступ до сайту можна отримати за посиланням: http://ec2-18-194-196-49.eu-central-1.compute.amazonaws.com/ або QR-кодом (рис. 3.32).

Рисунок 3.32 – QR-код посилання на сайт

Таким чином було створено та протестовано сайт інтернет-аптеки який повністю відповідає поставленим задачам. Мануальне тестування дало позитивні оцінки, всі знайдені помилки було виправлені.

ВИСНОВКИ

У роботі розроблено та створено веб-сайт онлайн аптеки.

Створений веб-додаток має реалізованим наступний функціонал:

- можливість слідкувати за існуючими позиціями товарів та контролювати їх зміст;
- ведення обліку замовлень;
- формування замовлення;
- перегляд найпопулярніших товарів;
- можливість робити підписку на товар який закінчився;
- механізми надсилання повідомлень на електронну пошту користувача.
- локалізація веб-сайту (українська мова, англійська мова);
- мультивалютність веб-сайту (гривня, долар, євро);
- контроль над обліковими записами користувачів;
- можливість редагування облікового запису користувачем.

В результаті виконання кваліфікаційної роботи було виконано наступні поставлені задачі:

1) виконано огляд існуючих інструментів для розробки сучасних веб-додатків;

2) створено веб-додаток який має інтуїтивний, зрозумілий та адаптивний інтерфейс;

3) спроектовано та реалізовано інформаційну систему для веб-сайту інтернет аптеки;

4) робота по створенню дипломного проекту велася згідно визначеному алгоритму;

5) розроблено та протестовано UI та UX дизайн веб-сайту;

6) було проведено літературний огляд сучасних джерел інформації по тематиці розробки веб-додатків та веб-сатйів і на основі отриманої інформації було виконано розробку веб-додатку інтернет-магазину.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Retail e-commerce sales worldwide from 2014 to 2025 [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/.

2. Global Ecommerce Sales (2020–2025) [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.shopify.com/blog/global-ecommerce-sales.

3. Підсумки ринку електронної комерції в Україні [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://ain.ua/ru/2020/12/25/pidsumky-2020-evo/.

4. Mobile E-commerce [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.statista.com/chart/13139/estimated-worldwide-mobile-e-commerce-sales/.

5. Find out how you stack up to new industry benchmarks for mobile page speed [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/.

6. Ecommerce Stats That Will Make You Change Your Entire Marketing Approach [Електронний ресурс] – Режим доступу до ресурсу: https://neilpatel.com/blog/5-ecommerce-stats/.

7. Ecommerce Email Marketing Statistics By Industry [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.smartrmail.com/blog/ecommerce-email-marketing-statistics-by-industry/.

8. Principles of a Great eCommerce Site [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.customfitonline.com/news/2019/10/30/principles-of-great-ecommerce-sites-infographic/.

9. Amazon Elastic Compute Cloud [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud#Amazon_CloudWatch.

10. Laravel [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Laravel.

11. Composer (software) [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Composer_(software).

12. Bootstrap (front-end framework)[Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework).

13. How to SEO an eCommerce Site Structure [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.reliablesoft.net/ecommerce-site-structure/#comments

14. Установка Linux, Nginx, MySQL, PHP (стека LEMP) в Ubuntu 20.04 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-on-ubuntu-20-04-ru

15. Laravel Queues [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://laravel.com/docs/8.x/queues#main-content

ДОДАТОК А

Діаграма зв’язків між таблицями бази даних проекту

ДОДАТОК Б

Програмний код

Основні контролери:

MainController:

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProductsFilterRequest;
use App\Http\Requests\SubscriptionRequest;
use App\Models\Category;
use App\Models\Currency;
use App\Models\Product;
use App\Models\Sku;
use App\Models\Subscription;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\App;

class MainController extends Controller
{
    public function index(ProductsFilterRequest $request)
    {
        $skusQuery = Sku::with(['product', 'product.category']);

        if ($request->filled('price_from')) {
            $skusQuery->where('price', '>=', $request->price_from);
        }

        if ($request->filled('price_to')) {
            $skusQuery->where('price', '<=', $request->price_to);
        }

        foreach (['hit', 'new', 'recommend'] as $field) {
            if ($request->has($field)) {
                $skusQuery->whereHas('product', function ($query) use
($field) {
                    $query->$field();
                });
            }
        }

        if ($request->filled('search')) {
            $skusQuery->join('products', 'products.id', '=',
'skus.product_id')->where('products.name', 'like',
                '%' . $request->search . '%');
        }

        if ($skusQuery->get()->isEmpty()) {
            session()->now('warning', __('main.not_found'));
        }

        $skus = $skusQuery->paginate(12)->withPath("?" . $request-
>getQueryString());

        return view('index', compact('skus'));
    }
```

```
    public function categories()
    {
        return view('categories');
    }

    public function category($code)
    {
        $category = Category::where('code', $code)->first();

        if (is_null($category)) {
            abort(404, 'Page not found');
        }

        return view('category', compact('category'));
    }

    public function sku($categoryCode, $productCode, Sku $sku)
    {
        if ($sku->product->code != $productCode) {
            abort(404, 'Product not found');
        }

        if ($sku->product->category->code != $categoryCode) {
            abort(404, 'Category not found');
        }
        return view('product', compact('sku'));
    }

    public function subscribe(SubscriptionRequest $request, Sku $sku)
    {
        Subscription::create([
            'email' => $request->email,
            'sku_id' => $sku->id,
        ]);
        return                          redirect()->back()->with('success',
__('mail/subscription.thanks_for_subscription'));
    }

    public function changeLocale($locale)
    {
        $availableLocales = ['ua', 'en'];
        if (!in_array($locale, $availableLocales)) {
            $locale = config('app.locale');
        }
        session(['locale' => $locale]);
        App::setLocale($locale);
        return redirect()->back();
    }

    public function changeCurrency($currencyCode)
    {
        $currency = Currency::byCode($currencyCode)->firstOrFail();
        session(['currency' => $currency->code]);
        return redirect()->back();
    }

    public function about_us()
    {
        return view('about_us');
    }
}
```

BasketController:

```
<?php

namespace App\Http\Controllers;

use App\Classes\Basket;
use App\Models\Order;
use App\Models\Product;
use App\Models\Sku;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class BasketController extends Controller
{
    public function basket()
    {
        $order = (new Basket())->getOrder();
        return view('basket', compact('order'));
    }

    public function basketConfirm(Request $request)
    {
        $email = Auth::check() ? Auth::user()->email : $request->email;
        if ((new Basket())->saveOrder($request->name, $request->phone,
$email)) {
            session()->flash('success',
__('basket.you_order_confirmed'));
        } else {
            session()->flash('warning',
__('basket.you_cant_order_more'));
        }

        return redirect()->route('index');
    }

    public function basketPlace()
    {
        $basket = new Basket();
        $order = $basket->getOrder();
        if (!$basket->countAvailable()) {
            session()->flash('warning',
__('basket.you_cant_order_more'));
            return redirect()->route('basket');
        }
        return view('order', compact('order'));
    }

    public function basketAdd(Sku $sku)
    {
        $result = (new Basket(true))->addSku($sku);

        if ($result) {
            session()->flash('success',         __('basket.added').$sku-
>product->__('name'));
        } else {
            session()->flash('warning',    $sku->product->__('name')    .
__('basket.not_available_more'));
        }

        return redirect()->route('basket');
    }

    public function basketRemove(Sku $sku)
    {
        (new Basket())->removeSku($sku);
```

```
            session()->flash('warning',  __('basket.removed').$sku->product-
>__('name'));

            return redirect()->route('basket');
        }
    }
```

OrderController:

```
<?php

namespace App\Http\Controllers\Person;

use App\Http\Controllers\Controller;
use App\Models\Order;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class OrderController extends Controller
{
    public function index()
    {
        $orders = Auth::user()->orders()->active()->paginate(10);

        return view('auth.orders.index', compact('orders'));
    }

    public function show(Order $order)
    {
        if (!Auth::user()->orders->contains($order)) {
            return redirect()->route('person.orders.index');
        }

        $skus = $order->skus()->withTrashed()->get();

        return view('auth.orders.show', compact('order', 'skus'));
    }
}
```

SkuController:

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\SkuRequest;
use App\Models\Product;
use App\Models\Sku;
use Illuminate\Support\Facades\Storage;

class SkuController extends Controller
{
    public function index(Product $product)
    {
        $skus = $product->skus()->paginate(10);
        return view('auth.skus.index', compact('skus', 'product'));
    }

    public function create(Product $product)
    {
        return view('auth.skus.form', compact('product'));
    }
```

```
        public function store(SkuRequest $request, Product $product)
        {
            $params = $request->all();
            unset($params['image']);
            if ($request->has('image')) {
                $path     =     $request->file('image')->store('images/prod-
ucts_apteka');
                $params['image'] = $path;
            } else {
                $path = 'images/image-not-found.png';
                $params['image'] = $path;
            }
            $params['product_id'] = $request->product->id;
            $sku = Sku::create($params);
            $sku->propertyOptions()->sync($request->property_id);
            return redirect()->route('skus.index', compact('product'));
        }

        public function show(Product $product, Sku $sku)
        {
            return view('auth.skus.show', compact('product', 'sku'));
        }

        public function edit(Product $product, Sku $sku)
        {
            return view('auth.skus.form', compact('product', 'sku'));
        }

        public  function  update(SkuRequest  $request,  Product  $product,  Sku
$sku)
        {
            $params = $request->all();
            unset($params['image']);
            if ($request->has('image')) {
                Storage::delete($sku->image);
                $path     =     $request->file('image')->store('images/prod-
ucts_apteka');
                $params['image'] = $path;
            }

            $params['product_id'] = $request->product->id;
            $sku->update($params);
            $sku->propertyOptions()->sync($request->property_id);
            return redirect()->route('skus.index', compact('product'));
        }

        public function destroy(Product $product, Sku $sku)
        {
            $sku->delete();

            return redirect()->route('skus.index', compact('product'));
        }
    }
```

CategoryController:

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Http\Requests\CategoryRequest;
use App\Models\Category;
use Illuminate\Contracts\Foundation\Application;
```

```
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Storage;

class CategoryController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return Application|Factory|View
     */
    public function index()
    {
        $categories = Category::paginate(10);
        return view('auth.categories.index', compact('categories'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return Application|Factory|View
     */
    public function create()
    {
        return view('auth.categories.form');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param Request $request
     * @return RedirectResponse
     */
    public function store(CategoryRequest $request)
    {
        $params = $request->all();
        unset($params['image']);
        if ($request->has('image')) {
            $path = $request->file('image')->store('images/categories');
            $params['image'] = $path;
        } else {
            $path = 'images/image-not-found.png';
            $params['image'] = $path;
        }

        Category::create($params);
        return redirect()->route('categories.index');
    }

    /**
     * Display the specified resource.
     *
     * @param Category $category
     * @return Application|Factory|View
     */
    public function show(Category $category)
    {
        return view('auth.categories.show', compact('category'));
    }

    /**
```

```
     * Show the form for editing the specified resource.
     *
     * @param Category $category
     * @return Application|Factory|View
     */
    public function edit(Category $category)
    {
        return view('auth.categories.form', compact('category'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param Request $request
     * @param Category $category
     * @return RedirectResponse
     */
    public function update(CategoryRequest $request, Category $category)
    {
        $params = $request->all();
        unset($params['image']);
        if ($request->has('image')) {
            Storage::delete($category->image);
            $path = $request->file('image')->store('images/categories');
            $params['image'] = $path;
        }

        $category->update($params);
        return redirect()->route('categories.index');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param Category $category
     * @return RedirectResponse
     */
    public function destroy(Category $category)
    {
        $category->delete();
        return redirect()->route('categories.index');
    }
}
```

ProductController:

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Http\Requests\ProductRequest;
use App\Models\Category;
use App\Models\Product;
use App\Models\Property;
use Illuminate\Contracts\Foundation\Application;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class ProductController extends Controller
{
```

```
        /**
         * Display a listing of the resource.
         *
         * @return Application|Factory|View
         */
        public function index()
        {
            $products = Product::paginate(7);
            return view('auth.products.index', compact('products'));
        }

        /**
         * Show the form for creating a new resource.
         *
         * @return Application|Factory|View
         */
        public function create()
        {
            $categories = Category::get();
            $properties = Property::get();
            return view('auth.products.form', compact('categories', 'proper-
ties'));
        }

        /**
         * Store a newly created resource in storage.
         *
         * @param ProductRequest $request
         * @return RedirectResponse
         */
        public function store(ProductRequest $request)
        {
            $params = $request->all();
            unset($params['instruction']);
            if ($request->has('instruction')) {
                $path    =    $request->file('instruction')->store('instruc-
tions');
                $params['instruction'] = $path;
            }

            Product::create($params);
            return redirect()->route('products.index');
        }

        /**
         * Display the specified resource.
         *
         * @param Product $product
         * @return Application|Factory|View
         */
        public function show(Product $product)
        {
            return view('auth.products.show', compact('product'));
        }

        /**
         * Show the form for editing the specified resource.
         *
         * @param Product $product
         * @return Application|Factory|View
         */
        public function edit(Product $product)
        {
            $categories = Category::get();
```

```
            $properties = Property::get();
            return   view('auth.products.form',   compact('product',   'cate-
gories', 'properties'));
        }

        /**
         * Update the specified resource in storage.
         *
         * @param Request $request
         * @param Product $product
         * @return RedirectResponse
         */
        public function update(ProductRequest $request, Product $product)
        {
            $params = $request->all();
            unset($params['instruction']);
            if ($request->has('instruction')) {
                Storage::delete($product->instruction);
                $path    =    $request->file('instruction')->store('instruc-
tions');
                $params['instruction'] = $path;
            }

            foreach (['new', 'hit', 'recommend'] as $fieldName) {
                if (!isset($params[$fieldName])) {
                    $params[$fieldName] = 0;
                }
            }

            $product->properties()->sync($request->property_id);

            $product->update($params);
            return redirect()->route('products.index');
        }

        /**
         * Remove the specified resource from storage.
         *
         * @param Product $product
         * @return RedirectResponse
         */
        public function destroy(Product $product)
        {
            $product->delete();
            $product->skus()->delete();
            return redirect()->route('products.index');
        }
    }
```

Web-routes:

```
<?php

use App\Http\Controllers\Admin\CategoryController;
use App\Http\Controllers\Admin\OrderController;
use App\Http\Controllers\Admin\ProductController;
use App\Http\Controllers\Admin\PropertyController;
use App\Http\Controllers\Admin\PropertyOptionController;
use App\Http\Controllers\Admin\UserController;
use App\Http\Controllers\Auth\LoginController;
use App\Http\Controllers\BasketController;
use App\Http\Controllers\ChangeProfileController;
use App\Http\Controllers\MainController;
use App\Http\Controllers\ResetController;
```

```
use App\Http\Controllers\SkuController;
use App\Http\Middleware\Authenticate;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('locale/{locale}', [MainController::class, 'changeLocale'])->name('locale');
Route::get('currency/{currencyCode}', [MainController::class, 'changeCurrency'])->name('currency');

Route::get('/logout', [LoginController::class, 'logout'])->name('get-logout');

Route::middleware(['set_locale'])->group(function () {

    Auth::routes([
        'confirm' => false,
        'verify' => false,
    ]);

    Route::middleware([Authenticate::class])->group(function () {
        Route::group([
            'prefix' => 'person',
            'as' => 'person.',
        ], function () {
            Route::get('/orders', [App\Http\Controllers\Person\OrderController::class, 'index'])->name('orders.index');
            Route::get('/orders/{order}',
                [App\Http\Controllers\Person\OrderController::class, 'show'])->name('orders.show');
        });

        Route::group([
            'prefix' => 'admin',
        ], function () {
            Route::group(['middleware' => 'is_admin'], function () {
                Route::get('/orders', [OrderController::class, 'index'])->name('home');
                Route::get('/orders/{order}', [OrderController::class, 'show'])->name('orders.show');
                Route::resource('categories', CategoryController::class);
                Route::resource('products', ProductController::class);
                Route::resource('products/{product}/skus', SkuController::class);
                Route::resource('properties', PropertyController::class);
                Route::resource('properties/{property}/property-options', PropertyOptionController::class);
                Route::resource('users', UserController::class);
```

```
                    Route::get('/users/{user}/orders',         [UserControl-
ler::class, 'orders'])->name('user_orders');
                    Route::get('reset', [ResetController::class, 'reset'])-
>name('reset');
                });
            });
            Route::resource('profile', ChangeProfileController::class);
        });


        Route::get('/', [MainController::class, 'index'])->name('index');
        Route::get('/categories',  [MainController::class,  'categories'])-
>name('categories');
        Route::post('subscription/{sku}',   [MainController::class,   'sub-
scribe'])->name('subscription');
        Route::get('/about-us',    [MainController::class,    'about_us'])-
>name('about_us');

        Route::group(['prefix' => 'basket'], function () {
            Route::post('/add/{sku}',   [BasketController::class,   'basket-
Add'])->name('basket-add');

            Route::group([
                'middleware' => 'basket_not_empty',
            ], function () {
                Route::get('/',    [BasketController::class,    'basket'])-
>name('basket');
                Route::get('/place',   [BasketController::class,   'basket-
Place'])->name('basket-place');
                Route::post('/remove/{sku}', [BasketController::class, 'bas-
ketRemove'])->name('basket-remove');
                Route::post('/place', [BasketController::class, 'basketCon-
firm'])->name('basket-confirm');
            });
        });

        Route::get('/{category}',   [MainController::class,   'category'])-
>name('category');
        Route::get('/{category}/{product?}/{sku}',   [MainController::class,
'sku'])->name('sku');
    });
```

Основні view:

layout master:

```
<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="utf-8">
    <meta    name="viewport"    content="width=device-width,    initial-
scale=1.0">

    <title>@lang('main.online_shop'): @yield('title')</title>

    <link           href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
          integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrB-
nVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor" crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="/bootstrap5_2/css/main-
navbar.css">
```

```
          <link rel="apple-touch-icon" sizes="180x180" href="/img/apple-touch-
icon.png">
          <link rel="icon" type="image/png" sizes="32x32" href="/img/favicon-
32x32.png">
          <link rel="icon" type="image/png" sizes="16x16" href="/img/favicon-
16x16.png">
          <link rel="manifest" href="/img/site.webmanifest">

          <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.8.2/font/bootstrap-icons.css">

      </head>
      <nav class="navbar navbar-expand-lg">
          <div class="container">
              <a       class="navbar-brand"       href="{{       route('index')
}}">@lang('main.online_shop')</a>
              <button class="navbar-toggler custom-toggler" type="button" data-
bs-toggle="collapse"
                      data-bs-target="#navbarSupportedContent"
                      aria-controls="navbarSupportedContent"          aria-ex-
panded="false" aria-label="Toggle navigation">
                  <span class="navbar-toggler-icon"></span>
              </button>
              <div  class="collapse  navbar-collapse  "  id="navbarSupportedCon-
tent">
                  <ul  class="navbar-nav  me-auto  mb-2  mb-lg-0  d-flex  justify-
content-lg-center align-items-lg-center">
                      <li class="nav-item">
                          <a class="nav-link" aria-current="page"
                             href="{{ route('index') }}">@lang('main.all_prod-
ucts')</a>
                      </li>
                      <li class="nav-item">
                          <a  class="nav-link"  href="{{  route('categories')
}}">@lang('main.categories')</a>
                      </li>
                      <li class="nav-item">
                          <a    class="nav-link"    href="{{    route('basket')
}}">@lang('main.cart')</a>
                      </li>
                      <li class="nav-item">
                          <a   class="nav-link"   href="{{   route('about_us')
}}">@lang('main.about_us')</a>
                      </li>
                      <li class="nav-item">
                          <a    class="nav-link"    href="{{    route('locale',
__('main.set_lang')) }}">
                              @if(Lang::locale() == 'ua')
                                  <img       src="/img/us.png"       alt="EN"
style="width:20px;height:20px;">
                              @else
                                  <img       src="/img/ua.png"       alt="RU"
style="width:20px;height:20px;">
                              @endif</a>
                      </li>
                      <li class="nav-item dropdown">
                          <a    class="nav-link    dropdown-toggle"    href="#"
id="navbarDropdown" role="button"
                             data-bs-toggle="dropdown" aria-expanded="false">
                              {{ $currencySymbol }}
                          </a>
                          <ul  class="dropdown-menu"  aria-labelledby="navbar-
Dropdown" id="currencys-dropdown">
                              @foreach ($currencies as $currency)
```

```
                                    <li><a class="dropdown-item"
                                           href="{{ route('currency', $currency-
>code) }}">{{ $currency->symbol }}</a></li>
                                @endforeach
                            </ul>
                        </li>
                    </ul>
                    <form   class="d-flex"   method="GET"   role="search"   ac-
tion="{{route("index")}}">
                        <input  class="form-control  me-2"  type="search"  place-
holder="@lang('main.search')" aria-label="Search"
                               name="search" id="search">
                    </form>
                    <ul class="navbar-nav d-flex justify-content-lg-center align-
items-lg-center">
                        @guest
                            <li  class="nav-item"><a  class="nav-link"  href="{{
route('login') }}">@lang('main.login')</a></li>
                        @endguest

                        @auth
                            @admin
                            <li  class="nav-item"><a  class="nav-link"  href="{{
route('home') }}">@lang('main.admin_panel')</a>
                            </li>
                        @else
                            <li class="nav-item"><a class="nav-link"
                                                    href="{{   route('person.or-
ders.index') }}">@lang('main.my_orders')</a></li>
                            @endadmin
                            <li  class="nav-item"><a  class="nav-link"  href="{{
route('get-logout') }}">@lang('main.logout')</a>
                            </li>
                        @endauth
                    </ul>
                </div>
            </div>
        </nav>

        <div id="main">
            <div class="container">
                @if(session()->has('success'))
                    <div  class="text-center  py-1"><p  class="alert  alert-suc-
cess">{{ session()->get('success') }}</p></div>
                @endif
                @if(session()->has('warning'))
                    <div  class="text-center  py-1"><p  class="alert  alert-warn-
ing">{{ session()->get('warning') }}</p></div>
                @endif
            </div>
            @yield('content')
        </div>
        <div class="container">
            <footer  class="row  justify-content-center  row-cols-1  row-cols-sm-2
row-cols-md-5 pb-5 mb-5 pt-3 mt-3 border-top">
                <div class="col mb-3">
                    <h5>@lang('main.categories')</h5>
                    <ul class="nav flex-column">
                        @foreach($categories as $category)
                            <li class="nav-item mb-2"><a href="{{ route('catego-
ry', $category->code) }}"
                                                         class="nav-link     p-0
text-muted">{{ $category->__('name') }}</a></li>
                        @endforeach
```

```
                </ul>
            </div>
            <div class="col mb-3">
                <h5>@lang('main.popular_products')</h5>
                <ul class="nav flex-column">
                    @foreach ($bestSkus as $bestSku)
                        <li class="nav-item mb-2"><a
                                href="{{  route('sku',  [$bestSku->product-
>category->code, $bestSku->product->code, $bestSku]) }}"
                                class="nav-link p-0 text-muted">{{ $bestSku-
>product->__('name') }}</a></li>
                    @endforeach
                </ul>
            </div>
        </footer>
    </div>
    <script              src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-
beta1/dist/js/bootstrap.bundle.min.js"
            integrity="sha384-pprn3073KE6tl6bjs2QrFaJGz5/SUsLqkti-
wsUTF55Jfv3qYSDhgCecCxMW52nD2"
            crossorigin="anonymous"></script>
    <script src="/bootstrap5_2/js/main-navbar.js"></script>
    </body>
    </html>
```

main index view:

```
@extends('layouts.master')

@section('title', __('main.title'))

@section('content')
    <header  class="container  text-center  fs-2  py-2">@lang('main.all_prod-
ucts')</header>

    <div class="container">
        <div class="pb-2 d-md-none text-end">
            <button  class="btn  btn-outline-dark"  type="button"  data-bs-tog-
gle="offcanvas"
                    data-bs-target="#offcanvasRight"
                    aria-controls="offcanvasRight">@lang('main.filter')
            </button>
        </div>

        <div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasRight"
aria-labelledby="offcanvasRightLabel">
            <div class="offcanvas-header">
                <h5            class="offcanvas-title"            id="offcan-
vasRightLabel">@lang('main.filter')</h5>
                <button type="button" class="btn-close" data-bs-dismiss="off-
canvas" aria-label="Close"></button>
            </div>
            <div class="offcanvas-body">
                <form method="GET" action="{{route("index")}}">
                    <div class="mb-3">
                        <label        for="price_from"        class="form-la-
bel">@lang('main.price_from')</label>
                        <input  type="text"  name="price_from"  id="price_from"
class="form-control"
                               value="{{ request()->price_from}}">
                    </div>
                    <div class="mb-3">
                        <label         for="price_to"         class="form-la-
bel">@lang('main.to')</label>
```

```
                        <input   type="text"   name="price_to"   id="price_to"
class="form-control"
                               value="{{ request()->price_to }}">
                    </div>
                    <div class="mb-3">
                        <div class="form-check">
                            <label class="form-check-label" for="new">
                                @lang('main.properties.new')
                            </label>
                            <input   type="checkbox"     name="new"   id="new"
class="form-check-input"
                                   @if(request()->has('new')) checked @endif>
                        </div>
                    </div>
                    <div class="mb-3">
                        <div class="form-check">
                            <input   type="checkbox"   class="form-check-input"
name="hit" id="hit"
                                   @if(request()->has('hit')) checked @endif>
                            <label class="form-check-label" for="hit">
                                @lang('main.properties.hit')
                            </label>
                        </div>
                    </div>
                    <div class="mb-3">
                        <div class="form-check">
                            <input   type="checkbox"   class="form-check-input"
name="recommend" id="recommend"
                                   @if(request()->has('recommend'))    checked
@endif>
                            <label class="form-check-label" for="recommend">
                                @lang('main.properties.recommend')
                            </label>
                        </div>
                    </div>
                    <div   class="btn-group"   role="group"   aria-label="Basic
mixed styles example">
                        <button     type="submit"     class="btn     btn-pri-
mary">@lang('main.filter')</button>
                        <a  href="{{  route("index")  }}"  class="btn  btn-warn-
ing">@lang('main.reset')</a>
                    </div>
                </form>
            </div>
        </div>

        <div class="d-none d-md-block">
            <form method="GET" action="{{route("index")}}">
                <div class="row py-3 align-items-center text-center">
                    <div class="col-4">
                        <div class="input-group">
                            <span                          class="input-group-
text">@lang('main.price_from')</span>
                            <input      type="text"      class="form-control"
name="price_from" id="price_from" value="{{ request()->price_from}}">
                            <span                          class="input-group-
text">@lang('main.to')</span>
                            <input      type="text"      class="form-control"
name="price_to" id="price_to" value="{{ request()->price_to }}">
                        </div>
                    </div>
                    <div class="col-auto">
                        <input    class="form-check-input"    type="checkbox"
name="new" id="new-lg"
```

```
                                @if(request()->has('new')) checked @endif>
                            <label class="form-check-label" for="new-lg">
                                @lang('main.properties.new')
                            </label>
                        </div>
                        <div class="col-auto">
                            <input    class="form-check-input"    type="checkbox"
name="hit" id="hit-lg"
                                @if(request()->has('hit')) checked @endif>
                            <label class="form-check-label" for="hit-lg">
                                @lang('main.properties.hit')
                            </label>
                        </div>
                        <div class="col-md-auto">
                            <input    class="form-check-input"    type="checkbox"
name="recommend" id="recommend-lg"
                                @if(request()->has('recommend'))  checked  @en-
dif>
                            <label class="form-check-label" for="recommend-lg">
                                @lang('main.properties.recommend')
                            </label>
                        </div>

                        <div class="col text-end">
                            <div class="btn-group-vertical d-lg-none" role="group"
aria-label="Basic mixed styles example">
                                <button    type="submit"    class="btn    btn-pri-
mary">@lang('main.filter')</button>
                                <a href="{{ route("index") }}" class="btn btn-warn-
ing">@lang('main.reset')</a>
                            </div>
                            <div class="btn-group d-none d-lg-block" role="group"
aria-label="Basic mixed styles example">
                                <button    type="submit"    class="btn    btn-pri-
mary">@lang('main.filter')</button>
                                <a href="{{ route("index") }}" class="btn btn-warn-
ing">@lang('main.reset')</a>
                            </div>
                        </div>
                    </div>
                </form>
            </div>
        </div>
        <div class="container">
            <div
                class="row  row-cols-2  row-cols-sm-2  row-cols-md-3  row-cols-lg-4
row-cols-xl-6
                g-3 g-sm-3 g-md-3 g-lg-3 g-xl-3">
                @foreach($skus as $sku)
                    @include('layouts.card', compact('sku'))
                @endforeach
            </div>
        </div>
        <div class="d-flex justify-content-center pt-4">
            {{$skus->links('pagination::bootstrap-4')}}
        </div>
@endsection
```

basket view:

```
@extends('layouts.master')

@section('title', __('basket.cart'))
```

```
@section('content')
    <div class="container  py-3">
        <h1>@lang('basket.cart')</h1>
        <p>@lang('basket.ordering')</p>

        <table class="table table-striped align-middle">
            <thead>
            <tr>
                <th>@lang('basket.name')</th>
                <th>@lang('basket.count')</th>
                <th>@lang('basket.price')</th>
                <th>@lang('basket.cost')</th>
            </tr>
            </thead>
            <colgroup>
                <col style="width: 30%"/>
            </colgroup>
            <tbody>
            @foreach($order->skus as $sku)
                <tr>
                    <td style="white-space: nowrap; text-overflow:ellip-
sis; overflow: hidden; max-width:1px;">
                        <div class="row row-cols-1 row-cols-sm-1">
                            <div class="col">
                                <a
                                    href="{{  route('sku',  [$sku->prod-
uct->category->code, $sku->product->code, $sku]) }}">
                                    <img  height="56px"  src="{{  Stor-
age::url($sku->image) }}">
                                </a>
                            </div>
                            <div class="col">
                                <div class="text-truncate"><a
                                        href="{{   route('sku',   [$sku-
>product->category->code, $sku->product->code, $sku]) }}">
                                        {{ $sku->product->__('name') }}
                                    </a></div>
                            </div>
                        </div>
                    </td>
                    <td>
                        <div class="d-flex flex-column flex-sm-row align-
items-center">
                            <span
                                class="badge  bg-secondary  align-items-
center justify-content-sm-center h-100 m-2">{{ $sku->countInOrder }}</span>
                            <div  class="btn-group-vertical  align-items-
center justify-content-sm-center h-100">
                                <form  action="{{  route('basket-remove',
$sku) }}" method="POST">
                                    <button   type="submit"   class="btn
btn-danger">
                                        <svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16"
                                             fill="currentColor"
                                             class="bi bi-dash-lg" view-
Box="0 0 16 16">
                                            <path fill-rule="evenodd"
                                                  d="M2 8a.5.5 0 0 1 .5-
.5h11a.5.5 0 0 1 0 1h-11A.5.5 0 0 1 2 8Z"/>
                                        </svg>@csrf
                                    </button>
                                </form>
```

```
                                        <form   action="{{   route('basket-add',
$sku) }}" method="POST">
                                            <button   type="submit"   class="btn
btn-success">
                                                <svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16"
                                                     fill="currentColor"
                                                     class="bi bi-plus-lg" view-
Box="0 0 16 16">
                                                   <path fill-rule="evenodd"
                                                         d="M8  2a.5.5  0  0  1
.5.5v5h5a.5.5 0 0 1 0 1h-5v5a.5.5 0 0 1-1 0v-5h-5a.5.5 0 0 1 0-1h5v-5A.5.5 0 0
1 8 2Z"/>
                                                </svg>
                                            </button>@csrf
                                        </form>
                                    </div>
                                </div>
                            </td>
                            <td>{{ $sku->price }} {{ $currencySymbol }}</td>
                            <td>{{ $sku->price * $sku->countInOrder }} {{ $cur-
rencySymbol }}</td>
                        </tr>
                    @endforeach
                    <tr>
                        <td colspan="3">@lang('basket.full_cost'):</td>
                        <td>{{ $order->getFullSum() }} {{ $currencySymbol }}</td>
                    </tr>
                    </tbody>
                </table>
                <br>
                <div class="btn-group pull-right" role="group">
                    <a type="button" class="btn btn-success" href="{{ route('bas-
ket-place') }}">@lang('basket.place_order')</a>
                </div>
                <div class="btn-group pull-right my-2 my-sm-0" role="group">
                    <a type="button" class="btn btn-primary" href="{{ route('in-
dex') }}">@lang('main.continue_shopping')</a>
                </div>
            </div>
        @endsection
```

product view:

```
@extends('layouts.master')

@section('title', __('main.product'))

@section('content')
    <div class="container">
        <h3 class="p-3">{{ $sku->product->category->__('name') }} / {{ $sku-
>product->__('name') }}</h3>
        <div class="row row-cols-1 row-cols-md-2">
            <div class="col-12 col-md-6">
                <div class="product-img">
                    <img  id="product-img"  src="{{  Storage::url($sku->image)
}}">
                </div>
                <div class="container p-3">
                    <div class="row">
                        <div class="col p-3 text-start">
                            <h2>{{ $sku->product->__('name') }}</h2>
                        </div>
                        <div class="w-100"></div>
```

```
                        <div class="col p-3">
                            @isset($sku->product->properties)
                                @foreach($sku->propertyOptions as $propertyOp-
tion)
                                    <h6  class="text-start">{{$propertyOption-
>property->__('name')}}
                                        : {{$propertyOption->__('name')}}</h6>
                                @endforeach
                            @endisset
                        </div>
                    </div>
                </div>

                <div class="row cols-3 row-cols-2 p-3">
                    <div class="col">
                        <p  class="text-start">@lang('product.price'):  <b>{{
$sku->price }} {{ $currencySymbol }}</b></p>
                    </div>
                    <div class="col text-end">
                        @if($sku->isAvailable())
                            <form  action="{{  route('basket-add',  $sku)  }}"
method="POST">
                                <button type="submit" class="btn btn-success"
                                        role="button">@lang('prod-
uct.add_to_cart')</button>

                                @csrf
                            </form>
                        @else
                            <div class="warning text-start">
                                @if($errors->get('email'))
                                    {!! $errors->get('email')[0] !!}
                                @endif
                            </div>
                            <form  method="POST"  action="{{  route('subscrip-
tion', $sku) }}">
                                <div class="mb-3 text-start">
                                    <label for="exampleInputEmail1"
                                           class="form-label">@lang('prod-
uct.not_available')</label>
                                    <label            for="exampleInputEmail1"
class="form-label">@lang('product.tell_me'):</label>
                                    <input      type="email"      name="email"
class="form-control" id="exampleInputEmail1"
                                           aria-describedby="emailHelp">
                                </div>
                                <button   type="submit"   class="btn   btn-pri-
mary">@lang('product.subscribe')</button>
                                @csrf
                            </form>
                        @endif
                    </div>
                </div>
                <hr>
                <div class="row">
                    <div   class="col   text-start   p-3"><p>{{   $sku->product-
>__('description') }}</p></div>
                </div>
            </div>
            <div class="col-12 col-md-6">
                @if(!is_null($sku->product->instruction) || !empty($sku->prod-
uct->instruction)  ||  !is_null($sku->product->instruction_en)  ||  !empty($sku-
>product->instruction_en))
```

```
                    <iframe style="width: 100%; height: 80vh; border: 1px solid
#D1D1D1;"
                            src="{{   Storage::url($sku->product->__('instruc-
tion')) }}"></iframe>
                @else
                    <iframe style="width: 100%; height: 80vh; border: 1px solid
#D1D1D1;"
                            src="{{   Storage::url('instructions/not_found_in-
struction.html') }}"></iframe>
                @endif
            </div>
        </div>
    </div>
@endsection
```

about_us view:

```
@extends('layouts.master')

@section('title', 'About us')

@section('content')
    <div class="px-4 pt-5 mt-5 text-center">
        <img   class="d-block   mx-auto   mb-4"   src="/img/logo.svg"   alt=""
style="width: 10vh; height: 10vh">
        <h1 class="display-5 fw-bold">@lang('main.online_apteka')</h1>
        <div class="col-lg-6 mx-auto">
            <p class="lead mb-4">@lang('main.about_us_txt_0')</p>
            <div class="d-grid gap-2 d-sm-flex justify-content-sm-center">
                <button type="button" class="btn btn-primary btn-lg px-4 gap-
3">@lang('main.to_goods')</button>
            </div>
        </div>
    </div>
    <div class="container px-4 py-5" id="featured-3">
        <div class="row g-4 py-5 row-cols-1 row-cols-lg-3">
            <div class="feature col">
                <div class="feature-icon d-inline-flex align-items-center jus-
tify-content-center fs-2 mb-3">
                    <i class="bi bi-hand-thumbs-up"></i>
                </div>
                <h2>@lang('main.about_us_txt_1')</h2>
                <p>@lang('main.about_us_txt_1.1')</p>
                <a href="#" class="btn btn-success d-inline-flex align-items-
center">
                    @lang('main.to_goods')
                </a>
            </div>
            <div class="feature col">
                <div class="feature-icon d-inline-flex align-items-center jus-
tify-content-center fs-2 mb-3">
                    <i class="bi bi-hourglass-split"></i>
                </div>
                <h2>@lang('main.about_us_txt_2')</h2>
                <p>@lang('main.about_us_txt_2.1')</p>
                <a href="#" class="btn btn-success d-inline-flex align-items-
center">
                    @lang('main.to_goods')
                </a>
            </div>
            <div class="feature col">
                <div class="feature-icon d-inline-flex align-items-center jus-
tify-content-center fs-2 mb-3">
                    <i class="bi bi-line"></i>
```

```
                </div>
                <h2>@lang('main.about_us_txt_3')</h2>
                <p>@lang('main.about_us_txt_3.1')</p>
                <a href="#" class="btn btn-success d-inline-flex align-items-
center">
                    @lang('main.to_goods')
                </a>
            </div>
        </div>
    </div>
@endsection
```