МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МЕЛЬНИЧУК КИРИЛ ВОЛОДИМИРОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ О.В.Зелінська
«__» _____2024р.

РОЗРОБКА ВЕБКАТАЛОГУ НА ОСНОВІ SVELTE

Спеціальність 122 «Комп’ютерні науки»

Кваліфікаційна (бакалаврська) робота

Керівник:
О. В Зелінська , доцент кафедри
інформаційних технологій,
к. т. н., доцент
_____

Оцінка: _____/ _____/ _____
(бали за шкалою ЄКТС / за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2024

АНОТАЦІЯ

Мельничук К. В. Розробка вебкаталогу на основі Svelte. Спеціальність 122 «Комп’ютерні науки», освітня програма «Сучасні інформаційні технології та програмування». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У рамках бакалаврської роботи було створено вебкаталог, ціллю створення є навчання веброзробці та набуття практичного досвіду. Для розробки зберігання та контролю версій коду були використані: HTML, CSS, SCSS, Bootstrap, JavaScript, Svelte, Git, GitHub, Netlify, Node.Js

Ключові слова: вебкаталог, вебсайт , SCSS, JavaScript, Svelte, HTML, CSS.

Сторінок: 53. Рисунків: 43. Додатків: 1. Джерел літ.: 40

ABSTRACT

Melnychuk K. V. Development of a web catalogue based on Svelte. Speciality 122 "Computer Science", educational programme "Modern Information Technologies and Programming". Vasyl' Stus Donetsk National University, Vinnytsia, 2024.

As part of the bachelor's thesis, a web catalogue was created, the purpose of which is to learn web development and gain practical experience. For the development, code storage and version control were used: HTML, CSS, SCSS, Bootstrap, JavaScript, Svelte, Git, GitHub, Netlify, Node.Js.

Keywords: web catalogue, website, SCSS, JavaScript, Svelte, HTML, CSS.

Pages: 53. Figures: 43. Appendices: 1. References: 40.

ЗМІСТ

ВСТУП

У сучасному світі інтернет відіграє важливу роль, що дає змогу створювати та публікувати різноманітний контент у мережі. Використання вебсайтів є звичайною справою для багатьох у повсякденному житті. Люди послуговуються ними як для пошуку інформації та роботи, так і для замовлень товарів. Розробка навіть звичайного вебкаталогу вимагає ретельного планування, враховуючи фактори зручності використання, продуктивності та масштабованості.

Актуальність теми дослідження. Розробка такого ресурсу, присвяченого каталогу різних видів піц, їхніх цін, розмірів і складу, з можливістю фільтрації, сортування, додавання товарів в кошик, видаляти їх і оформляти замовлення, є актуальною задачею, оскільки це може значно полегшити процес замовлення піци та забезпечити зручний користувацький досвід і виконати вимоги бізнесу що замовляє подібні проєкти для себе.

Вебкаталоги - це сайти або платформи які містять список продуктів або послуг, доступних для покупки або перегляду. Вони широко використовуються для інтернет-торгівлі, а також для інформаційного представлення товарів і послуг. Вони забезпечують зручний спосіб для користувачів здійснювати пошук і порівняння продуктів та можуть включати різноманітні функції, такі як фільтри пошуку, рейтинги товарів, відгуки користувачів та інші.

Метою цієї дипломної роботи є розробка нескладного вебкаталогу з використанням фреймворку Svelte, що поєднує в собі зручний користувацький інтерфейс, та високу продуктивність. Вивчення та вдосконалення навичок програмування мовою JavaScript та використання різних бібліотек. Створення продукту, що може бути корисним як для користувача, так і для бізнесу [16][17]

Задачами дослідження є:

- Вивчення принципів роботи та особливостей фреймворку Svelte [17].
- Проєктування архітектури вебкаталогу та його функціональності

- Розробка користувацького інтерфейсу з акцентом на зручність
- Використання REST API для зберігання інформації та виведення Товарів [15].
- Навчання нових можливостей у веброзробці та отримання практичного досвіду в створені великих проєктів

Об'єктом дослідження є процес створення нескладного вебкаталогу для замовлення піци з використанням сучасних вебтехнологій.

Предметом дослідження є вивчення функціоналу Svelte, створення сортування та фільтрації, адаптивності та різних функцій для зручного користування.

Результатом роботи є нескладний але повністю задовільний вебкаталог розроблений на основі фреймворку Svelte.

Бакалаврська робота складається зі вступу, трьох розділів, висновків та списку використаних посилань кількістю 12 найменувань, 43 рисунка. Загальний об’єм роботи 48 сторінок.

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ РОЗРОБКИ ВЕБСАЙТІВ

1.1 Структура вебсайту

Вебсайти створюються простими, зручними у використанні та сприйнятті. Відвідувачі повинні легко переміщатися між сторінками і розуміти, де розташовані розділи, що їх цікавлять. Щоб відвідувач не витрачав багато часу на пошук інформації, гортаючи десятки сторінок вебсайту [21].

Головна мета вебсторінки зацікавити та продати, саме тому вона повинна мати привабливий для ока вигляд, мати цікаву інформацію або товар для відвідувача, мінімум реклами, а також є дуже важливим швидкість завантаження цієї сторінки, оскільки як показує практика, людина не буде довго чекати поки завантажується сайт, замість цього вона просто вийде з нього. Наукові дослідження підтверджують, що користувачі дійсно не люблять чекати під час завантаження веб-сторінок. Згідно з дослідженнями, опублікованими на сайті Think with Google, майже половина користувачів очікують, що вебсторінки завантажуватимуться менше 2-х секунд. Із збільшенням часу очікування збільшується ймовірність того, що користувач покине сайт. Отже, важливо робити все можливе, щоб зменшити час завантаження вебсторінок та підтримувати швидкість їх роботи [13] [21].

Спочатку розберемось з тим на які типи розділять вебсайти та розглянемо коротко кожен з них [1] [30]:

Візитна картка;

- Landing Page;
- Персональний;
- Корпоративний;
- Інтернет-магазин;
- Каталог

Візитна: це сторінка, яка представляє інформацію про особистість, бізнес або організацію. Вона містить контактну інформацію, короткий опис професійних або особистих досягнень, фотографії тощо.

Landing Page: односторінковий сайт, розроблений для конкретної мети, такої як збір контактних даних, продаж продукту чи послуги, або розкриття інформації про акцію.

Персональний: це вебсторінка, яка відображає особисту інформацію про конкретну особу, як правило, використовується в особистих блогах, портфоліо, або вебсайтах для особистого брендування.

Корпоративний: сторінка, яка представляє бізнес або організацію.

Вона може включати інформацію про продукти, послуги, історію компанії, контактну інформацію, а також новини і події.

Інтернет-магазин: вебсторінка, де користувачі можуть переглядати йпридбати товари чи послуги онлайн. Зазвичай має функціонал додавання товарів до кошика, оформлення замовлення, оплату тощо.

Каталог: вебсторінка, яка містить перелік товарів чи послуг, зазвичай з вказанням їх опису, характеристик, цін тощо. Часто використовується в інтернет-магазинах або на сайтах, що надають послуги.

Рисунок нижче зображує один з варіантів будови сторінки у графічному вигляді рис. 1.1.

Структура сайту – це план розташування блоків сторінки або товарів, а також навігація. Вона є ключовою у забезпеченні функціональності та зручності користування [2].

Залежно від потреб розділяють такі схеми вебсайтів:

Алфавітна структура: цей тип структури передбачає систематизацію контенту за алфавітним принципом, що сприяє швидкому пошуку інформації, коли відвідувач вебсайту точно знає назву необхідного об’єкта чи продукту.

Хронологічна структура: використовується для організації контенту в порядку його публікації, що є оптимальним рішенням для новинних

порталів та ресурсів, які регулярно оновлюються.

Деревоподібна структура: характеризується наявністю центральної, кореневої сторінки, від якої відгалужуються категорії та підкатегорії, формуючи ієрархічну систему навігації. Це ідеально підходить для електронних комерційних платформ.

Тематична структура: забезпечує групування контенту за спільними темами чи категоріями, що полегшує користувачам процес пошуку інформації за інтересами.

Лінійна структура: представляє контент як послідовний ланцюг, змушуючи користувача слідувати певному маршруту без можливості вільного переходу між сторінками. Це часто зустрічається на сайтах-візитках.

Ґратчаста структура: кожен елемент контенту розміщується у власній гілці, що дозволяє створити багаторівневу систему навігації, яка часто використовується в інформаційних каталогах.

Павутинна структура: це складна комбінована структура, яка інтегрує декілька типів організації контенту, але через свою складність використовується нечасто.

Гібридна структура: поєднує в собі елементи різних структур, надаючи користувачам гнучкість у пошуку та доступі до інформації.

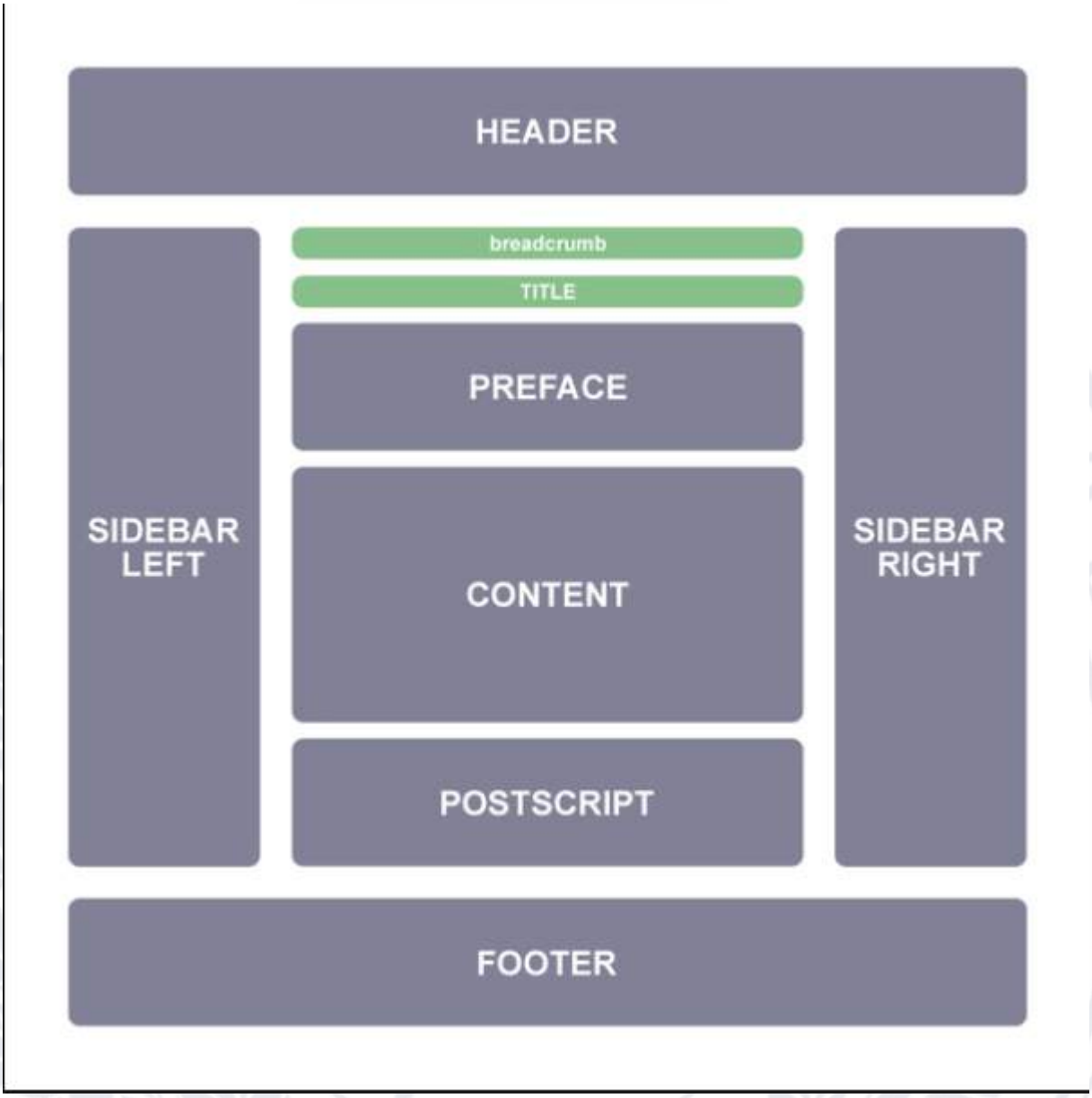

Рисунок 1.1 - Графічний приклад побудови сторінки

Будова вебсторінки не є суворо стандартизованою, тому можна розміщувати блоки за власними потребами. Наприклад у проєкті для цієї бакалаврської роботи виконується така будова головної сторінки рис. 1.2. [33] [34]

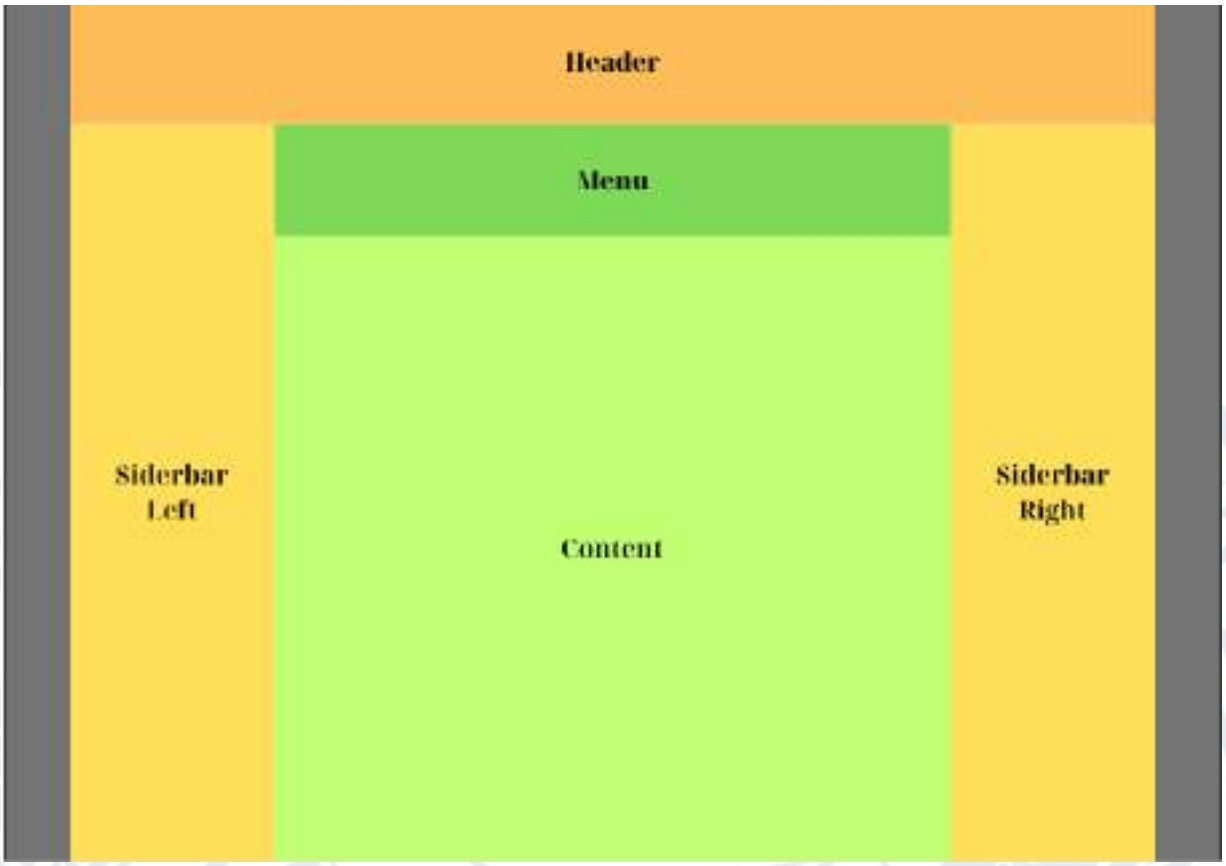

Рисунок 1.2 – Будова головної сторінки

Вебкаталог, що був створений для цієї бакалаврської роботи, має гібридну структуру, а саме:

Тематична структура: вебсайт має розділи для різних видів піц, таких як: “М’ясні”, “Вегетаріанські”, “Гриль”, “Гострі”, та інші, що вказує на тематичну організацію контенту.

Деревоподібна структура: є основне меню, з якого користувачі можуть переходити до різних категорій піц, що свідчить про наявність ієрархічної структури.

Гратчаста структура: кожен вид піци представлений окремо з можливістю вибору розміру та покупки, що формує гратчасту структуру з множинними варіантами вибору.

При плануванні та створенні вебсайту варто звернути увагу на такі важливі моменти [2, 3]:

1. Зрозумілість та логічність: структура сайту повинна бути інтуїтивно зрозумілою для користувачів. Розміщення контенту та навігація мають бути

логічними та відповідати очікуванням користувачів.

2. Ієрархія та категоризація: правильна ієрархія розділів та категорій допомагає користувачам легко знаходити потрібну інформацію. Контент повинен бути структурований за темами та типами для кращої навігації.

3. Навігація: ефективна навігація є ключовою для зручності користування сайтом. Головне меню, хлібні крихти, фільтри та пошук повинні бути інтуїтивними та доступними.

4. Адаптивний дизайн: структура сайту має бути адаптованою для різних пристроїв та екранів (комп'ютер, мобільні, планшети). Необхідно враховувати особливості перегляду контенту на різних розмірах екранів.

5. Зручність використання: структура повинна забезпечувати зручність пошуку та знаходження потрібної інформації. Мінімізувати кількість натискань, необхідних для доступу до основних розділів та функцій.

6. Пріоритети та акценти: найважливіший контент та функції повинні бути виділені та легко доступні на головній сторінці та у верхній навігації. Використовувати візуальні акценти для підкреслення ключових елементів.

7. Оптимізація для пошукових систем: структура сайту впливає на його ранжування в пошукових системах. Використовувати семантичні теги, структуру даних та правильну ієрархію.

8. Масштабованість та гнучкість: структура повинна бути гнучкою для майбутнього розширення та додавання нового контенту. Передбачити можливість легкого додавання нових розділів та категорій без порушення наявної ієрархії.

9. Послідовність та узгодженість: дизайн, навігація та розміщення елементів повинні бути послідовними на всіх сторінках сайту. Це забезпечує кращий користувацький досвід та запам'ятовуваність.

10. Зворотний зв'язок та тестування: збирати відгуки користувачів та аналізувати їхню поведінку на сайт. Проводити тестування користування для виявлення проблемних місць і вдосконалення структури.

Переваги вебкаталогів [4] [34]

1. Зручна навігація та пошук: вебкаталоги упорядковані за категоріями та підкатегоріями, що полегшує пошук потрібних товарів чи послуг. Користувачі можуть швидко звузити пошук, застосовуючи фільтри та сортування.

2. Огляд усіх доступних варіантів: каталоги дозволяють побачити весь асортимент товарів або послуг, що пропонується, в одному місці. Це допомагає користувачам порівнювати різні варіанти та приймати обґрунтовані рішення.

3. Детальна інформація про продукти: вебкаталоги містять детальні описи, специфікації, зображення та відгуки про товари, щоб допомогти користувачам приймати зважені рішення.

4. Просування товарів та послуг: каталоги є ефективним інструментом маркетингу для компаній, дозволяючи їм представити свої товари та послуги потенційним клієнтам.

5. Економія часу та зусиль: замість відвідування окремих сайтів чи магазинів, користувачі можуть знайти потрібні товари в одному місці, економлячи час і зусилля.

6. Актуальність інформації: вебкаталоги зазвичай регулярно оновлюються, забезпечуючи користувачів актуальною інформацією про ціни, наявність товарів та новинки.

7. Підвищення пізнаваності бренду: присутність у популярних вебкаталогах може підвищити пізнаваність бренду та допомогти збільшити трафік на власний вебсайт компанії.

8. Зручність електронної комерції: багато вебкаталогів інтегровані з системами електронної комерції, дозволяючи користувачам легко здійснювати покупки в одному місці.

9. Персоналізація та рекомендації: сучасні вебкаталоги можуть використовувати алгоритми для надання персоналізованих рекомендацій на основі переваг користувачів.

10. Мобільна зручність: більшість вебкаталогів мають адаптивний або мобільний дизайн, що забезпечує зручний доступ з різних пристроїв.

Ці переваги роблять вебкаталоги потужним інструментом як для покупців, що шукають товари чи послуги, так і для продавців, які прагнуть ефективно представити та просувати свої пропозиції.

Недоліки вебкаталогів

Незважаючи на численні переваги вебкаталогів, вони також мають певні недоліки, які варто враховувати:

1. Затрати на створення та підтримку: розробка та постійне оновлення вебкаталогу може бути досить затратним процесом, особливо якщо він містить величезну кількість товарів чи послуг.

2. Складність категоризації: для деяких товарів або послуг може бути складно визначити належну категорію, що може призвести до неузгодженості у структурі каталогу.

3. Потенційна неактуальність даних: якщо вебкаталог не оновлюється регулярно, інформація про наявність, ціни та характеристики товарів може застаріти.

4. Проблеми з пошуковою оптимізацією: вебкаталоги з величезною кількістю сторінок можуть мати проблеми з оптимізацією для пошукових систем (SEO), що може вплинути на їхню видимість.

5. Обмежена інформація про продукт: у вебкаталогах часто міститься лише основна інформація про товар, тоді як потенційні покупці можуть потребувати додаткових деталей.

6. Конкуренція з іншими каталогами: якщо на ринку є кілька великих вебкаталогів, це може ускладнити просування і залучення трафіку для нових або менш відомих каталогів.

7. Необхідність регулярного оновлення: каталоги повинні постійно оновлюватися, щоб забезпечити актуальність інформації, що вимагає значних зусиль і ресурсів.

8. Складність інтеграції з системами електронної комерції: інтеграція

вебкаталогу з системами електронної комерції, обробки платежів та логістики може бути складним і дорогим процесом.

9. Безособистість: вебкаталоги можуть зробити процес покупки більш безособистим, оскільки вони не забезпечують того рівня персонального обслуговування, якого очікують деякі покупці.

10. Залежність від технологій**:** вебкаталоги повністю залежать від технологій та інтернету, тому будь-які збої або перебої в роботі можуть негативно вплинути на доступність та функціонування каталогу.

Незважаючи на ці недоліки, багато компаній все ж використовують вебкаталоги як ефективний спосіб представлення та просування своїх товарів і послуг. Але важливо ретельно розглянути всі переваги та недоліки перед створенням вебкаталогу, щоб забезпечити максимальну ефективність та задоволеність користувачів.

1.2 Аналіз вебкаталогів

Важливо проводити аналіз подібних продуктів підчас створення власного каталогу і ось чому [36]:

1. Розуміння ринку: можна визначити, які вебкаталоги є популярними та чому, що допоможе вам зрозуміти потреби та інтереси вашої цільової аудиторії.

2. Виявлення ніші: аналізуючи наявні каталоги, ви можете виявити незадоволені потреби або сегменти ринку, які ви можете заповнити своїм унікальним пропозицією.

3. Оцінка функціональності: вивчення того, як інші вебкаталоги організовують інформацію та які функції вони пропонують, може надихнути на впровадження інноваційних рішень у вашому проекті.

4. Уникнення помилок: вчитися на помилках інших, аналізуючи їхні слабкі сторони та проблеми, з якими вони стикалися.

Загалом, аналіз конкурентів та ринку дозволяє створити більш

ефективний та корисний вебкаталог, який відповідатиме потребам користувачів та виокремлюватиметься серед інших.

Розглянемо декілька прикладів:

1. dominos.ua

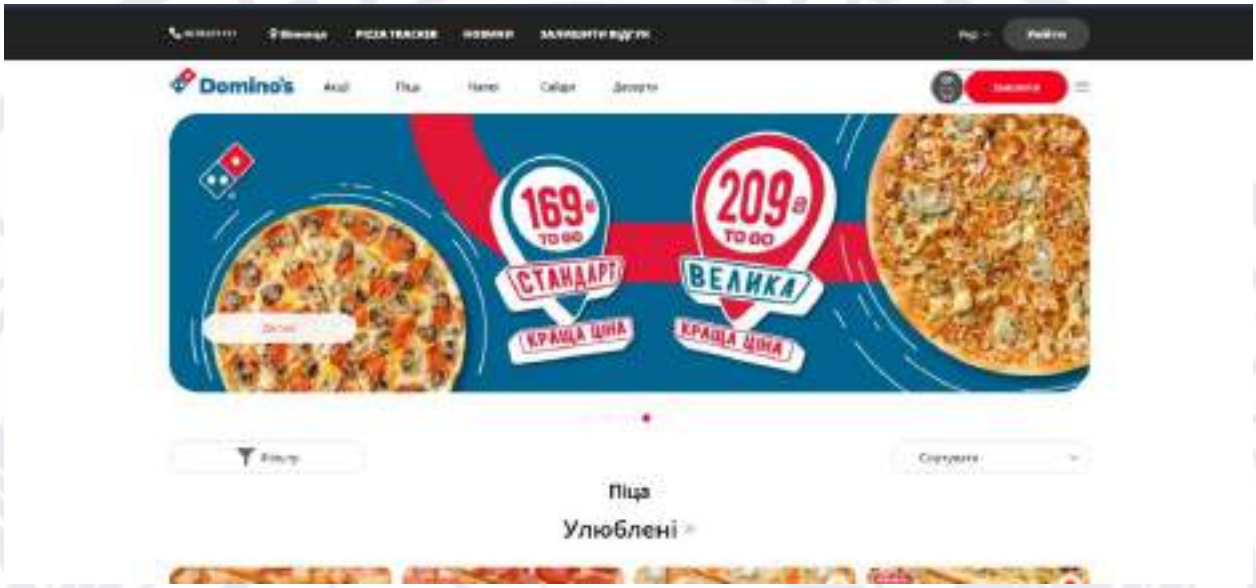

Рисунок 1.3 – Сайт dominos.ua

dominos.ua - одна з найбільших мереж піцерій у світі. Їхній вебкаталог дозволяє переглядати меню, вибирати та замовляти піцу онлайн.

Сайт має таку структуру:

- Тематична структура: вебсайт має розділи, такі як “Акції”, “Піца”, “Напої”, “Сайди”, “Десерти”, що вказує на тематичну організацію контенту.
- Деревоподібна структура: є основне меню, з якого користувачі можуть переходити до різних категорій продуктів, що свідчить про наявність ієрархічної структури.
- Ґратчаста структура: рекламний контент та продукти представлені окремо з можливістю вибору та покупки, що формує ґратчасту структуру з множинними варіантами вибору.

Навігація:

- Головна сторінка з основним меню, акціями та можливістю швидкого замовлення.

- Окремі розділи для перегляду меню.
- Інформація про доставлення, самовивіз, способи оплати.
- Контактні дані ресторану
- Чітка навігація з використанням верхнього меню та хлібних крих

Функціональність:

- Інтерактивний конструктор піци
- Система замовлення з вибором адреси доставки чи самовивозу, способу оплати.
- Інформація про акції, знижки та спеціальні пропозиції.
- Можливість залишати відгуки про ресторан та обслуговування.

Загалом сайт завантажується досить швидко, навіть на мобільних пристроях. Навігація між розділами та перехід до оформлення замовлення є плавними та зручними. Проте з погляду користувача, який щойно зайшов на сайт, може виникнути дискомфорт через те нагромадження різних візуальних ефектів та різного роду інформації.

2. express-pizza

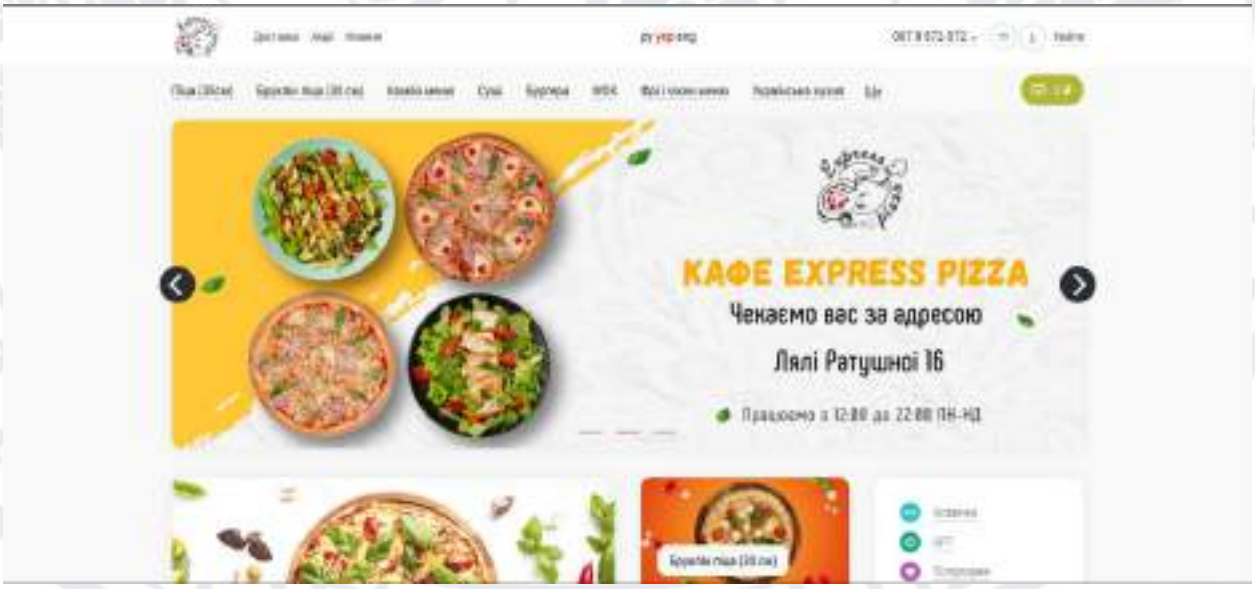

Рисунок 1.4 – Сайт express-pizza

Express Pizza - це великий регіональний вебкаталог піцерій, що охоплює міста Вінницької області України. На сайті представлено меню, умови доставлення та інформація про численні піцерії, доступні для

замовлення в конкретному місті.

Сайт має гібридну структуру. Вона поєднує елементи тематичної структури, оскільки контент групується за категоріями, і ґратчастої структури, де кожен елемент контенту розміщений у власній гілці, що дозволяє створити багаторівневу систему навігації. Це забезпечує користувачам гнучкість у пошуку та доступі до інформації. Також присутній лінійний елемент у вигляді "слайдера" з акціями на головній сторінці, який веде користувача через послідовність пропозицій. А також ґратчаста структура

Навігація:

- Головна сторінка з пошуком піцерій за містом та категоріями меню.
- Сторінки окремих піцерій з повним асортиментом піц, закусок, напоїв та десертів.
- Розділи для перегляду популярних страв, акцій та новинок.
- Інформація про умови доставлення, оплати, повернення та контактні дані.
- Чітка навігація з використанням головного меню, пошуку та хлібних крихт.

Функціональність:

- Розширений пошук піцерій за містом, кухнею, рейтингом та іншими фільтрами.
- Детальні сторінки піцерій з повним меню, описом, оцінками та відгуками.
- Можливість додавати товари до кошика та оформлювати замовлення онлайн.
- Особистий кабінет користувача для збереження історії замовлень.
- Інтеграція з популярними способами оплати та доставлення.

Сайт Express Pizza завантажується достатньо швидко, особливо на

комп'ютерах. Навігація між розділами, пошук піцерій та перехід до оформлення замовлення виконується плавно. Розміщені зображення страв не створюють надмірного візуального навантаження.

Express Pizza надає користувачам всебічний вебкаталог піцерій в регіоні з можливістю пошуку, перегляду меню та онлайн-замовлення. Сильні сторони включають широкий вибір закладів, детальні описи меню, систему оцінок та відгуків. Проте сайту не вистачає функцій для створення власної піци, розширеної інформації про акції та оптимізації мобільної версії.

3. Carnemeat

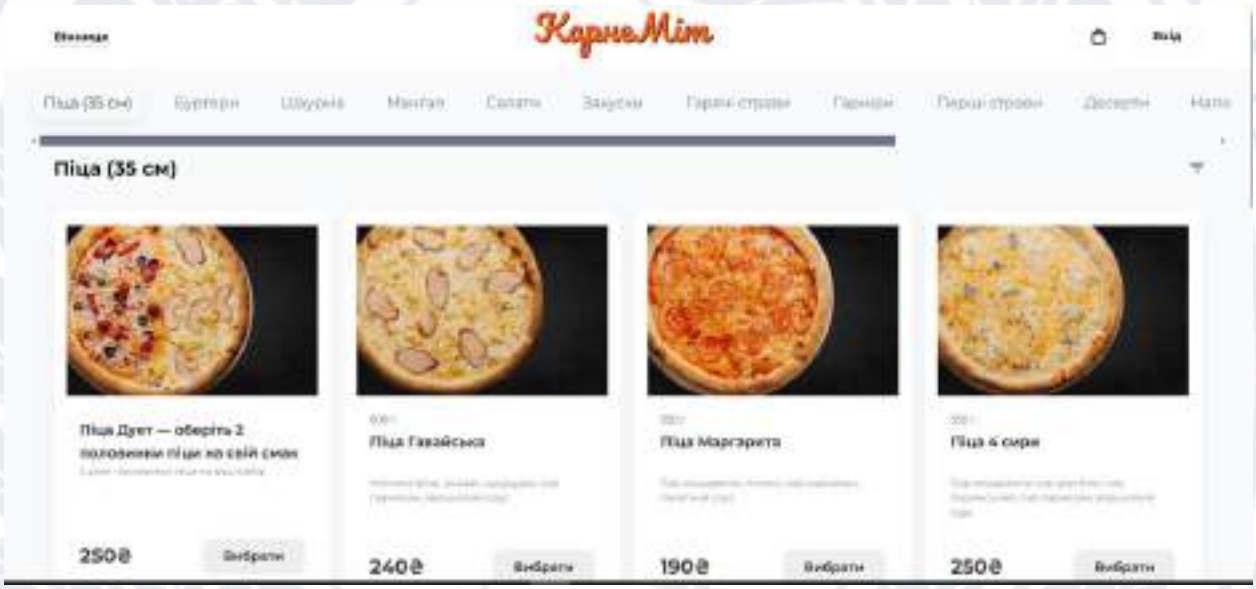

Рисунок 1.5 – Сайт carnemeat

Carne Meat - це вебкаталог, який спеціалізується на доставлені піци та інших страв від місцевих ресторанів у Вінниці, Україна. Сайт дозволяє переглядати меню, створювати замовлення та отримувати доставлення обраних страв.

Цей сайт також має гібридну структуру. Вона містить елементи тематичної структури, оскільки контент групується за категоріями й ґратчастої структури, де кожен елемент контенту розміщений у власній гілці, що дозволяє створити багаторівневу систему навігації. Це забезпечує користувачам гнучкість у пошуку та доступі до інформації. Крім того,

присутній лінійний елемент у вигляді меню піц, яке веде користувача через послідовність пропозицій. Така структура сприяє зручному вибору та замовленню страв

Навігація:

- Головна сторінка з пошуком піцерій за назвою або розділами меню.
- Сторінки окремих піцерій з повним асортиментом піц, закусок, напоїв та десертів.
- Детальні сторінки з описом, зображеннями та характеристиками кожної піци.
- Інформація про доставлення, способи оплати, акції та контактні дані.
- Зручна навігація з використанням верхнього меню, пошуку та хлібних крихт.

Функціональність:

- Наявність великого мені з різними категоріями та сортування.
- Можливість додавати обрані піци до кошика та оформлювати замовлення.
- Інтеграція з популярними способами оплати та доставлення.

Сайт Carne Meat завантажується швидко, як на комп'ютерах, так і на мобільних пристроях. Навігація між розділами, пошук піц та перехід до оформлення замовлення виконуються плавно. Зображення піц та інших візуальних елементів не створюють надмірного навантаження.

На жаль сайт має багато недоліків, порівнюючи з іншим прикладами, такі як: відсутність можливості зробити замовлення та додавати продукти до кошика без реєстрації, відсутність фільтрації, головне меню сайту займає забагато місця, що створює незручність

1.3 Застосування Node.js та SCSS для роботи зі Svelte

Розробка вебсайту з використанням Node.js є однією з

найпопулярніших технологій веброзробки. Node.js - це платформа на основі JavaScript, яка дозволяє розробникам створювати швидкі та ефективні вебсайти та застосунки. [5][17]

Node.js надає можливість виконувати JavaScript на сервері, що дозволяє розробникам створювати повноцінні вебзастосунки, які можуть взаємодіяти з базами даних, зберігати інформацію та відображати дані для користувачів. Фреймворки на основі Node.js дозволяють створювати сайти зі складними бізнес-логіками, що дозволяє підтримувати сайти з високим навантаженням та ефективно обробляти запити від користувачів [28].

Ще одним дуже корисним інструментом для фронт-енд розробки є препроцесор. SCSS (Sass - Syntactically Awesome Style Sheets) – це препроцемор написаний мовою Rubi, заснована на CSS, який надає додаткові можливості, що дозволяє писати більш структурований, модульний та масштабований CSS-код. SCSS є розширеною версією препроцесора Sass, який використовує синтаксис, схожий на традиційний CSS, що полегшує перехід для розробників, звиклих до стандартного CSS. [6]

SCSS надає можливість оголошувати змінні для зберігання значень, вкладати селектори один в одного, визначати повторювані блоки стилів у вигляді міксинів, підтримує визначення та використання власних функцій, дозволяє розбивати стилі на окремі модулі та імпортувати їх у головний файл, дає можливість використовувати механізм розширення, коли один селектор наслідує стилі іншого, підтримує як стандартні CSS-коментарі, так і коментарі в стилі JavaScript.

Svelte - це революційний фреймворк для побудови веб-інтерфейсів, який відрізняється від інших популярних фреймворків, таких як React або Vue.js. Замість використання віртуального DOM та техніки перезапису, Svelte здійснює компіляцію компонентів у високоефективний код, що виконується у браузері. Це дозволяє досягти кращої продуктивності та менших розмірів файлів порівняно з традиційними фреймворками [7].

Цикл написання вебсайту на основі Svelte [25, 26]

1. Вивчення основ Svelte:
- Ознайомлення з концепцією компонентів Svelte та їх особливостями.
- Вивчення синтаксису Svelte, включаючи шаблони, обробку подій, реактивні змінні тощо.
- Дослідження вбудованих директив, модифікаторів та бібліотек Svelte.
2. Розробка компонентів:
- Створення базових компонентів (наприклад, header, footer, sidebar).
- Реалізація логіки взаємодії між компонентами (передача даних, обробка подій).
- Застосування Svelte-специфічних концепцій, таких як реактивність, життєвий цикл, анімації.
3. Впровадження маршрутизації:
- Вибір та інтеграція бібліотеки маршрутизації (наприклад, svelt-spa router) [37].
- Налаштування динамічних маршрутів, параметрів та переходів між сторінками.
- Реалізація логіки обробки маршрутів у компонентах.
4. Підключення до API:
- Інтеграція з серверною частиною або зовнішніми API.
- Розробка компонентів для відображення, фільтрації та взаємодії з даними.
- Обробка асинхронних запитів та станів завантаження.
- Стилізація та оформлення:
- Вибір підходу до стилізації (CSS-файли, CSS-модулі, Styled Components).
- Створення глобальних та локальних стилів для компонентів.
- Адаптація дизайну до різних пристроїв та розмірів екрану.
5. Тестування та налагодження:

- Пошук та виправлення помилок, оптимізація продуктивності.
- Налаштування процесу збірки, автоматизації та розгортання.

6. Розгортання та супровід:

- Вибір платформи для хостингу та розгортання застосунку.
- Налаштування процесу безперервної інтеграції та розгортання.
- Моніторинг, оновлення та підтримка застосунку в робочому стані.

Під час написані вебкаталогу були використані певні команди Node.Js які, вони є корисними як для завантаження бібліотек, так і для запуску певних модулів. Ось команди та їхні складові які були використані підчас розробки [8] [27]:

npm - це система керування пакетами, яка використовується для інсталяції та керування залежностями проєкту.

node - ця команда використовується для запуску файлів JavaScript в середовищі Node.js.

npm i - використовується для інсталяції пакетів залежностей проєкту з пакетного менеджера npm. Ця команда зчитує список залежностей з файлу package.json у кореневій директорії проєкту та інсталює всі необхідні пакети.

npm init – створить файл package.json

npm run dev - використовується для запуску скрипту з налаштуваннями для розробки проєкту. Зазвичай, ця команда запускає сервер та робить доступним застосунок у режимі розробки.

npm run server - використовується для запуску локального сервера розробки вебзастосунка. Вона дозволяє перевірити проєкт на локальному сервері перед розгортанням на платформі.

В нашому випадку був використаний Node.js, SCSS та Svelte задля вивчення синтаксису цих технологій був створений невеликий сайт каталог.

9.4 Git та GitHub

Git - це система контролю версій, яка дозволяє зберігати та відстежувати зміни в коді програмного забезпечення, а також злиття цих змін між кількома розробниками. Git забезпечує зручну спільну роботу над проєктом, забезпечує збереження історії змін, дозволяє швидко відновлювати попередні версії коду та відстежувати проблеми, які виникають в процесі розробки [9].

GitHub - це хмарний сервіс, який надає безплатне зберігання коду та забезпечує зручний інтерфейс для спільної роботи над проєктами. GitHub використовує Git як базову технологію для контролю версій та надає додаткові функції, такі як інструменти для спільної роботи, зручний інтерфейс для перегляду та відстеження змін, можливість відгалужувати проєкти та внесення змін до цих відгалужень [10, 20].

Чим важливі ці технології для розробки проєкту?

Git та GitHub важливі для створення сайту, оскільки вони дозволяють розробникам зберігати та відстежувати зміни в коді сайту, працювати над проєктом в команді, зберігати історію змін, а також ділитися кодом з іншими розробниками та співпрацювати з ними. Крім того, Git та GitHub дозволяють швидко відновлювати попередні версії коду, які працювали до виникнення проблем, а також легко відстежувати проблеми та помилки, які виникають в процесі розробки. Таким чином, використання Git та GitHub допомагає розробникам ефективно та організовано працювати над проєктом сайту, що дозволяє значно збільшити продуктивність розробки та якість [20].

Команди що були використані [11]:

- git init: утворює новий репозиторій Git.
- git clone: клонує репозиторій з веб-сервера в локальний каталог.
- git add: додає зміни до індексу (також відомий як "staging area").
- git commit: зберігає зміни у репозиторії.
- git push: відправляє зміни до віддаленого репозиторію.
- git pull: отримує зміни з віддаленого репозиторію.

- git status: показує поточний стан робочого каталогу та індексу.
- git log: показує список попередніх комітів.
- git branch: показує список гілок.
- git checkout: перемикається між гілками та комітами.

Посилання на репозиторій GitHub - https://github.com/LirikTop/svelte-pizza-it-school.git

Висновок з розділу 1

У першому розділі було розглянуто загальні відомості про структуру та особливості вебсайтів, зокрема вебкаталогів. Було розкрито ключові елементи побудови ефективної структури вебсайту, такі як зрозумілість, логічність, ієрархія, навігація, адаптивний дизайн, зручність використання тощо. Також проаналізовано переваги та недоліки вебкаталогів як інструменту для представлення та просування товарів і послуг.

Для створення власного вебкаталогу був проведений аналіз наявних прикладів, що надало розуміння щодо застосування різних структурних схем, особливостей навігації, зручності використання та функціональних можливостей.

Окрім того, в розділі висвітлено використання технологій Node.js, SCSS та Svelte для розробки вебсайту, а також важливість застосування системи контролю версій Git та платформи GitHub у процесі розробки. Це дозволяє організувати ефективну розробку, забезпечити прозорість та збереження історії змін, а також сприяє спільній роботі над проєктом.

Загалом, перший розділ надає комплексне розуміння ключових аспектів проєктування структури вебсайту, зокрема вебкаталогів, та технологічних інструментів, що застосовуються для їх створення.

РОЗДІЛ 2
ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБКАТАЛОГУ НА ОСНОВІ SVELTE

2.1 Структура проєкту та опис головного файлу

Проєкт написаний за допомогою фреймворку Svelte та препроцесор SCSS як замінник CSS. Щоб почати роботу зі Svelte потрібно, після встановлення Node.JS та GIT, прописати в консолі редактора коду, в цьому випадку Visual Studio Code, такі команди [12, 32]:

npx degit sveltejs/template firstapp - так ми налаштуємо degit, який завантажить останню версію шаблона проєкту Svelte з репозиторію в щойно створену теку firstapp

Перейдемо до теки firstapp та запустимо npm install, щоб завантажити додаткові залежності проєкту.

Під час написання матеріалу, залежності були такі:

- "@fortawesome/fontawesome-free": "^6.5.1",
- "axios": "^1.6.2",
- "json-server": "^0.17.3",
- "rollup-plugin-scss": "^4.0.0",
- "sass": "^1.66.1",
- "sirv-cli": "^2.0.0",
- "svelte-spa-router": "^3.3.0"

Після цього в консолі прописуємо команду npm run dev Що дозволить перейти в режим розробки та розгорнути стартовий сайт для майбутнього його редагування.

Далі показано структуру реалізованого проєкту рис. 2.1

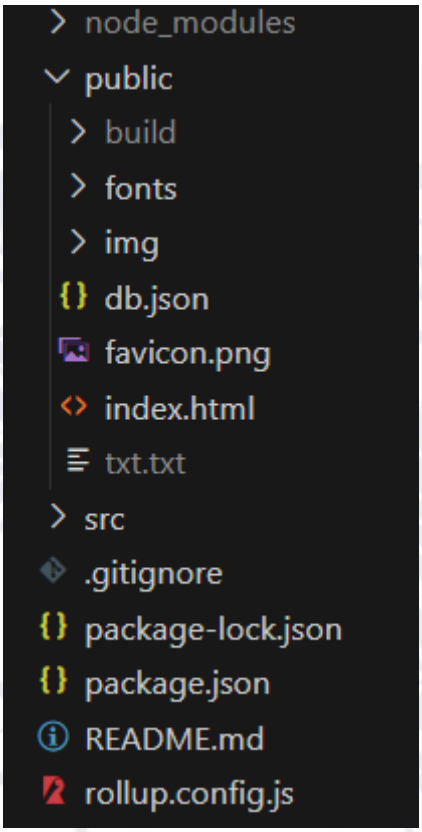

Рисунок 2.1 – Структура реалізованого проєкту

Файл rollup.config.js там прописані важливі процеси від самого його завантаження, оскільки при створенні сайту було бажання навчитись користуватись препроцесором SCSS, отже необхідно його встановити і прописати в цьому файлі. Для встановлення потрібно прописати *npm i sass*. Далі потрібно прописати в rollup.config.js наступний код рис. 2.2 та рис. 2.3. На рисунку рис. 2.2 показано які імпорти повинні бути прописані для правильного функціонування процесів. На рис. 2 вказано код, який прописаний для процесу ‘переписування’ SCSS коду в мову CSS в файл під назвою “bundle.css” його було створено окремо в директорії build рис. 2.1 [19]

```
rollup.config.js > ...
    Click here to ask Blackbox to help you code faster
1   import { spawn } from 'child_process';
2   import svelte from 'rollup-plugin-svelte';
3   import commonjs from '@rollup/plugin-commonjs';
4   import terser from '@rollup/plugin-terser';
5   import resolve from '@rollup/plugin-node-resolve';
6   import livereload from 'rollup-plugin-livereload';
7   import css from 'rollup-plugin-css-only';
8   import scss from 'rollup-plugin-scss';
9
```

Рисунок 2.2 – Вміст файлу rollup.config.js

На рисунку зображено вміст файлу rollup.cofig.js, який є “збирачем” модулів усього проєкту, при встановлені потрібної бібліотеки вона буде відображатися як на рисунку [24].

```
41      plugins: [
42          svelte({
43              compilerOptions: {
44                  // enable run-time checks when not in production
45                  dev: !production
46              }
47          }),
48          scss({
49              include: ["/**/*.css", "/**/*.scss"], // Які файли буде білдити scss
50              fileName: 'bundle.css', // В який файл треба білдити
51              failOnError: true, // Якщо щось неправильно написано, видитись не буде
52              import: ("sass") // Використовуй препроцесор sass
53          }),
54          // we'll extract any component CSS out into
55          // a separate file - better for performance
56          css({ output: 'bundle.css' }),
```

Рисунок 2.3 - Вміст файлу rollup.config.js

В теці src зберігаються компоненти, стилі основних сторінок, база даних для бібліотеки json-server [18] [22], також містить окремі скрипти router для перемалювання сторінок і скрипт згортання при натисненні на пусте місце рис. 2.4.

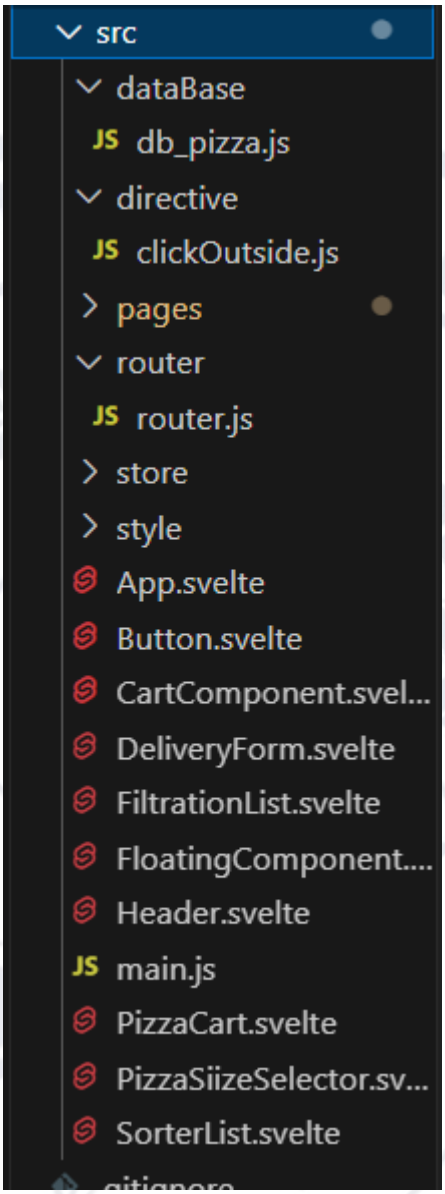

Рисунок 2.4 – Приклад наповнення теки src

Тека public містить шрифти, іконки, логотип, файл html, який зберігає в собі всі посилання та, після компіляції усіх сторінок, буде розгорнутий на веббраузері [15]. В середині теки public є ще одна build, що містить скомпільовані файли сторінок svelte в чистий js і файл з скомпільованим scss в чистий css, потім у файлі html завдяки посилання на ці файли буде розгорнутий сайт, компіляція необхідна, оскільки веббраузери розуміють тільки чистий js та css рис. 2.5.

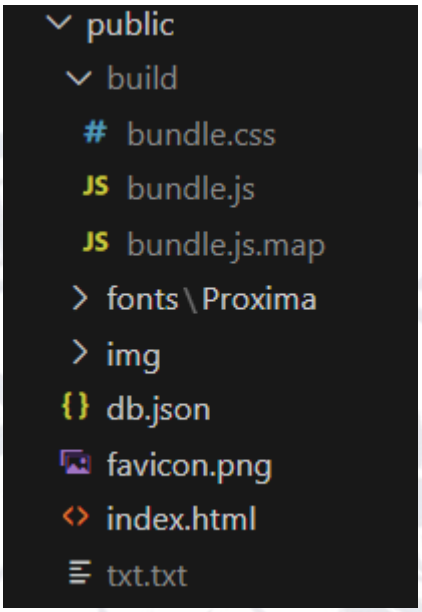

Рисунок 2.5 – Приклад наповнення теки public

Окрім цього також є: .gitignore він потрібен, щоб не було автоматично передано зазначені файли в репозиторій GitHub, зазвичай вказують ті файли які забирають забагато місця й не є необхідними для озгортання на сервері рис. 2.6, node_modules це файл який автоматично створюється та наповнюється коли ми завантажуємо якусь бібліотеку командою npm i, саме цей файл буде обов'язково написаний в .gitignore рис. 2.7.

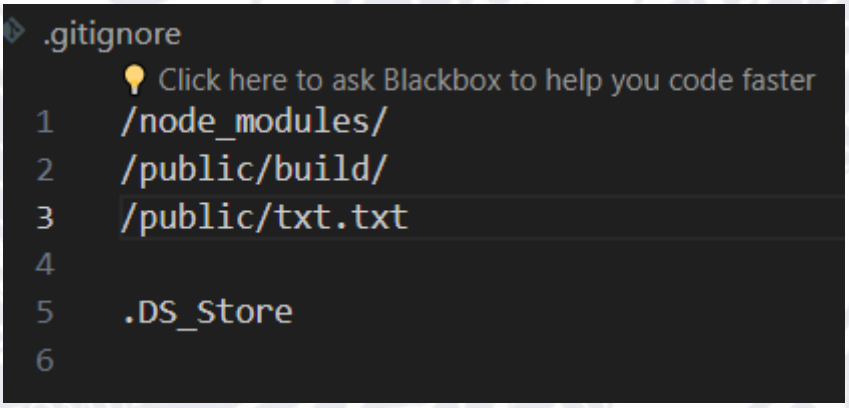

Рисунок 2.6 – Приклад наповнення для .gitignore

На рисунку видно зміст файлу .gitignore, вміст файлу каже програмі ігнорувати вказані файлі, тобто при застосуванні команди git push вказані файли передавитись не будуть, це робиться за необхідністю, якщо розмір файлів є завеликими або вони не є важливими [29].

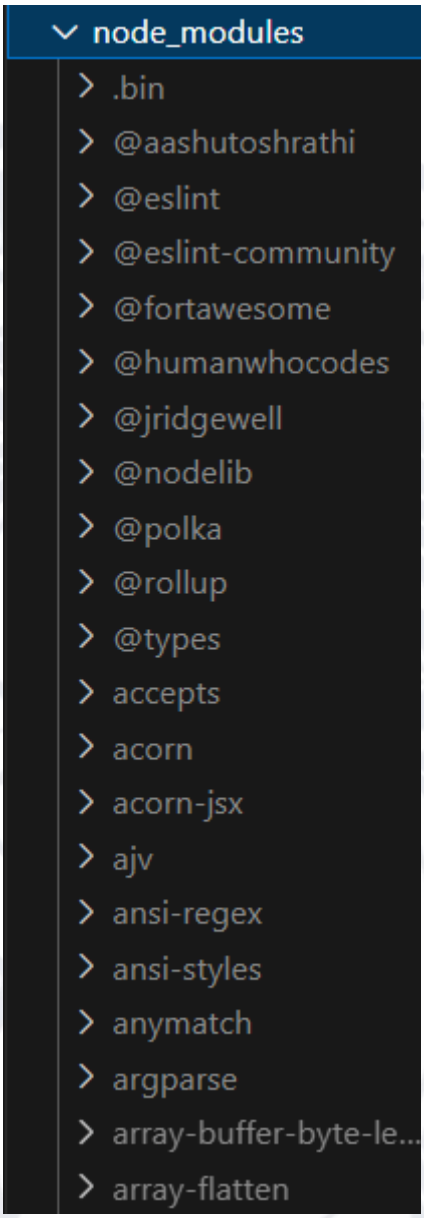

Рисунок 2.7 – Приклад наповнення node_modules

На рисунку зображено вміст теки node_modules, в ній зберігаються усі бібліотеки які були завантажені автоматично при створені проєкту або самим розробником.

2.2 Реалізація фільтрації та сортування та json файл

Однією з найскладніших завдань під час створення каталогу були реалізація функцій сортування та фільтрації рис. 2.8.

Реалізація фільтрації та сортування є важливим компонентом каталогу. Користувачі повинні мати змогу гнучко налаштовувати критерії пошуку та

впорядковувати результати відповідно до своїх уподобань. Ця функціональність дозволить підвищити зручність використання сайту, збільшити продуктивність користувачів та надати їм більший контроль над списком товарів.

Рисунок 2.8 – Фільтрація та сортування проєкту

Як видно на рисунку 2.4 елементи каталогу фільтруються за 6 критеріями, перший – всі елементи, другий – м’ясні піци, третій – вегетаріанські, четвертий – гриль піци, п’ятий – гострі піци, шостий – закриті піци. Всі категорії фільтрації можуть бути відсортовані за трьома критеріями: перши – за популярністю піц, другий за алфавітом, третій – ціна.

Усі елементи зберігаються в масиві об’єктів json файлу в кожному об’єкті міститься уся інформація яка потрібна для фільтрації та сортування рис. 2.9.

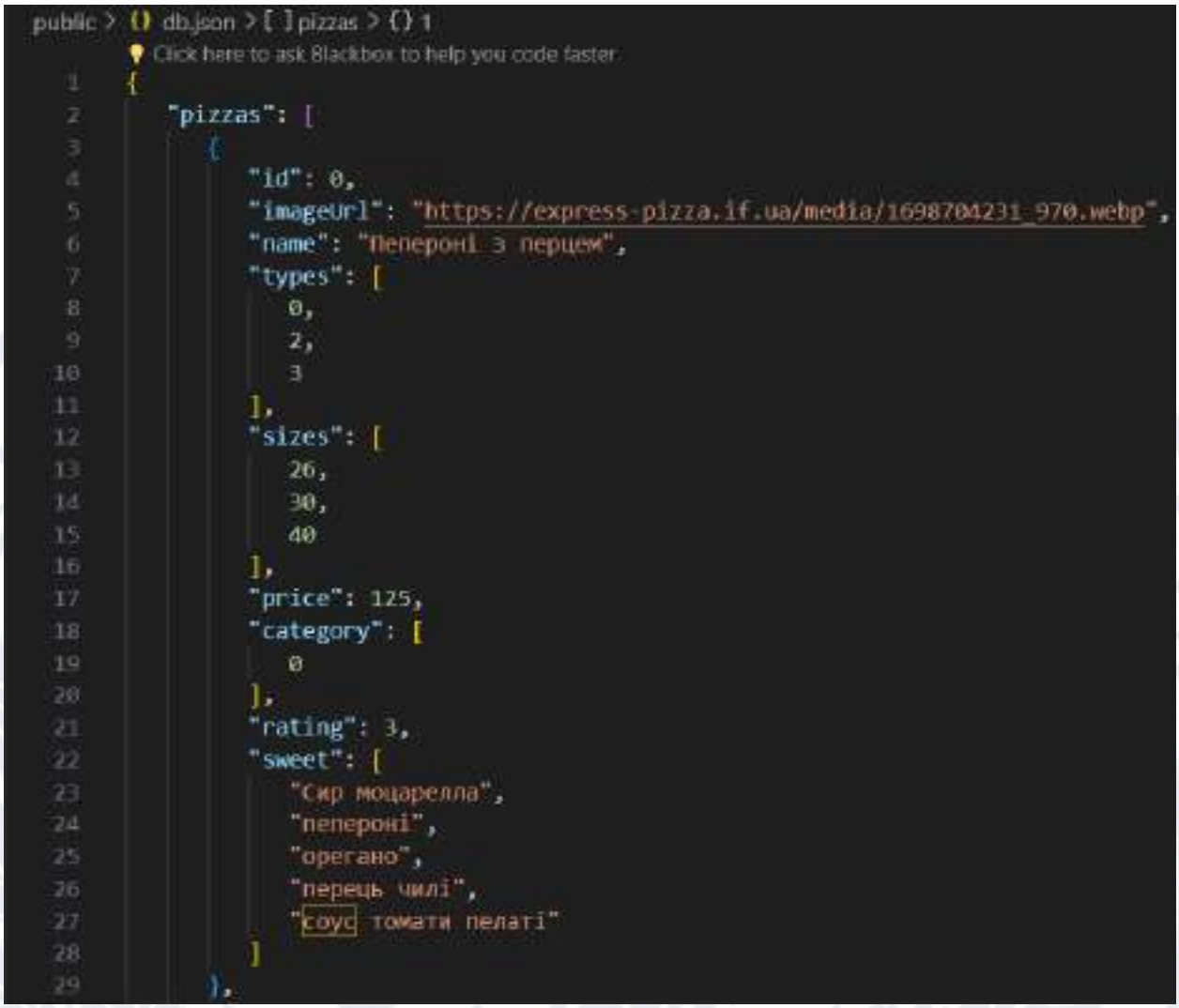

Рисунок 2.9 – Приклад об’єкту піци

Для виводу одного з елементів масиву об’єктів використовується бібліотека npm-json-server, вона з її допомогою ми перетворюємо json файл на локальний сервер та передаємо посилання в основний код, сторінки де відображатимуться піци, та робимо запит до елементів масиву рис. 2.10, далі фільтруємо та сортуємо [31].

```
let url = "http://localhost:3000/pizzas";

let arrayPizza = [];
let arrayFilter = [ ···
];

let selectFilter = arrayFilter[0].value;
let selectSort = 0;
$: updatePizzas(selectFilter, selectSort);

function updatePizzas(filter, sort) {
    axios({
        method: "get",

        url: url,
    })
        .then((response) => {
            let pizzas = response.data;
            if (filter !== "") {
                pizzas = pizzas.filter((pizza) =>
                    pizza.types.includes(Number(filter)),
                );
            }
            switch (sort) {
                case 0:
                    pizzas.sort((a, b) => b.rating - a.rating);
                    break;
                case 1:
                    pizzas.sort((b, a) => b.price - a.price);
                    break;
                case 2:
                    pizzas.sort((a, b) => a.name.localeCompare(b.name));
                    break;
            }
            arrayPizza = pizzas;
        })
```

Рисунок 2.10 - Приклад написання запиту та сортування

На рисунку зображено вміст одного з компонентів, цей код створює можливість сортувати файлі які зберігаються на локальному сервері.

Детальний код можна побачити в Додатку А.

2.3 Публікація сайту

Для успішної реалізації будь-якого вебпроєкту, публікація сайту є завершальним та надзвичайно важливим етапом. Це той момент, коли створений продукт стає доступним для широкої авдиторії та починає виконувати свої функції. Процес публікації сайту охоплює цілий ряд технічних та організаційних кроків, які забезпечують безперебійне функціонування вебресурсу в мережі Інтернет [35].

Буде проведено аналіз популярних гостингів провайдерів та порівняємо їх з тим на якому було опубліковано сайт

Отже, розглянемо декілька популярних гостингів:

1. DigitalOcean

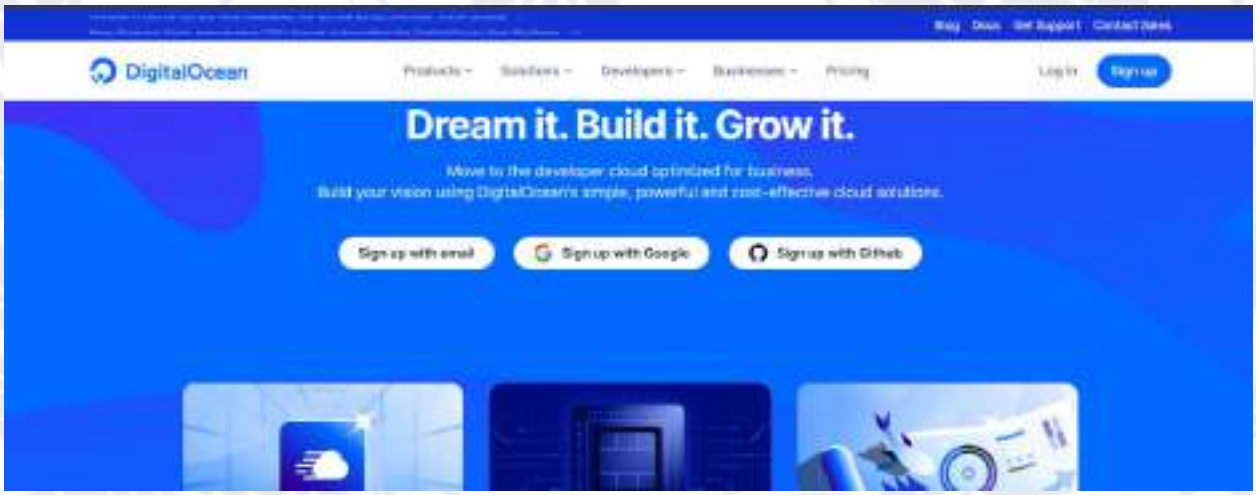

Рисунок 2.11 – Головна сторінка DigitalOcean

- Переваги: висока продуктивність, гнучкість, підтримка різних операційних систем, широкий набір конфігурацій
- Недоліки: складність налаштування для недосвідчених користувачів, відсутність безкоштовного тарифу

2. Amazon Web Services (AWS)

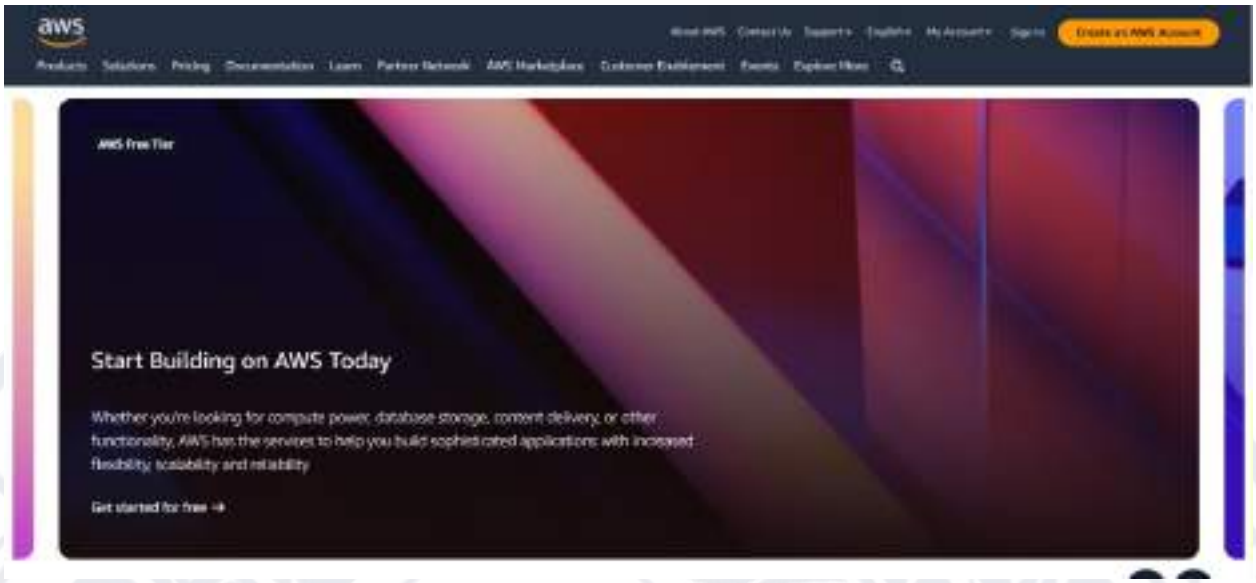

Рисунок 2.12 – Головна сторінка Amazon Web Services

- Переваги: масштабованість, широкий спектр сервісів, глибока інтеграція, наявність безкоштовного тарифу
- Недоліки: складність налаштування, крива навчання, висока вартість для оптимального використання

3. Google Cloud Platform (GCP)

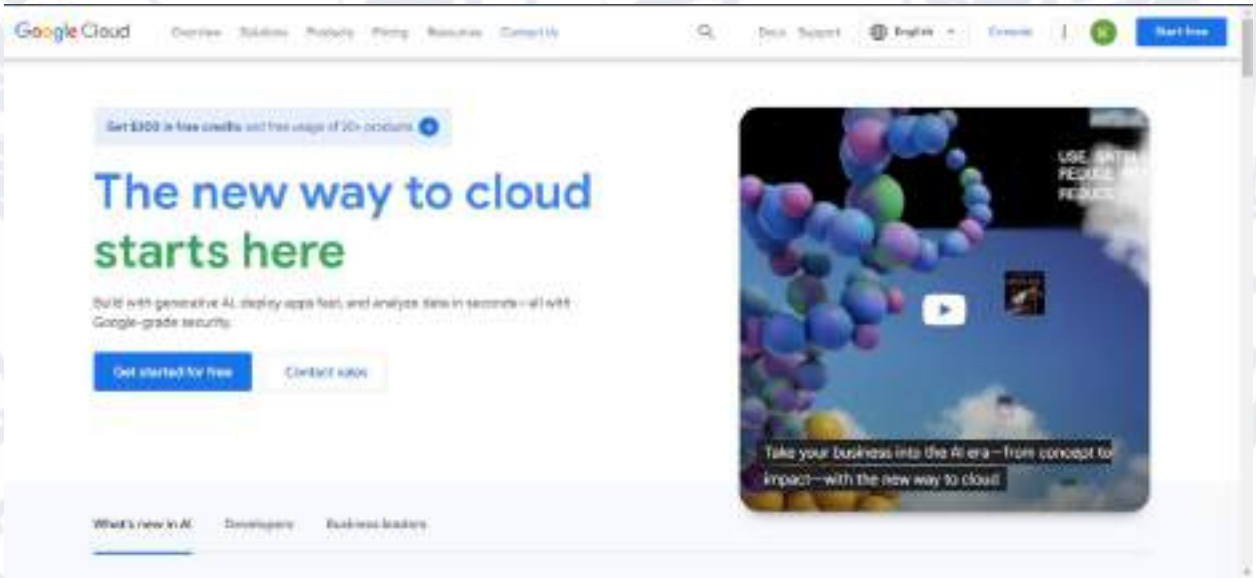

Рисунок 2.13 – Головна сторінка Google Cloud Platform

- Переваги: масштабованість, надійність, інтеграція з іншими Google сервісами, наявність безкоштовного тарифу
- Недоліки: складність налаштування, висока вартість для деяких сервісів

Для розгортання вебкаталогу, розробленого для цієї бакалаврської робити, було вибрано платформу Netlify [23]

Порівняння з Netlify:

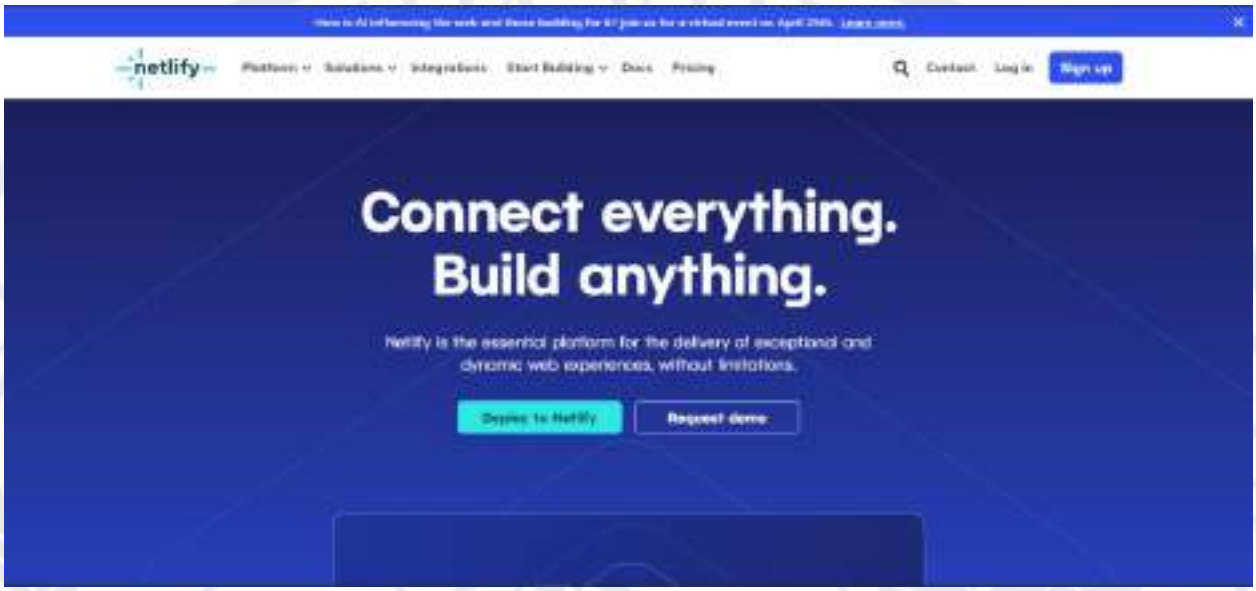

Рисунок 2.14 – Головна сторінка Netlify

Netlify - це хостинговий сервіс, що ідеально підходить для публікації статичних сайтів та безсерверних вебзастосунків. У порівнянні з іншими популярними хостингами, Netlify має наступні переваги:

- Простота використання: Netlify має дуже зручний та інтуїтивно зрозумілий інтерфейс, що робить процес публікації сайту максимально простим.
- Безкоштовний тариф: Netlify пропонує безкоштовний тарифний план, що дозволяє розробникам публікувати свої проєкти без додаткових витрат.
- Вбудована підтримка CI/CD: Netlify надає вбудовані можливості для безперервної інтеграції та розгортання, що значно спрощує процес публікації.
- Висока доступність та масштабованість: завдяки глобальній CDN-мережі, Netlify забезпечує високу доступність та масштабованість для вебсайтів.
- Додаткові функції: Netlify пропонує широкий спектр додаткових

можливостей, таких як підтримка SSL/TLS, функції безпеки, моніторинг та аналітика.

Тепер розглянемо як легко розгорнути сайт на цій платформі:

1. Створення облікового запису на Netlify рис 2.15

Рисунок 2.15 – Сторінка реєстрації netlify

- Перейдіть на вебсайт Netlify (https://www.netlify.com/) рис. 2.14 та натисніть
- "Get Started" для створення нового облікового запису.
- Виберіть спосіб авторизації, наприклад, через GitHub, GitLab або email.

2. Підключення сайту до Netlify

- Після авторизації, натисніть "New site from Git" для підключення вашого вебсайту рис. 2.16.

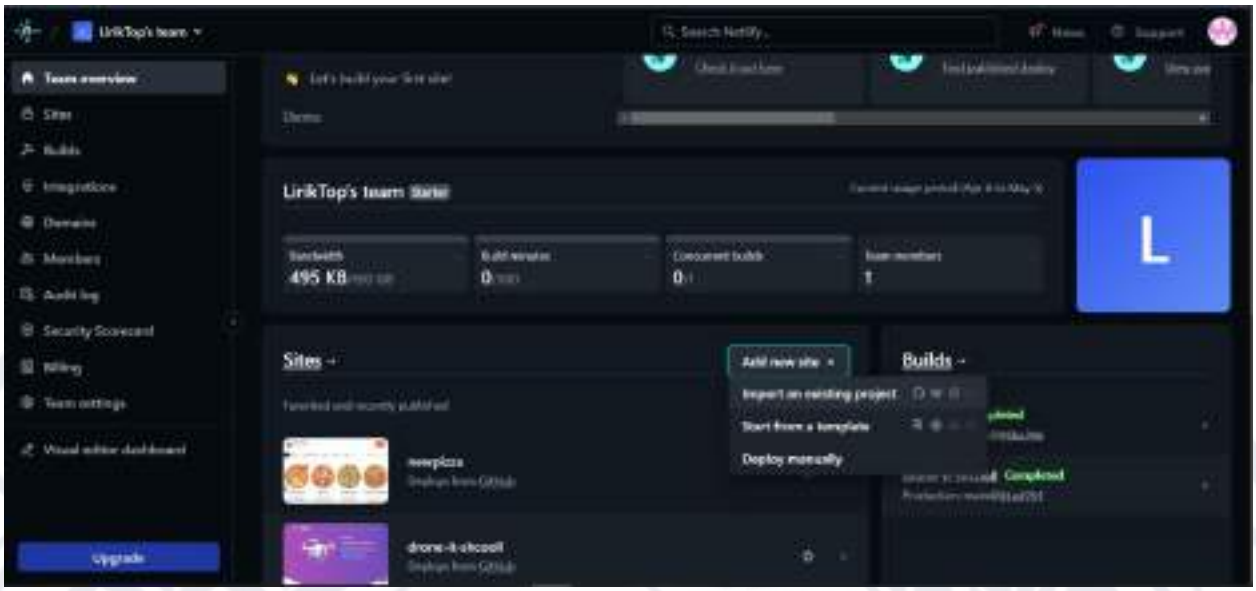

Рисунок 2.16 – Сторінка додавання нового сайту

- Виберіть Git-провайдера, де знаходиться ваш сайт (наприклад, GitHub), і виберіть необхідний репозиторій рис. 2.17.

Рисунок 2.17 – Сторінка для вибору репозиторію

3. Налаштування розгортання

- На сторінці налаштувань розгортання, Netlify автоматично визначить основні параметри вашого сайту, такі як команда збірки та директорія публікації рис. 2.18.
- За потреби, ви можете налаштувати ці параметри відповідно до ваших потреб.

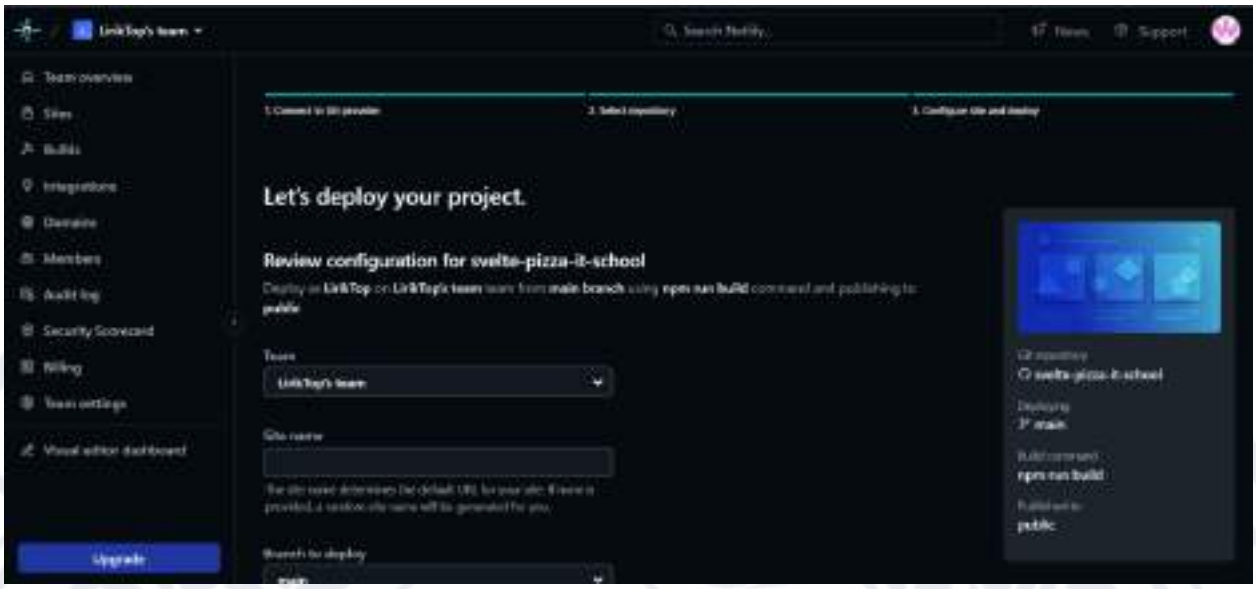

Рисунок 2.18 – Останній етап перед публікацією

- Обов’язково прописати команду *npm run build*, щоб сайт міг запуститись рис. 2.19.

Рисунок 2.19 – обов’язкова команда

4. Публікація сайту

- Після налаштування всіх необхідних параметрів, натисніть "Deploy site" для публікації вашого вебсайту.
- Netlify автоматично розгорне ваш сайт і наддасть вам посилання на нього.

5. Подальше управління

- У панелі адміністратора Netlify ви можете відстежувати стан розгортання, переглядати логи, налаштовувати автоматизацію та керувати всіма аспектами публікації вашого вебсайту рис. 2.20

Покликання на вебкаталог - https://newpizza.netlify.app/

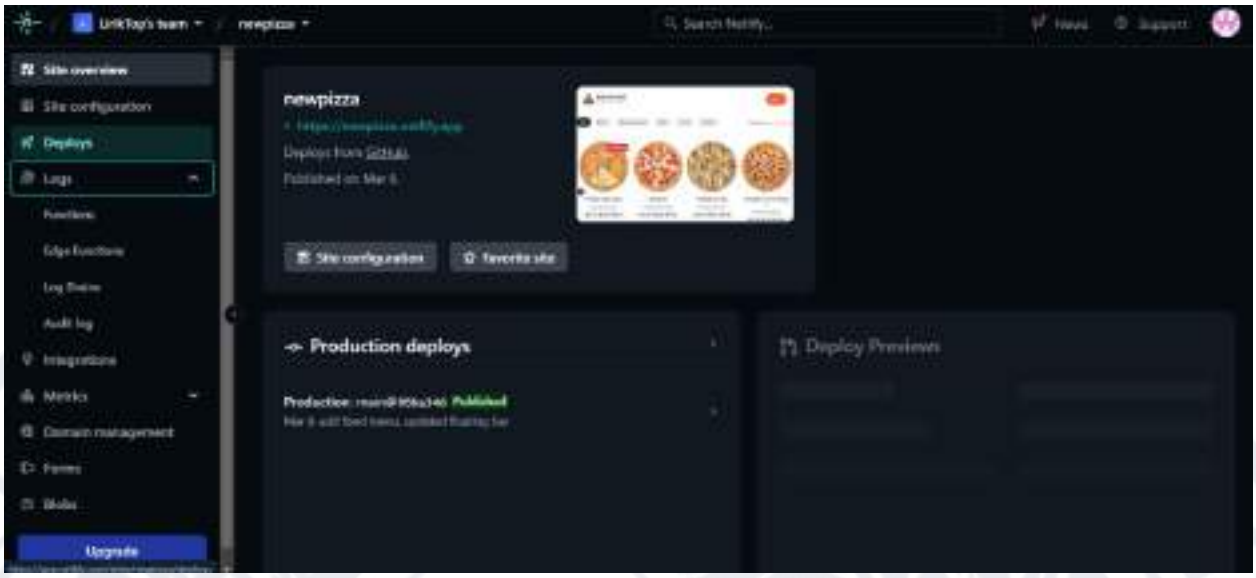

Рисунок 2.20 – Сторінка з налаштуваннями сайту

Висновок з розділу 2

У другому розділі дипломної роботи було детально розглянуто процес програмної реалізації вебкаталогу. Основною складовою цього розділу стала розробка функціонала фільтрації та сортування даних в каталозі.

Було продемонстровано структуру проєкту, включаючи головні файли та налаштування препроцесора SCSS. Детально описано процес реалізації фільтрації за різними критеріями, а також сортування за популярністю, алфавітом та ціною. Ці механізми дозволяють користувачам гнучко налаштовувати перегляд товарів відповідно до їхніх потреб, що підвищує зручність використання системи.

Окрім того, у другому розділі було приділено увагу процесу публікації розробленого вебсайт. Проведено аналіз популярних хостингових рішень, таких як DigitalOcean, AWS та GCP, та порівняно їх з обраною платформою Netlify. Було наведено покрокову інструкцію щодо розгортання вебкаталогу на Netlify, яка продемонструвала простоту, гнучкість та потужні можливості цього хостингового сервісу.

Таким чином, другий розділ дипломної роботи всебічно висвітлив технічні аспекти програмної реалізації та розгортання вебкаталогу, закладаючи основу для подальшого впровадження розробленого рішення.

РОЗДІЛ 3
ОГЛЯД ВЕБКАТАЛОГУ

3.1 Зовнішній вигляд вебкаталогу

Зовнішній вигляд сторінки відіграє важливу роль у зручному користуванню будь-якого сайту, від нього залежить де будуть розташовані ті чи інші функції сайту, а також чи захоче користувач взагалі послуговуватись цим сервісом. Функції що будуть на рисунках ми протестуємо трохи згодом.

Перша сторінка вебкаталогу є головною, тому на ній одразу розташовані усі товари які є в наявності рис. 3.1. Користувач має змогу фільтрувати та сортувати товари, продивитись ціну, яка змінюється залежно від розміру піци, прочитати склад і додати в кошик. Також є можливість подзвонити або написати на гарячу лінію сервісу.

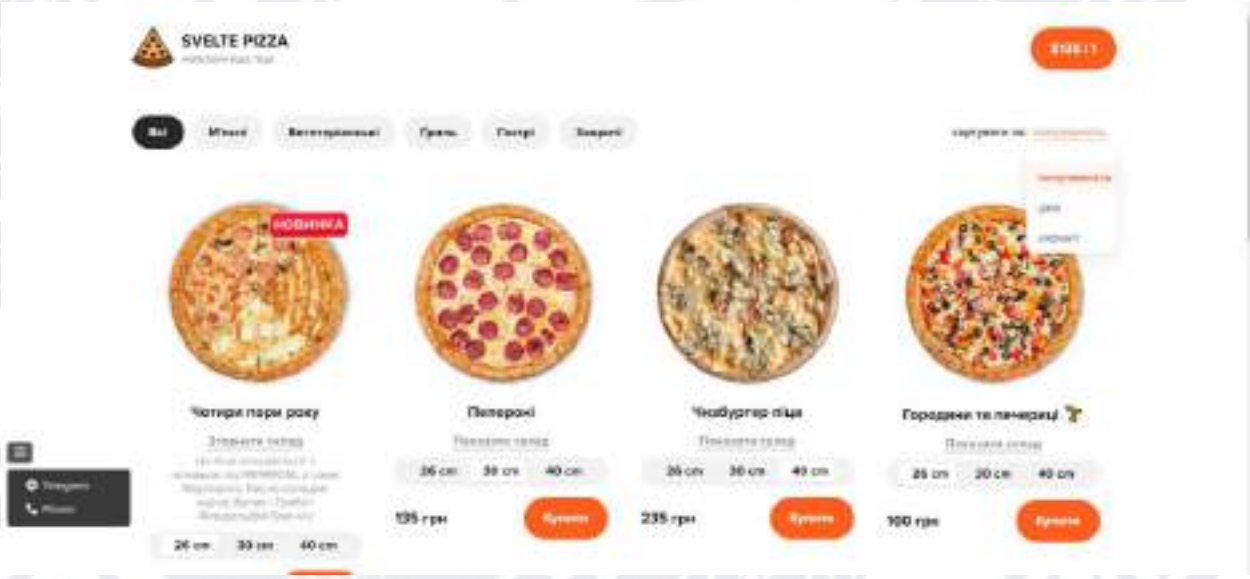

Рисунок 3.1 – Головна сторінка вебкаталогу

Друга сторінка це «кошик», вона зберігає товари які вибрав користувач на головній сторінці рис. 3.2. На сторінці «кошик» користувач може: збільшити або кількість піц, побачити ціни, видалити вміст кошика, повернутись до головного меню або оформити доставлення рис. 3.3.

Рисунок 3.2 – Сторінка «кошик»

На рисунку ми бачимо одну вибрану піцу «Пепероні», користувач може робити певні маніпуляції які наведено вище.

Рисунок 3.3 – Форма доставки

Друга сторінка може мати інший вигляд якщо користувач не вибрав жодної піци рис. 3.4. Тоді кошик не буде містити жодного товару, замість нього буде відображено візуальний елемент та напис «Кошик пустий. Ви не додали піцу». Можна повернутись назад, натиснувши на відповідну кнопку.

Рисунок 3.4 – Пуста сторінка кошику

Усі сторінки при переході від одної до іншої використовують бібліотеку svelte-spa-router яка дозволяє «перемалювати» сторінки, тобто перехід між сторінками буде відбуватися без перезавантаження, вона забезпечуючи при цьому швидку та плавну взаємодію користувача [37] [38]. Код можна переглянути в Додатку А.

3.2 Огляд функцій вебкаталогу

Ми вже розглянули які функції присутні на усіх сторінках, тому зараз перевіримо як вони працюють.

Вебкаталог має адаптивний режим, тому для зручності перегляду, рисунки ниже будуть показані в розмірі 800 px на 1280 px (розмір планшета) [39] [40].

Коректність виконання більшості функцій можна звірити з інформацією яка записана в файлі json, Додаток А.

Повернемося до першої, головної, сторінки. Як видно на рис. 3.1 на головній сторінці є багато різних дій.

1. Фільтрація

- за замовчуванням фільтрація стоїть на категорії «всі» як на рис. 3.1. Змінимо категорію на «вегетаріанські» рис. 3.5 та «гострі» рис. 3.6

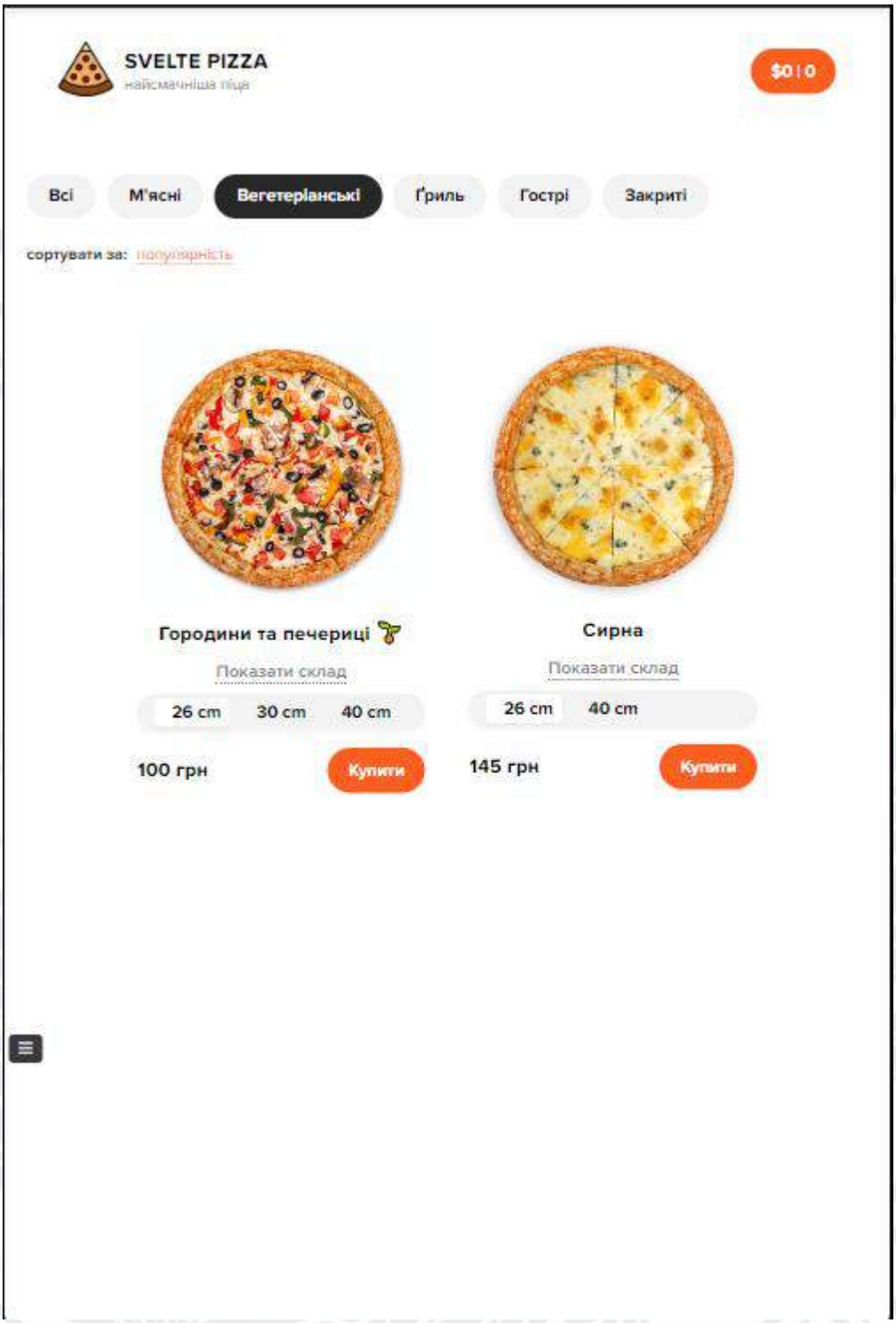

Рисунок 3.5 – Фільтрація категорія вегетаріанські

Через дію фільтрації за цією категорією кількість піц зменшилась, тепер їх усього дві

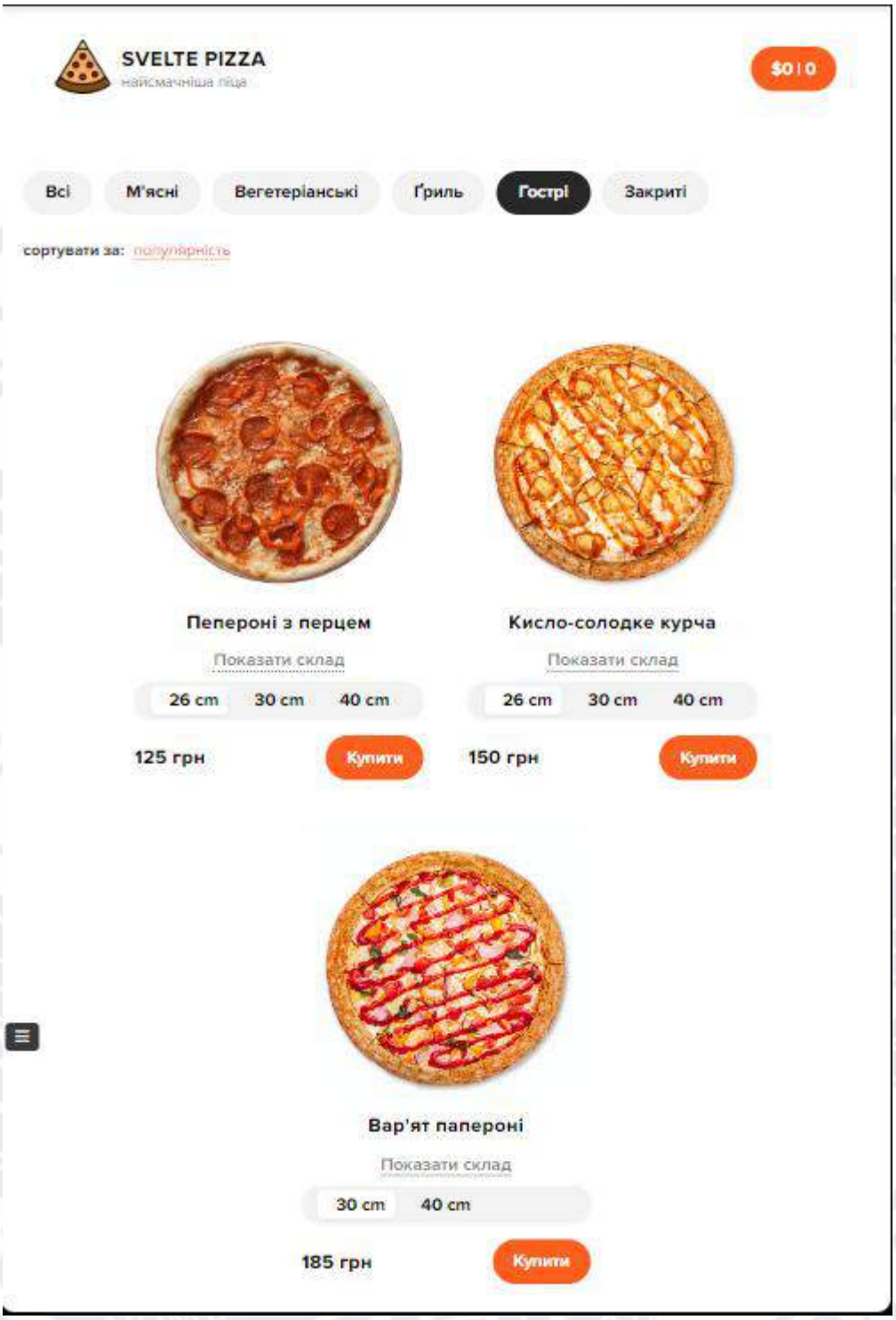

Рисунок 3.6 – Фільтрація категорії «гострі»

Знову змінивши категорію фільрації, також змінилась кількість піц, тепер їх усього три.

2. Сортування

- За замовчуванням встановлено на категорії «популярне» рис 3.1. Змінимо на категорію «ціна» рис. 3.7 та на категорію «алфавіт» рис. 3.8.

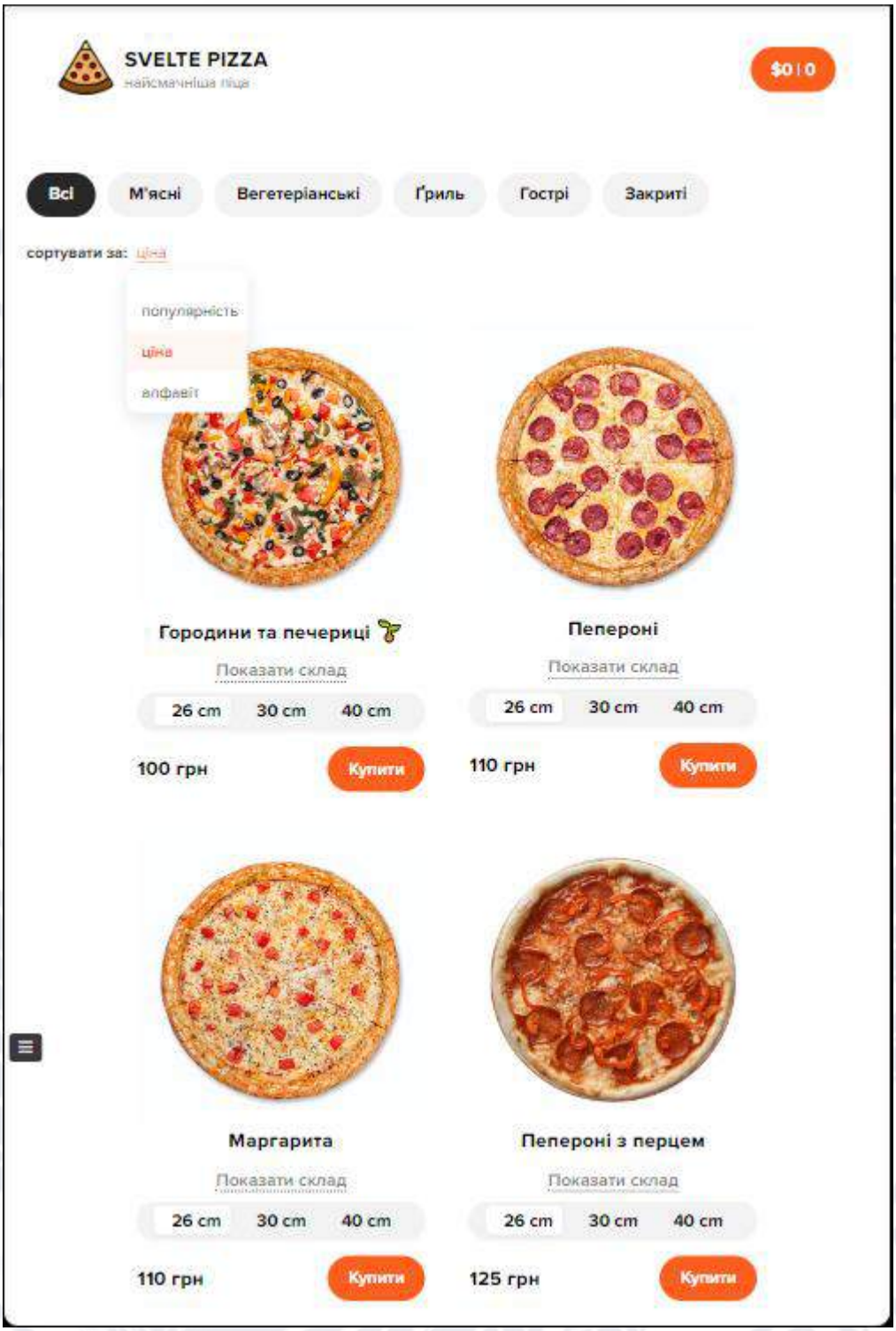

Рисунок 3.7 – Сортування категорії «ціна»

Вибравши категорію «ціна», усі наявні піци будуть відсортовані як на рисунку 3.7, де видно, що піци відсортувалися від найбільшої до найменшої ціни

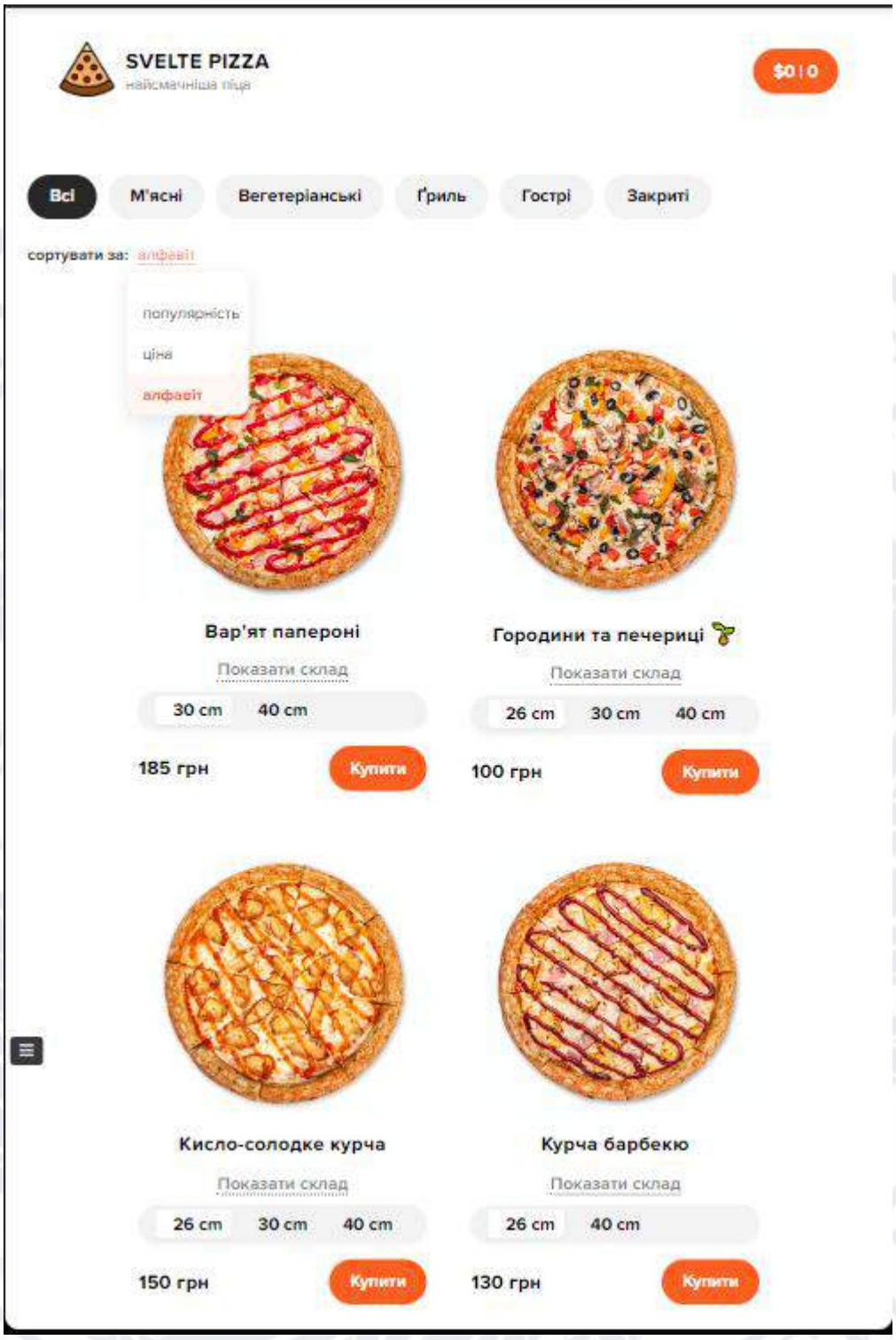

Рисунок 3.8 – Сортування за алфавітом

Вибравши категорію «алфавіт», усі наявні піци будуть відсортовані як на рисунку 3.8, де видно, що піци відсортувалися згідно з алфавітом

3. Зміна ціни та показ складу піци

- На рис. 3.9 склад піци згорнутий, а ціна встановлена на першій позиції розміру піци, тобто найменший розмір з наявних для кожної піци,

змінимо ці показники рис. 3.9

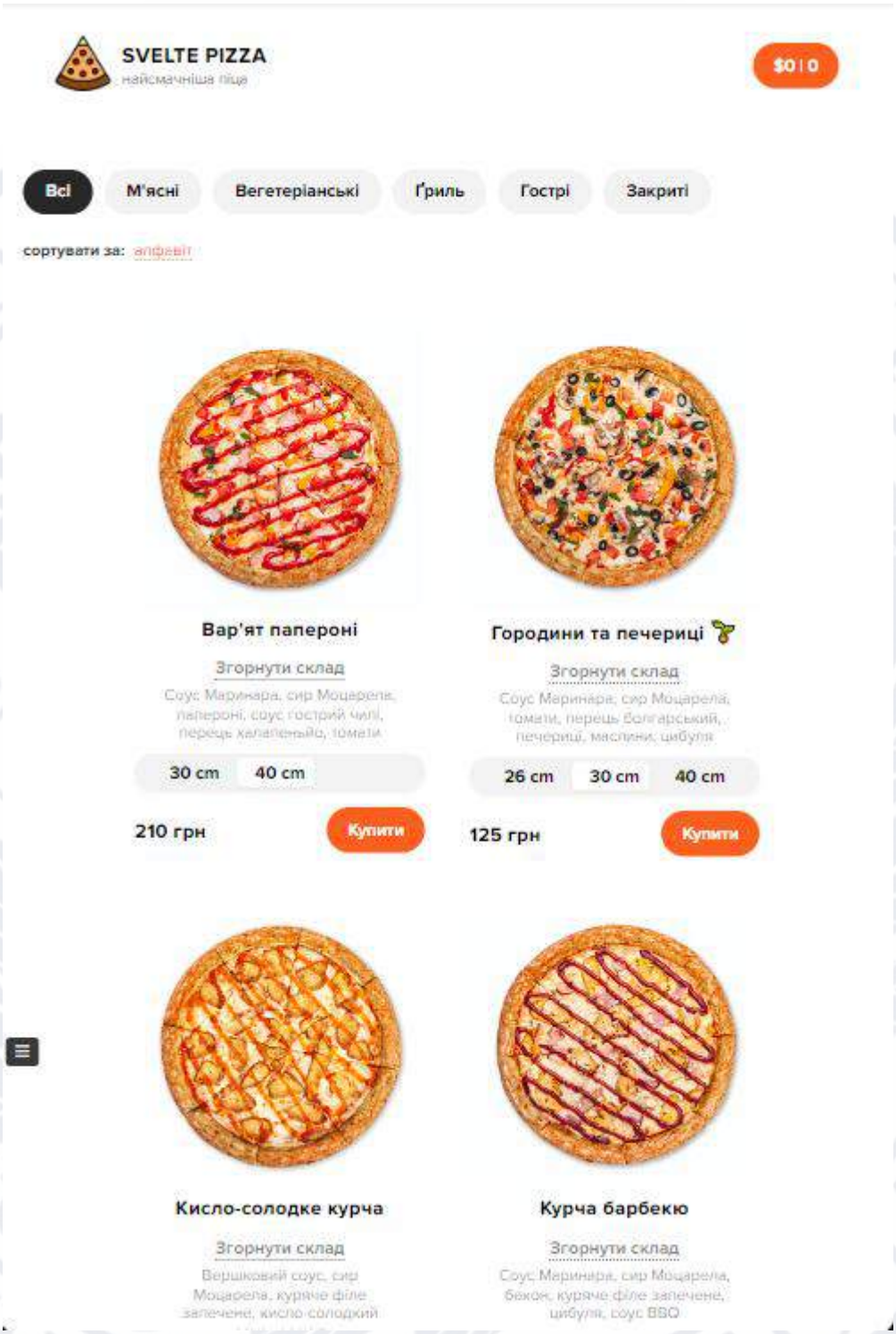

Рисунок 3.9 – Склад піци та змінена ціна

На рисунку 3.9 розгорнуто склад кожної піци, користувач може розгорнути, прочитати та згорнути. Склад кожної піци індивідуальний.

4. Контакти

- На усіх сторінках ліворуч присутня кнопка що плаває, натиснувши на

неї розгортається список можливих способів зв’язку рис. 3.10.

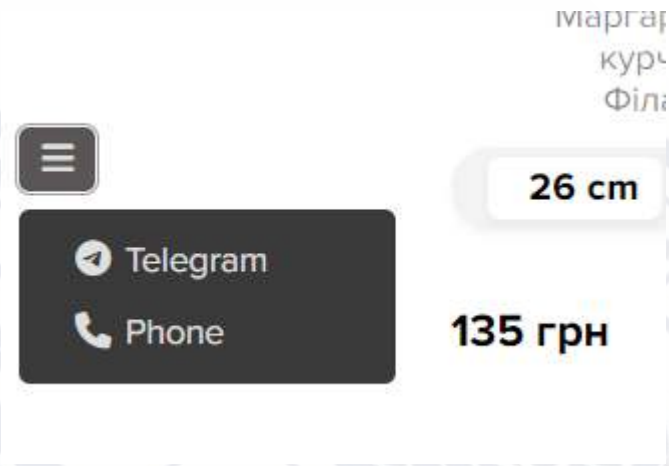

Рисунок 3.10 – Розгорнуте мені для контактної інформації

- Якщо натиснути на «Telegram» користувача перенесе до того каналу, бота чи конкретного користувача, це має вказувати сама піцерія, на цю мить відвідувача перенесе до особистих повідомлень автора вебкаталогу рис. 3.11. Так само і з «Phone» - користувача перенесе до контактів де автоматично запишеться номер телефону автора вебкаталогу, в реальності там може бути вказаний номер телефону для швидкого оформлення доставлення.

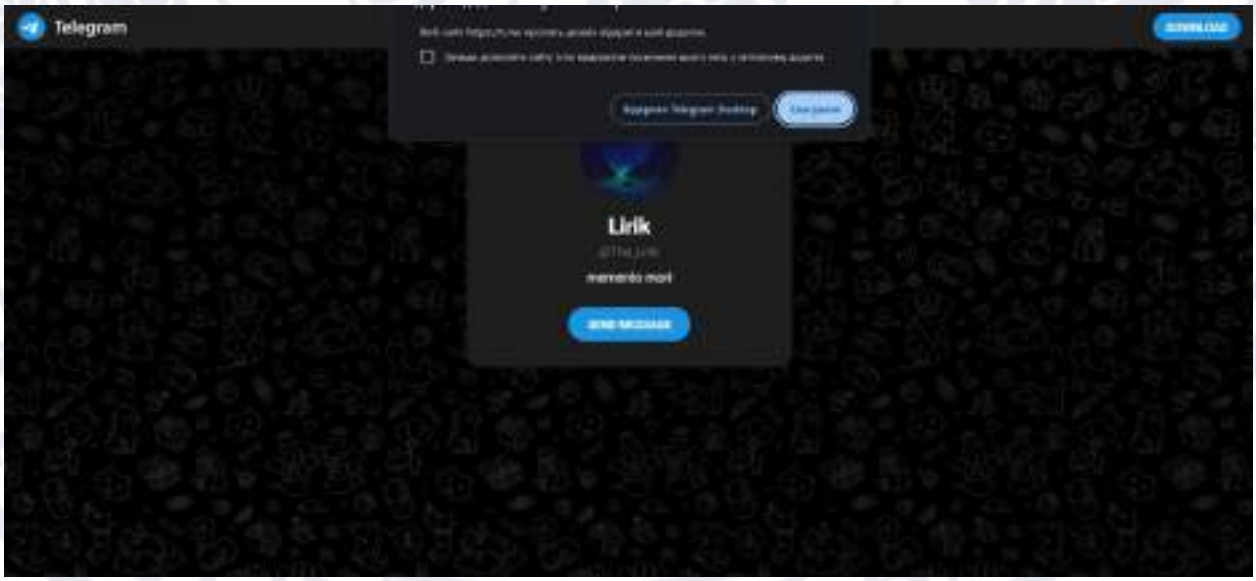

Рисунок 3.11 – Перенос користувача до вказаного каналу Telegram

На рисунку 3.11 видно, що натиснувши на іконку з Telegram, користувача перенесе автоматично до цього каналу.

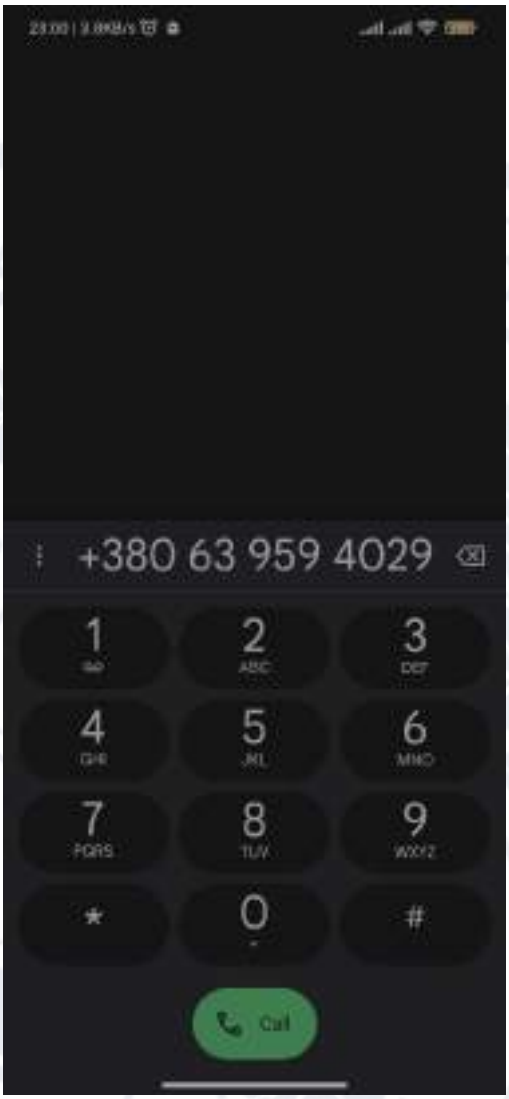

Рисунок 3.12 – Перенос користувача до вказаного номера телефону

На рисунку 3.12 видно, що схожим чином як і з Telegram, натиснувши на іконку телефону, користувача перенесе до відповідної програми з якої користувач може дзвонити.

5. Додавання товару в кошик

- Виберемо одну піцу і натиснемо кнопку «купити» рис. 3.13

Рисунок 3.13 – Кнопка «купити»

- Після натиснення «купити» в меню праворуч відобразиться змінена ціна рис. 3.14, за замовчуванням ціна дорівнює нулю.

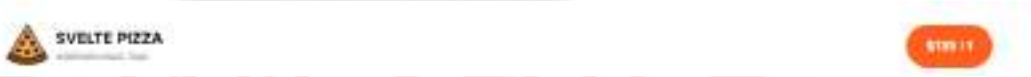

Рисунок 3.14 – Меню вебкаталогу після додавання піци в кошик

Тепер перейдемо до другої сторінки «кошик». Як видно на рисунку 3.4 окрім можливості повернутися до головної сторінки нічого нового для користувача немає, оскільки не було додано жодної піци. Розглянемо варіант події коли користувач додав піцу. Отже:

1. Оплатити замовлення

- На рисунку 3.3 ми бачимо форму для доставлення замовлення, тому краще розберемось як саме заповнювати форму. Форма містить сім рядків заповнення, перший рядок це ім’я, другий – номер телефону, в третьому треба вибрати місто де проживає користувач рис. 3.15, четвертий і п’ятий рядки – потрібно вибрати спосіб оплати рис. 3.16, останній рядок – користувач може написати повідомлення / побажання і натиснути «замовити»

Рисунок 3.15 – Список міст для форми

На рисунку 3.15 бачимо розгорнутий список міст з яких може здійснюватися доставлення, користувач повинен вибрати одне з цих міст.

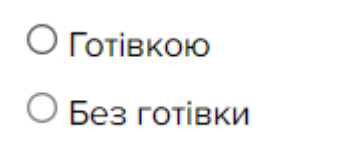

Рисунок 3.16 – Вибір способу оплати

Далі на рис. 3.15 користувач повинен вибрати спосіб оплати, доступно лише два способи: готівкою або без.

2. Збільшення або зменшення кількості піц

- Коли користувач додав хоча б одну піцу до кошика він може збільшити або зменшити кількість цих піц, натиснувши на плюс або мінус, буде показана нова кількість піц, а також оновлена ціна. На рис. 3.17 показана кількість піц до збільшення та їхня ціна, а на рис. 3.18 оновлені.

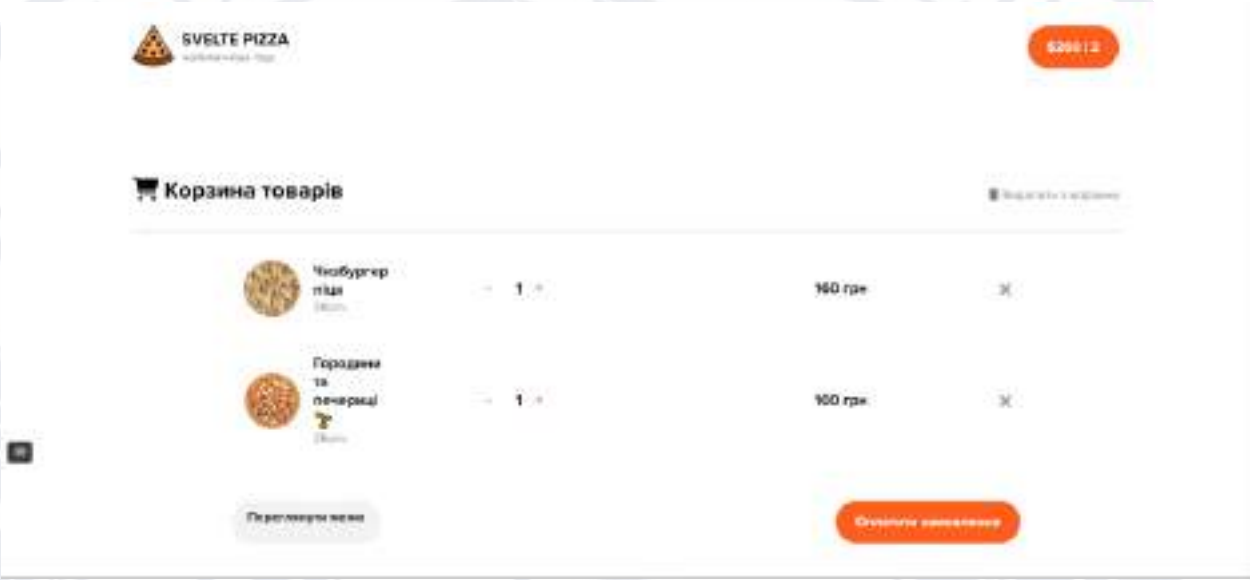

Рисунок 3.17 – Поточні кількість піц та ціни

Рис. 17 демонструє подію коли піци були додані в кошик, але їхня кількість ще не збільшена.

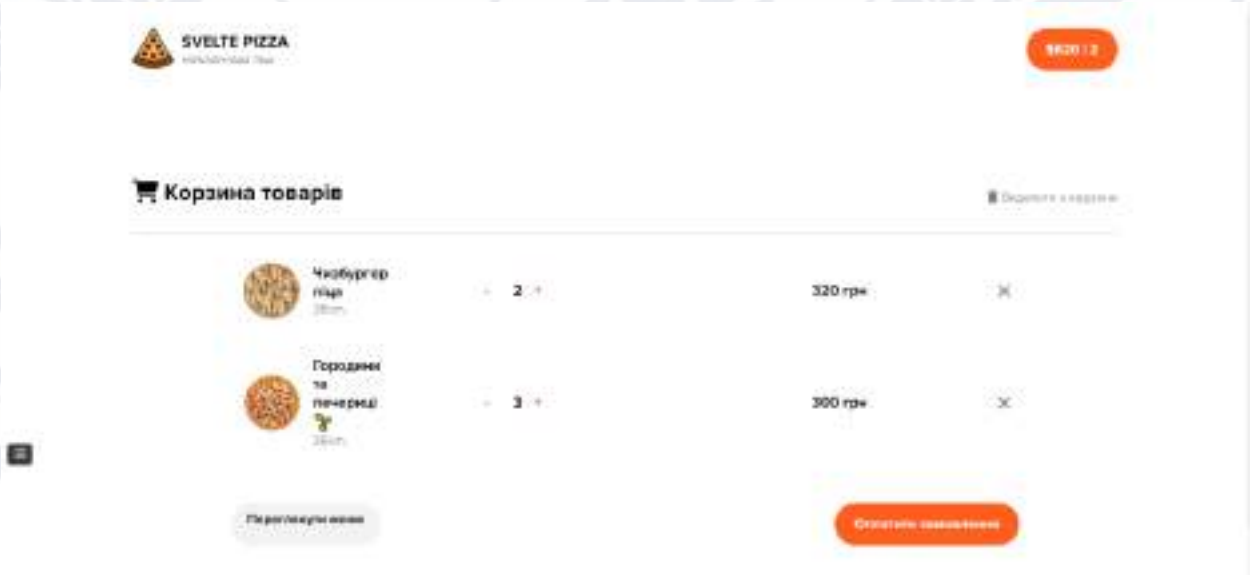

Рисунок 3.18 – Оновлені кількість та ціни

Рис. 3.18 демонструє випадок коли користувач, після додання піц в

кошик, вирішивши збільшити кількість піц, натиснувши іконку плюсика.

3. Поступове видалення

- Праворуч від ціни кожної піци є хрестик, натиснувши на який буде видалене одне замовлення рис. 3.19. Ми бачимо, що на рис. 3.18 було два замовлення, але тепер на рис. 19 залишилось лише одне.

Рисунок 3.19 – Видалення замовлення

На рисунку 3.19 показано випадок коли користувач видали одну піцу, натиснувши іконку хрестика.

4. Масове видалення

- Якщо замовлення зроблено або клієнт передумав брати вибрані піци він може видалити усі замовлення одним натисненням. Праворуч поточної сторінки над хрестиком є «видалити з кошика», після натиснення усі елементи в кошику зникнуть, тоді користувач побачить пусту сторінку кошика така як на рис. 3.4.
-

Висновок з розділу 3

У третьому розділі дипломної роботи було детально розглянуто зовнішній вигляд та функціональність розробленого вебкаталогу. Проведено ретельний аналіз усіх ключових сторінок та елементів інтерфейсу, підтверджено їхню працездатність і відповідність поставленим вимогам.

Особлива увага була приділена першій, головній сторінці каталогу. Було продемонстровано наявність інтуїтивно зрозумілих інструментів для фільтрації та сортування товарів, зміни ціни та перегляду складу піц. Також було описано зручні методи взаємодії з каталогом, такі як додавання товарів до кошика та можливість зв'язку з адміністраторами сервісу.

На сторінці «Кошик» було показано, як користувач може управляти своїми замовленнями – збільшувати/зменшувати кількість, видаляти окремі позиції або очищати кошик повністю. Крім того, було детально розглянуто процес оформлення замовлення, включаючи заповнення форми доставлення та вибір способу оплати.

Загалом, проведений аналіз дозволяє зробити висновок, що розроблений вебкаталог відповідає поставленим вимогам і забезпечує зручний, інтуїтивно зрозумілий та функціональний користувацький інтерфейс. Широкі можливості фільтрації, сортування та управління замовленнями надають користувачам повний контроль над процесом вибору та придбання товарів.

ВИСНОВКИ

У цій бакалаврській роботі було розроблено вебкаталог для замовлення піци. Робота складалася з трьох основних розділів, де детально розглянуто процес проєктування, програмної реалізації та практичного застосування розробленого рішення.

У першому розділі надано теоретичні відомості про предметну область, включаючи огляд технологій веброзробки, принципів проєктування користувацького інтерфейсу та архітектури вебсайтів. Ці знання утворили міцне підґрунтя для ефективної реалізації поставленого завдання.

У другому розділі було висвітлено ключові етапи програмної реалізації вебкаталогу з використанням фреймворку Svelte та препроцесора SCSS. Детально описано процес реалізації функцій фільтрації та сортування товарів, що є критично важливими для забезпечення зручної навігації та пошуку в каталозі. Окрім того, було розглянуто процес публікації вебсайту на платформі Netlify, проаналізовано інші популярні хостинг-провайдери та наведено покрокову інструкцію з розгортання проєкту.

У третьому розділі проведено докладний огляд розробленого вебкаталогу, включаючи аналіз зовнішнього вигляду та функціональності кожної сторінки. Було підтверджено, що вебкаталог забезпечує зручний користувацький інтерфейс, широкі можливості фільтрації, сортування та управління замовленнями. Крім того, демонструвалися процеси оформлення замовлення та способи зв’язку з адміністраторами сервісу.

Результати бакалаврської роботи підтверджують успішну реалізацію поставлених завдань. Розроблений вебкаталог є функціональним, зручним у використанні та відповідає вимогам. Застосування передових технологій та ретельне планування забезпечили створення якісного програмного продукту.

Отриманий досвід під час виконання цієї бакалаврської роботи, безсумнівно, буде корисним для подальшого професійного зростання у сфері веброзробки, а також стане надійною основою для майбутніх проєктів.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Види сайтів URL: https://apri-code.com/uk/razrabotka/vidy-sajtov/ (дата звернення: 19.04.2024).
2. Структура сайту URL: https://webtune.com.ua/statti/web-rozrobka/struktura-sajtu/ (дата звернення: 19.04.2024).
3. Речі на які варто звернути увагу при створені вебсайту URL: http://surl.li/stouo (дата звернення: 19.04.2024).
4. Переваги вебкаталогу URL: https://ampifire.com/blog/benefits-of-an-online-digital-product-catalog-in-e-commerce/ (дата звернення: 19.04.2024).
5. Node.JS URL: https://dan-it.com.ua/uk/blog/chto-jeto-takoe-node-js-prostymi-slovami/ (дата звернення: 19.04.2024).
6. SCSS URL: https://highload.today/uk/scss/ (дата звернення: 19.04.2024).
7. Svelte URL: https://wearecommunity.io/communities/EbmXcPOiYx/articles/1637 (дата звернення: 19.04.2024).
8. Команди Node.JS URL: https://robotdreams.cc/uk/blog/271-chto-takoe-npm-i-zachem-on-nuzhen (дата звернення: 19.04.2024).
9. Git URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-git/ (дата звернення: 19.04.2024).
10. GitHub URL: https://training.qatestlab.com/blog/technical-articles/what-is-github-and-how-to-work/ (дата звернення: 19.04.2024).
11. Команди Git: http://surl.li/smvfi (дата звернення: 19.04.2024).
12. Посібник зі Svelte: https://devzone.org.ua/post/posibnyk-zi-svelte (дата звернення: 21.04.2024).
13. Стаття про цикл життєдіяльності вебпродукту: http://surl.li/uaptt (дата звернення 29.05.2024)
14. Стаття про засоби веброзробки: http://surl.li/uapwz (дата звернення 29.05.2024)
15. Стаття про Modern tools and current trends in web-development:

http://surl.li/uaqel (дата звернення 29.05.2024)
16. DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte: https://www.doria.fi/handle/10024/177433 http://surl.li/uaqel (дата звернення 29.05.2024)
17. Svelte and Sapper in Action: http://surl.li/uassk (дата звернення 29.05.2024)
18. DataGen: JSON/XML Dataset Generator: http://surl.li/uaswl (дата звернення 29.05.2024)
19. Comparison of JavaScript Bundlers: http://surl.li/uaszv (дата звернення 29.05.2024)
20. Implementing Version Control With Git and GitHub as a Learning Objective in Statistics and Data Science Courses: https://www.tandfonline.com/doi/full/10.1080/10691898.2020.1848485 (дата звернення 29.05.2024)
21. Дослідження структури організації вебсайту при просуванні в мережі: http://surl.li/uatef (дата звернення 29.05.2024)
22. Json-server: https://www.npmjs.com/package/json-server (дата звернення 29.05.2024)
23. JAMStack Continuous Integration and Continuous Deployment with CircleCI and Netlify:https://www.theseus.fi/bitstream/handle/10024/342452/Hoang_Tri.pdf?sequence=2 (дата звернення 29.05.2024)
24. Навчальний сайт по rollup.cofig.js: https://rollupjs.org/tutorial/ (дата звернення 29.05.2024)
25. Вступ до Svelte: https://svelte.dev/docs/introduction (дата звернення 29.05.2024)
26. Getting started with Svelte: http://surl.li/uatxg (дата звернення 29.05.2024)
27. Introduction to Node.js: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (дата звернення 29.05.2024)
28. Node.js Get Started: https://www.w3schools.com/nodejs/nodejs_get_started.asp (дата звернення 29.05.2024)
29. Як використовувати .gitignore: https://git-scm.com/docs/gitignore (дата

звернення 29.05.2024)

30. Види сайтів – основного інструменту інтернет-маркетингу: https://www.economyandsociety.in.ua/index.php/journal/article/view/1006/964 (дата звернення 29.05.2024)
31. JSON Object Literals: https://www.w3schools.com/js/js_json_objects.asp (дата звернення 29.05.2024)
32. Sass Tutorial: https://www.w3schools.com/sass/default.php (дата звернення 29.05.2024)
33. Основні принципи будови сайту: https://www.taina.com.ua/osnovni-pryncypy-budovy-horoshoho-sajtu/ (дата звернення 29.05.2024)
34. Основні інформаційні джерела: https://dptnzspl101.ucoz.net/dokumenty/informacijni_tekhnologiji.pdf (дата звернення 29.05.2024)
35. Публікація сайту: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/Publishing_your_website (дата звернення 29.05.2024)
36. Аналіз вебкаталогів: https://webtune.com.ua/statti/internet-marketing/analiz-sajtiv-konkurentiv/ (дата звернення 29.05.2024)
37. Використання svelte-spa-router: https://github.com/ItalyPaleAle/svelte-spa-router/blob/main/Advanced%20Usage.md (дата звернення 29.05.2024)
38. Lesson for svelte-spa-router: https://svelte.dev/repl/6e7546119cff41fabe35df3ef999ff7e?version=4.2.17 (дата звернення 29.05.2024)
39. Що таке адаптив для сайту: https://hostiq.ua/blog/ukr/adaptive-design/ (дата звернення 29.05.2024)
40. CSS Media Queries: https://www.w3schools.com/css/css3_mediaqueries.asp (дата звернення 29.05.2024)

ДОДАТКИ

ДОДАТОК А

ЛІСТИНГИ ПРОГРАМИ

Код головного компонента App.svelte

```
<script>
  import FloatingComponent from "./FloatingComponent.svelte";
  import Header from "./Header.svelte";
  import Router from "svelte-spa-router";
  import routes from "./router/router";
  import { onMount } from "svelte";
  import { cart } from "./store/store.js";

  onMount(function () {
    let cartL = JSON.parse(localStorage.getItem("pizzaCart"));
    if (cartL) {
      cart.set(cartL);
    }
  });
</script>

<Header />
<FloatingComponent />
<Router {routes} />
```

Компонент головної сторінки MainPage.svelte

```
<script>
  import axios from "axios";
  import FiltrationList from "../FiltrationList.svelte";
  import SorterList from "../SorterList.svelte";
  import PizzaCart from "../PizzaCart.svelte";
  import { onMount } from "svelte";
```

```
import FloatingComponent from "../FloatingComponent.svelte";

    let url = "http://localhost:3000/pizzas";

    let arrayPizza = [];
    let arrayFilter = [
      {
        name: "Всі",
        value: "",
      },
      {
        name: "М'ясні",
        value: "0",
      },
      {
        name: "Вегетеріанські",
        value: "1",
      },
      {
        name: "Гриль",
        value: "2",
      },
      {
        name: "Гострі",
        value: "3",
      },
      {
        name: "Закриті",
        value: "4",
      },
```

```
];
        let selectFilter = arrayFilter[0].value;
        let selectSort = 0;
        $: updatePizzas(selectFilter, selectSort);

        function updatePizzas(filter, sort) {
          axios({
            method: "get",

            url: url,
          })
            .then((response) => {
              let pizzas = response.data;
              if (filter !== "") {
                pizzas = pizzas.filter((pizza) =>
                  pizza.types.includes(Number(filter)),
                );
              }
              switch (sort) {
                case 0:
                  pizzas.sort((a, b) => b.rating - a.rating);
                  break;
                case 1:
                  pizzas.sort((b, a) => b.price - a.price);
                  break;
                case 2:
                  pizzas.sort((a, b) => a.name.localeCompare(b.name));
                  break;
              }
              arrayPizza = pizzas;
```

```
})
            .catch((e) => {
                console.warn(e);
            });
    }
    function getSorter(e) {
        selectSort = e.detail.value;
    }
</script>

<section id="main">
    <div class="container filter">
        <FiltrationList arrayList={arrayFilter} bind:selectFilter />
        <SorterList on:select={getSorter} />
    </div>
    <div class="container">
        {#each arrayPizza as pizza (pizza.id)}
            <PizzaCart {...pizza} />
        {/each}
    </div>
</section>
```

Друга сторінка «кошик» CartPage.svelte

```
<script>
    import { link } from "svelte-spa-router";
    import CartComponent from "../CartComponent.svelte";
    import DeliveryForm from "../DeliveryForm.svelte";
    import { cart } from "../store/store.js";
    import { onDestroy } from "svelte";
```

```
function clearCart() {
    cart.set([]);
    localStorage.setItem("pizzaCart", JSON.stringify([]));
}

let cartList;
let showDeliveryForm = false; // нова змінна стану для відображення форми доставки

let unsub = cart.subscribe((value) => {
    cartList = value;
});

function updatePizzaCart(name, size, count) {
    if (count > 0) {
        let index = cartList.findIndex(
            (fn) => fn.name == name && fn.size == size,
        );

        if (index >= 0) {
            cartList[index].count = count;
            cart.set([...cartList]);
        }
    } else {
        let newArray = cartList.filter(
            (fn) => fn.name != name || fn.size != size,
        );
        cart.set(newArray);
    }
    localStorage.setItem("pizzaCart", JSON.stringify(cartList));
```

```
    }

    function openDeliveryForm() {
        showDeliveryForm = true; // відкрити форму доставки
    }

    function closeDeliveryForm() {
        showDeliveryForm = false; // закрити форму доставки
    }

    onDestroy(unsub);
</script>

<section id="cart">
    {#if cartList.length > 0}
        <div class="container list">
            <div class="header">
                <div class="title">
                    <i class="fa-solid fa-cart-shopping" />
                    Корзина товарів
                </div>
                <!-- svelte-ignore a11y-click-events-have-key-events -->
                <div class="delete-all" on:click={clearCart}>
                    <i class="fa-solid fa-trash" />
                    Видалити з корзини
                </div>
            </div>
            <div class="list">
                {#each cartList as pizza (pizza.name + pizza.size)}
                    <CartComponent
```

```
                name={pizza.name}
                size={pizza.size}
                startPrise={pizza.price}
                number={pizza.count}
                img={pizza.img}
                on:change={(e) => {
                  updatePizzaCart(
                    pizza.name,
                    pizza.size,
                    e.detail.count,
                  );
                }}
                on:delete={() => {
                  updatePizzaCart(pizza.name, pizza.size, 0);
                }}
              />
            {/each}
          </div>
          <div class="button">
            <a href="/" class="btn black" use:link>Переглянути меню</a>
            <button class="btn orange" on:click={openDeliveryForm}>
              Оплатити замовлення
            </button>
          </div>
        </div>
        {#if showDeliveryForm}
          <DeliveryForm on:close={closeDeliveryForm} />
          <!-- форма доставки -->
        {/if}
      {:else}
```

```
        <div class="container">
          <div class="title">Корзина пуста 💔</div>
          <div class="description">Ви не додали піцу</div>
          <div class="image">
            <img src="/img/shopping-cart-colour.svg" alt="Enpty-cart" />
          </div>
          <div class="button">
            <a href="/" class="btn black" use:link> Назад </a>
          </div>
        </div>
      {/if}
    </section>
```

Скрипт для перемальовування сторінок router.js

```
import MainPage from "../pages/MainPage.svelte"
import CartPage from "../pages/CartPage.svelte"
const routes = {
  '/': MainPage,
  '/cart': CartPage
}

export default routes;
```

База даних db.json

```
{
  "pizzas": [
    {
      "id": 0,
      "imageUrl": "https://express-
pizza.if.ua/media/1698704231_970.webp",
```

```
"name": "Пепероні з перцем",
            "types": [
              0,
              2,
              3
            ],
            "sizes": [
              26,
              30,
              40
            ],
            "price": 125,
            "category": [
              0
            ],
            "rating": 3,
            "sweet": [
              "Сир моцарелла",
              "пепероні",
              "орегано",
              "перець чилі",
              "соус томати пелаті"
            ]
          },
          {
            "id": 1,
            "imageUrl": "https://paparoni.com.ua/wp-
content/uploads/2022/02/20_4_%D1%81%D0%B8%D1%80%D0%B8.jpg",
            "name": "Сирна",
            "types": [
```

```
1       ],
            "sizes": [
              26,
              40
            ],
            "price": 145,
            "category": [
              1
            ],
            "rating": 6,
            "sweet": [
              "Вершковий соус",
              "сир Моцарела",
              "сир Чедер",
              "сир Гауда",
              "сир Дорблю"
            ]
          },
          {
            "id": 2,
            "imageUrl": "https://paparoni.com.ua/wp-
content/uploads/2021/12/05_%D0%91%D0%B0%D1%80%D0%B1%D0%B5%D
0%BA%D1%8E-1.jpg",
            "name": "Курча барбекю",
            "types": [
              0
            ],
            "sizes": [
              26,
              40
```

```
    ],
    "price": 130,
    "category": [
      0
    ],
    "rating": 4,
    "sweet": [
      "Соус Маринара",
      "сир Моцарела",
      "бекон",
      "куряче філе запечене",
      "цибуля",
      "соус BBQ"
    ]
  },
  {
    "id": 3,
    "imageUrl": "https://paparoni.com.ua/wp-
content/uploads/2022/01/17%D0%9A%D1%83%D1%80%D0%BE%D1%87%D0
%BA%D0%B0-%D0%BF%D0%BE-
%D0%BA%D0%B8%D1%82%D0%B0%D0%B9%D1%81%D1%8C%D0%BA%
D1%96.png",
    "name": "Кисло-солодке курча",
    "types": [
      0,
      3
    ],
    "sizes": [
      26,
      30,
```

```
      40
    ],
    "price": 150,
    "category": [
      0
    ],
    "rating": 2,
    "sweet": [
      "Вершковий соус",
      "сир Моцарела",
      "куряче філе запечене",
      "кисло-солодкий соус",
      "кунжут"
    ]
  },
  {
    "id": 4,
    "imageUrl": "https://express-
pizza.if.ua/media/1698704060_493.webp",
    "name": "Чизбургер піца",
    "types": [
      0,
      4
    ],
    "sizes": [
      26,
      30,
      40
    ],
    "price": 160,
```

```
"category": [ 3 ],
"rating": 8,
"sweet": [
"Сир моцарелла",
"Бекон",
"Помідори",
"Огірок маринований",
"Фарш яловичий маринований",
"Сир чеддер",
"Салат айсберг",
"Соус бургер",
"Соус Бешамель"
]
},
{
"id": 5,
"imageUrl": "https://paparoni.com.ua/wp-content/uploads/2021/12/07_%D0%BC%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D1%8C%D0%BA%D0%B0-1.jpg",
"name": "Вар'ят пароні",
"types": [
0,
2,
3
],
"sizes": [
30,
40
],
```

```
    "price": 185,
    "category": [
      2
    ],
    "rating": 2,
    "sweet": [
      "Соус Маринара",
      "сир Моцарела",
      "пепероні",
      "соус гострий чилі",
      "перець халапеньйо",
      "томати"
    ]
  },
  {
    "id": 6,
    "imageUrl": "https://paparoni.com.ua/wp-content/uploads/2021/12/06-%D0%9F%D0%B5%D0%BF%D0%B5%D1%80%D0%BE%D0%BD%D1%96-1.jpg",
    "name": "Пеперoні",
    "types": [
      0
    ],
    "sizes": [
      26,
      30,
      40
    ],
    "price": 110,
    "category": [
```

```
      0
    ],
    "rating": 9,
    "sweet": [
      "Соус Маринара",
      "сир Моцарела",
      "пепероні(свинина)"
    ]
  },
  {
    "id": 7,
    "imageUrl": "https://paparoni.com.ua/wp-
content/uploads/2021/12/15_%D0%9C%D0%B0%D1%80%D0%B3%D0%B0%D
1%80%D0%B8%D1%82%D0%B0-1.jpg",
    "name": "Маргарита",
    "types": [
      0
    ],
    "sizes": [
      26,
      30,
      40
    ],
    "price": 110,
    "category": [
      4
    ],
    "rating": 6,
    "sweet": [
      "Соус Маринара",
```

```
      "сир Моцарела",
      "томати",
      "італійські трави"
    ]
  },
  {
    "id": 8,
    "imageUrl": "https://paparoni.com.ua/wp-content/uploads/2023/05/27_%D0%A7%D0%BE%D1%82%D0%B8%D1%80%D0%B8_%D1%81%D0%B5%D0%B7%D0%BE%D0%BD%D0%B8.jpg",
    "name": "Чотири пори року",
    "types": [
      2,
      0
    ],
    "sizes": [
      26,
      30,
      40
    ],
    "price": 135,
    "category": [
      5
    ],
    "rating": 10,
    "sweet": [
      "Ця піца складається з чотирьох піц PAPARONI",
      "а саме: Маргарита",
      "Кисло-солодке курча",
      "Балик і Гриби і Філадельфія Блю-чіз"
```

```
    ]
  },
  {
    "id": 9,
    "imageUrl": "https://paparoni.com.ua/wp-content/uploads/2021/12/13_%D0%BE%D0%B2%D0%BE%D1%87%D0%B5%D0%B2%D0%B0-1.jpg",
    "name": "Городини та печериці 🌱",
    "types": [
      2,
      1
    ],
    "sizes": [
      26,
      30,
      40
    ],
    "price": 100,
    "category": [
      5
    ],
    "rating": 7,
    "sweet": [
      "Соус Маринара",
      "сир Моцарела",
      "томати",
      "перець болгарський",
      "печериці",
      "маслини",
      "цибуля"
```

```
      ]
    }
  ]}
```

Компонент меню Header.svelte

```
<script>
  import Button from "./Button.svelte";
  import { link } from "svelte-spa-router";
  import { cart } from "./store/store.js";

  let cartList;
  let unsub = cart.subscribe((value) => {
    cartList = value;
  });

  $: sumPrice = countSumPrice(cartList);
  function countSumPrice(array) {
    let price = 0;
    array.forEach((element) => {
      price += element.price * element.count;
    });

    return price;
  }

  // Для фіксації меню
  let isScrolled = false;

  window.addEventListener("scroll", () => {
    isScrolled = window.scrollY > 0;
  });
```

```
    $: headerClass = isScrolled ? "fixed" : "";
    $: copyHeaderClass = isScrolled ? "show" : "";

    $: headerStyle = isScrolled ? "height: 10;" : ""; // Зменшити висоту header при фіксації

    $: copyHeaderStyle = isScrolled ? "height: auto;" : "height: 10;"; // Показати copyheader при фіксації

    $: copyHeaderTop = isScrolled ? "top: 0;" : ""; // Встановити copyheader у верхню частину екрану при фіксації
  </script>

  <div
    id="copyheader"
    class={copyHeaderClass}
    style={copyHeaderStyle + copyHeaderTop}
  >
    <div class="container"></div>
  </div>
  <header id="header" class={headerClass} style={headerStyle}>
    <div class="container">
      <div class="logo">
        <div class="img">
          <a href="/"><img src="/img/logo.png" alt="logo.png" /></a>
        </div>
        <div class="text">
          <div class="name">Svelte pizza</div>
          <div class="description">найсмачніша піца</div>
        </div>
```

```
        </div>
        <div class="card-button">
          <a href="/cart" class="btn orange" use:link>
            <b>$</b>{sumPrice} | <span>{cartList.length}</span>
          </a>
        </div>
      </div>
</header>
```

Компонент для фільтрації FiltrationList.svelte

```
<script>
    export let arrayList = [];

    export let selectFilter;
</script>

<div class="filter-list">
    {#each arrayList as filter}
        <label>
            <input
                type="radio"
                name="filter"
                value={filter.value}
                bind:group={selectFilter}
            />
            <span class="btn black">{filter.name}</span>
        </label>
    {/each}
</div>
```

Компонент сортування SorteList.svelte

```
<script>
    import { createEventDispatcher } from "svelte";
    import { fade } from "svelte/transition";
    import { clickOutside } from "./directive/clickOutside.js";

    let dispatcher = createEventDispatcher();

    let sorterList = [
        {
            name: "популярність",
            val: 0,
        },
        {
            name: "ціна",
            val: 1,
        },
        {
            name: "алфавіт",
            val: 2,
        },
    ];

    let selectSort = 0;

    let openList = false;

    $: selectName = sorterList[selectSort].name;

    function sendSelectSort(index) {
```

```
        selectSort = index;

        dispatcher("select", {
            value: sorterList[index].val,
        });
    }
</script>

<div
    class="sorter-list"
    use:clickOutside
    on:click-outside={() => {
        openList = false;
    }}
>
    <div class="selected">
        <!-- svelte-ignore a11y-click-events-have-key-events -->
        сортувати за:
        <span
            class="name"
            on:click={() => {
                openList = !openList;
            }}>{selectName}</span
        >
    </div>

    {#if openList}
        <div class="list" transition:fade={{ duration: 200 }}>
            <ul>
                {#each sorterList as sort, index}
```

```
                <!-- svelte-ignore a11y-click-events-have-key-events -->
                <li
                    class:active={selectSort === index}
                    on:click={() => {
                      sendSelectSort(index);
                        // selectName = sort.name;
                    }}
                >
                    {sort.name}
                </li>
              {/each}
            </ul>
          </div>
      {/if}
</div>
```

Конфігураційний файл для налаштування збірки JavaScript-модулів

```
import { spawn } from 'child_process';
import svelte from 'rollup-plugin-svelte';
import commonjs from '@rollup/plugin-commonjs';
import terser from '@rollup/plugin-terser';
import resolve from '@rollup/plugin-node-resolve';
import livereload from 'rollup-plugin-livereload';
import css from 'rollup-plugin-css-only';
import scss from 'rollup-plugin-scss';

const production = !process.env.ROLLUP_WATCH;

function serve() {
```

```
let server;    function toExit() {
                  if (server) server.kill(0);
               }

               return {
                  writeBundle() {
                     if (server) return;
                     server = spawn('npm', ['run', 'start', '--', '--dev'], {
                        stdio: ['ignore', 'inherit', 'inherit'],
                        shell: true
                     });

                     process.on('SIGTERM', toExit);
                     process.on('exit', toExit);
                  }
               };
            }

            export default {
               input: 'src/main.js',
               output: {
                  sourcemap: true,
                  format: 'iife',
                  name: 'app',
                  file: 'public/build/bundle.js'
               },
               plugins: [
                  svelte({
                     compilerOptions: {
                        // enable run-time checks when not in production
```

```
            dev: !production
        }
    }),
    scss({
        include: ["/**/*.css", "/**/*.scss"], // Які файли буде білдити scss
        fileName: 'bundle.css', // В який файл треба білдити
        failOnError: true, // Якщо щось неправильно написано, видитись
не буде
        import: ("sass") // Використовуй препроцесор sass
    }),
    // we'll extract any component CSS out into
    // a separate file - better for performance
    css({ output: 'bundle.css' }),

    // If you have external dependencies installed from
    // npm, you'll most likely need these plugins. In
    // some cases you'll need additional configuration -
    // consult the documentation for details:
    // https://github.com/rollup/plugins/tree/master/packages/commonjs
    resolve({
        browser: true,
        dedupe: ['svelte'],
        exportConditions: ['svelte']
    }),
    commonjs(),

    // In dev mode, call `npm run start` once
    // the bundle has been generated
    !production && serve(),
```

```
    // Watch the `public` directory and refresh the
    // browser on changes when not in production
    !production && livereload('public'),

    // If we're building for production (npm run build
    // instead of npm run dev), minify
    production && terser()
  ],
  watch: {
    clearScreen: false
  }
};
```