

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Створення вебсайту інтернет-магазину та адаптування його під CMS систему

Здобувача вищої освіти другого (магістерського) рівня, II курсу, КМІ-23-1 групи, спеціальності 122 «Комп'ютерні науки», освітня програма «Інформаційні управляючі системи і технології в економіці»


(підпис)

Танигіна Ігоря Всеволодовича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Грищенко Світлана
Миколаївна, к.пед.н., ст.
досл.
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Горбовий Артур Юліанович,
д.т.н, професор
(прізвище, ім'я, по батькові, науковий
ступінь, вчене звання)

Допущено до захисту за рішенням кафедри протокол від «10» січня 2025 №11

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н.,
доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Реєстрація № 1

27.01.2025

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)

27.01.2025

(дата)



(підпис)

Результат захисту

4 (добре)
(оцінка)

05.02.2025

(дата)



(підпис)

Голова комісії:



(підпис)

В.М. Франчук

Члени комісії:



(підпис)

А.Ю. Горбовий



(підпис)

М.М. Філоненко



(підпис)

І.Д. Погореловська

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	20		<div style="text-align: right;">05.02.2025</div> <div style="display: flex; justify-content: space-between;"> <div> Франчук В.М. Горбовий А.Ю. Філоненко М.М. Погореловська І.Д. Суховерхівська Л.П. </div> <div>      </div> </div>
2. Дотримання вимог щодо оформлення	від 0 до 10	6		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	35		

Оцінювання кваліфікаційної роботи науковим керівником - 37 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

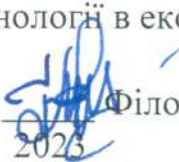
Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
72	C	4 (добре)	05.02.2025	<div style="display: flex; justify-content: space-between;"> <div> Франчук В.М. Горбовий А.Ю. Філоненко М.М. Погореловська І.Д. Суховерхівська Л.П. </div> <div>      </div> </div>

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую

Завідувач кафедри  Філоненко М.М.

« 23 » грудня 2023

**Завдання на кваліфікаційну роботу
здобувачу вищої освіти**

Танигіну Ігорю Всеволодовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Створення вебсайту інтернет-магазину та адаптування його під CMS систему

Затверджена наказом ректора від «20» грудня 2023 № 1701

2. Строк здачі закінченої роботи «07» січня 2025

3. Мета роботи є дослідження процесу створення вебсайту з впровадженням фронтендної частини та процес адаптування на русій WordPress.

Об'єкт дослідження розробка інтернет-магазину та адаптація до CMS системи WordPress

Предмет дослідження розробка інтернет-магазину та адаптація до CMS системи WordPress

Орієнтовна база дослідження Інтернет-магазин доставки ролів

4. Вихідні дані до роботи методи та підходи до розробки та створений вебсайт, що пройшов процес адаптації під CMS систему

5. Перелік графічного матеріалу: 28 знімків екранних вікон

6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)

ЗМІСТ

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.

1.1. Аналіз та перспективи електронної комерції ресторанного бізнесу в Україні.

1.2 Аналіз обраних інструментів та засобів розробки вебсайту.

РОЗДІЛ 2. ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ТА СТВОРЕННЯ ВЕБСАЙТУ.

2.1. Впровадження дизайну та верстка вебсайту.

2.2 Фронтендтна складова вебсайту.

РОЗДІЛ 3. АДАПТАЦІЯ ДО CMS СИСТЕМИ

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	За фактом
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.11.2023	15.11.2023
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	08.05.2024	08.05.2024
3	II розділ (аналіз експериментальної частини про методики досліджень)	24.09.2024	24.09.2024
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	10.11.2024	10.11.2024
5	Висновки	15.12.2024	16.12.2024
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	26.12.2024	26.12.2024
7	Отримання відгуку керівника	07.01.2025	28.01.2025
8	Отримання зовнішньої рецензії	08.01.2025	31.01.2025

8. Дата видачі завдання «23» грудня 2023

9. Керівник випускної кваліфікаційної (дипломної) роботи


(підпис)

Грищенко С.М.

(прізвище, ініціали)

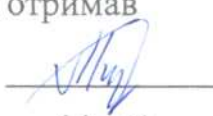
10. Керівник освітньо-професійної програми


(підпис)

Горбовий А.Ю.

(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Танигін І.В.

(прізвище, ініціали)

АНОТАЦІЯ

Створення вебсайту інтернет-магазину та адаптування його під CMS систему

Мета роботи полягає у дослідженні процесу створення вебсайту з впровадженням повноцінного фронтенду та адаптуванні до систему контролю контенту WordPress.

Завданнями роботи є:

- Проведення аналізу предметної області,
- Створення архітектури та стилізація вебсайту,
- Впровадження фронтендтної складової,
- Створення адаптивного дизайну.
- Перенесення на систему WordPress.

Предметом дослідження є розробка інтернет-магазину та адаптація до CMS системи WordPress.

Об'єктом дослідження є розробка інтернет-магазину та адаптація до CMS системи WordPress.

Базою дослідження виступає інтернет-магазин доставки їжі RollDelivery.

За результатами дослідження сформульовані методика та етапи реалізації вебсайту.

Одержані результати можуть бути використані при оптимізації бізнес процесів ресторану та залученню нових клієнтів завдяки використанню вебсайту.

Кваліфікаційна робота містить 53 сторінок, 0 таблиць, 28 рисунків, список використаних джерел 42 найменувань, 8 додатка.

Ключові слова: вебсайт, інтернет-магазин, фронтенд, CMS система, бізнес, користувацький інтерфейс, дизайн, функціональність.

Рік виконання кваліфікаційної роботи 2024

Рік захисту роботи 2025

ABSTRACT

Creating an online store website and adapting it to the CMS system

The purpose of the work is to study the process of creating a website with the introduction of a full-fledged frontend and adaptation to the WordPress content control system.

The tasks of the work are:

Carrying out an analysis of the subject area.

- Carrying out an analysis of the subject area,
- Creating the architecture and styling of the website,
- Implementation of the frontend component,
- Creation of adaptive design.
- Migration to the WordPress system.

The subject of the research is the development of an online store and adaptation to the WordPress CMS system.

The object of the research is the development of an online store and adaptation to the WordPress CMS system.

The basis of the research is the online real estate store RollDelivery.

Based on the results of the research, the methodology and stages of website implementation were formulated. The obtained results can be used to optimize the business processes of the restaurant and attract new customers through the use of the website.

The qualification work consists of 53 pages, 0 tables, 28 images, a list of 42 references, and 8 appendices.

Keywords: website, online store, frontend, CMS system, business, user interface, design, functionality.

Year of qualification work: 2024

Year of work defense: 2025

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	5
1.1. Аналіз та перспективи електронної комерції ресторанного бізнесу в Україні.....	5
1.2 Аналіз обраних інструментів та засобів розробки вебсайту	7
РОЗДІЛ 2. ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ ТА СТВОРЕННЯ ВЕБСАЙТУ	17
2.1. Впровадження дизайну та верстка вебсайту.....	17
2.2 Фронтендтна складова вебсайту.....	22
РОЗДІЛ 3. АДАПТАЦІЯ ДО CMS СИСТЕМИ	37
ВИСНОВКИ.....	42
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	44
ДОДАТКИ.....	48

ВСТУП

Сучасний рівень розвитку інформаційного суспільства та інтелектуальних сфер діяльності людини базується на взаємодії цифрових технологій. Постійний зріст бізнесу та підприємницької діяльності, призводить до потреби з власному інтернет-середовищі, яким являється вебсайт. Задача вебсайтів полягає у поширенні та просуванні власних послуг у інтернет сфері. У цих умовах розробка вебсайту та адаптація його до CMS системи є досить актуальною. Зокрема сутність даної проблеми розкривали такі автори як R. Nicson, B. Williams, H. Stern, D. Demstra та інші.

На сьогоднішній день є досить багато способів щодо комерційних підходів до Інтернету. Мережа Інтернет дозволяє здійснювати рекламу та просувати свої послуги різноманітними способами, або займатися продажем власних товарів та послуг. Комерційний вебсайт вважається тоді, коли основним завданням якого являється отримання прибутку.

Раніше переважна більшість вебсайтів не приносили на пряму дохід, а тільки сприяли до його зростання, одержуваного звичайними засобами. Мета таких сайтів полягала лише у приверненні уваги до клієнтів та власної реклами щодо певної комерційної діяльності. В даний час все більше і більше компаній та підприємств просто вимушені продавати свої товари та послуги через Інтернет аби розвиватися.

Метою роботи є дослідження процесу створення вебсайту з впровадженням фронтендної складової та процес адаптування до системи контролю контенту WordPress.

Для досягнення мети необхідно вирішити такі **завдання**:

- дослідити та проаналізувати теоретичну базу досліджуваного питання;
- вивчити предметну область і провести аналіз досліджуваної області;
- розробити дизайн;

- розробити верстку інтернет-магазину по продажі суші;
- застилізувати вебсайт включно з роботою над адаптивним дизайном.
- впровадити всі необхідні скрипти.
- адаптувати вебсайт до перенесення на WordPress.

Об’єкт дослідження – вебсайт, система контролю контенту WordPress.

Предмет дослідження – інтернет-магазин по продажі страв японської кухні.

Для досягнення поставленої мети було використано наступні методи дослідження:

Методи дослідження – *теоретичні* – аналіз, узагальнення, систематизація законодавчої бази, матеріалів конференцій, Інтернет-ресурсів для виділення теоретичних засад дослідження; *емпіричні* – аналіз досвіду роботи в даній галузі, для констатування стану розв’язання проблеми.

База досліджень – Інтернет-магазин по продажі страв японської кухні.

Наукова-практична значимість одержаних результатів. Розробка інтернет-магазину продажі нерухомості з впровадженням повноцінного фронтенду та адаптуванням щодо перенесення до системи управління контентом дозволяє вийти на більш широкий рівень розробки вебсайту, та підвищити якість обслуговування підприємства аба замовників. Завдяки перенесенню вебсайту на CMS WordPress власник зможе дуже зручно та комфортно екрувати змістом та наповненням вебсередовища.

Апробація результатів роботи. Матеріали висвітлено в збірнику тез IV міжнародної науково-практичної інтернет-конференції (Дубляни, 16 травня 2024 р.). Частина II. Дубляни: ЛНУП, 2024. с. 317-321. URL: <http://surl.li/ehxlcy>.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз та перспективи електронної комерції ресторанного бізнесу в Україні

При стрімкому розвитку сфери інформаційних технологій змінюються умови ведення бізнесу. Формування цифрової економіки призвело до зміни парадигми щодо функціонування підприємницької діяльності. Завдяки доступності мережі Інтернет та з розвитком інформаційних технологій, збільшувався попит на онлайн-торгівлю (e-commerce). На сьогоднішній день переважна кількість компаній по можливості переводять свою діяльність в мережі Інтернет, тобто переходять на онлайн-платформи для продажу власної продукції. В Україні онлайн-торгівля займає досить широкий спектр пропонуючих послуг та товарів. Дана комерційна діяльність в Україні являється перспективною з великим потенціалом для подальшого розвитку.

По даним опитування Deloitte[12] в Україні, не за довго до війни, два роки поспіль спостерігалася тенденція у двократному переважанні темпів зростання режиму онлайн над офлайн. Дана тенденція мала характер лише посилюватися. Такий ефект можна співвіднести до наслідків пандемії, яка вимушила людей обирати послуги в режимі онлайн. Як показало дослідження Soul Partners[7], в Україні обсяг ринку електронної комерції зріс на 41% в 2020 році та досягнув \$4 млрд, що становило 8,8% від загального обсягу роздрібної торгівлі в Україні. У наступних роках дана частка тільки збільшувалася. Ринок e-commerce виріс майже втричі порівняно з 2016 роком та аналізуючи дану статистику, аналітики прогнозували її подвоєння протягом наступних п'яти років.



Рис. 1.1. Динаміка ринку електронної комерції та її частки у роздрібній торгівлі.

Незважаючи на те що придбані товари та послуги поступово кількісно зростають в Україні, але все одно дані показники є нижчими порівнюючи їх з середнім значенням по країнах Європейського Союзу.

На даний момент можна сказати що український ресторанний бізнес проходить справжні випробовування та у свою ж чергу демонструє своєю стійкістю. Подія війни в Україні викликала надзвичайно складну ситуацію на ринку індустрії гостинності, таким чином що виникла неспроможність рестораторів стратегічно планувати розвиток свого закладу на довгострокову перспективу. На даний момент ринок ресторанного бізнесу зменшився майже на чверть порівняно з лютим 2022 року. У певних областях, переважно східних, зниження ринку становило понад 50%, у центральних областях – падіння до 30%. В той же час у західних областях спостерігається навпаки позитивна динаміка. Так у Львівській області кількість об'єктів ресторанного бізнесу зросла на 30%, у Закарпатській, Чернівецькій, Івано-Франківській областях зростання сягає приблизно на 20%. Ресторанні інвестори очікувано стали вважати місцевість Західної України вигідними для відкриття нового бізнесу у даній сфері послуг. Таким чином ресторанний бізнес має позитивні тенденції.

1.2 Аналіз обраних інструментів та засобів розробки вебсайту

Загальноприйнятим фактом у сучасному світі є те, що аби мати можливість отримати певну інформацію в мережі Інтернет, необхідно взаємодіяти безпосередньо через вебсайт.

Поняття вебсайту – це певна кількість логічно зв'язаних вебсторінок, що як правило складаються з гіпертекстової інформації та об'єднуються за своїм змістом та типами даних, оформленні у вигляді окремих сторінок і є доступними в мережі Інтернет. По іншому вебсайт являється певною базою даних, що містить програмні, інформаційні та медійні засоби, що систематизуються і є логічно зв'язаними.

Поняття сайту – це лише сукупність певних інформаційних та медіа файлів, що об'єднані гіпертекстовими посиланнями і спільним доменним ім'ям.

Вебсайт виконує ряд основних завдань:

- безпосередня реклама самого продукту або послуги, що компанія надає. Вебсайт компанії дозволяє якнайшвидше залучити клієнта придбати та обрати товар, послугу, саме на цьому сайті не переходячи на інші;
- продаж певного товару, послуг, інформаційних матеріалів чи ідей. У наші часи, у сучасної людини є багато важливих справ та обов'язків, тому буває не вистачає часу оф-лайн взаємодію з магазином, що в свою чергу робить актуальним можливість взаємодії з інтернет-послугами, придбання продукту через мережу інтернет;
- безкоштовне надання послуг. Надання послуг через Інтернет надає змогу залучати клієнтів для придбання товару, без фізичної взаємодії з магазином, та користуючись тільки інтернетом;
- підтримка клієнтів.

Поняття вебсторінки – засіб отримання інформації, де інформація має представлення у HTML форматі та вміщає гіпертекст та навігаційні

гіперпосилання на інші окремі сторінки. Розповсюдженими типи вебсторінок представлено на (рис. 1.1).



Рис. 1.2. Схема розповсюджених типів вебсторінок

Сукупність вебсторінок, що об'єднані однією темою чи метою та концепційно сформованих як єдине ціле – є вебсайтом. Найпоширенішими типами вебсайтів є: – статичні (лендінги); – веб-спільноти; – інтернет-магазини; – портали; – блоги.



Рис. 1.3. Схема типів сайтів

Проаналізуємо основні поняття та функції інтернет-магазину та електронного бізнесу.

З появою комп'ютерних мереж, в свою чергу, стрімко покращилися умови життя у сучасному суспільстві. З часом мережа інтернет все більше домінує у різних сферах та стає основним каналом соціальних комунікацій. Зокрема, самою перспективною формою бізнесу, що динамічно розвивається, являється електронний бізнес.

Електронний бізнес позитивно впливає на підприємства, що дозволяє значно збільшувати економічну ефективність за рахунок просування товарів та послуг, розширення ринків збуту та розвитку взаємовідносин із

клієнтами. Важливим фактором є зниження витрат, економія часу при обслуговуванні клієнтів та збільшення їхньої кількості.

Перш за всього потрібно розібрати визначення електронного бізнесу та електронної комерції, аби розуміти зону застосування.

Електронний бізнес – термін, який використовується для позначення діяльності, спрямованої на підвищення ефективності організації за рахунок використання інформаційних технологій. Електронний бізнес включає взаємодію ділових партнерів, виробничі та організаційні відносини між працівниками, підприємствами, установами, некомерційними громадськими організаціями, та іншими об'єктами. До прикладів електронного бізнесу можна віднести електронну та мобільну комерцію або електронний консалтинг. Електронний бізнес дозволяє здійснювати продаж, купівлю, дистанційне обслуговування, проведення різних маркетингових програм за рахунок використання комп'ютерних технологій через інтернет.

У свою чергу, електронну комерцію можна визначити як підприємницьку діяльність щодо здійснення комерційних операцій із застосуванням електронних засобів обміну даними, в тому числі й інтернету. Електронна комерція являється окремою складовою електронного бізнесу. Як правило, характерною ознакою електронної комерції є наявність усіх етапів укладання угоди, а саме: продаж, оплата та доставка.

Слід звернути увагу на те що здійснювати продажу товарів та послуг у мережі Інтернет можна різними способами.

Інтернет-магазин – самий ефективний та привабливий спосіб електронного бізнесу, який призначений для продажу різноманітного майна або послуг покупцям та використовує у процесі ведення бізнесу електронний засіб комунікації – Інтернет.

Поступове трансформування Інтернету з набору лише інформаційних ресурсів, у платформу розробки, керування та ведення бізнесу, такі технології як створення вебсайтів досить змінилися та вдосконалились. На сьогоднішній день існує дуже багато різноманітних методів створення та

реалізації вебсайтів. Кожний метод розробки відрізняється за своїм типом і призначенням сайту, а також за професійними навичками самого розробника.

Такими методами слід виділити:

- повна розробка;
- застосування онлайн-конструктора;
- використання системи управління сайтом.

Аби пройти повний процес розробки вебсайту, необхідно володіти такими навичками й засобами розробки та веб-програмування як HTML, CSS, в залежності від задачі і функціоналу - знанням мов програмування PHP та Java Script. Також будуть умісні різноманітні допоміжні програми, інструменти та збірки. Для повної розробки в режимі «під ключ» також необхідно використовувати CMS систему (Система Управління Контентом), найпопулярнішою з яких є WordPress.

За змістовне наповнення надання структурованості тексту у вебсторінках відповідає HTML (Hyper Text Markup Language). Дана технологія являє передову мову розмітки тексту, яка слугує системою для створення гіпертекстових документів.

Visual Studio Code — редактор коду, зручний інструмент для створення та редагування програм та веб-застасунків. VS Code є безкоштовним і доступним для всіх операційних систем, крім мобільних.

Для роботи над проектом було обрано саме цей редактор коду завдяки його зручності щодо написання коду, вбудованим плагінам, таким як «Emmet», що дозволяють досить ефективно пришвидшити процес роботи та встроєному терміналу.

HTML - мова програмування гіпертексту, що будує каркас сайту, використовується для розмітки тексту, засобом чого є теги. Ці оператори потрібні для того, аби розташовувати та вказувати на той чи інший текстовий чи графічний елемент на вебсторінці. Основою функціональності HTML – є гіперпосилання. Гіперпосилання являються базовим інструментом HTML-документу, що забезпечує зв'язок між об'єктом вебсторінки та іншим

документом. Для створення гіперпосилання використовується як фрагмент тексту, так і графічний елемент, де гіперзв'язок встановлюється як між елементами одного вебсайту, так і між елементами розміщеними на інших сайтах мережі Інтернету.

CSS – це каскадна таблиця стилів, технологія що застосовується щоб мати вплив на зовнішній вигляд документа[39].

В разі роботи з веб-сторінками, CSS є технологією що використовується для задання кольору, шрифту, розташування всіх елементів на сторінці та маніпуляційних дій щодо них, відображення анімацій, певних ефектів та мінімального дизайну. Але в даній технології є невеликий недолік – інші браузері можуть інтерпретувати задані правила CSS по своєму, і в свою чергу, по різному відображати ті чи інші фрагменти сторінки.

Bootstrap – переважно CSS фреймворк під стандартні шаблони верстки адаптивних дизайнів. Як правило застосовується при розробці вебсайтів та веб-інтерфейсів. Даний фреймворк значно пришвидшує процес розробки завдяки наявності шаблонів дизайну для всіх тегів та елементів на веб-сторінці[23]. Головною перевагою є те що він надає можливість значно легше робити адаптивні дизайни.

Фреймворк Bootstrap має відкритий вихідний код, тож він має вільний доступ на Github, також має ліцензію MIT. Таким чином він є безкоштовним та доступним у користуванні.

Bootstrap являється ще доволі популярним серед розробників та є рекомендованим скілом у багатьох компаніях. По бажанню та необхідності даний фронтенд фреймворк можна досить легко налаштувати під себе через редагування змінних та використанню міксинів. Початківцю і без глибоких познань та достатньої практики є можливим швидко створювати якісні макети.

Фреймворк Bootstrap складається з набору CSS та JavaScript файлів, які необхідно підключити до проекту, як і всі плагіни. Зробити це можна

локольно просто підключити їх до сторінки або підключити з CDN (Content Delivery Network).

До переваг слід відзначити:

1. Простота у застосуванні: будь який новачок, котрий має базові знання, може з легкістю його використовувати;
2. Висока швидкість процесу створення адаптивної верстки;
3. Можливість налаштування на свій лад;
4. Єдиний стиль всіх компонентів. Широкий функціонал адаптиву від широких комп'ютерів до малих телефонів;
5. Кросбраузерність – Сумісність з усіма роружарними браузерми. Тобто коректне відображення у всіх підтримуваних цим фреймворком браузерах.

Серед мінусів даного фреймворку є те що існує дуже велика складність верстати сайти не по загальним шаблонам та з унікальним дизайном, та на виході більший за розміром кінцеві css та js-файли проекту.

JavaScript це динамічна, високорівнева мова програмування, що активно використовується у веб-розробці та відповідає за функціонал та надає інтерактивності вебсайтам. JS контактує з діями користувача, робить анімації, модальні вікна та спливаючі повідомлення, оновлює контент та валідує форми. Саме JavaScript відповідає за повноцінний фронтенд у веб-застосунках та вебсайтах. Послідовність виконання інструкцій (скриптів), здійснюється завдяки вбудованим у звичайні веб-браузери інтерпретатори.

Основні переваги та особливості мови програмування JavaScript:

- **Інтерпретованість.** Код JavaScript зчитується в режимі реального часу в середовищі браузера, без необхідності у попередній компіляції. Тож при розробці вносені зміни в коді відразу становляться наявними.
- **Інтеграція з HTML і CSS.** JavaScript підходить до спільної роботи з даними технологіями розробки, даючи повноцінну змогу створювати повнофункціональні вебсторінки.

- **Динамічна типізація.** Типи змінних визначається у автоматичному режимі.
- **Модифікованість.** Існує величезна кількість плагінів та бібліотек що присвідшують то чи інше завдання або функціонал. Також досить багато фреймворків які обумовлені наданням зумовлених функцій для прискорення процесу створення програм.
- **Кросплатформеність.** Коректна робота коду незалежно від платформи чи пристрою користувача.
- **Автоматичний клінінг.** Видалення не потрібних об'єктів у автоматичному режимі, таким чином полегшуючи процес керування пам'яттю.
- **Об'єктно-орієнтований принцип програмування та імперативність.** Мова JavaScript включає в собі основні парадигми програмування, дозволяючи проектувати складні структури даних, покращуючи організаційний процес кодингу.
- **Асинхронність.** Можливість мови JavaScript продовжувати опрацьовувати завдання незалежно від завершення попереднього завдання.
- **Підтримка DOM (Document Object Model).** Надає можливості до динамічної взаємодії з елементами вебсторінки, оброблюючи події та вносячи зміни.
- **Велика спільнота.** Навколо мови JavaScript сформувалася досить велика спільнота розробників, що постійно розвивають її, створюючи нові фреймворки, бібліотеки, плагіни, підручники на інші навчальні матеріали.
- **Відкритий код.** JavaScript являється доступним для використання та створенню модифікацій без ліцензійних обмежень.

jQuery представляє найпопулярнішу бібліотеку JavaScript, тобто фреймворк, що розроблений на основі технології JS. Дане програмне забезпечення дозволяє спростити написання досить великих фрагментів коду і включає дуже велику кількість вже заготовлених функцій, що дуже допомагає при розробці різних інтерактивних елементів на сторінці[33]. JQ прагне відмежуватися своєю поведінкою від загальної HTML-документа. Тобто коли потрібно зробити обробку подій під час натискання на кнопки, то замість цього зазначення, управління надається безпосередньо JQuery, де він спершу визначає саму кнопку, після передає цей сигнал безпосередньо до обробки події. Такий процес відокремленої поведінки дозволяє зробити JS гнучкішим.

З серед різноманітних модулів бібліотеки jQ існує велика кількість вигідних компонентів, що використовуються розробником у застосуванні широкого спектру завдань. Натомість, розробники даної бібліотеки не мали на меті створити набір інструментів, що дозволяють поєднувати велику кількість jQ-функцій, так як це в свою чергу призведе до утворення досить об'ємних фрагментів коду. Тому обрано архітектуру універсального ядра, що має в собі бібліотеку та плагіни. Такий підхід надає здібність об'єднувати лише ті JS-функції, що дійсно затребувані при розробці.

CMS (Content Management System) — представляє собою систему керування вмістом вебсайту. Дане програмне забезпечення призначено для організації вебсайтів чи інших інформаційних ресурсів. В колі фронтенд розробників CMS ще має назву «двигун сайту». За даними дослідження агентства W3Techs[3], за допомогою CMS працює більше половини вебсайтів. Найпопулярнішими представниками CMS є WordPress та OpenCart.

Wordpress – це найпопулярніший представник CMS, за допомогою якого успішно функціонують різні платформи, починаючи від блогів і закінчуючи найпопулярнішими майданчиками світу.

PHP – це мова програмування загального призначення з відкритим вихідним кодом. Основне завдання та здібність даної мови полягає у створенні вебдодатків та сайтів, де її код може прописуватися безпосередньо у розмітку HTML. На відміну від JavaScript PHP застосовується на стороні бекенд, маючи властивість керувати серверними файлами, обробляти різні функції та змінюючи бази даних.

По завершенню роботи даний проект буде закомічено у віддалений git-репозиторій у Github, завдяки системі Git, аби додати його до списку проектів та залишити посилання у списку джерел для наявного демонстрування.

Git – це програмне забезпечення, що систематизує керування та зберігання файлів та їх версій. Завдяки даній системі, розробники можуть відстежувати кожен версію розроблюючого проекту, та за потребою «відкатитися» на більш ранню версію. Також дана технологія забезпечує надійне шифрування, та асинхронні робочі процеси, тому аби працювати з потрібними розділами проекту немає залежності від місцезнаходження.

Github представляє найпопулярніше на даний час віддалене сховище для git-репозиторіїв. Головні його особливості та призначення полягає в тому що він дозволяє встановлювати права доступу до проектів, відстежувати помилки, приймати запити на поліпшення та використовувати графічний інтерфейс, а не командний рядок[16].

Git-репозиторій – це певний проект із файлів та папок, що використовується під час взаємодії з git. Дане віддалене сховище складається з усіх версій змін проекту та всього матеріалу що було додано вами.

З перерахованих вище технологій та засобів я обираю для реалізації проекту як через їх пряму доцільність, так і для отримання нового досвіду та експериментування.

РОЗДІЛ 2. ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ТА СТВОРЕННЯ ВЕБСАЙТУ

2.1. Впровадження дизайну та верстка вебсайту

Веброзробка являється складним та комплексним процесом, який включає декілька етапів. Початковим етапом є робота над дизайном вебсайту, де спершу необхідно визначитися з загальною концепцією дизайну та планом розташування елементів контенту з урахуванням того як всі блоки розмітки будуть групуватися в залежності від різноманітної зміни екрану пристроїв користувача. Тобто кінцевою стадією початкового етапу є планування адаптивного дизайну. Також під час роботи над дизайном вебсайту плануються різні мінімальні інтерактивні елементи такі як ховерні ефекти та анімації.

Надалі, по завершенню проектування дизайну, слід переходити до верстки вебсайту, де всі ідеї дизайну мають бути відображеними безпосередньо у вікні браузера та на різноманітних пристроях. Під час цього етапу роботи краще працювати поетапно, тобто виконати спершу один блок розмітки в html-документі та одразу його застилізувати через css-документ, а після переходити до інших.

По завершенню етапу верстки треба переходити до адаптування вебсайту під різні пристрої від широкоекранного комп'ютера до компактного мобільного телефону. Під час цього процесу необхідно взаємодіяти з інструментами розробника, що є вбудованим інструментом у браузерах, та одразу тестувати під найпоширеніші пристрої.

Це був загальний план та структура роботи над даним розділом. Тож переходимо до безпосереднього процесу.

Дизайн вебсайту буде мати дещо мінімалістичний характер, але компактно спроектований та доречно спроектований відносно психології дизайну, де каталог розташовується по центру, а другорядна інформація з правого боку екрану. Планується ховерний ефект над кнопками зміни

кількості товару та додавання до кошику у вигляді плавної зміни кольору фону та шрифтів.

Щодо шрифтів, то в більшості ключових елементів буде використовуватися шрифт «Merriweather»[5], до головного заголовку буде підключено шрифт «Presentscript-cyrillic» так як роли асоціюються з японським колоритом, тож було б доречно застосувати адаптований під кириличну писемність японський шрифт. До невиразних елементів додано по дефолту стандартний шрифт у вигляді «Sans-serif».

У вигляді логотипу виступає векторне зображення у вигляді парних суші у форматі SVG.

Під фонове зображення було обрано картину в рожевих барвах у японському настрої у форматі JPG.

Переходячи до розмітки слід підключити всі необхідні файли, скрипти та посилання такі як файли ссс-стилів, бутстрапу, файли js-скриптів з бібліотекою jQuery та посилання на шрифти.

Першим блоком іде шапка сайту (header) з логотипом, заголовком та підзаголовком котрі розташовані один за одним. Так як специфічного шрифту під японську писемність немає у наявності у GoogleFonts, тож прийшлося його скачати на сторонньому сайті та підключити власноруч[10].

```
@font-face {
  font-family: "presentscript-cyrillic";
  font-style: normal;
  font-weight: 400;
  src: local("presentscript-cyrillic"),
       url("/fonts/presentscript-cyrillic.woff2") format("woff2"),
       /* url("/fonts/presentscript-cyrillic.woff") format("woff"); */
}
.title {
  font-family: "presentscript-cyrillic", sans-serif;
  font-size: 52px;
  font-weight: 500;
}
```

Рис. 2.1. Налаштування локального шрифту.

Після розтошовується контейнер з головним контентом де розтошовуються каталог з наявними стравами з вибором категорій щодо них та інформативний блок для користувача щодо його замовлення та статусу доставки.

```
<div class="container mb-5">
  <div class="row">
    <div class="col-md-8">
      <div class="row">
        <div class="col-md-6">
          <div class="card mb-4">
            
            <div class="card-body text-center">
              <h4 class="item-title">Філадельфія хіт рол</h4>
              <p><small class="text-muted">6 шт.</small></p>

              <div class="details-wrapper">
                <div class="items">
                  <button class="items__control" id="minus">-</button>
                  <div class="items__current" id="number">1</div>
                  <button class="items__control" id="plus">+</button>
                </div>

                <div class="price">
                  <div class="price__weight">180г.</div>
                  <div class="price__currency">160 ₴</div>
                </div>

                <button type="button" class="btn btn-block btn-outline-warning" id="goBasket">+ до кошика</button>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Рис. 2.2. Приклад одного елементу товару в каталозі.

Дана розмітка на етапі верстки слугує як початкове креслення, щоб мати уявлення як елементи розвертаються на html-сторінці та застилізовані засобами CSS включно з адаптивом звичайно. Бо надалі всі дані товарів будуть підбератися з масиву даних у JSON форматі, а самі товари викликатися через функції рендерингу з підстановкою даних методами інтерполяції.

Для ключових елементів розмітки прописуються власні ідентифікатором ("id"), для більш точного параметру пошуку даного елемента. Також надалі головним елементам буде надано data-атрибути для більш широкої взаємодії з ними у файлі скриптів js.

```

<!-- cart-wrapper -->
<div class="cart-wrapper">
  <div id="cartItemsHolder">
    <div class="cart-item" data-id="1">
      <div class="cart-item__top">
        <div class="cart-item__img">
          
        </div>
        <div class="cart-item__desc">
          <div class="cart-item__title">Філадельфія хіт рол</div>
          <div class="cart-item__weight">6 / 180</div>

          <div class="cart-item__details">
            <div class="items items--small counter-wrapper">
              <div class="items__control" data-action="minus">-</div>
              <div class="items__current" data-counter="">1</div>
              <div class="items__control" data-action="plus">+</div>
            </div>

            <div class="price">
              <div class="price__currency">180</div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

```

Рис. 2.3. Приклад одного елементу товару у корзині.

Для товарів в корзині принцип розмітки є аналогічний каталогу, де в html файлі залишиться лише обгортка для контенту, а всі елементи перенесуться до функцій рендиру, та надалі дещо зміняться.

В момент коли користувач ще не визначився з вибором товару, на сторінці це має бути зазначено – тобто блок порожньої корзини.

```

<!-- Корзина Порожня -->
<div id="cart" class="card mb-4">
  <div class="card-body">
    <h5 class="card-title">Ваше замовлення</h5>

    <div id="cartEmpty" class="alert alert-secondary" role="alert">
      Корзина порожня
    </div>
  </div>
</div>

```

Рис. 2.4. Приклад одного елементу товару в корзині

Але для сценарію коли користувач таки обрав що він планує замовляти, елемент порожньої корзини з відподним ідентифікатором необхідно буде приховати, а інформаційні контентні блоки замовлення навпаки відобразити. Де такі елементи як форма оформлення замовлення, тарифу доставку та загальної ціни спешу приховані через клас «none» з параметрами «display: none»

```


Доставка:



Всього:


₴



##### Оформити замовлення


```

Рис. 2.5. Приклад одного елементу товару в корзині

Основна стилізація розмітки зроблена засобами bootstrap. Адаптивний дизайн в ньому теж передбачен але як і все не ідеально в ньому. Тож варто переходити до адаптивної стилізації. Код стилів під різну ширину та висоту екрану, тобто під пристрій користувача, можна прописувати одразу у головному файлі css, але під цю задачу краще виділити окремий документ в організаційних міркуваннях. Де під конкретні налаштування екрану буде відповідати певний блок коду. Також не треба забувати підключати даний

файл у тезі head службових посилань в кінці списку, щоб основні стилі не перебивали пріоритет адаптивних.

```
@media (max-width: 767px){
  .catalog-menu{
    flex-direction: column;
  }
  .catalog-menu-item button {
    width: 100%;
  }
}
@media (max-width: 600px){
  .catalog-menu {
    gap: 20px;
  }
}
```

Рис. 2.6. Приклад медіа запитів для адаптиву

По завершенню роботи над адаптивною стилізацією, тобто фінальною стадією верстки, необхідно себе перевірити та протестувати застосунок на різноманітних пристроях. Для цього зручно скористатися інструментами розробника, що є вбудованим засобом браузерів. Щоб його викликати та взаємодіяти з ним треба натиснути f12 або обрати через панель задач на сторінці.

2. 2. Фронтендтна складова вебсайту

По завершенню роботи з розміткою вебсайту та повністю застилізувавши його, слід переходити до фронтендтної частини – тобто програмні вказівки щодо взаємодії користувача з інтерфейсом інтернет-магазину.

Даний код буде написано мовою JavaScript формату ES6 та з дотриманням патерну MVC.

Патерн MVC (Model View Controller) являється архітектурним шаблоном написання коду, що дозволяє структурно розбити ваш застосунок

на три логічні компоненти, з чого і складається назва, і від цього ефективніше керувати даними, їх відображенням та безпосередньою логікою користувача[35].



Рис. 2.7. Організаційна структура патерну MVC

Модель (Model) має бути першим компонентом та відповідати за обробку даних програми. Саме вона зберігає інформацію та вирішує як її обробляти та змінювати.

Далі, коли дані в моделі було якимось чином змінено, це має бути відображено візуально. За це відповідає компонент Вигляду (View), котрий представляє інформацію з Моделі у зручному вигляді для користувача.

Мозговим компонентом являється компонент Контролер (Controller), який взаємодіє з користувачем та відправляє команди до Моделі та Вигляду. Він контролює усі дії та потік інформації між Моделлю та Виглядом.

Тобто можна зазначити що даний патерн здатен перетворити складну розробку застосунків на більш зручний та керований процес. Також MVC на перспективі є дуже зручним для командної роботи, де декілька розробників можуть працювати окремо над кожним компонентом програми.

Тож переходячи до розробки слід спершу створити json файл який буде зберігати масив даних до товарів. В структурному вигляді це масив з об'єктами, де в кожному з яких прописано інформація у вигляді ключ-значення. Під ключами зберігається така інформація як id, назва товару, його ціна, вага, кількість в комплекті та назва картинки.

Для роботи з каталогом товарів створено окрему папку де переходимо до файлу Моделі. Створюється клас з відповідною назвою, який одразу експортується щоб надалі Контролер мав змогу взаємодіяти з ним. В конструкторі класу зазначаємо властивість «products» який на старті рівний порожньому масиву. Аби заповнити цей порожній масив даними товарів, необхідно застосувати асинхроне програмування. Так як мова програмування JavaScript є синхроною – тобто код виконується послідовно але одночасно, то треба викликати асинхрону функцію, що чекає поки виконується мережевий fetch запит до створеного файлу з даними json, що в свою чергу повертає проміс який треба повернути до формату js коду через метод json() – результатом цього і будуть дані товарів. Отже перевизначаємо масив товарів під отримані дані. В цій функції загрузки товарів одразу варто зазначити таку властивість до кожного товару як лічильник, бо хранити цю інформації в масиві даних не є доцільно, так як на початку у всіх він рівний статичному значенню що дорівнює одиниці. Тож через цикл for надаємо до кожного товару нову властивість лічильника.

```

export default class Model {
  constructor() {
    this.products = []
  }

  async loadProducts(){
    const response = await fetch ('./js/products.json');
    const data = await response.json()
    this.products = data;

    for (const product of this.products){
      product.counter = 1;
    }
  }
}

```

Рис. 2.8. Отримання товарів в Моделі.

Після отримання товарів в Моделі слід відобразити їх на сторінці створивши в компоненті View функцію рендерингу.

Спершу прописується об'єкт з DOM елементами, тобто з структурними елементами HTML документа. Внесемо туди контейнер для товарів. Вміст компоненту Відображення, як і Моделі, експортуються.

Прописуємо функцію що приймає масив продуктів, та до кожного з яких перебираючи методом сортування створюємо константу з HTML розміткою та через інкапсуляцію підставляємо значення даних для перебираючого товару. Далі звертаємося до контейнера товарів та вказуємо команду відображення цих елементів.

```

export const elements = {
  productsContainer: document.querySelector('#products-container')
}

export function renderProducts(productsArray){
  productsArray.forEach(item => {
    const productMarkup = `<div class="col-md-6">
      <div class="card mb-4" data-id="${item.id}">
        
        <div class="card-body text-center">
          <h4 class="item-title">${item.title}</h4>
          <p><small data-items-in-box="" class="text-muted">${item.itemsInBox} шт.</small></p>

          <div class="details-wrapper">
            <div class="items counter-wrapper">
              <div class="items__control" data-action="minus">-</div>
              <div class="items__current" data-counter="">1</div>
              <div class="items__control" data-action="plus">+</div>
            </div>

            <div class="price">
              <div class="price__weight">${item.weight}г.</div>
              <div class="price__currency">${item.price} ₴</div>
            </div>
          </div>

          <button
            data-action="add-to-cart"
            type="button"
            class="btn btn-block btn-outline-warning"
          >
            + до кошика
          </button>
        </div>
      </div>
    `;
    elements.productsContainer.insertAdjacentHTML('beforeEnd', productMarkup)
  });
}

```

Рис. 2.9. Функція відображення товарів.

Далі, коли функції в Моделі та Відображенні прописані, слід їх запустити до дії через компонент Контролер. В ньому імпортуємо всі діючі компоненти та створюємо константу на основі класу Моделі. Як і в Моделі викликаємо асинхронну функцію для загрузки товарів з послідуочим рендерингом їх на сторінці.

```

import ProductsModel from './products/model.js';
import * as ProductsView from './products/view.js';

const productsModel = new ProductsModel();

async function getAndRenderProducts() {
  await productsModel.loadProducts();
  ProductsView.renderProducts(productsModel.products);
}
getAndRenderProducts();

```

Рис. 2.10. Загрузка та відображення товарів на сторінці

Після загрузки товарів слід призвести до працездатності перемикання лічильника одиниці товару. Для даної задачі буде прослуховуватися подія кліку по всьому блоку товару, де буде визначено де саме було здійснено клік користувачем для запуску функцій щодо обробки дій. Тож в Контролері, при умові взаємодії з показниками лічильника, буде знайдено саме обраний товар де було здійснено клік, на основі дата-атрибуту «id» обраного товару переведеного зі строчного типу даних в ціле число, отримано змінені дані лічильника з функції в Моделі, відповідно до обраної кнопки перемикання, та запущено функцію відображення цього лічильника на основі цих даних.

```
ProductsView.elements.productsContainer.addEventListener('click', function(e){
  const action = (e.target.dataset.action);

  // клік по лічильнику
  if (action === 'plus' || action === 'minus'){
    const productId = +e.target.closest('.card').dataset.id;

    const product = productsModel.updateCounter(productId, action);
    ProductsView.updateCounter(product)
  }
});
```

Рис. 2.11. Прослушка події кліку по товару та забезпечення перемикання лічильника

Переходячи до Моделі то функція має приймати значення id товару та обраної події перемикання. Активний товар знаходимо методом пошуку find та надалі відповідно до події значення лічильника збільшувати або зменшувати його значення. Слід зазначити що перемикання лічильника має бути з мінімальним значенням що дорівнює одиниці, то таким чином в умові на подію мінуса буде додаткова перевірка на дану умову. Та в кінці повертаємо об'єкт даного товару щоб надалі було можливим посилатися на нього.

```

updateCounter(id, action){
  const product = this.products.find((item)=>{
    if (item.id === id){
      return true
    }
  })
  if (action === 'plus'){
    product.counter++
  }
  if (action === 'minus'){
    if(product.counter > 1){
      product.counter--
    }
  }
  return product
}

```

Рис. 2.12. Функція перемиання лічильника в Моделі

Змінивши дані в Моделі слід відобразити їх на сторінці, прописавши дану функцію в компоненті Відображення, що приймає продукт. В тілі ж функції зберігає актуальні до даного товару DOM елементи перемикавання виходячи на його значення, та замінюючи на значення лічильника приймаючого продукту (counter), що було створено для кожного товару на етапі загрузки.

```

export function updateCounter(product){
  const counterWrapper = document.querySelector(`[data-id="${product.id}"]`);
  const counterElement = counterWrapper.querySelector('[data-counter]');
  counterElement.innerText = product.counter
}

```

Рис. 2.13. Функція відображення взаємодії з лічильном.

Тепер прийшов час роботи над корзиною. Першим етапом буде додавання товарів в корзину. Корзина представлена окремою сутністю, тобто вона має власні компоненти Моделі та Відображення.

Переходячи до Моделі, створюємо її клас, в конструкторі зазначаємо властивість «cart», що поки на старті дорівнює порожньому масиву.

Прописуємо метод що буде добавляти до масиву корзини об'єкти з товарами. Метод має приймати товар, у своєму тілі розуміти чи було вже додано до корзини цей самий товар, щоб не було зайвих повторень, натомість просто збільшувалася його кількість. Просто добавити до масиву корзини приймаючий товар не вийде, так як це буде пересилання одного і того ж об'єкту, тож щоб обійти цей ефект потрібно перетворити методами JSON до строчного вигляду даний продукт, а після розпарсити до javascript коду. Таким чином отримуємо нове індивідуальне посилання на товар. Для цих дій потрібно відстежити чи є вже товар в корзині, і виходячи з цього робити перевірку – якщо так, то до властивості лічильника товару в корзині додати обране значення з каталогу, якщо ж товару немає, то добавляємо його як було описано вище.

```
export default class Model {
  constructor(){
    this.cart = []
  }

  addToCart(product){
    let productInCart;
    // знаходимо товар в корзині
    productInCart = this.cart.find((item)=>{
      if (item.id == product.id){
        return true
      }
    })

    if (productInCart){ // якщо вже є цей товар у корзині то сумуємо його лічильник
      productInCart.counter = productInCart.counter + product.counter
    } else {
      // JSON.parse(JSON.stringify(product)) // копія старого об'єкта
      this.cart.push(JSON.parse(JSON.stringify(product)))
    }
    // console.log(this.cart)
  }
}
```

Рис. 2.14. Компонент Моделі корзини з методом додавання

В компоненті Моделі товарів зазначено метод збросу показника лічильника, який буде викликатися по події додавання товару до кошика, а також окремий метод отримання товару щоб не повторювати код.

```
products > JS model.js > Model > resetCounter

  getProduct(id){
    return this.products.find((item)=> item.id === id);
  }

  resetCounter(product){
    product.counter = 1;
  }
}
```

Рис. 2.15. Допоміжні методи Моделі товарів.

Переходячи ж до Контролера зазначимо логіку взаємодії. В прослушці каталогу по події кліку надамо нову умову взаємодії з кнопкою додавання до кошика, де знаходимо id обраного продукту, на основі якого далі знаходимо сам продукт, котрий у свою чергу приймається аргументом в методі додавання в Моделі. Результат цих дій буде доданий об'єкт товару в корзині, що буде відрендерений на сторінці відповідною функцією в компоненті Відображення. Скидаємо значення лічильника до початкового.

```
if (action === 'add-to-cart'){
  const productId = +e.target.closest('.card').dataset.id
  const product = productsModel.getProduct(productId);

  cartModel.addToCart(product)
  CartView.renderCart(cartModel.cart);

  productsModel.resetCounter(product)
  ProductsView.updateCounter(product)
}
```

Рис. 2.16. Логіка додавання товару до кошика.

В компоненті View зберігаємо об'єкт з DOM елементами щодо кошика.

Підготуємо функцію відображення товару в корзині, що приймає вхідний товар. В тілі функції зазначено змінну в якій має бути присвоєно

html розмітку, що на старті відповідає пустій строці. Для всіх вхідних товарів призначаємо константу розмітки, що відповідає кошику, дані розтавлено через інкапсуляцію даних товарів. Змінну-каркас перезаписуємо на отриману розмітку. Таким чином, при умові наявності вхідних товарів, контейнер по вмісту кошика буде перезаписуватися з порожньої сторчки на html-розмітку товару в кошику.

```
export const elements = {
  productsContainer: document.querySelector('#products-container'),
  cartWrapper: document.querySelector('.cart-wrapper'),
  cartEmptyBadge: document.querySelector('[data-cart-empty]'),
  orderForm: document.querySelector('#order-form'),
  totalPrice: document.querySelector('.total-price'),
}

export function renderCart(product){
  let cartHTML = '';

  product.forEach(item => {
    const cartItemHTML = `<div class="cart-item" data-id="${item.id}">
      <div class="cart-item_top">
        <div class="cart-item_img">
          
        </div>
        <div class="cart-item_desc">
          <div class="cart-item_title">${item.title}</div>
          <div class="cart-item_weight">${item.itemsInBox} / ${item.weight}</div>

          <div class="cart-item_details">
            <div class="items items--small counter-wrapper">
              <div class="items__control" data-action="minus">-</div>
              <div class="items__current" data-counter="">${item.counter}</div>
              <div class="items__control" data-action="plus">+</div>
            </div>

            <div class="price">
              <div class="price_currency">${item.price} ₪</div>
            </div>
          </div>
        </div>
      </div>
    `;

    cartHTML += cartItemHTML;
  });

  elements.cartWrapper.innerHTML = cartHTML;
}
```

Рис. 2.17. Логіка додавання товару до кошика.

Слід врахувати момент коли товари в кошику будуть прибрані за бажанням користувача. Для цього нюансу прописано наступну функцію що перевіряє умову присутності об'єкту товару в кошику. Відповідно від результатів перевірки інформативні блоки будуть зникати чи з'являтися

завдяки вказівкам з CSS. Дану функцію викликаємо у функції рендерингу товарів корзини.

```
toggleCart();  
}  
  
function toggleCart(){  
  if (elements.cartWrapper.children.length > 0){  
    console.log('full')  
    elements.cartEmptyBadge.classList.remove('none');  
    elements.orderForm.classList.add('none');  
  } else {  
    console.log('empty')  
    elements.cartEmptyBadge.classList.add('none');  
    elements.orderForm.classList.remove('none');  
  }  
}
```

Рис. 2.18. Перемикання контентного блоку корзини.

Для отримання загальної ціни що змінюється в залежності від кількості та вартості товарів у кошику зазначимо наступний метод в Моделі. На старті дана величина буде дорівнюватиме нулю, так як користувачем товар ще не обрано, але після того як перепройдовши увесь масив товарів корзини, її значення буде перезаписано на своє попереднє значення та суму добутку ціни та кількості товару. Отримане значення має повертатися для подальшої передачі щодо виведення його на сторінку.

```
getTotalPrice(){  
  let totalPrice = 0;  
  this.cart.forEach((item)=>{  
    totalPrice = totalPrice + item.price * item.counter  
  })  
  return totalPrice;  
}
```

Рис. 2.19. Перерахування загальної ціни в Моделі.

Для відображення загальної ціни на сторінці напишемо функцію у View, що приймає значення з Моделі, та у своєму тілі змінює текстове значення відповідного елементу розмітки на приймаюче значення у відформатованому форматі.

```
export function updateTotalPrice(price){  
  elements.totalPrice.innerText = new Intl.NumberFormat().format(price)  
}
```

Рис. 2.20. Функція відображення загальної ціни обраних товарів.

В Контролері, в тілі асинхронної функції отримання та рендарингу товарів, викликаємо цю функцію аргументами якої буде отриманий результат з методу в Моделі.

Також важливим інтерактивним елементом є перемикання лічильника в корзині товарів. Для цього, по стратегії MVC, створюється метод Моделі що схожий з тим який було прописано для товару в каталозі, який приймає id для відстеження поточного товару, та дію – тобто збільшення або зменшення, в залежності від якої значення лічильника буде пропорційно змінюватися. Його допустимим значенням має бути одиниця, але при умові рівності до нуля даний товар має зникати з кошика. По закінченню перевірок даний метод має повертати змінений товар.

Для точного представлення дій розпишемо логіку в Контролері. Через DOM елементи знаходимо обгортку до контентного змісту товарів в якій прослуховуємо подію кліка. По відстеженню дата-атрибутів перимикачів лічильника знаходимо актуальний товар що приймається першим аргументом в метод Моделі, другим з яких ця подія що викликає зміни. Функція відображення лічильника має запускатися при умові його значення в Моделі в інтервалі додатніх чисел. В інакшому ж випадку має викликатися функція видалення товару. При події видалення товару з корзини має

змінюватися обгортка кошика до початкового стану. Для цього в тіло функції видалення корзини додається функція переключання стану корзини.

```

CartView.elements.cartWrapper.addEventListener('click', function(e){
  const action = e.target.dataset.action;

  if (action === 'plus' || action === 'minus'){
    const productId = +e.target.closest('.cart-item').dataset.id

    const productInCart = cartModel.updateCounterInCart(productId, action);

    if (productInCart.counter > 0){
      CartView.updateCounter(productInCart)
    } else {
      CartView.removeItemFromCart(productInCart)
    }
  }
})

```

Рис. 2.21. Логіка перемикавання лічильника товару в корзині.

В компоненті Відображення зазначаємо функцію зміни значення лічильника що є аналогічною до товару в каталозі, та функцію видалення товару з корзини. Функція приймає продукт на основі якого знахотить його контентну обгортку та видаляє.

```

export function updateCounter(product){
  const productWrapper = elements.cartWrapper.querySelector(`[data-id="${product.id}"]`);
  const counterElement = productWrapper.querySelector('[data-counter]');
  counterElement.innerText = product.counter;
}

export function removeItemFromCart(product){
  const productWrapper = elements.cartWrapper.querySelector(`[data-id="${product.id}"]`);
  productWrapper.remove();
}

```

Рис. 2.22. Функції відображення взаємодії з лічильником.

При зміні кількості товару в корзині має і змінюватися його загальна ціна. Для цього в умові взаємодії з лічильником отримуємо дані загальної

ціни через метод з Моделі та підставляємо їх в аргумент функції відображення загальної ціни.

Завершальним етапом фронтенду буде задача збереження прогресу користувача на сайті. Щоб коли він випадково закрив вкладку вибору товару у своєму браузері, або просто переходячи зі сторінки на сторінку, його дії збережувалися на сайті. У випадку інтернет-магазину це будуть маніпуляції з товарами, а саме взаємодія з кошиком замовлення. Для цього технічного завдання можна використати середовище локальних даних LocalStorage.

LocalStorage є об'єктом веб-сховища що дозволяє зберігати дані в браузері користувача у вигляді пар ключ – значення, які зберігаються при повному закритті вікна браузера.

Тож в компоненті Моделі корзини необхідно додати два нових методи. Перший має безпосередньо зберігати дані під відповідним ключем у локальне середовище у JSON форматі та викликатися під час будь якої зміни стану товару. Другий метод має отримувати ці збережені дані по ключу та підгружати їх на сторінку. При умові якщо дані є, то корзина, що на старті рівна порожньому масиву, має перезаписатися на отримані дані. Виклик даного метода має бути здійснено з конструктора класу Моделі, в межах видимості самої корзини.

```

constructor(){
  this.cart = [];
  this.loadCartFromLocalStorage();
}
loadCartFromLocalStorage(){
  const data = localStorage.getItem('cart');
  if (data){
    this.cart = JSON.parse(data)
  }
}
saveToLocalStorage(){
  localStorage.setItem('cart', JSON.stringify(this.cart))
}

```

Рис. 2.23. Збереження та загрузка корзини в Моделі

Аби відобразити збережені товари при оновленні сторінки, дані кошика варто передати функції рендерингу товарів кошику, що у свою чергу зазначено у головній функції Контролера. Також в ній запускаємо роботу плагіну `mixitub`[9], прописавши класи-тригери до назв категорій, щоб товари ефектно сортувалися в залежності від своєї категорії.

По закінченню роботи над безпосереднім робочим кодом потрібно також зробити ретельну перевірку, вдосконалившись що всі функції коректно працюють. Так само як і в адаптивному дизайні, для більш швидкого процесу перевірки та усуненні несправностей, слід скористатися панеллю у браузері – «інструменти розробника». Надалі даний проект слід додати у віддалене середовище для зберігання та можливості демонстрації. Для цієї задачі було обрано вебсервіс GitHub[4].

Таким чином, обравши сервіс, створюємо віддалений git-репозиторій у котрий потрібно подбавляти усі необхідні файли та документи для коректного відображення проекту[17] - <https://igortan14.github.io/roll-delivery/>.

РОЗДІЛ 3. АДАПТАЦІЯ ДО CMS СИСТЕМИ

Зробивши основну роботу над інтернет-магазином, у даному розділі буде переноситися вся інформація по ньому на обрану CMS систему та налаштовуватися власна панель адміністратора для передачі потенційному контент-менеджеру.

Першим етапом буде встановлення хатрр[42], що буде спершу виступати локальним сервером. В папці htdocs створюється папка каталогу вебсайту. Після чого скачується WordPress у вигляді архіву документів, розпакувавши котрий, необхідно перенести усі дані до створеної папки. Перевіряємо в налаштуваннях активність останніх версій MySQL та PHP.

Створюємо базу даних в режимі admin, де зазначаємо її назву та формат кодировки. Після чого, налаштувавши WordPress[26], можна відкрити адмін-панель сайту. Таким чином, процес адаптації готового вебсайту до CMS системи WordPress, можна назвати створення своєї власної теми.

Структура сайту та скрипти дещо будуть змінені та покращені, на приціл того, що тематика може змінитися та бути більш широкою у застосуванні. Так наприклад, в каталозі буде три слоти товару на великому екрані, під загальним описом товару з'явиться його перемикаючі параметри.

Тож переходячи до папки тем, переносимо туди проект зі статичною версткою. Файл стилів переноситься в корень даної теми за правилами WordPress, де прописується в коментарях назву теми. Головному файлу «index.html» змінюємо розширення на «php». Також робиться скріншот до теми, що буде виступати її прев'ю, розмірами 1200 на 900 пікселів, та розтошовується у корені. Усі додаткові файли до проекту переносяться в єдину папку «assets». Для їх підключення створюється файл «functions.php», який по більшій частині буде центром управління по процесу адаптації та налаштувань. У ньому застосуємо хуки, тобто прив'язки до певної події і

після виклику якої виконується зазначена функція. Таким чином, при загрузці WordPress, буде запущено функцію в тілі якої йде підключення усіх ресурсів та файлів та змінну назву версії що підставляється до кожного ресурсу, так як є однаковим параметром. Функція вордпрес «wp_enqueue_style» для стилів, що в аргументах має назву, src-шлях, залежність від файлів (якщо не має порожній масив) та версію. Для скриптів аналогічна функція але з приставкою «scripts» та з додатковим аргументом true, аби вони підставилось у футер, згідно семантики.

Далі робиться розбивання файлів, де вміст футера та хедера у окремих файлах, для підключення яких у головному файлі прописується php функція отримання, а у самих файлів wp-функцію тег шаблона, що перед закриваючим головним тегом, аби можна було прив'язуватись до них за допомогою хуків.

Перейшовши до етапу створення інформативних полів, необхідно застосувати плагін «Carbon Fields»[38], котрий дозволяє робити додаткові налаштування сайту в панелі адміністратора. Для цього завантажуюмо його у ручному режимі та у окрему папку, поруч з якою буде папка з його налаштуваннями. У файлі функцій зазначається хук «після установки теми», де визиваємо функцію котра підключає плагін та ініціалізує його до проекту. У файлі налаштувань зазначаємо його роботу з перевіркою, що якщо звертання до нього йде напрямку то має здійснитися вихід, також підключивши його у функціях. Зазначаємо поступово налаштування сайту, зробивши таби у вигляді ключ-значення, де ключ це назва з типом інформації, а значення масив з полями цих даних. У файлі функцій зазначається хук «init», тобто буде першочерговий пріоритет виконання функції що виводить поля. Для спрощення написання зазначається глобальний масив, елементами якого є назви полів що привласнюють введену інформації щодо них через файл налаштувань (Додаток А. Рис. 1.7).

Так як більшість сучасних логотипів у форматі svg, необхідно ще встановити плагін «SVG Support», що робиться з панелі управління

WordPress. Надалі, скрізь де зазначено інформаційні дані у вигляді тексту або картинок, їх треба перезаписати на функції отримання в аргументі яких ідентифікатор даного поля інформації, у випадку ж логотипу така функція виведе текстовий ідентифікатор зображення, тому вона має бути аргументом іншої функції що повертає її url. Далі зазначаємо таб контактів з такою інформацією як: телефон – у звичайному режимі та машинному (тобто без пробілів та плюсу), адресу, координати карти та соціальну мережу, та заповнюємо створені поля потрібною інформацією, перезаписуючи минулий статичний варіант в документі за допомогою коду, як було зазначено вище. Таким чином створюються логічні секції контенту, що здібні до дуже зручного редагування вмістом. Основними з яких, котрі можуть змінюватись, ще є «Перший екран», «Підвал», «Каталог», «Про нас» (Додаток А. Рис. 1.8). В секції «Про нас» в створюючому полі тексту, так як він є блогового формату, замість атрибуту «text» прописується «rich_text», аби було можливо зручно його змінювати в лажежності від стилю та формату.

Переходячи до ключової секції з товарами каталогу, необхідно застосувати функцію «register_post_types», яка налаштовується завдяки своїм методам відповідно документації. За замовчуванням додавання зображення в ній не підтримується, тож де йде хук «after_setup_theme» треба це виправити, дописавши відповідну команду «add_theme_support('post-thumbnails');».

В налаштуваннях «Carbon-fields» створюється новий контейнер з назвою «Додаткові поля» який буде відображати свої опції для товарів функцією «show_on_post_type('product')». Далі таб з інформацією товару, в якому у полях йде ціна та інші атрибути, де треба застосувати комплексний підхід функцією, що створює комплексне поле аби підлаштувати параметри для кожного товару. В панелі WordPress з'явилася вкладка «товари», перейшови в яку обираємо «категорії», та заповнюємо її. Для наявного прикладу створимо нову категорію «Італійська кухня» окрім існуючої японської, з підкатегорією «піца» та «паста». Повертаючись до файлу налаштувань, створюється таб каталогу та застосовується

асоціативний підхід, на виході чого бачимо додані товари в панелі WordPress. Тепер треба зробити можливим їх відображення на сторінці, для цього у головному файлі видаляємо усю розмітку з товарами окрім одної, що буде взята за основу, та переноситься у окремий файл, в якому проводиться підстановка даних. В основному ж документі виводимо отримані товари циклом в тіло розмітки блоку каталогу.

```
<?php
    $catalog_products = carbon_get_theme_option( $page_id, 'catalog_products' );
    $catalog_products_ids = wp_list_pluck($catalog_products, 'id');

    $catalog_products_args = [
        'post_type' => 'product',
        'post__in' => $catalog_products_ids
    ];
    $catalog_products_query = new WP_Query( $catalog_products_args );
?>

<?php if ( $catalog_products_query->have_posts() ) : ?>
    <div class="catalog">

        <?php while ( $catalog_products_query->have_posts() ) : $catalog_products_query->the_post(); ?>
            <?php echo get_template_part('product-content')?>
        <?php endwhile; ?>

        <?php wp_reset_postdata( ); ?>
    </div>

    <?php endif; ?>
```

Рис. 3.1. Процес отримання відфільтрованих товарів за ідентифікатором та їх виведення циклом в блоці каталогу при умові їх наявності.

В окремому файлі елементу товару проводиться підстановка інформації зі статичних, на ту що була вказана в панелі вордпрес у табі каталогу та класу активності для поточного атрибуту товару з відповідними перевірками на наявність.

```

<?php
$product_id = get_the_ID();

$product_price = carbon_get_post_meta($product_id, 'product_price');
$product_attributes = carbon_get_post_meta($product_id, 'product_attributes');
$product_img_scr = get_the_post_thumbnail_url($product_id, 'thumbnail');
?>
<div class="catalog_item" data-category="roles">
    <div class="product catalog_product">
        <picture>
            
        </picture>
        <div class="product_content">
            <h3 class="product_title"><?php the_title();?></h3>
            <div class="product_description">
                <?php the_excerpt(); ?>
            </div>
        </div>
        <div class="product_footer">
            <?php if ($product_attributes) : ?>
                <div class="product_sizes">
                    <?php $is_first_item = true ?>
                    <?php foreach ($product_attributes as $attribute) : ?>
                        <?php $attribute_active_class = $is_first_item ? ' is-active' : '' ?>
                        <button data-product-size-price="<?php echo $attribute['price']; ?>"
                            class="product_size<?php echo $attribute_active_class; ?>" type="button"><?php echo $attribute['name'] ?></button>
                        <?php $is_first_item = false;
                    </foreach> ?>
                </div>
            <?php endif; ?>
            <div class="product_bottom">
                <div class="product_price">
                    <span class="product_price-value"><?php echo $product_price ?></span>
                    <span class="product_currency">₴</span>
                </div>
                <button class="btn product_btn" type="button" data-popup="popup-order">заказати</button>
            </div>
        </div>
    </div>
</div>

```

Рис. 3.2. Підготовчий процес щодо відображення усіх товарів.

Таким чином увесь зміст сайту інтернет-магазину було повністю оновлено та адаптовано під будь який контент адміністратора WordPress даного сайту.

ВИСНОВКИ

Отже, кваліфікаційну роботу на тему "Створення вебсайту інтернет-магазину та адаптування його під CMS систему" було розбито на логічні етапи та поступово успішно реалізовано. На виході було створено інтернет-магазин доставки японської кухні. Проєкт включав в себе розробку інтуїтивно зрозумілого інтерфейсу, де користувачі можуть з легкістю переглядати актуальне меню та обирати своє замовлення (Додаток А. Рис. 1.1-1.3).

В першому розділі було досліджено предметну область. Аналізуючи історію та сучасний стан ресторанного бізнесу в Україні, стає очевидним, що створення вебсайту доставки їжі є актуальним кроком для розвитку цього бізнесу. Ресторанний сектор в Україні попри все переживає розквіт, зростання конкуренції та зміну попиту клієнтів. Завдяки вебсайту, каже чи ресторан може ефективно привертати нових клієнтів, поліпшувати обслуговування та підтримувати зв'язок зі своїми поточними клієнтами.

Особливості бізнес-процесів ресторанного бізнесу, такі як замовлення їжі онлайн, оптимізація доставки та самовивозу. Це дає ресторану зручність, ефективність та конкурентну перевагу на ринку.

Аналіз обраних інструментів розробки додатку показав, що створення вебсайту доставки ролів можливе з використанням сучасних технологій та платформ, таких як HTML, CSS, JavaScript. Ці інструменти дозволяють створити привабливий, функціональний та зручний вебсайт, який задовольнить потреби як клієнтів, так і власника сайту.

В другому розділі було проаналізовано етапи безпосереднього створення вебсайту, тобто спроектовано дизайн, зверстано необхідну архітектуру з адаптивною стилізацією та розроблено фронтендтну складову вебсайту.

Етапи створення вебсайту ресторану, такі як розробка архітектури, створення дизайну та програмування, є ключовими для створення

функціонального та привабливого вебсайту. Кожен з цих етапів вимагає уважного планування та виконання з метою створення зручного та ефективного інтерфейсу для користувачів.

В останньому розділі було проаналізовано та здійснено процес перенесення вебсайту на рушій системи WordPress, дозволивши у дуже зручному та зрозумілому форматі редагувати усю необхідну інформацію на вебсайті на власне бажання власника, аж до повної зміни тематики ресторану, чи в принципі діяльності.

Аналізуючи стан та перспективи електронного бізнесу в Україні, можна зробити висновок, що наявність швидко контролюючого сайту постачаємих послуг, значно впливає на збільшення конверсії діяльності підприємця.

Тож результати даної роботи на перспективі сприяють розвитку економічної ситуації, завдяки електронній комерції підприємницької діяльності.

В результаті виконання кваліфікаційної роботи було отримано та засвоєно ключові навички для створення вебсайту та здійснено два основних підхода щодо розробки. Таким чином, отриманий досвід дозволить бути більш професійним в подальшій професійній діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адаптивна верстка сайту: основи та принципи. URL: <https://atriples.com.ua/adaptivna-verstka-saytu-osnovi-ta-printsipi/> (Дата звернення: 09.10.2024).
2. Вільямс Б., Стерн Х., Демстра Д. WordPress для професіоналів. Розробка і дизайн сайтів. 2015. 412 с.
3. Веб-програмування. jQuery у веб-додатках. URL: <https://webstudio2u.net/ua/programming/120-jquery.html> (Дата звернення: 26.10.2024).
4. Веб-сервіс GitHub. URL: <https://github.com/> (Дата звернення: 30.10.2024).
5. Гугл шрифти. URL: <https://fonts.google.com> (Дата звернення: 15.09.2024).
6. Дослідження агентства W3Techs щодо популярності CRM систем. URL: https://w3techs.com/technologies/overview/content_management (Дата звернення: 01.08.2024).
7. Дослідження Soul Partners щодо електронної комерції в Україні. URL: <https://soulpartners.com.ua/news/tpost/x2dve03v71-rinok-elektronno-komerts-v-ukran-dosyagn> (Дата звернення: 16. 10.2024).
8. Єгорова І.М. Проектування та розробка Web-документів: навч. Посібник. Харків: ХНУРЕ, 2018. 264 с.
9. Кознова Олександра. Як відновлюється та адаптується український бізнес: результати дослідження | Think brave. *Think brave* | *Останні новини бізнесу України*. URL: https://biz.ligazakon.net/news/214925_yak-vdnovlyutsya-ta-adaptutsya-ukranskiy-bznes-rezultati-doslidzhennya
10. Конвектор шрифтів. URL: <https://anyconv.com/ru/konverter-ttf-v-woff2/> (Дата звернення: 08.10.2024).

11. Маріков Ф. Самовчитель PHP 5. 2013. 420 с.
12. Опитування Deloitte. URL: https://www2.deloitte.com/content/dam/Deloitte/ua/Documents/Press-release/Consumer-sentiment-of-the-Ukrainians-over-the-pre-war-period_UA.pdf (Дата звернення: 16.10.2024).
13. Основні етапи розробки вебсайту. Ігор Кобилянський. URL: <https://impulse-design.com.ua/ua/etapy-razrobotki-sajta.html> (Дата звернення: 01.08.2024).
14. Правильний шлях: PHP. URL: <https://php.org.ua/php-the-right-way> (Дата звернення: 04.10.2024).
15. Петро Ц. Асинхронне програмування у JavaScript. URL: <https://buki.com.ua/blogs/asinxronne-programuvannia-u-javascript/>
16. Підручник для розробників GitHub. URL: <https://uk.myservername.com/github-tutorial-developers-how-use-github> (Дата звернення: 16.11.2024)
17. Посилання на проект (звичайний підхід). URL: <https://igortan14.github.io/roll-delivery/> (Дата звернення: 16.11.2024).
18. Семантична верстка: детальний гайд і важливість для SEO. URL: <https://netpeak.net/uk/blog/semantichna-verstka-detel-niy-gayd-i-vazhlivist-dlya-seo/> (Дата звернення: 15.09.2024).
19. Трофименко О. Г. Веб-дизайн та HTML-програмування / О. Г. Трофименко. – Одеса: Фенікс, 2018. – 194 с.
20. Що таке WordPress. URL: <https://interkassa.com/blog/oglyad-cms-wordpress-plyusi-minusi-prikladi> (Дата звернення: 11.10.2024).
21. Що це таке CMS і як працює, види та приклади. URL: <https://hostiq.ua/wiki/ukr/cms/> (Дата звернення: 16.10.2024).
22. An Introduction to JavaScript. *The Modern JavaScript Tutorial*. URL: <https://javascript.info/intro>

23. Bootstrap - The most popular HTML, CSS, and JS library in the world. URL: <https://getbootstrap.com/> (Дата звернення: 03.08.2024).
24. Can I use. URL: <https://caniuse.com/> (Дата звернення: 16.11.2024).
25. Chawre H. Understanding web development process: full guide in 6 steps. *Hire the World's Most Deeply Vetted Remote Developers | Turing*. URL: <https://www.turing.com/resources/web-development-process> (Дата звернення: 17.08.2024).
26. CMS Wordpress. URL: <https://wordpress.com/> (Дата звернення: 01.11.2024).
27. Daber J. Html, Css, Website development and design. Exemo, 2022. 480 p.
28. Documentation on updating and downloading new versions of themes and plugins and general interaction in WordPress. URL: <https://wp-kama.com/> (Дата звернення: 19.10.2024).
29. Duckett, J. (2014). HTML & CSS: Design and Build Websites. John Wiley.
30. Elliott, J., & Freeman, E. (2020). Head First JavaScript: A Brain-Friendly Guide (2nd ed.). O'Reilly Media.
31. Flanagan, D. (2018). JavaScript: The Definitive Guide (7th ed.). O'Reilly Media.
32. Freeman, A., Robson, E., & Bates, B. (2020). Head First JavaScript Programming: A Brain-Friendly Guide (2nd ed.). O'Reilly Media.
33. JQuery documentation. URL: <https://jquery.com/> (Дата звернення: 10.09.2024).
34. MixItUp - A CSS3 and jQuery Filter & Sort Plugin. URL: <https://cdnjs.com/libraries/mixitup> (Дата звернення: 16.11.2024).
35. MVC - MDN Web Docs Glossary. URL: <https://developer.mozilla.org/en-US/docs/Glossary/MVC> (Дата звернення: 07.10.2024).

36. MySQL documentation. URL: <https://dev.mysql.com/doc/> (Дата звернення: 01.11.2024).
37. Nixon R. Learning PHP, MySQL & JavaScript with jQuery, CSS & HTML5. URL: <https://tinyurl.com/3kj8s97h> (Дата звернення: 07.10.2024).
38. WordPress Custom Fields Library. URL: <https://carbonfields.net/> (Дата звернення: 01.11.2024).
39. What is CSS, how does it work and what is it used for? - skillcrush. *Skillcrush*. URL: <https://skillcrush.com/blog/css/> (Дата звернення: 14.08.2024).
40. What is PHP? How PHP is Used in WordPress? URL: <https://www.wpbeginner.com/glossary/php/> (Дата звернення: 04.11.2024).
41. What is web design? The ultimate guide to website design [2020] - pagecloud blog - web design & small business tips for your website. *Pagecloud.com*.
URL: <https://www.pagecloud.com/blog/web-design-guide>
42. XAMPP is the most popular PHP development environment. URL: <https://www.apachefriends.org/index.html#> (Дата звернення: 15.10.2024).

ДОДАТКИ

Додаток А

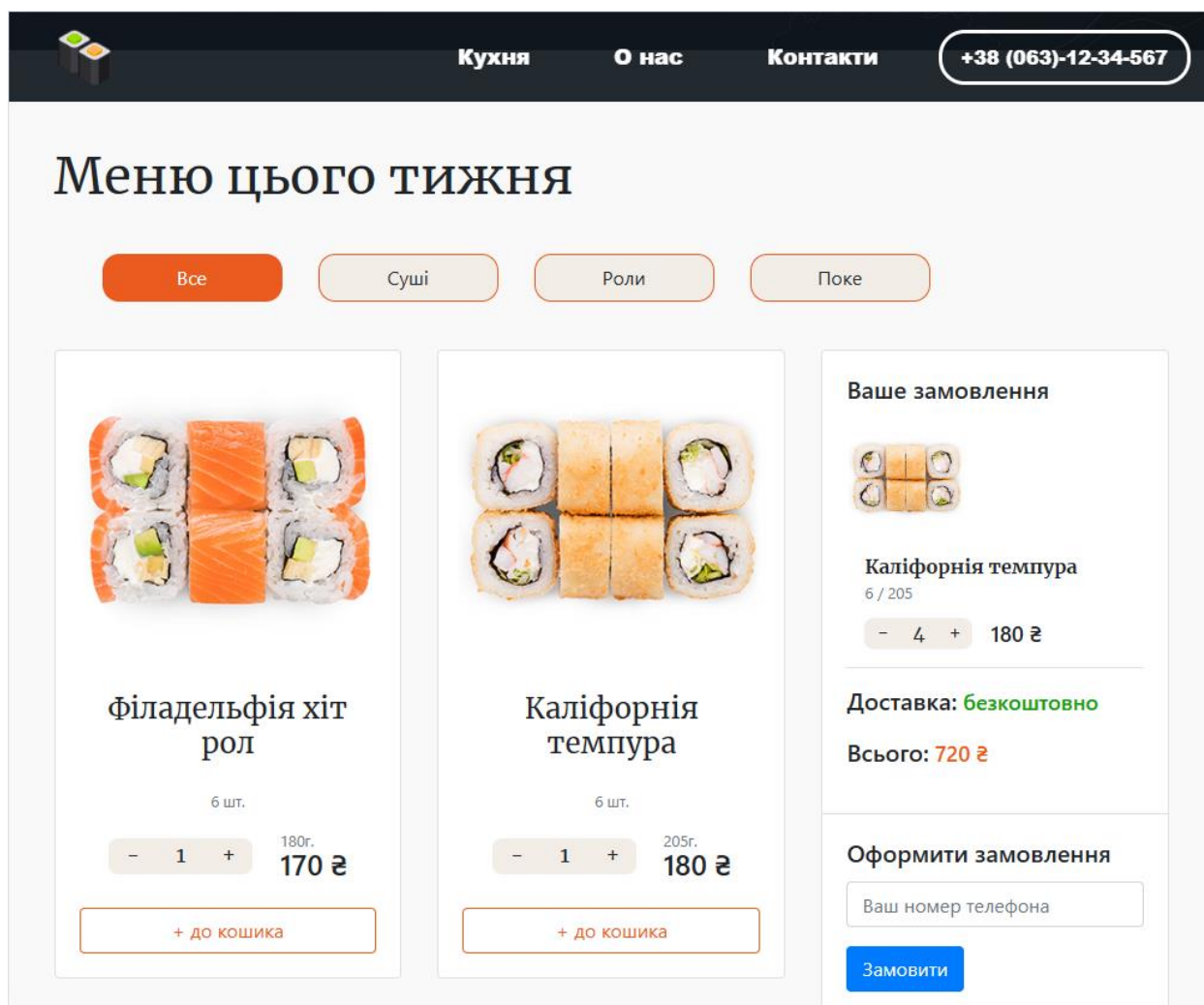


Рис. 1.1. Каталог товарів з корзиною – реалізація JavaScript патерном MVC.

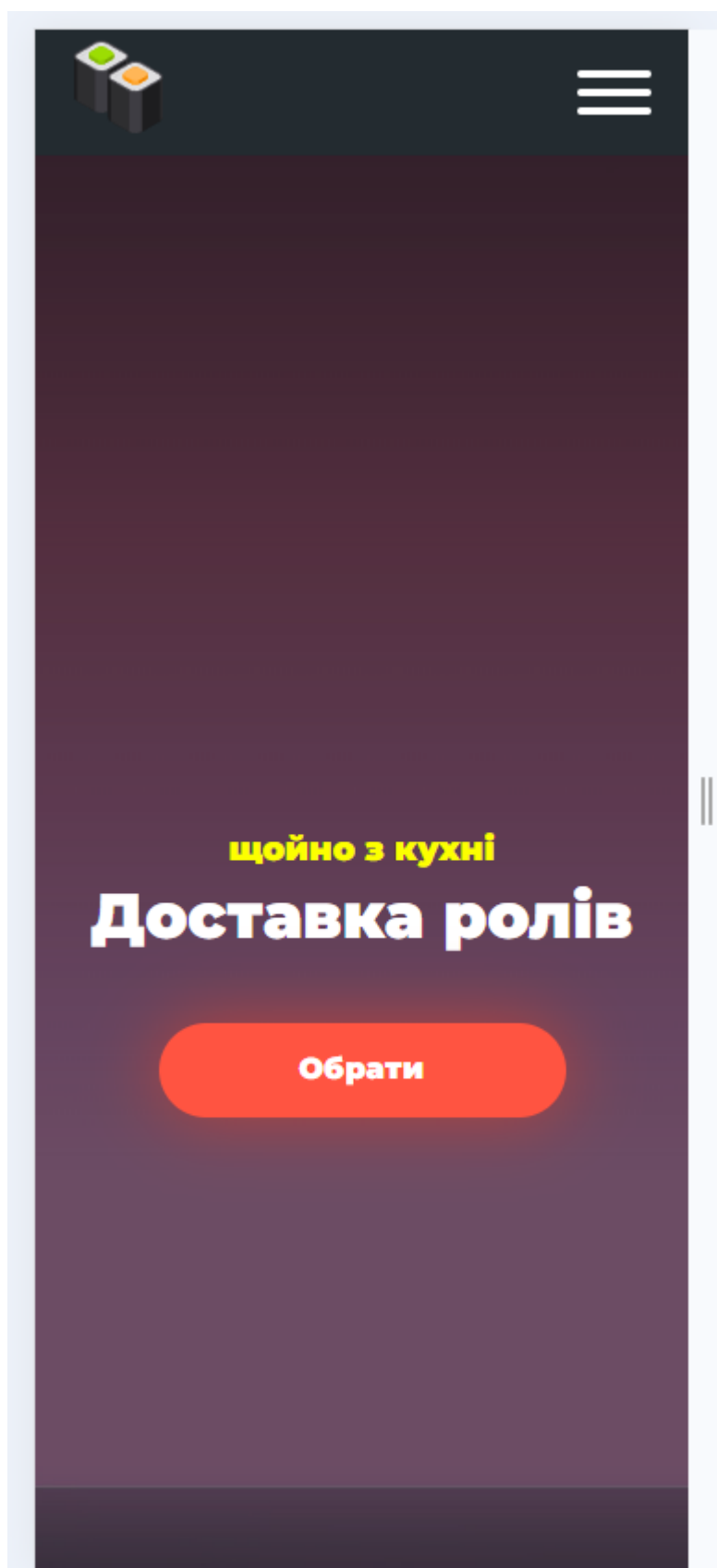


Рис. 1.2. Приклад відображення на мобільному пристрою.

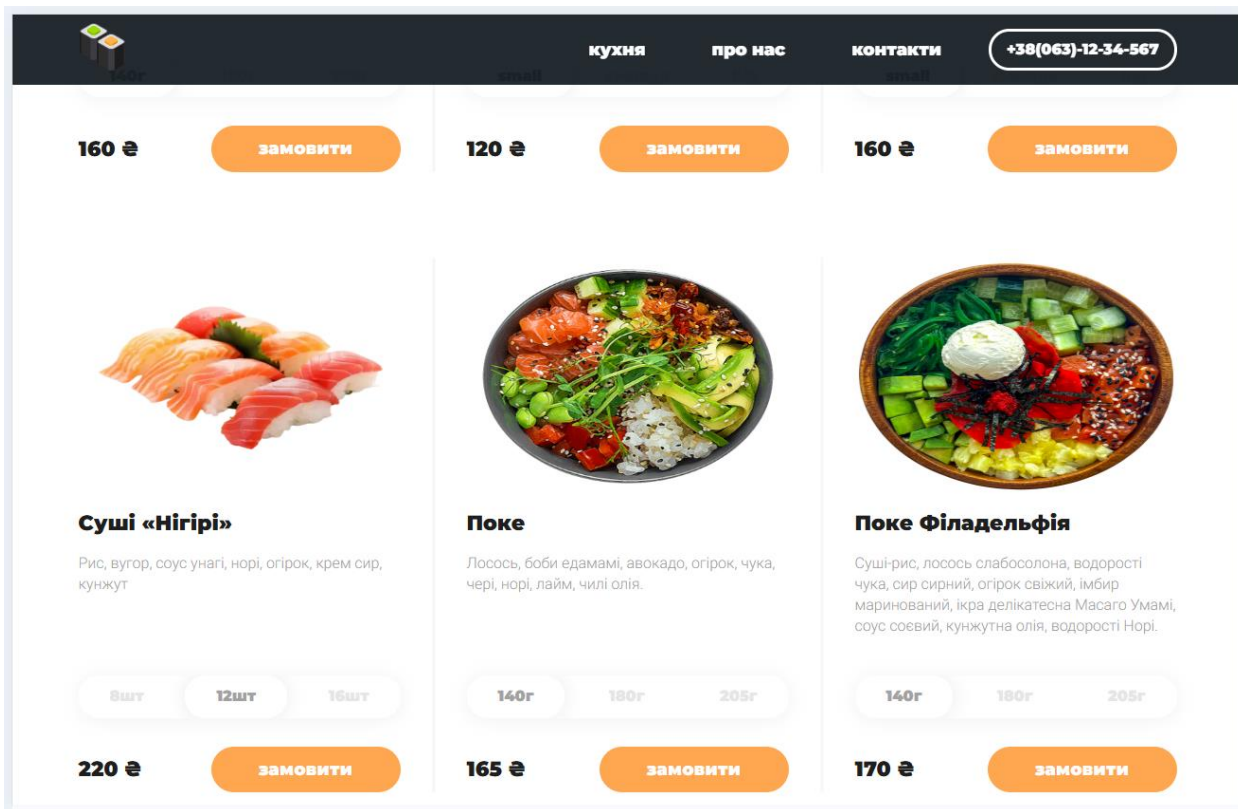


Рис. 1.3. Змінений сайт під систему WordPress з генеруючим контентом.

The screenshot shows the 'Контакти' (Contacts) settings page in a WordPress CMS. The page includes a sidebar with navigation links (Загальні налаштування, Контакти, Перший екран, Про нас, Підвал, Каталог) and a main content area with fields for contact information.

Назва	Значення
Телефон	+38(063)-12-34-567
Телефон без пробілів у форматі +380631234567	380631234567
Адреса	місто Буча, вулиця Нове шосе 120
Координати карти	https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d10140.958689859148!2d30.20408420005415!3d50.548312
Instagram	your_instagram_url

Рис. 1.4. Приклад управління контентом до адаптованого сайту під CMS систему WordPress.

Заповніть форму

Опертивно доставимо за 45 хвилин

Філадельфія хіт рол 135 ₪
6шт

Ім'я

Телефон

Адреса

Оплата готівкою

Відправити

Рис. 1.5. Форма замовлення.

Каталог

Заголовок

Меню цього тижня

Категорії товарів

Search... Showing 7 of 7 results

Категорія	Ім'я	Дія
Італійська кухня	КАТЕГОРІЯ ТОВАРА	✎ +
Паста	КАТЕГОРІЯ ТОВАРА	✎ +
Піца	КАТЕГОРІЯ ТОВАРА	✎ +
Поке	КАТЕГОРІЯ ТОВАРА	✎ +
Роли	КАТЕГОРІЯ ТОВАРА	✎ +

Товари

Search... Showing 10 of 10 results

Ім'я	Категорія	Дія
Запечений рол «Каліфорнія»	ТОВАР	✎ +
Каліфорнія темпура	ТОВАР	✎ +
Паста з помідорами	ТОВАР	✎ +
Піца «Маргарита»	ТОВАР	✎ +
Піца Сицилійська	ТОВАР	✎ +
Запечений рол «Каліфорнія»	POST	✕
Каліфорнія темпура	POST	✕
Поке	POST	✕
Поке Філадельфія	POST	✕
Суші «Нігірі»	POST	✕

Рис. 1.6. Головна панель управління товарами.

```

add_action('init', 'create_global_variable');
function create_global_variable() {
    global $roll_delivery;
    $roll_delivery = [
        'phone' => carbon_get_theme_option('site_phone'),
        'phone_digits' => carbon_get_theme_option( 'site_phone_digits' ),
        'address' => carbon_get_theme_option( 'site_address' ),
        'map_coordinates' => carbon_get_theme_option( 'site_map_coordinates' ),
        'inst_url' => carbon_get_theme_option( 'site_inst_url' ),
        'subtitle' => carbon_get_theme_option( 'top_info' ), '
        title' => carbon_get_theme_option( 'top_title' ),
        'top_bg' => carbon_get_theme_option( 'top_img' ),
        'about_text' => carbon_get_theme_option( 'about_text' ),
        'about_img' => carbon_get_theme_option( 'about_img' ),
        'footer_text' => carbon_get_theme_option( 'footer_text' ),
        'menu_title' => carbon_get_theme_option( 'catalog_title' ),
    ]; }

```

// Приклад ініціалізації контентних полів.

```

<?php if (!defined('ABSPATH')) {
    exit; }
use Carbon_Fields\Container;
use Carbon_Fields\Field;

Container::make( 'theme_options', 'Налаштування сайта' )
->add_tab( 'Загальні налаштування', [
    Field::make( 'image', 'site_logo', 'Логотип' ),
])
->add_tab( 'Контакти', [
    Field::make( 'text', 'site_phone', 'Телефон' ),

```

```
Field::make( 'text', 'site_phone_digits', 'Телефон без пробілів у форматі  
+380631234567' ),  
Field::make( 'text', 'site_address', 'Адреса' ),  
Field::make( 'text', 'site_map_coordinates', 'Координати карти' ),  
Field::make( 'text', 'site_inst_url', 'Instagram' ),  
// Field::make( 'text', 'site_fb_url', 'Facebook' ),  
])  
?>  
  
// Приклад створення табів та панелі управління контентом.
```