Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ОНЛАЙН-МАГАЗИНУ СОЛОДОЩІВ «ТАЙМЧАК»

DEVELOPMENT OF THE ONLINE CANDY STORE "TAYMCHAK"

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗ-41
Тимчук Владислав Володимирович
_____
(підпис)

Керівник:
д.т.н., професор
Андрущак Ігор Євгенович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Н. Ліщина
«2» 02 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Тимчуку Владиславу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *Розробка онлайн-магазину солодощів «Таймчак»*

Керівник роботи: *д.т.н., професор, Андрущак Ігор Євгенович*
затверджені наказом закладу вищої освіти від «*30*» *грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу: *21 рисунок.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Андрущак І. Є.*		
Специфікації вимог до розробленої системи	*Андрущак І. Є.*		
Розробка об'єкта проєктування	*Андрущак І. Є.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту		4,0 %	
Академічна доброчесність	*Яцук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	вик
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2024 р.*	вик
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	вик
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2024 р.*	вик
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	вик
6	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	вик
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*05.06.2024 р.*	вик

Здобувач вищої освіти _____ (підпис) *Тимчук В. В.* (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Андрущак І. Є.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: *Тимчук Владислав Володимирович*

Тема: *«Розробка онлайн-магазину солодощів «Таймчак»»*

Коротка характеристика кваліфікаційної роботи: *Кваліфікаційна робота бакалавра складається з трьох розділів, в яких було проведено аналіз сучасного стану проблеми, чітко поставлено завдання, описана структура онлайн-магазину, та методи її реалізації, здійснено розробку і проведено тестування результатів.*

Самостійні розробки і пропозицій автора: *Автором був самостійно розроблений інтернет-магазин солодощів «Таймчак».*

Практичне значення роботи: *Полягає в проектуванні і розробці власного інтернет магазину солодощів «Таймчак».*

Недоліки: *Суттєвих недоліків в роботі не виявлено.*

Загальний висновок: *Кваліфікаційна робота бакалавра виконана на високому рівні. Матеріал викладено чітко і зрозуміло. Вважаю, що кваліфакаційна робота відповідає усім вимогам і заслуговує оцінки «відмінно», а її авторові, студенту Тимчуку В.В., може бути присвоєний кваліфікаційний рівень "бакалавр" зі спеціальності «Інженерія програмного забезпечення».*

Рецензент: к.т.н., доц. Кошельник В.І.

Рецензент кваліфікаційної роботи _____
(підпис) «07» 06 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач вищої освіти *Тимчук Владислав Володимирович*
(прізвище, ім'я, по батькові)
група *ІПЗ-41*

Тема кваліфікаційної роботи *«Розробка онлайн-магазину солодощів «Таймчак»»*

Керівник *Андрущак Ігор Євгенович*

Актуальність теми *Робота актуальна та своєчасна. Сайти, в тому числі і онлайн-магазини допомагають розвитку біснесу та просуванню власного бренду.*

Об'єкт дослідження *Онлайн-магазин солодощів «Таймчак».*

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи *Автор аналізує різні методи та технології розробки веб-застосунків. Чітко описує структуру сайту, а саме fron-end та back-end частини. Зміст роботи відповідає обраній темі.*

Зауваження та недоліки: *Суттєвих недоліків в роботі не виявлено.*

Загальний висновок: *Кваліфікаційна робота бакалавра виконана на високому рівні. Всі вимоги було дотримано. Вся робота описана логічно та послідовно. Пояснювальна записка відповідає усім стандартам оформлення.*

Керівник_____ **(Андрущак І. Є.)**
(підпис) (прізвище, ім'я, по батькові)

«07» 06 2024 р.

АНОТАЦІЯ

Тимчук В. В. Розробка онлайн-магазину солодощів «Таймчак». Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел (згідно структури кваліфікаційної роботи, затвердженої кафедрою).

У першому розділі здійснено аналіз сучасного стану проблеми розробки онлайн-магазинів і сформульовано завдання. В другому розділі визначено вимоги до розроблюваної системи, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено та проведено тестування онлайн-магазину. У висновках узагальнено інформацію, відображену в попередніх частинах. Результати розробки демонструють можливості обраних інструментів та технологій для веб-розробки.

Ключові слова: веб-застосунок, онлайн-магазин, back-end, front-end, база даних, веб-сервер, Docker, API, фреймворк, бібліотека, Laravel, React, Inertiajs.

ABSTRACT

Tymchuk V. V. Development of the Online Candy Store "Taymchak". Manuscript.

Bachelor's qualifying thesis of the educational program "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualifying thesis consists of an introduction, three sections, conclusions and proposals, a list of used sources and annexes.

In the first chapter, an analysis of the current state of the problem of developing online shops was carried out and the task was formulated. The second section defines the requirements for the developed system, as well as means, methods and development algorithms. In the third chapter, online shop was developed and tested. The conclusions summarize the information presented in the previous parts. The development results demonstrate the possibilities of used instruments and technologies for web developing.

Keywords: web application, online shop, back-end, front-end, database, webserver, Docker, API, framework, library, Laravel, React, Inertiajs.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні реалії вимагають від кожного бізнесу бути гнучким, слідкувати за останніми тенденціями ринку, невпинно шукати нові ідеї, технології та методи ведення бізнесу тощо. Будь-яка компанія працює на збільшення об’єму продажів, розширення клієнтської бази та охоплення максимальних обсягів продажу свого товару.

Інтернет-магазин дає можливість власнику бізнесу економити на оренді приміщень, оплаті праці персоналу, скорочувати витрати на локальну рекламу. Адже торгові точки в мережі можуть ефективно використовувати сучасні рекламні інструменти для розширення своєї клієнтської бази та розповсюдження інформації про себе.

Споживачі, в свою чергу, отримують зручний доступ до широкого асортименту солодощів, які можна замовити з доставкою додому, що значно економить час та зусилля.

За статистичними даними, минулого року загальний об’єм виручки з інтернет-продажів в Україні сягнув понад 150 млрд грн. На торгівлю в мережі наразі припадає понад третина продажів від всіх товарів у світі. Розширення продажів в мережі інтернет може стати важливим фактором успіху бізнесу. Саме тому створення інтернет-магазину солодощів стає надзвичайно актуальним та важливим.

Власний онлайн-магазин солодощів допомагає збільшити кількість покупців, об’єм та асортимент продажу, є зручним та сучасним інструментом ведення бізнесу. Торгівля в інтернеті є доступною цілодобово, що допомагає покупцям здійснювати покупки в зручний час та день тижня, не виходячи з дому.

Усі фактори обумовлюють постійно зростаючий попит на онлайн-магазини з продажу солодощів у сучасному світі.

Мета роботи – ідентифікація та вивчення ключових методів і стратегій розробки веб-сайтів та веб-додатків, зокрема тих, що стосуються інтернет-шопінгу.

Завдання роботи:

– визначити актуальність поставленої проблеми створення онлайн-магазину;

– сформувати структуру майбутнього веб-сайту, визначити вимоги та логіку роботи, обрати основну кольорову палітру кольорів інтерфейсу користувача;

– описати структуру архітектури веб-додатків;

– провести детальний аналіз та порівняти існуючі технології та інструменти для веб-розробки, обрати оптимальний спосіб реалізації інтернет-магазину відповідно до визначених вимог;

– розробити сайт онлайн-магазину «Таймчак»;

– описати типи тестування, які застосовувались на етапах розробки сайту;

– створити, налаштувати та підключити базу даних до проєкту;

– провести заходи з захисту розробленого веб-додатку;

– створити технічну документацію з покроковими інструкціями первинного запуску системи.

Об’єктом кваліфікаційної роботи бакалавра є інструменти та технології для розробки веб-додатків.

Предметом кваліфікаційної роботи є онлайн-магазин солодощів «Таймчак».

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ СИТУАЦІЇ НА РИНКУ ІНТЕРНЕТ-ШОПІНГУ

1.1 Аналіз сучасного стану проблеми

З кожним днем інтернет все більше інтегрується в життя людей. Особливо в останні роки, вплив розвитку технологій дає змогу спростити нам звичні раніше речі чи справи. Це і відображається у багатьох сферах діяльності: освіті, транспорті та логістиці, комерції, рекламі та маркетингу, та, зокрема, торгівельній сфері.

Багато людей цінують саме зручність використання технології, інструменту, приладу, інших речей, та свій комфорт. Це можна побачити в популярних зараз онлайн-сервісах різного роду, зокрема онлайн-магазинах. Популярність сайтів, де продають речі чи продукти, обумовлена простими чинниками – це зручно та просто.

Стрімкий «бум» онлайн-шопінгу (покупок в інтернеті) стався під час пандемії COVID-19, починаючи з 2020 року [1]. Тоді майже всі традиційні магазини перейшли на дистанційну торгівлю, щоб забезпечити максимально можливий обіг товарів. У цей період люди зрозуміли, що це дуже зручно, через що частота покупок в інтернеті не згасає, а лише далі продовжує рости.

Онлайн-шопінг зручний тим, що маючи доступ до інтернету, звичайній людині не потрібно покидати свій дім та їхати кудись для покупок, можна просто замовити потрібну річ на сайті. Ви можете це зробити в будь-який час та з будь-якого місця, використовуючи девайси з доступом у мережу. Це неабияк зручно для усіх груп населення, зокрема людей з обмеженими можливостями, для яких звичайний похід у магазин складає великі труднощі.

Створення веб-сайту для підприємства відкриває безліч переваг, що сприяють розвитку бізнесу. Одна з ключових переваг полягає в збільшенні онлайн присутності. Це дозволяє підприємству стати більш доступним і видимим для потенційних клієнтів, що розширить коло можливостей для залучення нових клієнтів і партнерів.

Крім того, веб-сайт є ефективним інструментом для маркетингу і збільшення продажів. Він дозволяє ефективно презентувати продукти і послуги, підвищуючи інтерес і перетворюючи відвідувачів сайту на клієнтів. Професійно розроблений веб-сайт також сприяє зміцненню довіри споживачів до бренду, створюючи враження надійності і професіоналізму компанії.

Зокрема, через веб-сайт підприємство може підтримувати комунікацію з клієнтами, надаючи легкий доступ до інформації про продукти, послуги та контактні дані. Це сприяє покращенню обслуговування клієнтів і збільшує загальну задоволеність від взаємодії з бізнесом. Для багатьох підприємств дуже вигідним є створення онлайн-магазину.

Інтернет-магазин – це сайт, де користувачі можуть купувати або просто переглядати товари, порівнювати їх, і все це завдяки мережі інтернет. Зазвичай, інтернет-магазин представляє цифрову версію звичайного магазину, власник якого вирішив розміститися у веб-просторі. Проте є і такі магазини, торгові майданчики яких існують лише у хмарі, і товари з яких можна отримати лише за допомогою мереж доставки [2].

Простими словами, інтернет-магазин – це певний каталог продукції або різноманітних товарів. Він може містити категорії, ціни, описи та зображення товарів. Також, інтернет-магазини можуть бути і складнішими, тут функціонал розширюється різними фільтрами пошуку, системою рекомендацій, онлайн-консультаціями і так далі.

Онлайн-магазини, як і звичайні, можуть ґрунтуватися на вузькій спеціальності, або ж пропонувати ширший спектр товарів, як це роблять гіпермаркети.

Також існує поняття «маркетплейс». Основна його відмінність від звичайного онлайн-магазину полягає в методах поставки товару на торговий майданчик, тобто окрім власної мережі складів, або як повний замінник, існують контрактні відношення з іншими постачальниками, які можуть надавати власні послуги та товари на таких маркетплейсах [3].

Таким чином, онлайн-шопінг є важливою частиною сучасної економіки та займають окреме місце в житті людей, адже пропонують зручність, економію часу та доступність [4]. Інтернет-магазини, як і інші веб-технології, продовжують розвиватися та вдосконалюватись, а їх популярність невпинно зростає. У цьому полягає актуальність обраної теми.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

У ході виконання кваліфікаційної роботи буде розроблено веб-сайт онлайн-магазину солодощів. На сайті повинно бути розміщено: категорії солодощів, списки товарів, сторінка товару, кошик, створено функціонал додавання солодощів у кошик та оформлення замовлень, адміністративна панель.

Спираючись на основне завдання, було поставлено наступні задачі:

– визначити актуальність поставленої проблеми створення інтернет-магазину;

– сформувати структуру майбутнього веб-сайту, визначити вимоги та логіку роботи, обрати основну кольорову палітру кольорів інтерфейсу користувача;

– описати структуру архітектури веб-додатків;

– провести детальний аналіз та порівняти існуючі технології та інструменти для веб-розробки, обрати оптимальний спосіб реалізації інтернет-магазину відповідно до визначених вимог;

– розробити сайт онлайн-магазину «Таймчак»;

– описати типи тестування, які застосовувались на етапах розробки сайту;

– створити, налаштувати та підключити базу даних до проєкту;

– провести заходи з захисту розробленого веб-додатку;

– створити технічну документацію з покроковими інструкціями первинного запуску системи.

Висновки до розділу 1

У розділі 1 було проаналізовано сучасний стан онлайн-торгівлі та її вплив на повсякденне життя людей. Окрім того, було проаналізовано швидке зростання популярності інтернет-магазинів у різних сферах діяльності. Також були поставлені завдання на кваліфікаційну роботу бакалавра, які включають у себе: літературний аналіз даних, порівняння інструментів та технологій веб-розробки, розробка онлайн-магазину та проведення тестування продукту розробки.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО ІНТЕРНЕТ-МАГАЗИНУ

2.1 Аналіз, визначення вимог до інтернет-магазину та проєктування системи

Було сформовано структурну UML-діаграму схеми сайту, показану на рисунку 2.1, для візуального відображення структуру сторінок.

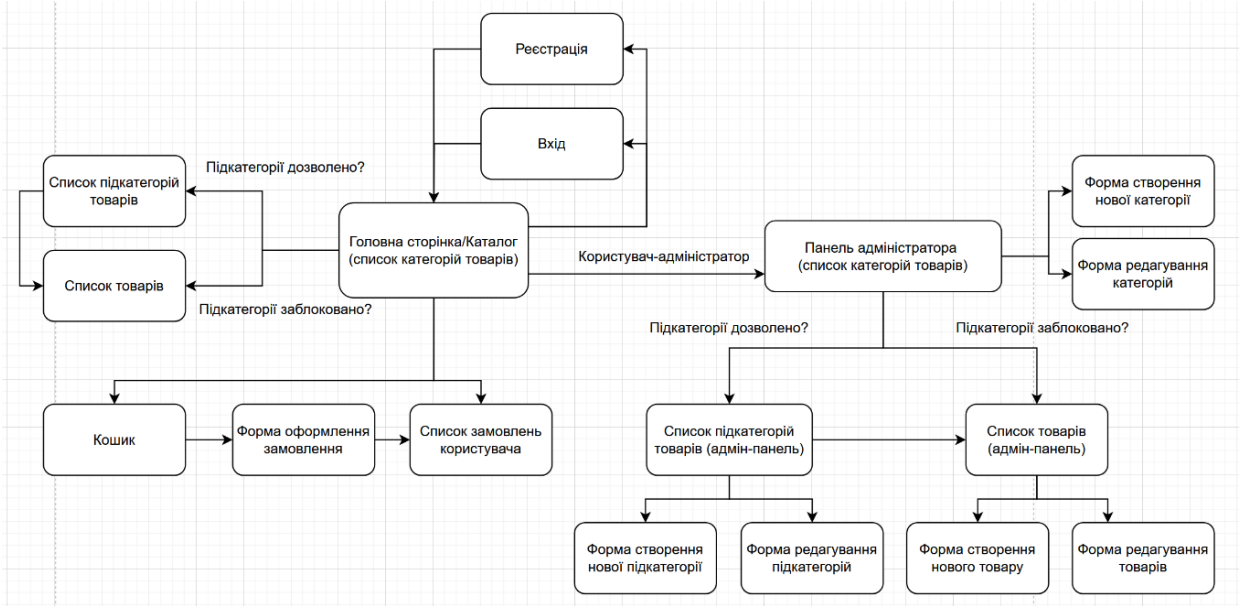

Рисунок 2.1 – Схема онлайн-магазину солодощів «Таймчак»

Структура каталогу інтернет-магазину містить: категорії товарів, підкатегорії товарів та самі товари. Категорії розділені на два типи: з підкатегоріями, та без підкатегорій, за це відповідатиме поле «has_subcategories». В залежності від значення цього поля («true» або «false»), категорії повинні посилатися або на список підкатегорій, або відразу на список товарів у категорії. Звідси виходять дві ситуації, які потрібно було розглянути:

– є підкатегорії;

– немає підкатегорій.

Для них було детально розібрано та створено логіку:

1) є підкатегорії:

– значення поля «has_subcategories» рівне «1», або «true», перемикач на відповідній категорії на сторінці зі списком категорій «увімкнений»;

– посилання на рух вниз по дереву каталогу товарів веде на список підкатегорій;

2) немає підкатегорій:

– значення поля «has_subcategories» рівне «0», або «false», перемикач на відповідній категорії на сторінці зі списком категорій «вимкнений»;

– посилання на рух вниз по дереву каталогу товарів веде на список усіх товарів у категорії.

Для доступу до сторінок категорій, підкатегорій та товарів, формування шляху адреси сайту, щоб уникнути перевантаження посилання числовими ідентифікаторами, передбачено формулу повного шляху «host/category_slug/subcategory_slug/good_id», де:

– «host» – ім’я хоста, або домен сайту;

– «category_slug» – представлення категорії у вигляді текстового ідентифікатора («slug»), тому що, за стандартом RFC 3986, посилання має містити лише символи ASCII [5, 6];

– «subcategory_slug» – представлення підкатегорії за ідентичним принципом, як представлення категорії;

– «good_id» – номер ідентифікатора товару, адже їх може бути багато, і їх можна ідентифікувати саме таким чином.

Додатково потрібно було розглянути ситуацію, коли значення «has_subcategories» у категорії було змінене. Виходячи з цього, було описано ще декілька умов та приміток:

1) у кожної категорії повинна бути невидима підкатегорія «no_subcategory»;

2) дана підкатегорія, при зміні значення «has_subcategory» у категорії, до якої належить, змінює своє представлення в адресному рядку (власний «slug»), для полегшення обробки даних;

3) у випадку, коли значення «has_subcategory» змінено на «true»:

– представлення категорії «no_subcategory» – «uncategorized»;

– підкатегорія «no_subcategory» буде недоступною у каталозі на сайті магазину;

– на адмін-панелі ця категорія відображатиметься лише коли у ній є нерозподілені товари;

4) у випадку, коли значення «has_subcategory» змінено на «false»:

– представлення категорії «no_subcategory» дублює представлення категорії-батька;

– усі товари, які були розподілені по підкатегоріях, одночасно виводяться при перегляді товарів основної категорії;

– нові товари, які додаються, записуються в підкатегорію «no_subcategory», і при зміні значення «has_subcategory» назад на «true» – всі новостворені товари показувати в «нерозподілених товарах».

Управління значенням «has_subcategories» для категорій має бути реалізованим за допомогою перемикача у панелі адміністратора (у кожної категорії власний перемикач).

Доступ до адмін-панелі має бути лише у користувачів з роллю «Адмін», а перехід на неї організований за допомогою додаткового посилання у спадному меню керування автентифікацією для авторизованого користувача (саме посилання видиме лише якщо користувач є адміністратором.

У кошику повинна бути можливість вибору певних товарів зі списку, видалення товару зі списку, зміни вибраної кількості відповідного товару та кнопка переходу на оформлення замовлення на вибрані товари.

Сторінка оформлення замовлення повинна містити список обраних товарів та форму для заповнення даних, які потрібні для забезпечення доставки. Список

замовлень повинен містити в собі основну інформацію усіх замовлень користувача.

Кольорова палітра має містити різні відтінки кремового кольору, оскільки саме цей колір найкраще асоціюється з солодкою продукцією.

2.2 Вибір інструментів для виконання поставленого завдання

Існує багато різних інструментів та технологій для реалізації проєктів веб-розробки, і всі вони різняться між собою різними можливостями, функціоналом, своєю гнучкістю тощо. Але всі вони мають своє призначення. Розглянемо найпопулярніші інструменти для створення веб-застосунків, орієнтуючись на трьох базових елементах архітектури більшості веб-застосунків:

– front-end;
– back-end;
– база даних.

Усі ці елементи мають абсолютно різну логіку роботи, і відповідають за власні частини усієї системи. Розглянемо кожну окремо.

Front-end – це клієнтська сторона додатку, яка відповідає за інтерфейс користувача [7]. Іншими словами, front-end – це зовнішня частина сайту для взаємодії з відвідувачем. Логотипи популярних фреймворків та бібліотек показано на рисунку 2.2.

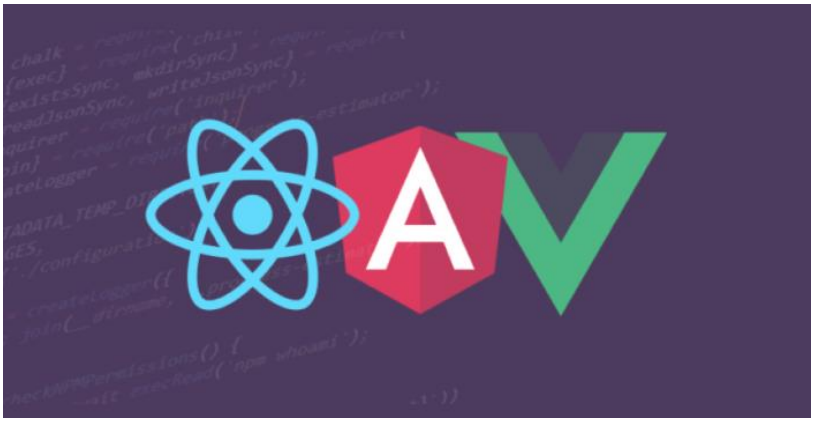

Рисунок 2.2 – Логотипи React, Angular та Vue, відповідно [8]

Нижче описано найпопулярніші інструменти для розробки «обличчя» сайту.

Angular – це високопродуктивна платформа з відкритим кодом на основі TypeScript, створена компанією Google. Вважається найкращою для створення стримінгових сервісів, додатків для прогнозу погоди, додатків для подорожей та інших типів додатків, в інтерфейсі яких є елементи, які не завжди є легкими для візуалізації іншими технологіями розробки front-end [9].

До переваг можна віднести:

– це є повноцінний фреймворк з інструментами та шаблонами для розробки об’ємних додатків;

– використання TypeScript допомагає виявляти помилки ще на стадії розробки.

Щодо недоліків:

– занадто громіздкий для невеликих проєктів;

– складний для початкового освоєння;

– повільніший у порівнянні з аналогами.

Vue.js є універсальним фреймворком для використання у розробці як невеликих компонентів сторінки, так і масштабних односторінкових додатків. Відомий тим, що його можна легко інтегрувати в структуру додатку [10].

Переваги:

– легкий для вивчення та інтеграції з іншими проєктами;

– підходить для проєктів різного масштабу, може використовуватися для окремих елементів на сторінці;

– високопродуктивний за рахунок оптимізованого віртуального DOM.

Недоліки:

– менша база спільноти, порівняно з React та Angular;

– тяжко справляється з надвеликими проєктами;

– погана вбудована підтримка типізованого TypeScript.

React є однією з найпопулярніших бібліотек JavaScript для розробки інтерфейсів користувача. Він особливо підходить для створення динамічних веб-додатків, де важлива висока продуктивність і гнучкість [11].

Переваги:

– компонентна архітектура допомагає полегшити розробку за рахунок зручного повторного використання коду;

– величезна кількість бібліотек різного масштабу, які мають пряму підтримку React;

– наявність віртуального DOM.

Недоліки:

– для роботи з React, потрібно мати навички та досвід роботи з JavaScript ES6 та «jsx» форматом;

– часті оновлення, які не завжди є зворотньо-сумісними;

– це бібліотека, а не фреймворк, тому потребує частого підключення інших бібліотек для керування різними процесами.

Back-end – це частина сайту, яка відповідає за бізнес-логіку системи та обробку даних, які використовуються на сайті, але не відображаються користувачу напряму [12]. Це включає в себе взаємодію з базою даних, обробку запитів від клієнтів, розрахунки, автентифікацію та авторизацію користувачів та інші аспекти, що забезпечують функціональність сайту. Щодо популярних технологічних рішень реалізації back-end можна виділити ті, що описані нижче.

Node.js є середовищем виконання JavaScript, яке дозволяє розробникам створювати back-end застосунки з використанням мови JavaScript. Він популярний у розробці серверних додатків, API та мікросервісів.

Переваги:

– зручний для швидкої full-stack (обох, back-end та front-end) розробки за рахунок використання JavaScript;

– наявність об’ємної екосистеми дозволяє знаходити вирішення для різних задач;

– асинхронний підхід дозволяє сягнути високої продуктивності та масштабованості.

Недоліки:

– використання JavaScript призводить до проблем з однопотоковими обробками;

– проблема з використанням публічних модулів може призвести до проблем різного роду.

Django – це високорівневий фреймворк для розробки веб-додатків на мові програмування Python. Він надає широкий спектр інструментів для розробки логіки веб-додатків.

Переваги:

– за рахунок бази на Python – на Django можна легко і швидко створювати системи різного масштабу;

– вбудована безпека від SQL-ін’єкцій, CSRF атак тощо.

Недоліки:

– занадто об’ємний для більшості малих проєктів;

– як і інші інструменти поглибленого вивчення на базі Python, Django являється доволі складним для вивчення «з нуля».

Ruby on Rails є фреймворком для швидкої розробки веб-додатків на мові програмування Ruby. Він надає стандартизовану структуру та широкий набір інструментів для розробки.

Переваги:

– використовує конвенції, дотримуючись яких уникається витрачання часу на конфігурацію;

– стандартна структура є достатньо зручною, а можливе використання генераторів лише збільшує можливості розробників;

– активна велика спільнота, яка продовжує розширятися.

Недоліки:

– для деяких розробників тяжко працювати строго за конвенціями;

– велика кількість автоматичних механізмів, які можуть давати неочікувані збої в роботі.

Laravel (логотип на рис. 2.3) є високорівневим фреймворком для розробки веб-додатків на мові програмування PHP. Він відомий своєю елегантністю, простотою використання та широким спектром функціоналу для розробки back-end частини веб-додатків [13]. Має деякі спільні моменти з Ruby on Rails. Також, має власну підтримку front-end.

Рисунок 2.3 – Логотип Laravel [14]

Переваги:

– дозволяє писати чистий код, який легко підтримувати;
– широкий високопродуктивний функціонал;
– активна власна спільнота «Laracasts».

Недоліки:

– велика кількість магічних методів;
– ресурсовибагливий до серверу;
– велика початкова файлова система.

База даних є центральним елементом будь-якого веб-сайту або додатку, що зберігає дані та дозволяє їх обробляти. Вона відповідає за збереження та організацію інформації, яка використовується в програмі [15].

Щодо найпопулярніших систем керування базами даних (СКБД), можна виділити описані нижче.

MySQL (логотип на рис. 2.4) є однією з найпопулярніших реляційних систем управління базами даних з відкритим кодом. Вона широко використовується для зберігання даних у веб-додатках та сайтах.

Переваги:

– широка підтримка різними хостинг-платформами;

– велика кількість різних систем адміністрування, таких як PhpMyAdmin та MySQL Workbench.

Рисунок 2.4 – Логотип MySQL [16]

Недоліки:

– порівняно з іншими СКБД, дещо падає продуктивність при обробці великого об’єму даних;

– має дещо спрощений функціонал, через що інколи не підтримує деякі рішення.

PostgreSQL відома своєю надійністю, функціональністю та розширюваністю та відкритим кодом.

Переваги:

– є підтримка складних запитів, транзакцій та інших складних функцій на високому рівні;

– легко розширяється плагінами та модулями;

– висока надійність даних та стійкість до збоїв;

– не зазнає суттєвого впливу на продуктивність при роботі з великими обсягами даних.

Недоліки:

– складніша у налаштуванні, порівняно з іншими СКБД;

– вимоглива до ресурсів;

– при роботі з меншою кількістю даних, дещо поступається швидкодії MySQL.

MongoDB – нереляційна СКБД, дані якої зберігаються в файлах JSON-подібного формату, доволі популярна за рахунок швидкості обробки великої кількості даних.

Переваги:

– є можливість зберігати дані у вигляді документів, що вважається більш гнучким способом у порівнянні з традиційними реляційними моделями баз даних;

– структура цих документів може змінюватися, після чого адаптація даних можлива без необхідності змінювати структуру самої бази даних;

– висока швидкість маніпуляції даними за рахунок внутрішніх механізмів індексації.

Недоліки:

– відсутність підтримки складних маніпуляцій даними;

– неефективне використання пам’яті через часті дублювання записів у документах;

– високий ризик втрати даних при роботі з конфігурацією.

І хоч базу даних можна віднести до back-end, оскільки це серверна частина застосунку, і саме вони двоє спілкуються між собою, проте, зазвичай, вони вважаються окремими елементами.

Також, важливо відмітити Системи Керування Контентом (CMS). CMS відіграють важливу роль у сфері розробки простих веб-застосунків [17]. Дані системи дозволяють відносно легко та швидко створювати невеликі та прості веб-додатки, при цьому процес майже не потребує поглиблених технічних знань.

Застосунки, зроблені на базі цих систем, забезпечують зручне керування контентом для користувачів. Самий популярний тип застосунків, які зазвичай створюються за допомогою CMS – блоги. Проте, вартує зазначити, що простота може відобразитись на безпеці, відсутності детальної конфігурації системи та затраті серверних ресурсів. Серед найпопулярніших CMS можна виділити:

WordPress – це одна з найпопулярніших CMS у світі, яка використовується для створення блогів, корпоративних сайтів, портфоліо та новинних ресурсів.

Переваги:

– інтуїтивно зрозумілий інтерфейс, легкість у встановленні та налаштуванні;

– велика кількість безкоштовних та платних тем і плагінів для розширення функціональності;

– активна спільнота користувачів та розробників, часті оновлення та введення нових технологій.

Недоліки:

– за рахунок популярності, додатки, створені за допомогою WordPress часто попадають під мішень злому, через що розробникам потрібно займатися постійною підтримкою своїх продуктів;

– швидкодія залежить від оптимізації та належного налаштування;

– не завжди підходить для великих проєктів.

Joomla! – CMS з відкритим вихідним кодом, яка підходить для створення різних типів веб-сайтів, включаючи новинні портали та соціальні мережі.

Переваги:

– наявна підтримка модульної структури;

– є вбудовані можливості для створення багатомовних сайтів, управління користувачами та багато іншого.

Недоліки:

– може бути складною для новачків;

– стадія підтримки в життєвому циклі додатку може бути проблематичною;

– невеликий репозиторій плагінів та тем.

TYPO3 – гнучка та масштабована CMS з відкритим вихідним кодом, яка підходить для створення корпоративних веб-сайтів та порталів.

Переваги:

– модульна структура;

– є можливість підключення сторонніх додатків та сервісів.

Недоліки:

– може бути складною у конфігурації;

– вимагає більше серверних ресурсів для стабільної роботи.

Для локальної роботи над веб-додатком зазвичай використовуються такі технології як: Kubernetes або Docker. За допомогою цих технологій в рази полегшується процес переносу або копіювання додатку між хостами, і не важливо, чи це робочий комп'ютер розробника, чи це серверний кластер. Що Kubernetes, що Docker базуються на методиці «оркестрації контейнерами». Кожен додаток базується на кількох контейнерах, які, в свою чергу, містять елементи для роботи веб-додатку, наприклад: веб-сервер, або базу даних.

Для взаємодії між компонентами додатку, а саме можливості отримування запитів від користувача, та відправлення йому відповідей використовуються веб-сервери. Серед найпопулярніших серверів можна розглянути nginx та Apache:

– nginx створений для забезпечення швидкого обслуговування та високої масштабованості. Він відомий своєю здатністю підтримувати з'єднання кінцевих користувачів з низькими витратами ресурсів. nginx добре справляється з балансуванням навантаження на хост. На рисунку 2.5 можна побачити логотип nginx;

Рисунок 2.5 – Логотип nginx [18]

– Apache HTTP Server, часто просто Apache, є одним з найпопулярніших і найстаріших веб-серверів. Відомий своєю гнучкістю та великою кількістю модулів, Apache підтримує широкий спектр функцій, які можна легко розширювати і налаштовувати. Проте, порівняно з nginx, має проблеми з продуктивністю при більших навантаженнях та набагато більш ресурсозатратний.

Провівши детальний аналіз більшості популярних інструментів та технологій, для створення інтернет-магазину відразу було відкинуто варіант використання CMS систем з рядку причин: можливість використання лише стандартизованих шаблонів, що ускладнює можливість створення власного інтерфейсу, можливість погіршення продуктивності при великій кількості запитів, відсутність впевненості в безпеці програми тощо. Після цього, обираючи основний інструмент серед back-end фреймворків (як основи інтернет-магазину), вибір впав на використання Laravel, потужного фреймворку на базі мови програмування PHP, який слідує архітектурному шаблону Модель-Вигляд-Контролер (MVC), та є широкий вибір інструментів, за допомогою яких можна зручно реалізовувати багато різного функціоналу.

Одна із цікавих особливостей Laravel – це те, що цей фреймворк може бути використаним не лише для back-end частини веб-додатку, а й front-end, що робить його базою для full-stack застосунку. «З коробки», Laravel має вбудований шаблонізатор Blade.

Laravel Blade працює так: створюється файл-шаблон, який містить в собі DOM-структуру HTML, але з прямими вставками інформації, яка передається з відповідного методу класу-контролеру. Коли користувач переходить на сторінку, за яку відповідає цей контролер – у нього відображається сторінка HTML, із потрібними даними. Всередині цих шаблонів, точно так само як в HTML документах, можна підключати таблиці стилів CSS та скріпти JavaScript.

Оскільки Laravel – це фреймворк на базі PHP, то у проєкті використовується менеджер залежностей Composer, за допомогою якого можна підтягувати різні модулі: або офіційні, або ті, що створені іншими розробниками. Серед них є

модулі, які дозволяють розширяти можливості подачі front-end складової веб-додатку. Наприклад, модуль Laravel Livewire, який робить можливість робити шаблони Blade динамічними, та і загалом, значно розширити функціонал шаблонізатору. Окрім цього модуля, можна ще підключити бібліотеку Intertia, логотип якої можна побачити на рисунку 2.6.

Рисунок 2.6 – Логотип Inertia

Головне призначення Inertia – створення сучасного односторінкового сайту, з використанням React, Vue, Svelve, використовуючи саме серверну основу проєкту. Це дуже зручно за рахунок того, що будується лише один проєкт, в середині якого, в зручній формі знаходяться всі потрібні елементи для роботи веб-застосунку.

Виходячи з цього, для основи front-end було вирішено вибрати один з двох: React, або Vue, інші варіанти були відкинуті через меншу популярність, вищий рівень складності, або з інших причин. Між React та Vue, часу на вибір теж не було затрачено багато, з тої причини, що протягом останнього року, дані бібліотека та фреймворк оновилися до нових версій, у яких вони були сильно змінені, і станом на сьогодні, React значно легший з боку вивчення та роботи з ним.

Для роботи з абсолютною більшістю фреймворків та бібліотек, основаних на JavaScript, використовується менеджер пакетів JS, npm. Даний менеджер встановлюється одночасно з встановленням Node.js на сервер, чи робочий комп’ютер.

Серед баз даних було вибрано СКБД MySQL, тому що її головна перевага полягає саме в швидкості обробки даних, а оскільки в інтернет-магазині немає дуже великих об’ємів даних для обробки, швидкодії MySQL нічого не загрожуватиме.

Виходячи з того, що основою проєкту являється фреймворк Laravel, то було обрано середовище розробки JetBrains PhpStorm, оскільки воно є спеціалізованим саме на мові програмування PHP, зокрема має хорошу інтеграцію з Laravel (за можливості, можна оформити передплату на розширення, яке повністю інтегрує підтримку Laravel в середовище розробки) та підтримує інші основні технології веб-розробки. Як і всі продукти компанії JetBrains, PhpStorm має дуже зручні додаткові можливості, серед яких слід виділити підтримку інтеграції бази даних (підключення, можливість перегляду та редагування даних), що дозволяє не використовувати інші аналогічні рішення, такі як: PhpMyAdmin, MySQL Workbench тощо [19].

Для роботи над веб-додатком обрано Docker, оскільки він простіший у налаштовуванні та використанні, і він є більш популярним серед розробників.

Веб-сервером було вибрано nginx, тому що він не є затратним по ресурсах.

Додаткове програмне забезпечення, яке було використане у процесі розробки інтернет-магазину:

– iTerm2 – замінник системного терміналу операційної системи macOS. Зручний за рахунок можливого налаштування зовнішнього вигляду, має можливість логування, історію, та інші інструменти;

– Docker Desktop – зручний інструмент для роботи з Docker-інфраструктурою за допомогою графічного інтерфейсу користувача.

Висновки до розділу 2

У цьому розділі було детально розглянуто та проаналізовано логіку майбутнього інтернет-магазину, що включає структуру та взаємодію різних компонентів системи. Представлена схема сторінок дозволяє зрозуміти, як

користувачі будуть переміщатися по сайту та взаємодіяти з різними його елементами.

Було описано та проведено порівняння популярних front-end фреймворків та бібліотек, таких як React, Angular, Vue.js, допомогли визначити найбільш підходящі інструменти для розробки інтерфейсу користувача. Проведено аналіз back-end фреймворків, таких як Laravel, Django, Node.js, зокрема, було враховано такі фактори, як простота використання, продуктивність, гнучкість та підтримка спільноти. Рохглянуто та проведено порівняння різних СКБД, у тому числі MongoDB, яка є однією з баз даних NoSQL-типу. Розглянуто веб-сервери та додаткові програмні рішення.

Обрано найбільш зручні та ефективні рішення для створення онлайн-магазину.

РОЗДІЛ 3
РОЗРОБКА ОНЛАЙН-МАГАЗИНУ СОЛОДОЩІВ

3.1 Практична реалізація об’єкта проєктування

Перед початком розробки програмного забезпечення потрібно було створити базовий проєкт Laravel [20]. Фактично, це процес копіювання репозиторію з бази Laravel. Це відбувається за допомогою команди менеджера залежностей Composer, яку наведено в лістингу 3.1.

Лістинг 3.1 – Команда створення проєкту

```
composer create-project laravel/laravel timechak-candy-shop
```

Кінець лістингу 3.1

Після успішного створення потрібно було налаштувати файл середовища «.env».

Файл середовища (зазвичай з назвою «.env») – файл конфігурації, який служить для зберігання конфіденційних даних, які не повинні бути включені в код додатку. У «.env» для Laravel передбачено багато значень, які потрібно змінити перед запуском системи, чи то на сервері, чи то на локальній машині окремого учасника процесу створення продукту (зазвичай це розробник або тестувальник). Більшість з цих значень бути індивідуальними для кожного серверу.

Основні значення, що були змінені у даному випадку – це (ключ-значення):

- «APP_NAME» – «timechak-candy-shop»;
- «APP_TIMEZONE» – «Europe/Kyiv»;
- «FILESYSTEM_DISK» – «public»;
- «DB_CONNECTION» – «mysql»;
- «DB_HOST» – «0.0.0.0»;
- «DB_PORT» – «3306»;
- «DB_DATABASE» – «timechak»;

- «DB_USERNAME» – «root»;
- «DB_PASSWORD» – «root».

Всі ці значення використовуються для конфігурування саме Laravel back-end, і зчитуються файлами конфігурації відповідних підсистем, які знаходяться у папці «/config».

Далі було проведено процес створення і налаштування Docker-контейнерів, та їх функціоналу.

У корені файлової системи проєкту було створено файл «docker-compose.yml», який являється конфігураційним файлом створення та роботи Docker-контейнерів. Сервіси всередині цього файлу – це і є контейнери, кожен з яких відповідає за те, що в ньому подано. Всього було створено три сервіси:

– «backend-app» – серверна частина для PHP;
– «webserver» – nginx;
– «database» – СКБД з даними.

Для конфігурації контейнерів використовувалися вже наявні дистрибутиви з репозиторію Docker. Для ширшого налаштування було використано: атрибути сервісів в конфігураційному файлі «docker-compose.yml», «Dockerfile» для «backend-app» сервісу (у таких файлах зазвичай знаходиться список команд, які потрібно виконати після першого запуску контейнера), та конфігураційний файл nginx.

Після цього було введено серію команд в терміналі, яку наведено в лістингу 3.2.

Лістинг 3.2 – Серія команд налаштування проєкту

```
docker-compose up -d --build         // build and run docker containers
composer require laravel/breeze –dev // requiring Laravel Breeze
php artisan install:api              // api interface and routing
php artisan breeze:install           // installing Breeze
```

Кінець лістингу 3.2

Перша команда займається побудовою Docker-контейнерів, запускає їх та проводить їхню конфігурацію при першого запуску.

Друга команда робить запит на пакет з Laravel Breeze, робить його необхідним лише для середовища з назвою «dev». Laravel Breeze встановлює вже готову систему автентифікації, на базі Sanctum. Він зручний тим, що при його інсталляції вибирається стек, на базі якого потрібно зробити автентифікацію, і він відразу встановлює всі потрібні для цього пакети.

Третя команда відповідає за встановлення та основного налаштування системи автентифікації Laravel Sanctum, створює правила (надалі – Middleware) для гостів та авторизованих користувачів, які можна пізніше використовувати у налаштуванні маршрутизації, та створює файл маршрутизації API.

Четверта команда відповідає саме за встановлення та публікацію пакету Laravel Breeze. При виконанні цієї команди, Laravel консоль artisan запитує додаткову інформацію, а саме: стек-базис автентифікації, додаткові рішення (такі як темний режим сайту), та вибір тестового фреймворку, на якому він відразу публікує готові автотести автентифікації, які, за потреби, пізніше можна використовувати.

Публікація – це копіювання деяких файлів з встановленого відповідного пакету Composer безпосередньо у файлову систему проєкту.

Серед пропонованих стеків було вибрано React with Inertia, завдяки чому було автоматично встановлено пакети, та підключено бібліотеки для роботи React та Inertia, було опубліковано базовий набір front-end компонентів та сторінок.

Після виконання всіх цих операцій, проєкт готовий до початку процесу розробки веб-додатку. Можна протестувати, зайти на локальний хост, та глянути на вікно входу (рис. 3.1).

Спершу було змінено надписи та назви полів та інших елементів, опублікованих Laravel Breeze. Додатково відразу було проставлено потрібну кольорову палітру на елементах (рис. 3.2). Стилізація зовнішнього вигляду елементів проводилася за допомогою фреймворку таблиці стилів Tailwind. Він дозволяє підключити таблиці стилів, які знаходяться на віддаленому хмарному

сховищі, через що їх не потрібно зберігати додатково у файловій системі проєкту. Підключення Tailwind відбувається за допомогою npm. У проєкт даний фреймворк був інтегрований автоматично, при інсталляції Laravel Breeze.

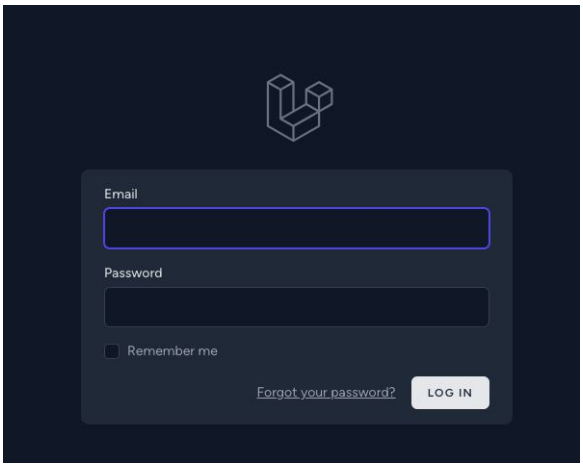

Рисунок 3.1 – Стандартне вікно автентифікації Laravel Breeze

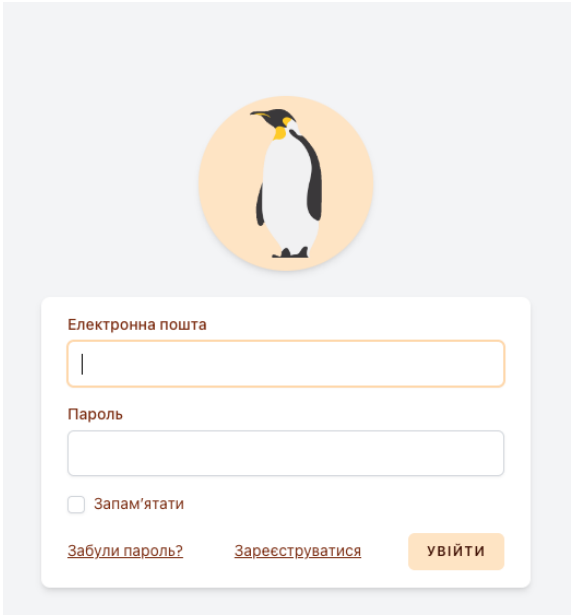

Рисунок 3.2 – Стилізоване вікно входу у систему під вибраний дизайн

Далі було проведено рефакторинг макетів сайту. Було змінено базовий макет «GuestLayout», та створено три інших, на базі до того існуючого «AuthenticatedLayout»:

– «ProfileLayout» – для майбутніх сторінок корзини та замовлень;

– «ShopLayout» – для самого інтернет-магазину;

– «AdminLayout» – для адміністративної панелі.

Макети «Layout» використовуються, як шаблони, що містять «Header» та «Footer» елементи, які використовуються на усіх сторінках, які обгорнуті відповідним макетом.

Макети для сторінок профілю та магазину відрізняються тим, що в залежності від того, чи увійшов користувач у систему, справа зверху будуть відображені або два посилання на форми входу та реєстрації, або спадне меню з іменем користувача, через яке можна вийти з системи. Частину реалізації показано в лістингу 3.3.

Лістинг 3.3 – Реалізація компоненту авторизації на макеті «ShopLayout»

```
 <{user ? (
<div className="flex items-center px-4">
    <Dropdown>
        <Dropdown.Trigger>
    <span className="inline-flex rounded-md">
       <button
            type="button"
            className="inline-flex items-center px-3 py-2 border
border-transparent text-sm leading-4 font-medium rounded-md text-orange-
900 opacity-80 bg-amber-50 hover:opacity-100 focus:outline-none transition
ease-in-out duration-150">
         {user.name}

            <svg
                className="ms-2 -me-0.5 h-4 w-4" xmlns =
http://www.w3.org/2000/svg viewBox="0 0 20 20" fill="currentColor">
                <path
                    fillRule="evenodd"
                    d= "M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-
3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z"
                    clipRule="evenodd"
                />
            </svg>
        </button>
    </span>
        </Dropdown.Trigger>
```

Кінець лістингу 3.3

За допомогою вкладеного елементу «user» через сторінку, іде пряма перевірка, чи цей «user» існує, і в залежності від цього відображаються відповідні елементи. Візуальну різницю зображено на рисунку 3.3.

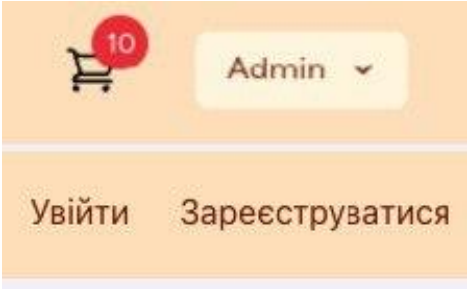

Рисунок 3.3 – Різниця в інтерфейсі користувача для авторизованого користувача та гостя сайту

Але макети профілю та магазину теж мали відрізнятися між собою. Макет для сторінок профілю повинен був мати два навігаційних посилання на кошик та список замовлень користувача. У той же час, коли на макеті для магазину біля спадного меню з іменем користувача повинна бути присутня кнопка «кошик» із рахунком кількості товарів у ньому, а з лівого боку біля логотипу повинне бути присутнім ще одне спадне меню «Каталог».

Отримання даних для показу категорій в спадному меню каталогу та для обчислення кількості товарів у кошику (ще для перевірки, чи товар уже в кошику) було організовано за допомогою створеного Middleware «ShareShopCatalogData» (команда для створення Middleware наведена в лістингу 3.4).

Лістинг 3.4 – Команда для створення Middleware

```
php artisan make:middleware ShareShopCatalogData
```

Кінець лістингу 3.4

Middleware – це класи, які прив’язуються до маршрутів, і виконуються перед переходом на те, чи інше посилання, ще перед виконанням відповідних методів у прив’язаних контролерах. Коли до фреймворку прив’язана бібліотека

Inertia, Middleware може використовуватися для того, щоб передавати якісь дані. Новоствореному Middleware було прив’язано ім’я «catalog», для цього його було записано у відносини «Midlleware alias», у файлі конфігурації роутинга «bootstrap/app.php» (лістинг 3.5).

Лістинг 3.5 – Код присвоєння короткої назви для Middleware

```
$middleware->alias([
    'admin' => Admin::class,
    'catalog' => ShareShopCatalogData::class
]);
```

Кінець лістингу 3.5

З того моменту можна використовувати магічний метод «middleware» на маршруті, на який потрібно подати дані цим Middleware.

Після цього було розпочато роботу над панеллю адміністратора.

Спочатку було створено додатковий файл маршрутизації з назвою «admin.php», у якому було описано маршрути для адмін-панелі. Частину з них наведено в лістингу 3.6.

Лістинг 3.6 – Маршрути адмін-панелі для керування списком категорій

```
// Categories administration routes
Route::get('/categories', [GoodCategoryController::class, 'index'])
->name('categories.index');
Route::get('/categories/new', [GoodCategoryController::class, 'create'])
->name('categories.create');
Route::post('/categories', [GoodCategoryController::class, 'store'])
->name('categories.store');
Route::get('/{category}/edit', [GoodCategoryController::class, 'edit'])
->name('categories.edit');
Route::post('/{category}', [GoodCategoryController::class, 'update'])
->name('categories.update');
Route::delete('/{category}', [GoodCategoryController::class, 'destroy'])
->name('categories.destroy');
```

Кінець лістингу 3.6

Маршрути для адміністрування підкатегоріями та товарами були також наведені у даному файлі.

Було створено контролери, зазначені у файлі маршрутизації за допомогою команди, приклад якої наведено в лістингу 3.7.

Лістинг 3.7 – Команда для створення контролера «GoodCategoryController»

```
php artisan make:controller Admin/GoodCategoryController
```

Кінець лістингу 3.7

У кожному з цих контролерів було створено методи для керування відповідними записами. Кожен з них отримав наступні методи:

– «index» – рендер сторінки із списком та передача усіх записів відповідних даних;

– «create» – рендер сторінки із формою створення нового запису;

– «store» – перевірка та збереження нового запису;

– «edit» – рендер сторінки із формою для зміни існуючого запису, із передачею даних запису, що змінюється;

– «update» – перевірка та зміна даних існуючого запису;

– «destroy» – видалення запису.

В лістингу 3.8 наведено код методу «create», на прикладі контролера товарів.

Лістинг 3.8 – Код методу «create» у контролері «GoodController»

```
public function create(GoodCategory $category, GoodSubcategory
$subcategory): Response
{
    return Inertia::render('Admin/Goods/Create', [
        'category' => $category,
        'subcategory' => $subcategory
    ]);
}
```

Кінець лістингу 3.8

Для категорій, підкатегорій та товарів було створено по 3 власні сторінки:

– для виводу списку (таблиці) усіх записів;

– для створення нового запису;

– для редагування існуючого запису.

Для кожної сторінки зі списком записів було створено власну таблицю, у яку передаються дані через сторінку. Це можливо завдяки функціоналу React.

У списку категорій, окрім назви, фотографії та посилань, потрібно було додати перемикач, що керує полем «has_subcategories». Для роботи цього перемикача було створено функцію «handleCheckboxChange» (лістинг 3.9), яка передається в компонент-таблицю, якою виводиться список категорій товарів.

Лістинг 3.9 – Код функції «handleCheckboxChange»

```
 const handleCheckboxChange = async (id) => {
    try {
        const category = categoriesList.find(category => category.id ===
 id);

        await axios.put(
            route('api.admin.categories.update', category.slug),
            {"hasSubcategories": !category.has_subcategories}
        );

        setCategoriesList(categoriesList.map(category =>
            category.id === id ? {...category, has_subcategories:
 !category.has_subcategories} : category
        ));
    } catch (err) {
        console.error(err);
    }
};
```

Кінець лістингу 3.9

Дана функція викликається при кожному переключені перемикача (параметр «onChange», приклад коду показано в лістингу 3.10). Вона робить асинхронний API запит типу PUT на маршрут з назвою «api.admin.categories.update», який заданий у файлі маршрутизації «api.php».

Лістинг 3.10 – Код реалізації елементу перемикача

```
<input type="checkbox"
      value=""
      className="sr-only peer"
      checked={category.has_subcategories}
      onChange={(e) => handleCheckboxChange(category.id)}
/>
```

Кінець лістингу 3.10

У лістингу 3.11 можна побачити, що цей маршрут зсилається на контролер «Api\Admin\GoodCategoryController».

Лістинг 3.11 – Маршрут на обробку API запиту на зміну значення поля «has_subcategories»

```
Route::put('/admin/{category}',
[App\Http\Controllers\Api\Admin\GoodCategoryController::class,
'update'])->name('api.admin.categories.update');
```

Кінець лістингу 3.11

На рисунку 3.4 показано зовнішній вигляд таблиці категорій із фотографіями, назвами, перемикачами та потрібними посиланнями.

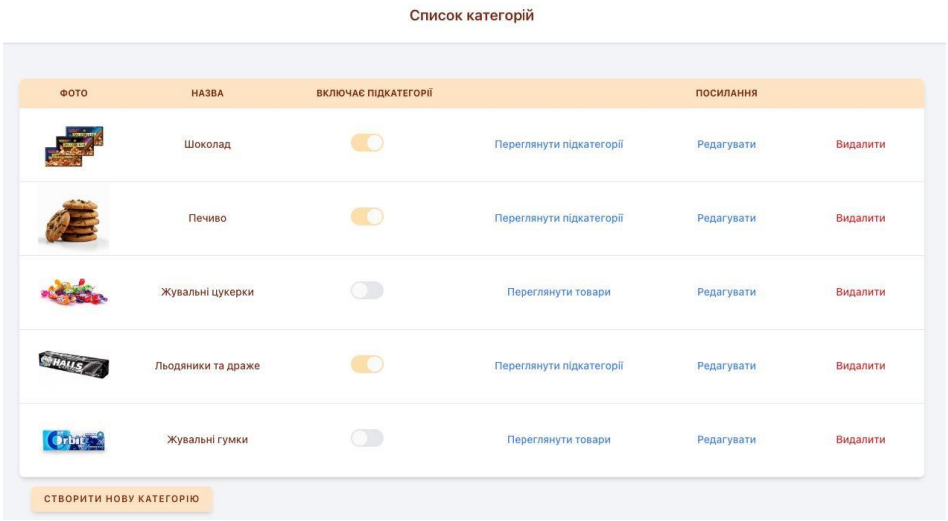

Рисунок 3.4 – Вигляд панелі адміністратора, на сторінці списку категорій

Після створеної адмін-панелі був розпочатий процес створення самого інтернет-магазину.

Спираючись на таку ж послідовність, яка слідувалася при створенні панелі адміністратора, спочатку був створений файл маршрутизації із назвою «shop.php», у якому було наведено список маршрутів самого онлайн-мазазину. Тут сторінок буде менше, а саме:

– домашня сторінка;
– список підкатегорій;
– список товарів;
– сторінка товару.

Домашня сторінка – це, фактично, список категорій. Три перших сторінки з цього списку були створені із дуже подібним дизайном. На рисунку 3.5 можна побачити головну частину списку категорій.

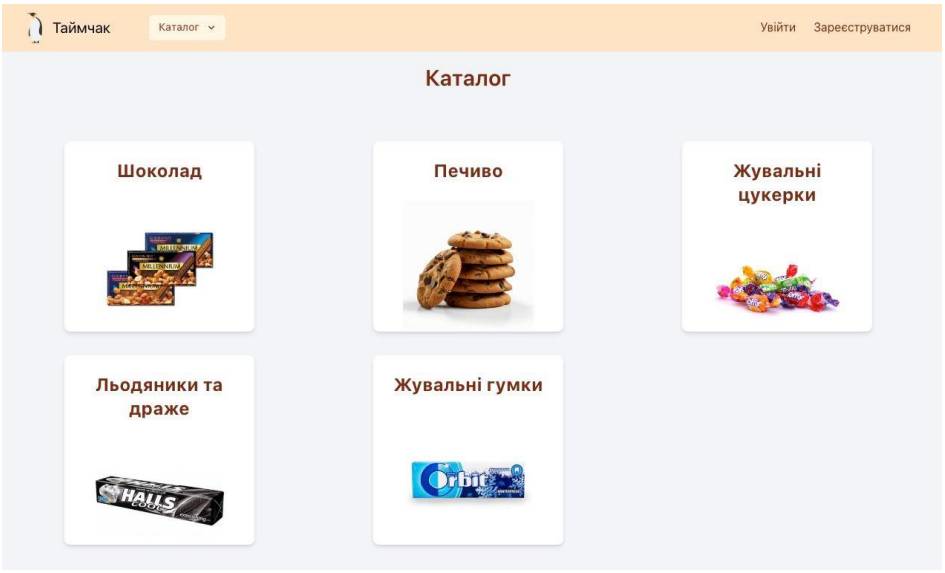

Рисунок 3.5 – Головна сторінка зі списком доступних категорій

Сторінка товару (рис. 3.6) містить на собі велику фотографію товару, ціну товару, та кнопку «В кошик», яка в залежності від наявності товару, та від того, чи добавлений вже цей товар у кошик, може блокуватися.

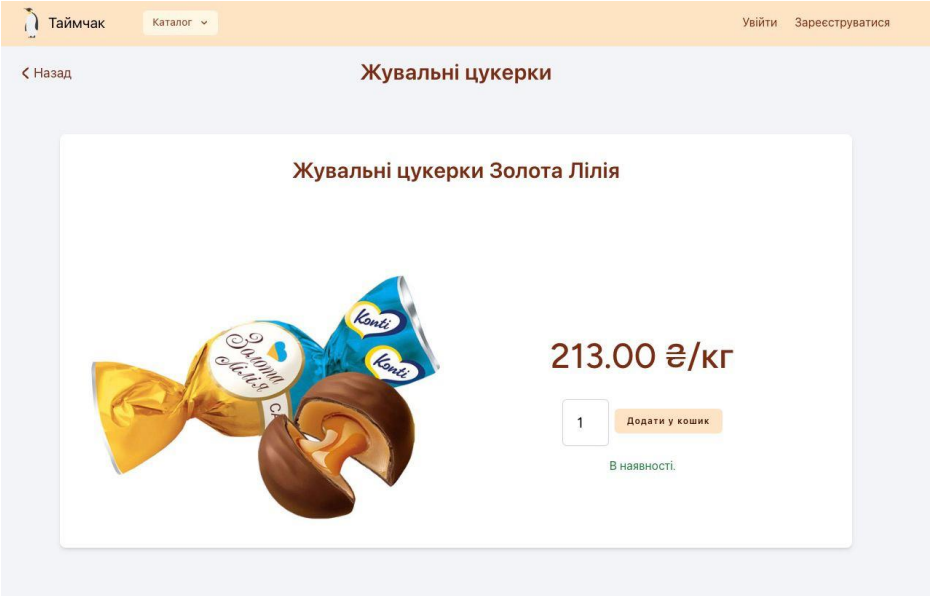

Рисунок 3.6 – Зовнішній вигляд сторінки товару

Поки кошик пустий, кнопка переходу у кошик повинна була бути не активною. Це було реалізовано перевіркою кількості товарів у кошику (код елементу посилання для переходу наведено в лістингу 3.12).

Лістинг 3.12 – Код елемента «кошик» у елементі «Header»

```
 <Link href={route('cart.index')} className="px-4 flex justify-center
items-center" disabled={isShoppingCartEmpty}>
    <div className="relative py-2">
        <div className="absolute bottom-5 left-7">
            {!isShoppingCartEmpty &&
                <p className="flex h-2 w-2 items-center justify-center
rounded-full bg-red-500 p-3 text-xs text-white">
                    {shoppingCartItems.length}
                </p>
```

Кінець лістингу 3.12

Найскладніший елемент кожного з інтернет-магазинів – кошик. Він завжди має велику кількість функціоналу, який потрібно реалізувати як на front-end, так і на логічній частині онлайн-магазину.

Спочатку було добавлено всі необхідні елементи на сторінку (рис. 3.7).

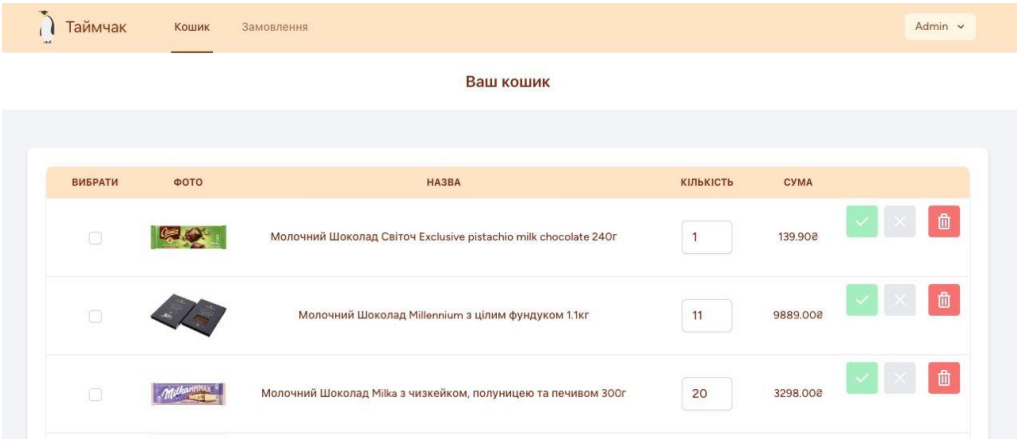

Рисунок 3.7 – Зовнішній вигляд верхньої частини сторінки кошика

Після цього потрібно було заблокувати будь-яку можливість взаємодії з товаром, який закінчився. Для цього робиться проста перевірка, чи кількість на складі дорінює нулю. Фрагмент коду з такою перевіркою наведено в лістингу 3.13.

Лістинг 3.13 – Код перевірки кількості продукції на складі

```
 {shoppingCartItems.map((item) => (
    <tr key={item.id}
        className={"border-b " + (item.good.stock === 0 ? "bg-gray-100" :
"bg-
 white")}>
        <td className="px-4 items-center text-center">
            <Checkbox
                checked={selectedItems.includes(item.id)}
```

Кінець лістингу 3.13

Після цього потрібно було зробити можливість збереження зміненої кількості товару у кошику. Для цього було додано значення «quantities» та «originalQuantities». Запис, і подальші перезаписи другого значення стаються при зміні кількості товарів у кошику (лістинг 3.14).

Лістинг 3.14 – React-хук, який перезаписує значення «originalQuantities»

```
useEffect(() => {
    const initialQuantities = {};
    shoppingCartItems.forEach(item => {
```

```
        initialQuantities[item.id] = item.quantity;
    });
    setQuantities(initialQuantities);
    setOriginalQuantities(initialQuantities);
}, [shoppingCartItems]);
```

Кінець лістингу 3.14

Тепер потрібно було організувати функціонал вибору товарів, для можливості переходу до оформлення замовлення. Це було осягнуто функцією «handleCheckboxChange» (лістинг 3.15).

Лістинг 3.15 – Тіло функції «handleCheckboxChange»

```
const handleCheckboxChange = (itemId) => {
    setSelectedItems(prevSelected => {
        if (prevSelected.includes(itemId)) {
            return prevSelected.filter(id => id !== itemId);
        } else {
            return [...prevSelected, itemId];
        }
    });
};
```

Кінець лістингу 3.15

І ще один, не менш необхідний функціонал – перехід на оформлення замовлення вибраних товарів. Для цього було використано HTTP запит POST. Функція, яка викликається при натисканні на кнопку «Перейти до оформлення замовлення» наведена в лістингу 3.16.

Лістинг 3.16 – Тіло функції, яка відповідає за передачу даних про вибрані товари у форму створення замовлення

```
{
e.preventDefault();
const itemsList = [];
selectedItems.forEach((id) =>
    itemsList.push(shoppingCartItems.find(
      cartItem =>
        cartItem.id === id)))
            router.post(route('orders.create'), {
```

```
      items: itemsList
      }
    );
}
```

Кінець лістингу 3.16

Сторінка з оформленням замовлення була зроблена простою, щоб вона складалася з багатьох полів, які потрібно заповнити. Зовнішній вигляд отриманої сторінки зображено на рисунку 3.8.

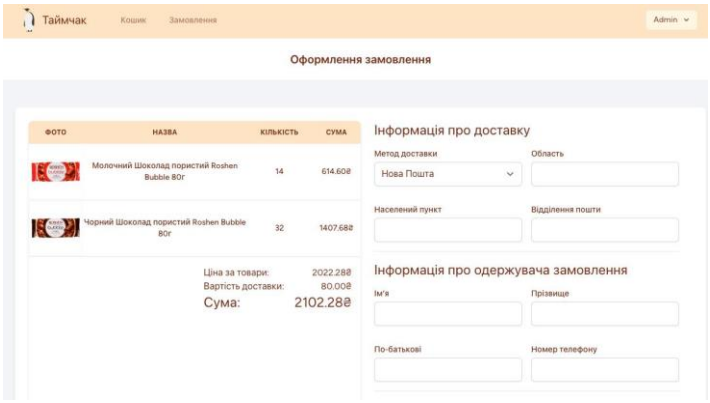

Рисунок 3.8 – Зовнішній вигляд сторінки оформлення замовлення

3.2 Тестування та налагодження веб-сайту

Протягом періоду розробки проводилися постійні мануальні юніт-тестування вводу-виводу, перевірки отриманої зовнішньої інформації. Для цього икористовувались прямі вставки в код інструментів для дебагінга. У випадку front-end це популярна команда JavaScript, наведена в лістингу 3.17.

Лістинг 3.17 – Команда для дебагінгу в JavaScript

```
console.log(data);
```

Кінець лістингу 3.17

У випадку з back-end, часто використовується одна з команд, які наведені в лістингу 3.18.

Лістинг 3.18 – Команди для дебагінгу в PHP

```
dd($data);
d($data);
```

Кінець лістингу 3.18

Приклад інформації, що виводиться при виклику першої з цих команд, щоб аналізувати отримані дані від клієнта показано на рисунку 3.9.

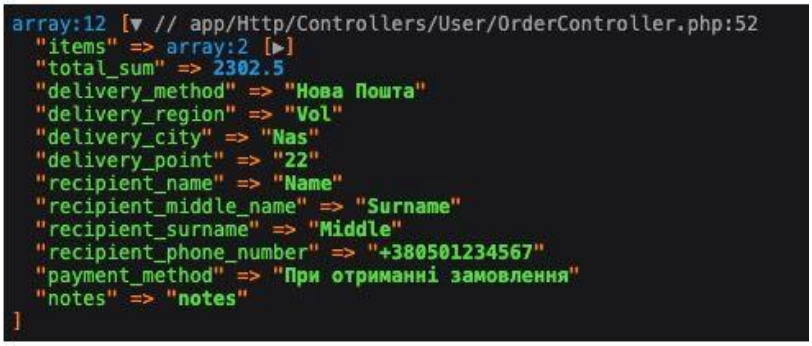

Рисунок 3.9 – Приклад інформації, отриманої при дебагінгу процесу відправлення даних у форму оформлення замовлення

Окрім юніт-тестування, проводилися і інші види тестування, такі як:

– тестування на відмову – це багаторазова одночасна спроба відправки запитів на сервер, і перевірка його реакції. Один з прикладів такого тестування – багаторазова спроба вводу пароля для автентифікації користувача (Результат на рисунку 3.10);

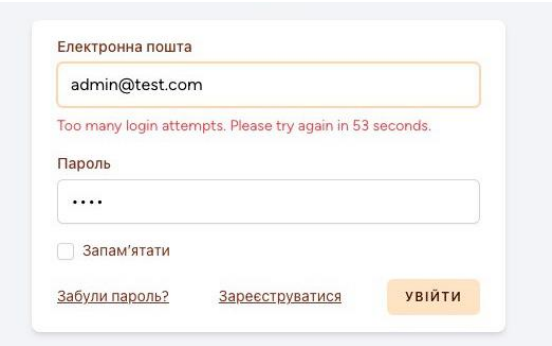

Рисунок 3.10 – Відтворення відмови при багатьох спроб входу у систему

– тестування безпеки – це перевірка реакції системи на несанкціоновані дії. Один із тестів такого типу, що проводився під час розробки – перевірка, чи можуть неавторизовані користувачі, або користувачі без ролі «Адмін» зайти на сторінку адміністративної панелі;

– E2E тестування – це тестування того, чи система покаже очікуваний кінцевий результат при проходження всього циклу випадку історії користувача. В основному проводилися два тести, тест на оформлення замовлення, та тест на додавання нової категорії в каталог.

Кроки E2E тестування функціоналу додавання нової категорії в каталог наведено у наступних кроках:

1) адміністратор заходить на сторінку авторизації, та входить у систему під обліковим записом адміністратора;

2) проводиться запис категорій, які доступні на сайті в той момент, для цього він переходить на головну сторінку магазину, отримує початковий стан системи (рис. 3.11);

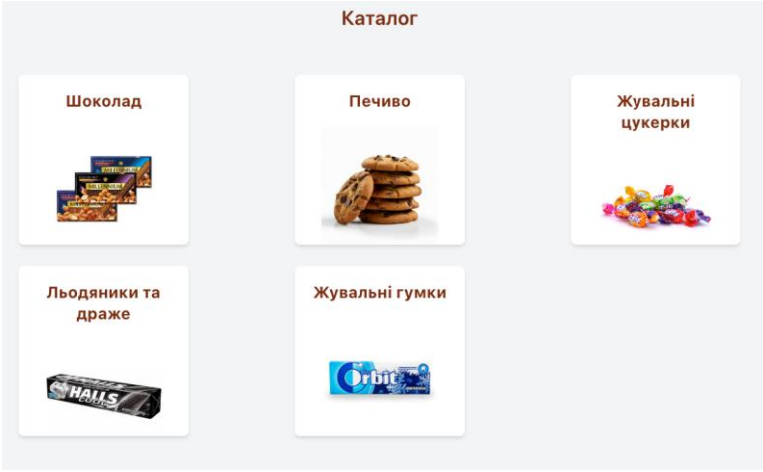

Рисунок 3.11 – Початковий список доступних категорій

3) здійснюється перехід назад на адміністративну панель, та через форму створюється нова категорія (рис. 3.12). Після цього нова категорія появляється у списку категорій на адмін-панелі (рис. 3.13);

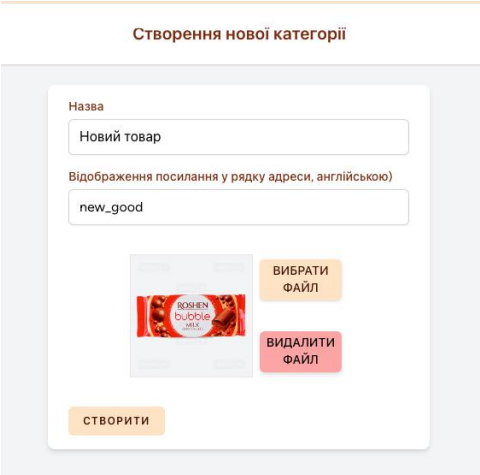

Рисунок 3.12 – Форма створення нової категорії

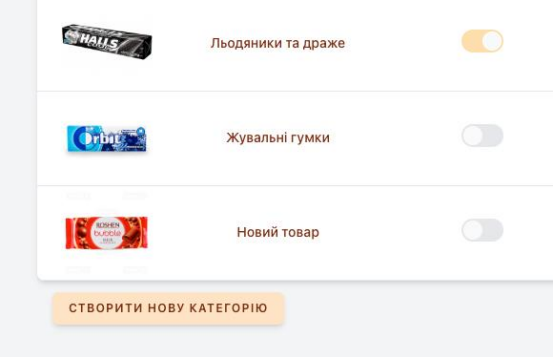

Рисунок 3.13 – Нова категорія, яка з’явилася серед інших категорій на адмін-панелі

4) далі адміністратор переходить на головну сторінку сайту, де видно щойно створену категорію (рис. 3.14);

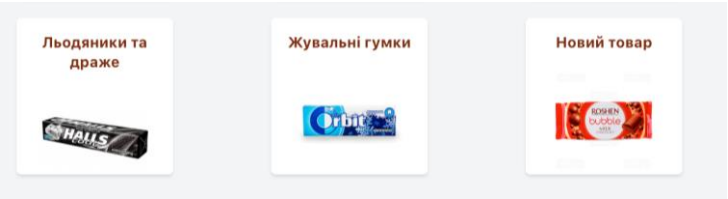

Рисунок 3.14 – Нова категорія на головній сторінці сайту

5) після цього він проводить повернення системи в початковий стан (у той же час, перевіряє, чи коректно видаляються категорії). У результаті він отримує очікуваний результат, який відповідає початковому стану.

E2E тести проводяться під час фінальних етапів розробки програмного забезпечення, а саме на етапах тестування та підтримки.

Таким чином показано, що система доволі ретельно тестувалася різними видами тестування та у періоди різних циклів розробки ПЗ.

3.3 Розробка бази даних

У Laravel є зручний інструмент для управління схемою бази даних – міграції. Вони дозволяють створювати, видаляти, вносити зміни у структуру таблиць, і все це використовуючи спеціальні PHP класи. Створюються дані класи за допомогою artisan-команди, наведеної в лістингу 3.19.

Лістинг 3.19 – Команда для створення класу міграції

```
php artisan make:migration create_shopping_cart_items_table
```

Кінець лістингу 3.19

Клас міграції складається з двох методів: «up» і «down», які створюють та видаляють таблиці, відповідно.

Всю структуру бази даних, яка використовується, було побудовано за допомогою міграцій. У лістингу 3.20 наведено приклад тіла методу «up».

Лістинг 3.20 – Приклад тіла методу міграції «up», яка створює таблицю «shopping_cart_items»

```
Schema::create('shopping_cart_items', function (Blueprint $table) {
    $table->id();
    $table->foreignIdFor(User::class)->constrained()->cascadeOnDelete();
    $table->foreignIdFor(Good::class)->constrained()->cascadeOnDelete();
    $table->float('quantity');
    $table->timestamps();
});
```

Кінець лістингу 3.20

Для безпосередньої роботи з даними, у Laravel є об’єктно-реляційний обробник (ORM) Eloquent [21], що дозволяє використовувати об'єктно-орієнтований підхід. З Eloquent розробники можуть працювати з базою даних, використовуючи моделі, які представляють записи в базі даних як екземпляри класів PHP. Це спрощує виконання запитів до бази даних, тому що дозволяє писати їх у стилі PHP замість SQL. Для створення моделей використовується команда, наведена в лістингу 3.21.

Лістинг 3.21 – Команда для створення моделі «CartItem»

```
php artisan make:model CartItem
```

Кінець лістингу 3.21

Серед доволі широкого вибору функціоналу моделей, при розробці було використано:

– видиме присвоєння назви таблиці до моделі: Модель «CartItem» очікує бачити таблицю «cart_items», проте, вручну було присвоєно назву «shopping_cart_items»;

– доступ до зв’язаних даних за допомогою «магічних методів»;

– створення додаткових атрибутів, базуючись на даних з таблиці та, навіть, використовуючи інформацію інших моделей (лістинг 3.22).

Лістинг 3.22 – Код для створення та присвоєння додаткових атрибутів

```
protected $appends = [
    'availability',
    'isOnDiscount'
];
public function getAvailabilityAttribute(): string
{
    if ($this->stock === 0) {
        return 'Немає в наявності';
    } elseif ($this->stock <= 10) {
        return 'Закінчується';
    }
    return 'В наявності';
}
```

```
public function getIsOnDiscountAttribute(): bool
{
   return (bool)$this->discount_price;
}
```

Кінець лістингу 3.22

3.4 Захист інформаційно-комп'ютерної системи

У підрозділі 3.1 уже згадувалось таке поняття, як Middleware. За доппомогою правил Middleware можна не лише передавати дані, а й забезпечувати захист веб-додатку. Наприклад, в Laravel є влаштовані Middleware, до яких прив'язані назви «auth» та «guest». Middleware «auth» працює так, що він перенаправляє користувачів з захищених маршрутів на форму автентифікації, у той час як «guest» використовується як захист маршрутів саме від авторизованих користувачів. «auth» Middleware хоч є і примітивним, проте дуже надійним захисником від несанкціонованого доступу на захищені маршрути.

У процесі розробки онлайн-магазину, були використані обидва: «auth» та «guest» Middleware. Також було створено Middleware, який був відразу віднесений до імені «admin», і ним захищено всі маршрути адмін-панелі. Він перевіряє роль авторизованого користувача, і якщо цей користувач не адміністратор, то його пересилає на головну сторінку сайту. Код перевірки вказано в лістингу 3.23.

Лістинг 3.23 – Метод-посередник між запитом та обробкою запиту на back-end

```
public function handle(Request $request, Closure $next): Response
{
   if(Auth::user()->role==”Admin”) {
       return $next($request);
   }
   return to_route('home');
}
```

Кінець лістингу 3.23

Ще одна технологія, що була використана – це валідація вхідних даних. Вона була використана в усіх методах контролерів, які приймають хоч якісь дані. Приклад валідації в лістингу 3.24 наведено з методу переходу на сторінку оформлення замовлення.

Лістинг 3.24 – Приклад використання валідації вхідних даних

```
 $data = $request->validate(
  [
    "total_sum" => "required|numeric|min:0",
    "delivery_method" => "required|in:Нова Пошта",
    "delivery_region" => "required|string",
    "delivery_city" => "required|string",
    "delivery_point" => "required|numeric",
    "recipient_name" => "required|string",
    "recipient_middle_name" => "required|string",
    "recipient_surname" => "required|string",
    "recipient_phone_number" => "required|string|regex:/^\+380\d{9}$/",
    "payment_method" => "required|in:При отриманні замовлення",
    "notes" => "string"
  ]
);
```

Кінець лістингу 3.24

3.5 Розробка документації для програмного продукту

При створенні проєкту Laravel автоматично генерується файл «README.md», який, за стандартом, використовується для технічної документації. Така документація, як правило, пишеться англійською мовою з використанням специфічної термінології, яка відноситься до предметної області проєкту. Важливо, щоб документація була конкретною, чіткою та описовою, надаючи розробникам всю необхідну інформацію для встановлення, налаштування та використання програмного продукту.

Документація до розробленого програмного продукту включає:

– передумови для локального запуску онлайн-магазину. Цей розділ містить перелік необхідних програмних та апаратних засобів, таких як

операційні системи (Windows, Linux, macOS), система контролю версій Git, Docker Engine та менеджер пакетів npm;

– покрокова інструкція з запуску онлайн магазину, такі як «скопіюйте .env.example у .env», «встановіть пакети npm» тощо.

На рисунку 3.15 показано частинку з написаної документації.

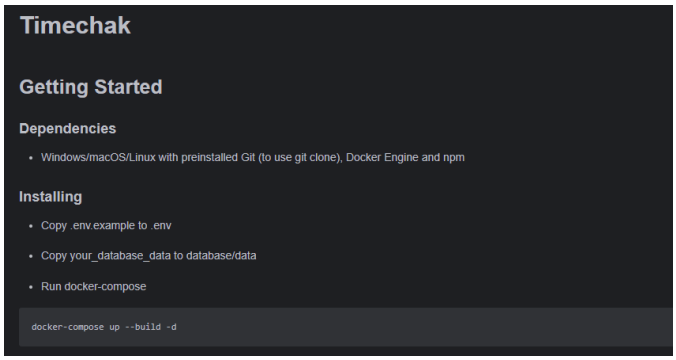

Рисунок 3.15 – Вирізка з файлу «README.md»

Висновки до розділу 3

Основою даного розділу був безпосередній процес розробки веб-застосунку (інтернет-магазину «Таймчак»). Було повністю налаштовано середовище розробки та налаштовано конфігурацію файлу середовища. Було об’єднано front-end та back-end бібліотекою-посередником Inertia. Налаштовано та піднято Docker-контейнери, які відповідають за роботу онлайн-магазину. Створено різні файли маршрутизації для веб та API-запитів, створено та заповнено класи-контролери маршрутів, налаштовано контролери для API-запитів. Створено сторінки та компоненти на базі бібліотеки React.

У процесі розробки постійно проводилися тести різних типів.

Створено базу даних, створено та налаштовано таблиці за допомогою інструмента Laravel – міграцій. Створено та налаштовано моделі Eloquent ORM.

Використано Middleware та метод валідації вхідних даних для захисту даних веб-застосунку.

Створено технічну документацію первинної розгортки системи.

ВИСНОВКИ

Результатом виконання даної кваліфікаційної роботи бакалавра є розроблений онлайн-магазин солодощів «Таймчак». У ході розробки було використано сучасні технології та інструменти, щоб покращити якість процесу та результату розробки.

Під час виконання кваліфікаційної роботи було проведено аналіз поточної ситуації на ринку онлайн-шопінгу. Описано терміни онлайн-магазину та маркетплейсу. На основі отриманих даних визначено актуальність розробки інтернет-магазину солодощів.

Сформовано схему онлайн-магазину солодощів, яка описує структуру сайту:

– каталог товарів;
– сторінка товару;
– кошик;
– форма оформлення замовлення;
– список замовлень користувача;
– панель адміністратора.

Описано структуру каталогу: категорії, підкатегорії та товари. Також описано логіку обробки даних та їх відображення при переключенні видимості підкатегорій (на випадок, коли вони не потрібні). Визначено вимоги до структури адреси сайту, а саме щоб вона відповідала стандарту RFC 3986. Обрано кремовий колір як основний в палітрі кольорів дизайну сайту.

Було розглянуто структуру архітектури веб-додатків. До неї належать:

– back-end;
– front-end;
– база даних.

Проведено детальний аналіз існуючих інструментів та методів, які використовуються у процесі розробки веб-сайтів. На основі отриманих даних було обрано наступні технології:

– фреймворк Laravel – для реалізації back-end складової архітектури системи;

– бібліотеку React, як основу для створення інтерфейсу користувача;

– систему керування базами даних MySQL;

– веб-сервер nginx;

– інструментарій для управління Linux-контейнерами Docker – для об’єднання елементів додатку у зв’язану систему.

Розглянуто типи тестування, які проводилися у процесі створення веб-сайту. Зокрема, розглянуто:

– юніт-тести;

– тести інтеграції;

– тестування безпеки;

– тестування на відмову;

– E2E тестування.

Результати тестування дозволяли вчасно знаходити помилки в структурі коду, проводити перевірку вхідних даних та перевіряти реакцію системи при різних ситуаціях.

Було запущено базу даних MySQL у власному Docker-контейнері. Налаштованно базу даних за допомогою інструментів фреймворку Laravel. Організовано маніпуляцію записами за допомогою ORM Eloquent та створення інформативних атрибутів на основі записів у БД.

Використано функціонал фреймворку Laravel для захищення маршрутів та вхідної інформації за допомогою Middleware та влаштованої механіки валідації даних.

Створено технічну документацію для первинного запуску системи додатку. У ній зазначено інструменти, які потрібні бути на комп’ютері чи сервері, на якому повинна працювати дана система та описано кроки, які слід пройти для коректної роботи додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як відкрити інтернет-магазин: покрокова інструкція створення бізнесу з нуля. URL: https://fondy.ua/uk/knowledge/online-store-creation/ (дата звернення: 12.02.2024).

2. Плюси та недоліки онлайн шопінгу. URL: http://surl.li/uorva (дата звернення: 13.02.2024).

3. Що таке маркетплейс і як він працює. URL: http://surl.li/uorvb (дата звернення: 13.02.2024).

4. Дослідження «Делойт» про споживацькі настрої українців. http://surl.li/uorvd (дата звернення: 14.02.2024).

5. Link [link] attribute Google Merchant Center Help. URL: http://surl.li/uorvg (дата звернення: 21.02.2024).

6. RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax. URL: https://datatracker.ietf.org/doc/html/rfc3986 (дата звернення: 21.02.2024).

7. Що таке фронтенд розробка: складові, етапи та технології. URL: http://surl.li/uorvi (дата звернення: 23.02.2024).

8. The Most Popular Front-end Frameworks. URL: https://stackdiary.com/front-end-frameworks/ (дата звернення: 23.02.2024).

9. Angular Roadmap. URL: https://angular.dev/roadmap (дата звернення: 27.02.2024).

10. Vue.js Introduction. URL: https://vuejs.org/guide/introduction.html (дата звернення: 27.02.2024).

11. Comparing the Best Front End Frameworks in 2024. URL: http://surl.li/uorwq (дата звернення: 04.03.2024).

12. Бійці невидимого фронту. Хто такий Backend-розробник та як ним стати? URL: http://surl.li/uorwi (дата звернення: 06.03.2024).

13. What is Laravel? A Beginner’s Introduction. URL: https://www.fastfwd.com/what-is-laravel/ (дата звернення: 06.03.2024).

14. Чому Laravel? URL: https://ensocore.com/uk/blog/chomu-laravel (дата звернення: 06.03.2024).

15. What is a database? URL: https://blog.airtable.com/what-is-a-database/ (дата звернення: 13.03.2024).

16. The Importance of MySQL Database: Advantages, Disadvantages, and Influence in AI and Data. URL: http://surl.li/uorwy (дата звернення: 14.03.2024).

17. What is a CMS? URL: http://surl.li/uorxa (дата звернення: 16.03.2024).

18. What is Nginx? URL: http://surl.li/uorxc (дата звернення: 18.03.2024).

19. JetBrains IDE Services. URL: https://www.jetbrains.com/ide-services/ (дата звернення: 12.04.2024).

20. Laravel official documentation. URL: https://laravel.com/docs/11.x (дата звернення: 14.04.2024).

21. Eloquent ORM. URL: http://surl.li/uotgv (дата звернення: 22.05.2024).