

Факультет комп'ютерних наук та технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ**
ПРОДУКТОВОГО МАГАЗИНУ

Виконав: студент 4 курсу, групи КНТ-520
спеціальності 123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація) _____
«Комп'ютерна інженерія»

РЮМІН О. О.
(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.
(ПРИЗВИЩЕ та ініціали)

Рецензент КОЗІНА Г. Л.
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерна інженерія
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
 Завідувач кафедри КУДЕРМЕТОВ Р.К.
 “ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

РЮМІНА Олександра Олександровича
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи продуктового магазину

керівник проєкту (роботи) СКРУПСЬКИЙ Степан Юрійович, к. т. н., доцент
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проєкту (роботи) 24 травня 2024 року

3. Вихідні дані до проєкту (роботи) опис наявних технологій розробки,

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Вибір технологій для реалізації системи;

2) Проєктування бази даних системи;

3) Програмна розробка комп'ютерної системи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

UML діаграми послідовностей;

ER-модель бази даних комп'ютерної системи

Сторінки веб-сайту комп'ютерної системи продуктового магазину

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	СКРУПСЬКИЙ К. М., к.т.н., доцент		
нормоконтроль	ПОЛЬСЬКА О. В., ст. викладач		

7. Дата видачі завдання 25.04.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних технологій для реалізації системи	20.03.2024 р.	
2	Розробка структури системи	25.03.2024 р.	
3	Розробка алгоритму роботи системи	01.04.2024 р.	
4	Розробка бази даних системи	12.04.2024 р.	
5	Розробка серверної частини	24.04.2024 р.	
6	Розробка клієнтської частини	13.05.2024 р.	
7	Оформлення пояснювальної записки	22.05.2024 р.	
8	Оформлення графічного матеріалу	22.05.2024 р.	

Студент (ка)


(підпис)

Олександр РЮМІН
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Степан СКРУПСЬКИЙ
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 78 сторінок, 23 рисунка, 5 таблиць, 5 лістингів, 24 джерел і 3 додатка.

ER-МОДЕЛЬ EXPRESS, HTML, CSS, JAVASCRIPT, NODE, АТРИБУТ, БАЗА ДАНИХ, ВЕБСТОРИНКА, ВЕРСТКА, ВИПРОБУВАННЯ, КОМП'ЮТЕРНА СИСТЕМА, РОЗРОБКА,

Об'єкт розробки – клієнт-серверна вебсистема продажу товарів продуктового магазину через мережу Інтернет.

Мета роботи – розробка вебсистеми для розширення послуг продуктового магазину у мережі Інтернет.

У пояснювальній записці описаний процес створення комп'ютерної системи, який складається з вибору технологій для реалізації комп'ютерної системи, моделювання предметної області і розробка комп'ютерної системи.

Для створення комп'ютерної системи використовувалася HTML, CSS, JavaScript та технології “Node.js” і “Express.js”. Для створення бази даних використовувалися СКБД SQL і клієнт для СКБД Heidi SQL.

Розроблена комп'ютерна система реалізована на клієнт-серверній архітектурі і містить серверну і клієнтську частину та базу даних. Серверна частина складається з серверу, за допомогою якого клієнт може взаємодіяти з базою даних, оброблюючи запити користувачів. Клієнтська частина містить три вебсторінки, які реалізують зв'язок з серверною частиною.

ЗМІСТ

Вступ.....	7
1 Вибір технологій для реалізації системи	9
1.1 Вибір архітектури програмного забезпечення	9
1.1.1 Клієнт-серверна архітектура	9
1.1.2 Багатокомпонентна архітектура	11
1.1.3 Монолітна архітектура.....	12
1.1.4 Мікросервісна архітектура	13
1.2 Вибір технологій для клієнтської та серверної частини	14
1.2.1 JavaScript	16
1.2.2 Python.....	17
1.2.3 PHP	18
1.2.4 Ruby	19
1.3 Вибір фреймворків для мови програмування.....	21
1.3.1 Next.js.....	21
1.3.2 React.js	22
1.3.3 Node.js.....	23
1.3.4 Angular.js	24
1.3.5 Express.js.....	25
1.4 Вибір СКБД.....	25
1.5 Висновки з вибору технологій	29
2 Моделювання предметної області	29
2.1 Загальні вимоги до комп'ютерної системи	29
2.2 Вимоги до апаратного забезпечення системи	30
2.3 Створення бази даних	30
2.4 Діаграми послідовностей.....	35
3 Програмна розробка системи	37

3.1 Розробка серверної частини	37
3.2 Розробка клієнтської частини	42
3.3 Проєктування і дизайн користувацького інтерфейсу.....	43
3.4 Верстка та розробка функціонала сторінок.....	48
3.6 Випробування комп'ютерної системи	52
3.7 Розгортання і обслуговування комп'ютерної системи.....	57
Висновки	58
Перелік джерел посилання	60
Додаток А Коди для реалізації функціону на головній сторінці	63
Додаток Б Коди для реалізації оформлення замовлення	69
Додаток В Коди для реалізації функціоналу адміністратора	75

ВСТУП

Використання комп'ютерних технологій стрімко зростає у всіх аспектах нашого життя. Вони стають ключовими маркетинговими інструментами, які допомагають в просуванні бізнесу. Розробка сайтів, спеціально налаштованих під потреби конкретного бізнесу, є найбільш ефективним підходом. Успішна робота створеної комп'ютерної системи залежить від комплексу заходів та етапів його розробки.

Велика кількість бізнесів мають свою комп'ютерну систему, яка відображає дійсність компанії, її послуги чи товари. Це стосується великих корпорацій, середніх компаній та малих бізнесів, кожен з яких ставить свої цілі та вигляд власної комп'ютерної системи, орієнтуючись на потреби, аудиторію, власні можливості тощо. На даний момент, створення комп'ютерної системи для власників бізнесу є актуальним, оскільки це слугує збільшенню аудиторії, приваблювання нових клієнтів, самореклами в Інтернеті та збільшення ефективності продажу послуг чи товарів і, відповідно, прибутку, що може позитивно відзначитися на процвітанні бізнесу, створити умови для правильного конкурування між бізнесами у однаковій сфері послуг і дати змогу новому бізнесу заявити про себе.

Розробка комп'ютерної системи у вигляді вебсайту продуктового магазину не є виключенням. Адже, це важливий крок, що спрямований на покращення стану магазину на всіх рівнях, включаючи отримання всіх переваг наявності комп'ютерної системи.

Типовим для вебсайтів з надання послуг є те, що дана система складатиметься з того, хто отримує послуги від організації чи підприємства і ті, хто ці послуги надає. Тобто, є клієнт і обличчя, що відповідає за надання послуг. Оскільки, за допомогою сучасних технологій можлива автоматизація, то всю роботу здатен виконувати сервер.

У ролі клієнта може бути користувач комп'ютеру або іншого пристрою, який бачить “обличчя” вебсторінки, має змогу отримати інформацію про послуги або товари та має здатність користуватися функціоналом, використовуючи один з багатьох видів браузерів.

Сервер, у свою чергу, надає користувачу потрібну інформацію і за допомогою нього виконується надання послуг.

Отже, метою дипломного проєкту є створення комп'ютерної системи продуктового магазину, яка має розширити функціонал магазину для отримання переваг і здатності до конкуренції серед інших продуктових магазинів, таких як: «Сільпо», «Делві», «ЕкоМаркет» тощо та опис етапів проєктування.

Для цього передбачається розробка двох компонентів: клієнтської і серверної частин. Клієнтська частина являє собою вебсайт, тому для розробки виконуватиметься процес створення вебсайтів, що буде проходити від HTML (для скелету) і CSS (для стилізування) до мови програмування або сценаріїв (для розробки функціонала), тоді як серверна частина буде розроблена за допомогою мови програмування або сценаріїв.

Таким чином, пройшовши всі етапи проєктування, які зазначені у даній пояснювальній записці, буде отримано комп'ютерну систему продуктового магазину, яка буде виконувати основні задачі, що були поставлені.

1 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1 Вибір архітектури програмного забезпечення

На початку створення комп'ютерної системи потрібно визначити її архітектуру. Для цього потрібно проаналізувати наявні та обрати необхідну, яка буде підлаштована під виконання поставлених цілей та вихідних даних.

Сама архітектура програмного забезпечення це високорівневий огляд структури системи програмного забезпечення. Вона визначає ключові елементи системи, їх взаємозв'язки та взаємодії з метою задоволення функціональних та нефункціональних вимог до системи.

До основних принципів архітектури програмного забезпечення відносять модульність, чистоту інтерфейсів, розширюваність, переносність і високу зручність та ефективність. При правильному обранні архітектури, можна дозволити легко розширювати та модифікувати систему і забезпечувати високу якість та ефективність розробки.

При проектуванні програмного забезпечення часто використовуються шаблони архітектур, що являють собою загальні рішення для організації. До найпопулярніших шаблонів відносять:

- клієнт-серверну;
- багатокomпонентну;
- монолітну;
- мікросервісну [1].

1.1.1 Клієнт-серверна архітектура

Клієнт-серверна архітектура - це модель взаємодії між комп'ютерами у мережі, де один комп'ютер (клієнт) надсилає запити до іншого комп'ютера (сервера) з метою отримання ресурсів або послуг.

Клієнт – це комп'ютер або пристрій, який знаходиться на стороні користувача і надсилає запити до сервера з метою отримання інформації або

виконання певних дій. До головних функцій клієнта відносять: надання користувацького інтерфейсу, формулювання запиту до сервера і його відправку, а також отримання результатів запиту і відправку додаткових команд.

Сервер – це більш потужний комп'ютер або обладнання, яке призначене для вирішення таких задач: зберігання, доступ, захист і резервне копіювання даних, обробку клієнтських запитів та надсилання результату клієнту.

Основна модель такої системи полягає в тому, що клієнт надсилає запит на сервер, де він обробляється, і готовий результат відправляється клієнту. Сервер може обслуговувати кілька клієнтів одночасно, і якщо одночасно приходить більше одного запиту, вони встановлюються в чергу і виконуються сервером послідовно, іноді з пріоритетами або правилами.

Клієнт-серверна архітектура має дві версії побудови комп'ютерної системи:

1. слабкий клієнт – потужний сервер. У цьому варіанті всі обчислення відбуваються на сервері, і права доступу до нього для клієнта обмежені. Сервер просто відправляє готову відповідь, яка не потребує додаткової обробки. Клієнт взаємодіє з користувачем, надсилаючи та отримуючи запити та виводячи інформацію на екран;

2. слабкий сервер – потужний клієнт. У даному варіанті сервер виступає в основному як сховище даних, а вся обробка та відображення інформації здійснюються на комп'ютері клієнта.

До переваг даної архітектури відносять:

- модульність. Здатність розділити функціональність системи між клієнтом і сервером, що сприяє більшій організованості коду та полегшує розробку системи;
- масштабованість. Завдяки можливості обслуговувати кілька клієнтів одночасно є здатність масштабувати систему залежно від потреб;
- централізоване управління даними. Сервер може зберігати та керувати базою даних, що спрощує доступ до даних та забезпечує їх цілісність;
- висока безпека та надійність.

Серед недоліків є такі:

- точкова вразливість. Якщо сервер виявиться недоступним через збої або атаки, це може призвести до тимчасової недоступності всієї системи;
- залежність від мережі. При поганому з'єднанні можливе погіршення продуктивності або повне припинення роботи системи;
- складність підтримки. Підтримка серверної інфраструктури може бути складною, особливо для великих систем з великою кількістю клієнтів;
- обробка трафіку на клієнті. У деяких випадках, велика частина даних може оброблюватися на клієнтському пристрої, що може призвести до перевантаження або неефективного використання ресурсів [2].

1.1.2 Багатокомпонентна архітектура

Багатокомпонентна архітектура – це модель розробки програмного забезпечення, яка складається з багатьох компонентів, які взаємодіють між собою для досягнення загальної мети. Кожен компонент відповідає за виконання певних функцій або сервісів, і вони спільно працюють для забезпечення повного функціонала системи.

Багатокомпонентна архітектура передбачає розділення функціональності системи на окремі модулі або компоненти, що дозволяє розробникам працювати над ними незалежно один від одного. Кожен компонент може мати своє власне API (інтерфейс програмування застосунків), яке визначає, як інші компоненти можуть з ним взаємодіяти.

Основні характеристики багатокомпонентної архітектури:

- розподілена природа. Компоненти можуть бути розташовані на різних фізичних машинах або в різних мережевих середовищах, що дозволяє розділити навантаження та забезпечити більшу масштабованість системи;
- модульність. Систему можна розбити на модулі, що полегшує розробку, випробування та підтримку програмного забезпечення;

- компонентна комунікація. Компоненти взаємодіють між собою через визначені інтерфейси або протоколи, що дозволяє їм обмінюватися даними та послугами;
- централізоване управління даними. Централізовані сервери баз даних можуть зберігати та керувати даними компонентів системи;
- масштабованість. Здатність легко масштабувати систему, додаючи або видаляючи компоненти залежно від потреб.

Однак ця архітектура має недоліки, такі як:

- складність розгортання. Багатокомпонентні системи можуть бути складними для розгортання та конфігурації через більшу кількість компонентів і їх взаємозв'язків;
- залежність від мережі. При поганому з'єднанні можливе погіршення продуктивності або повне припинення роботи системи;
- складність підтримки;
- можливість точкової вразливості [3].

1.1.3 Монолітна архітектура

Монолітна архітектура – це тип архітектури програмного забезпечення, в якій весь функціонал системи розглядається як один компактний блок. У такій архітектурі всі компоненти програми, такі як інтерфейс користувача, бізнес-логіка та база даних, розглядаються як єдина сутність, і вони розгортані на одному сервері чи в одному виконавчому середовищі.

Основні характеристики монолітної архітектури:

- один виконавчий модуль. Весь код програми розгорнутий в одному виконавчому середовищі;
- монолітне розгортання. Весь додаток разом з його залежностями розгортається як єдина сутність, що робить процес розгортання простим і зручним.
- внутрішня зв'язність. Всі компоненти додатка взаємодіють безпосередньо один з одним, що спрощує розробку та випробування;

- спільність завдань та технологій. Розробка, випробування та підтримка монолітного додатка зазвичай відбувається в межах однієї команди або організації [4].

Переваги монолітної архітектури:

- простота розробки та розгортання. Із-за простої структури можливе полегшення розробки та розгортання;
- простота налагодження та випробування. Загальна кодова база спрощує налагодження та випробування програми;
- висока продуктивність. За рахунок внутрішньої зв'язності та відсутності мережевого зв'язку між компонентами, монолітні додатки зазвичай демонструють високу продуктивність.

Недоліки монолітної архітектури:

- обмежена масштабованість. Підвищення продуктивності може бути складним через потребу збільшення ресурсів на одному сервері;
- послаблене відокремлення компонентів. Велика та однорідна кодова база не дає можливості редагувати один компонент без зміни іншого;
- складнощі в розробці у великих проєктах. На великих проєктах розробка та підтримка монолітного додатка може стати складнішою через розмір кодової бази та залежності між компонентами [5].

1.1.4 Мікросервісна архітектура

Мікросервісна архітектура є сучасним підходом у розробці програмного забезпечення, який полягає в розбитті додатка на невеликі, незалежні компоненти, що називаються мікросервісами. Кожен мікросервіс відповідає за виконання конкретної функції або послуги, і може бути розгорнутий, масштабований та оновлюваний незалежно від інших компонентів системи.

Основний принцип мікросервісної архітектури полягає в тому, щоб розділити складний додаток на менші, легко керовані частини, що дозволяє розробку, випробовувати та розгортку окремо, і, тим самим, дозволяє командам розробників працювати незалежно одна від одної. Кожен сервіс може бути

написаний мовою програмування або технологією, яка найкраще відповідає його конкретним потребам.

Переваги мікросервісної архітектури:

- гнучкість та швидкість розробки. Кожен мікросервіс може бути розроблений незалежно, що дозволяє зменшити час розробки та впровадження нових функцій;
- легкість масштабування. Кожен мікросервіс може бути масштабований окремо в залежності від навантаження, що дозволяє оптимізувати використання ресурсів;
- висока доступність. При відмові одного мікросервіса, інші можуть продовжувати працювати, що забезпечує високу доступність системи в цілому.

Недоліки мікросервісної архітектури включають:

- складність управління. Із-за великої кількості мікросервісів складно вести моніторинг, логування та керування конфігураціями;
- збільшення складності взаємодії. Підтримування великої кількості мікросервісів може призвести до ускладнення комунікації та взаємодії між ними;
- вартість. Розгортання та підтримка багатьох мікросервісів може бути вартісною, особливо для невеликих проєктів [6].

1.2 Вибір технологій для клієнтської та серверної частини

Оскільки клієнтська частина являє собою вебсайт, то для його розробки передбачено використання HTML, CSS (frontend) і однієї з мов програмування або сценаріїв (backend). Серверна частина складається лише з мов програмування, оскільки не передбачає наявності користувацького інтерфейсу.

HTML або HyperText Markup Language (мова розмітки гіпертексту) – це код, який використовується для структуризації та наповнення змістом вебсторінки.

Для цього використовують різні елементи, такі як блоки, абзаци, зображення, посилання, марковані та немарковані списки, таблиці тощо. Також HTML може поєднувати інші файли конфігурації та програми, що були написані мовами сценаріїв.

Елементи HTML є будівельними блоками сторінок HTML і створюються за допомогою тегів, які відповідають за кожний елемент. Теги складаються з параметрів (налаштування) і контенту. За допомогою різних параметрів можна видозмінювати елементи або їх наповнення, наприклад, змінити розмір елементу чи змінити колір тексту, його шрифт тощо. Однак для цих цілей доцільно використовувати CSS.

CSS або Cascading Style Sheets (каскадні таблиці стилів) – це мова, призначена для опису стилів та вигляду документа, який написаний у мовах розмітки, таких як HTML або XML, включаючи діалекти XML, такі як SVG, MathML або XHTML. CSS визначає, як елементи та вебсторінка в цілому повинні виглядати у різних умовах чи на екранах різного розміру.

CSS є однією з основних мов для веброзробки та є стандартизованою для веббраузера відповідно до специфікацій W3C. Раніше розробка CSS включала в себе синхронне випускання різних частин специфікації, що призводило до версій CSS, таких як CSS1, CSS2.1 тощо. Проте з моменту CSS 2.1 обсяг специфікації значно збільшився, що призвело до змін у підходах до випуску рекомендацій [7].

Дана мова, окрім стилізації елементів, також дозволяє використовувати анімацію, гнучкість у розміщенні елементів на сторінці, а також можливість створення адаптивних та реактивних вебдодатків. Окрім того, CSS постійно розвивається і покращується, додаючи нові можливості та специфікації, які можуть додати новий функціонал чи спростити використання наявного.

Для вибору мови програмування або сценаріїв можна провести аналіз популярних представників. До них відносяться JavaScript, Python, PHP та Ruby.

1.2.1 JavaScript

JavaScript – це високорівнева, інтерпретована мова програмування, яка використовується для створення динамічних вебсайтів та вебдодатків. Основним призначенням JavaScript є забезпечення взаємодії користувача з вебсторінкою, а також створення складних функцій та обробка даних на стороні клієнта. Javascript широко використовується завдяки своїй високій функціональності, продуктивності та об'єктно-орієнтованій поведінці.

Найчастіше JavaScript використовуються в таких галузях:

- веброзробка. Через його зручності можливе створення динамічного контенту на вебсторінці, такого як анімація, валідація форм, зміна стилів, робота з DOM (Document Object Model) та інше;
- вебдодатки. Широко застосовується для створення клієнт-серверних додатків за допомогою використання різних фреймворків та бібліотек;
- мобільна розробка. Також може бути використаний для мобільних застосунків, використовуючи необхідні для цього фреймворки та бібліотеки;
- ігрова розробка. Може бути використаний для створення мобільних та браузерних ігор.

Дана мова сценаріїв виділяється такими особливостями, як прототипне наслідування, асинхронність та динамічна типізація. Прототипне наслідування доволі цікава технологія, що дає змогу створювати прототипи методів об'єктів, що зменшує та спрощує написання та читання. Асинхронність дозволяє виконувати операції без блокування інших. Динамічна типізація означає те, що тип даних змінюється на потрібний під час виконання програми.

До переваг JavaScript відносять:

- досить широке застосування. JavaScript може використовуватися і на клієнтській, і на серверній стороні розробки, що робить його універсальним інструментом;
- легкість вивчення. Вивчення JavaScript відносно просте, особливо для початківців;

- динамічність. JavaScript здатна створювати динамічне наповнення вебсторінки;
- велике співтовариство. Для даної мови сценаріїв існує багато посібників та корисної літератури, створених для початківців, що зроблено вже існуючими розробниками, для підтримки та наставляння попередніх;
- швидкодія. На виконання потрібно мало часу, оскільки є інтерпретована мова.

Однак, дана мова сценарії має такі недоліки:

- безпека використання. JavaScript виконується на клієнтському пристрої, що може створювати проблеми з безпекою, такі як XSS (Cross-Site Scripting) атаки;
- різна поведінка браузерів. Різні браузери інтерпретують JavaScript по різному, наприклад, деякі браузери не підтримують певні динамічні об'єкти і це призводить до некоректного відображення усього вебсайту;
- відсутність підтримки налагодження. У разі виникнення помилки її складно виявити із-за відсутніх параметрів налагодження [8].

1.2.2 Python

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Вона широко використовується в різних галузях, таких як:

- веброзробка. Може бути використана для створення вебдодатків та вебсайтів за допомогою необхідних фреймворків;
- аналіз даних. За допомогою потрібних математичних бібліотек можливе легке проведення аналізу, обробки та візуалізації даних;
- штучний інтелект. Python є одним з найкращих мов програмування для створення штучного інтелекту, які призначені на розпізнавання мови, об'єктів, зображень;

- наукові обчислення. Окрім аналізу даних, також є можливість проводити складні розрахунки в різних наукових галузях, таких як хімія, фізика, вища математика тощо;

- робототехніка. Часто використовується для завдання логіки мікропроцесорам або мікроконтролерам для створення різних видів роботів.

З особливостей є динамічна типізація та інтерпретованість. Динамічна типізація працює за аналогією з JavaScript, а інтерпретованість означає те, що Python розуміє інші мови програмування такі як C і C++ і поєднувати їх зі своїм синтаксисом.

До основних переваг відносять такі:

- простота вивчення. Синтаксис Python легкий для розуміння, що робить його ідеальним вибором для початківців у програмуванні;
- широкий спектр бібліотек. Існує велика кількість бібліотек під різні задачі різного спектра використання;
- кросплатформенність. Є підтримка на різних операційних системах, таких як Windows, macOS та Linux;
- велике співтовариство. Через те, що як із JavaScript, Python має активне співтовариство розробників, які надають підтримку, допомогу та велику кількість документації для початківців.

Недоліки використання Python:

- відносна повільність. Python може бути повільним у порівнянні з іншими мовами програмування, такими як C++ або Java, особливо в області обробки великих обсягів даних;
- погана підтримка мобільної розробки. Не найкращим варіантом буде використання Python для розробки мобільних застосунків із-за скудності бібліотек і фреймворків у цьому напрямку [9].

1.2.3 PHP

PHP є серверною мовою програмування, призначеною переважно для розробки вебсайтів та вебдодатків. Дана мова програмування має відкритий

вихідний код. Це означає, що різні програмісти можуть доповнювати та покращувати мову, тому цей проєкт постійно розвивається і на сьогоднішній день має сім версій. PHP широко використовується для реалізації функціональності на вебсторінці, таких як обробка форм, робота з базами даних, створення сеансів користувачів та багато іншого.

Як було сказано раніше, найчастіше PHP використовують для вебсайтів та вебдодатків і як інструмент розробки інших застосунків чи проєктів помічений не був. Однак, PHP є одним з найкращих мов програмування серверної частини, яка взаємодіє з базою даних, оскільки має вбудовану підтримку баз даних. Також, мова програмування доволі гнучка для використання, що означає те, що її можна використовувати у всіх вебпроєктах від невеличких до складних та об'ємних.

Серед переваг PHP відносять:

- простота вивчення. Оскільки, PHP має схожий синтаксис до C-подібних мов, то вивчення цієї мови виявиться простіше та швидше;
- широке застосування. Дана мова використовується у більшості вебпроєктів. За статистикою, кожний шостий вебпроєкт створений на PHP;
- швидка розробка. PHP дозволяє швидко розробляти вебпроєкти за рахунок великої кількості готових рішень та великої спільноти розробників.

До недоліків можна віднести такі:

- безпека. Якщо готовий до використання код буде неправильно написано, то є вірогідність стати вразливим до атак, таких як SQL-ін'єкції та видозміна коду;
- швидкодія. PHP менш продуктивна зі сторони швидкодії ніж його конкуренти: JavaScript, C++ [10].

1.2.4 Ruby

Ruby – це мова програмування, яка зазвичай використовується для розробки вебдодатків та програмного забезпечення. Найчастіше можна побачити іншу його назву – Ruby on Rails, що є найпопулярнішим фреймворком для Ruby. Так як і

RНР, Ruby on Rails є мовою програмування з відкритим вихідним кодом, що має той же план розвитку, який також залежить і від інших програмістів-ентузіастів.

Дана мова має свої особливості, які вирізняють її серед інших мов програмування. Серед них є:

- динамічна типізація. Ця особливість дозволяє змінювати типи даних об'єктів під час виконання програми;
- метапрограмування. Ruby має потужну систему метапрограмування, що дозволяє розробникам динамічно змінювати структуру та поведінку програми під час її виконання;
- концентрація на принципах програмування. Є підтримка різних парадигм програмування, таких як об'єктно-орієнтоване, функціональне програмування та мета програмування [11].

Застосування Ruby on Rails має такі переваги:

- простий синтаксис. Ruby має простий синтаксис, що дозволяє зосередитися на розробці проєкту, а не на вивченні мови;
- розширюваність. Дана перевага впливає з особливості мови, а саме її здатності до метапрограмування.

До недоліків використання Ruby відносять:

- продуктивність. У порівнянні з іншими мовами програмування, Ruby може бути менш продуктивним з точки зору швидкодії виконання;
- споживання ресурсів. Ruby-програми вимагають більше ресурсів комп'ютера, таких як пам'ять та потужність процесора, ніж іншим мовам програмування;
- документація. Деякі бібліотеки та фреймворки для Ruby можуть мати неповну чи неточну документацію [12].

1.3 Вибір фреймворків для мови програмування

Після переліку і подальшого вибору мови програмування або сценарію можна приділити увагу на фреймворки.

Фреймворк – це набір інструментів, правил та бібліотек, який визначає структуру, яка буде притаманна для проєкту. Визначає, як компоненти застосунку повинні взаємодіяти між собою, задає правила для створення інтерфейсів, керування базою даних тощо.

В основному, фреймворки потрібні для спрощення та прискорення розробки, шляхом надання програмісту готових рішень для певних задач. Також, важливим аргументом для використання фреймворків є стандартизація. Використання стандартизації дозволяє полегшити роботу над проєктом для багатьох розробників шляхом стандартизації підходів до розробки. Окрім того, фреймворками користуються багато людей, і тим самим, є багато порад, документації, допомоги в розв'язанні проблеми чи задачі [13].

Передбачувано, що мовою розробки буде JavaScript, тому можна розглянути його найпопулярніші фреймворки. До таких відносяться Next.js, React.js, Node.js, Angular.js і Express.js.

1.3.1 Next.js

Next.js на сьогоднішній день є найпотужнішим фреймворком для мови програмування JavaScript. Цей фреймворк містить собою покращену версію React.js.

Next.js це здебільшого розробка для спрощення створення клієнтської частини проєкту, однак його можна використовувати і для серверної частини, оскільки він поєднує в собі також Node.js [14].

До основних можливостей Next.js відносять:

- Рендеринг на стороні сервера та генерація статичних сайтів. Завдяки ініціалізації серверного прошарку на Node.js, Next.js відмінно справляється з

передбачуваним рендерингом сторінок на стороні сервера, що дозволяє прискорити завантаження вебсторінки та покращити SEO (Search Engine Optimization/ Оптимізація пошукових систем) за рахунок передачі HTML-наповнення відразу до браузера. За допомогою цього покращується користувацький досвід та видимість у пошукових системах;

- автоматичне розподілення коду. Фреймворк автоматично розбиває код на менші, керовані частини. Це дозволяє завантажувати лише необхідні компоненти, скорочуючи початковий час завантаження та підвищуючи продуктивність сторінки;

- вбудована підтримка CSS. Next.js легко інтегрується з різними бібліотеками CSS-in-JS, що здатне спростити стилізування вебсторінки та може звільнити від необхідності використання додаткових налаштувань та інструментів;

- гаряча заміна модулів (HMR). HMR дозволяє розробникам бачити зміни в реальному часі без необхідності оновлювати сторінку, що робить процес розробки більш ефективним і приємним.

На цей момент Next.js є найперспективнішим і одним із кращих виборів для застосування у різних вебпроектах [15].

1.3.2 React.js

React.js – це фреймворк для JavaScript, що має мету покращити та зробити легше і зручніше написання клієнтської частини, а саме інтерфейсу користувача. Даний фреймворк, здебільшого, використовується для створення односторінкових вебсайтів.

Із основних особливостей React.js відзначають:

- компонентний підхід. React.js базується на концепції компонентів, які є складовими вебінтерфейсу. Компоненти можуть містити як HTML-код, так і JavaScript-логіку;

- віртуальний DOM. React.js використовує віртуальний DOM (Document Object Model/ Об'єктна модель документа), яка є відображенням реального DOM.

Із-за цього React.js може виконувати оптимізацію та оновлення інтерфейсу користувача без необхідності у взаємодії з реальним DOM;

- використання JSX (JavaScript XML). JSX – це розширення JavaScript, що може дозволити використовувати схожий на XML синтаксис для описування компонентів. Це дозволить краще розуміти, редагувати, доповнювати, взаємодіяти з кодом, що полегшує розробку;

- комбінування компонентів. Вебсторінку можна розбити на компоненти і комбінувати їх між собою для створення інтерфейсу користувача;

Використання даного фреймворку може покращити досвід користувача, надавши йому швидкий та ефективний інтерфейс, та розробника, спростивши йому розробку, при цьому, роблячи код чистіше і зрозуміліше. Також, не слід забувати про те, що React.js існує давно, у порівнянні з Nest.js, тому існує багато документації та порад від досвідних програмістів, що може допомогти в розумінні архітектури та вирішенні задач чи помилок.

З недоліків використання React.js і Next.js, так як останній є покращеною версією попереднього, визначають його складність вивчення, особливо для початківців, та потреба в наявності повної бібліотеки чи інструментів для забезпечення повного функціонала [16].

1.3.3 Node.js

Node.js є фреймворком JavaScript, що орієнтований на спрощення розробки серверної частини проєкту. Він використовується там, де потрібно оброблювати запити від клієнта та надсилати результати назад, до клієнта.

Особливостями Node.js є:

- асинхронність. Використання неблокуючого вводу/виводу, що дозволяє виконувати багато операцій без очікування попередніх операцій;
- незалежність від платформи. Є підтримка на різних операційних системах;

- повноцінний вебсервер. Node.js може використовувати інші фреймворки, наприклад Express.js, для створення повноцінної серверної частини проєкту;
- модульність. Мається на увазі розподіл коду на модулі, що дозволяє спростити роботу групі розробників;
- велика кількість бібліотек, що призначені для взаємодії з базою даних, роботи з мережею та аутентифікацією.

Якщо підбити підсумки, то використання цього фреймворку забезпечить швидкодію та продуктивність при обробці запитів на сервер, швидшу та простішу розробку серверної частини проєкту.

Однак, управління залежностями та версіями модулів доволі складне і при роботі з великими проєктами виникають складнощі з масштабуванням і обслуговуванням [17].

1.3.4 Angular.js

Angular.js є ще одним фреймворком, який націлений на спрощення розробки клієнтської частини, однак, він застосовується на мові програмування TypeScript або HTML. Як приклад, React.js може застосовуватися і на JavaScript (здебільшого), і на TypeScript або HTML.

Хоча Angular.js і React.js схожі за своєю метою один на одного, однак між ними є свої відмінності. Серед них є:

- архітектура. React.js базується на віртуальному DOM (Document Object Model), тоді як Angular.js має свою власну реалізацію DOM і використовує Hierarchical Dependency Injection для керування залежностями;
- компоненти. React.js використовує функціональні та класові компоненти, а Angular.js використовує лише класові компоненти;
- реактивність. Angular.js побудований на принципах реактивного програмування та спостереження, що краще підходить для роботи з асинхронними операціями. У React.js для роботи з асинхронним кодом можуть використовуватися бібліотеки, такі як Redux або RxJS;

- роутинг. У React.js для реалізації маршрутизації часто використовується бібліотека React Router, тоді як Angular.js має вбудований механізм маршрутизації;

- складність вивчення. React.js легший для вивчення, особливо для тих, хто вже знайомий з JavaScript. Angular.js, у свою чергу, складніше для вивчення, особливо для початківців [18].

1.3.5 Express.js

Express.js - це фреймворк-додаток для Node.js, який допомагає забезпечити простий та ефективний спосіб роботи з HTTP-запитами та відповідями, реалізацією маршрутизації, обробку запитів тощо.

До його властивостей відносять:

- простота. Його дуже просто використовувати у проєкті, він не вимагає детального вивчення, а базові можливості легкі до розуміння;
- middleware. Фреймворк може застосувати функцію middleware, за допомогою якої здійснюється обробка HTTP-запити та відповіді. При цьому є підтримка функції middleware від інших розробників;
- підтримка шаблонів для генерації вебсторінок;
- можливість бути розширеним за допомогою інших пакетів, які доступні через npm (менеджер пакетів Node.js) [19].

1.4 Вибір СКБД

При створенні бази даних, наступним етапом є вирішення питання про керування базою даних. При обранні системи керування базами даних (СКБД) потрібно врахувати зручність, багатофункціональність, простоту керування базами даних.

Одними з найпопулярніших СКБД є Oracle, MySQL, PostgreSQL, Microsoft Access. Для обрання СКБД потрібно порівняти між собою ці системи, знайти недоліки та переваги. Розглянемо кожну систему окремо.

Oracle DB – масштабована система управління базами даних з розділеною логічною та фізичною архітектурою. Вона підтримує ґрид-обчислення та забезпечує прозорість розташування даних для користувачів. Oracle DB є масштабованою та економічно ефективною для великих корпоративних баз даних. Вона пропонує ефективне кешування пам'яті, розбиття таблиць та потужні засоби резервного копіювання.

Найбільшою перевагою Oracle DB є те, що вона є більш масштабованою, ніж SQL, що може зробити її більш економічно ефективною в корпоративних екземплярах.

Також до переваг можна додати:

- ефективне кешування пам'яті для забезпечення оптимальної продуктивності дуже великих баз даних;
- високопродуктивне розбиття для розділення великих таблиць даних на кілька частин;
- наявність кількох методів резервного копіювання та відновлення, включаючи потужний інструмент Recovery Manager (RMAN).

З недоліків є:

- висока вартість на ліцензію;
- складність використання та адміністрування;
- вимога до потужного апаратного забезпечення та значних обсягів оперативної пам'яті [20].

База даних MySQL – це клієнт-серверна система, яка складається з багатопотокового SQL-сервера, який підтримує різні серверні частини, кілька різних клієнтських програм і бібліотек, інструменти адміністрування та широкий спектр інтерфейсів прикладного програмування.

Основні переваги MySQL:

- MySQL має відкритий код і, тим самим, його можна отримати та розповсюджувати безплатно;
- простий та зручний у використанні та зрозумілий SQL-синтаксис;
- висока продуктивність, тобто швидка та ефективна обробка запитів.

З недоліків є:

- MySQL може бути менш функціональним у порівнянні з деякими іншими базами даних, такими як Oracle DB або Microsoft SQL Server;
- є проблеми з некоректною обробкою певних запитів або управління пам'яттю, що може призвести до збоїв або втрати даних [21].

PostgreSQL – це передова система реляційних баз даних корпоративного класу з відкритим кодом. PostgreSQL підтримує як SQL (реляційні), так і JSON (нереляційні) запити.

Основні переваги PostgreSQL:

- надає широкий набір функцій та розширень, таких як географічні об'єкти, JSON-обробка, та вбудовану підтримку мов програмування, що дозволяє розробникам створювати складні та потужні додатки;
- має вбудовану підтримку транзакцій та відновлення після збоїв, що гарантує цілісність даних та високу доступність системи;
- надає широкі можливості налаштування, що дозволяє оптимізувати його під конкретні вимоги, проєкт та ресурси сервера.

Недоліки PostgreSQL:

- у порівнянні з деякими іншими системами управління базами даних, PostgreSQL може вимагати більшої експертизи та знань для ефективного використання, особливо для новачків;
- може мати обмежену кількість доступних засобів підтримки та консультування;
- деякі функції та опції можуть вимагати більших ресурсів апаратного забезпечення, таких як обсяг оперативної пам'яті та місце на диску [22].

Microsoft Access – СУБД від Microsoft, яка поєднує реляційну систему баз даних Access із графічним інтерфейсом користувача та інструментами розробки програмного забезпечення.

До переваг відносять:

- легкість та зручність використання СКБД завдяки зрозумілому інтерфейсу;
- Microsoft Access входить в пакет Microsoft Office, що робить його більш доступним з фінансової точки зору порівняно з деякими іншими системами управління базами даних;
- швидкість та ефективність розробки для невеликих проєктів.

До недоліків відносять:

- не підходить до великих проєктів;
- знижується продуктивність при роботі з великим обсягом даних [23].

Для зручності використання СКБД можна використовувати Heidi SQL. Heidi SQL – це безкоштовний клієнт для СКБД, який надає зручний графічний інтерфейс для управління базами даних. Даний інструмент підтримує багато різних типів баз даних, включаючи MySQL, PostgreSQL, Microsoft SQL Server та інші.

До переваг Heidi SQL можна віднести:

- легкість установки та використання;
- підтримка різних типів СКБД;
- можливість виконання різноманітних запитів та адміністративних операцій через інтуїтивний інтерфейс.

Незважаючи на це, є деякі недоліки, такі як обмежені функціональність та можливості, порівняно з іншими клієнтами, для баз даних. Однак, для невеликих або середніх проєктів, де зручність і простота важливіше, Heidi SQL може бути чудовим вибором [24].

1.5 Висновки з вибору технологій

Підбиваючи підсумки, постає необхідність в обранні архітектури та технологій для реалізації різних аспектів проєкту.

Проаналізувавши доступні архітектури програмного забезпечення, було обрано трирівневу (клієнт, сервер, база даних) клієнт-серверну архітектуру.

Для створення інтерфейсу користувача було обрано мови: HTTP, CSS та JavaScript. Для створення серверної частини було обрано мову JavaScript з використанням фреймворку Node.js з додатковим залученням Express.js.

Для розробки бази даних було обрано СКБД MySQL та клієнт для роботи з СКБД – HeidiSQL.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні вимоги до комп'ютерної системи

Оскільки метою дипломної роботи є розробка комп'ютерної системи продуктового магазину, то вона має містити функціонал і для клієнтів (користувачів) і адміністрації (менеджерів). Враховуючи це, можна виділити такі головні функції для клієнта і адміністратора:

Для клієнта:

- відображення змісту асортименту продуктового магазину, що має бути поділений на такі категорії: бакалія, молочні вироби, випічка, солодоці, м'ясо і риба, фрукти та овочі;
- можливість пошуку потрібного продукту за його ім'ям;
- здатність додавати продукт до кошика у потрібній кількості;

- перегляд і корегування вмісту кошика: видалення продукту або зміна його кількості;
- оформлення замовлення у вигляді форми;
- видача чека про успішно прийняте замовлення.

Для адміністрації:

- перегляд наявних записів у базі даних;
- додавання нових товарів до бази даних;
- зміна вже наявних записів у базі даних;
- вилучення записів з бази даних.

Окрім зазначеного, адміністрація має можливість користуватися функціями клієнта, а клієнт не має можливостей менеджера.

Таким чином, даний функціонал для обох ролей задовольнить потреби продуктового магазину.

2.2 Вимоги до апаратного забезпечення системи

Для виконання запланованої роботи комп'ютерної системи, у клієнта мають бути комп'ютер або інший мобільний пристрій (телефон, планшет тощо) з мінімально допустимими характеристиками, при яких можливий запуск браузера, та екран. Також, потрібне стабільне підключення до мережі Інтернет.

2.3 Створення бази даних

Як було зазначено у попередньому підрозділі, база даних є важливою складовою розроблюваної комп'ютерної системи. База даних потрібна для

зберігання у собі всієї необхідної інформації, що потрібна клієнту та адміністратору. За допомогою запитів, сервер взаємодіє з нею згідно з запитами та діями користувачів. Дана база даних містить у собі такі таблиці: “products”, “departments”, “orders”, “type_delivery”, “address_selfpickup”.

Розробка бази даних першочергово мала на меті сформувати структуру та зміст даних, що дозволило забезпечити основу для подальшої взаємодії серверної та клієнтської частин системи. Це рішення обґрунтоване необхідністю створення записів, з якими сервер та клієнт зможуть ефективно оперувати, забезпечуючи узгодженість та цілісність даних на всіх етапах розробки.

Таблиця “products” містить у собі всю інформацію про товари, а саме його торгову назву, відділ, назва файлу зображення, опис, кількість на складі та ціну за одиницю товару. Детальніше наведено у таблиці 2.1.

Таблиця 2.1 – Відношення “products”

Атрибут	Тип даних	Допустимі значення	Значення за умовчанням	Обов'язковість	Примітки
id_product	Лічильник	Натуральні числа	-	Так	Первинний ключ
name_of_product	Текстовий	Кирилиця/ Латиниця	-	Так	-
id_department	Числовий	Натуральні числа	-	Так	Зовнішній ключ
pictures	Текстовий	Латиниця	-	Так	Назва файлу зображення
description	Текстовий	Кирилиця/ Латиниця	-	Так	-
quantity	Числовий	Натуральні числа	-	Так	-
price	Числовий	Подвійне з плаваючою точкою	-	Так	

Таблиця “departments” зберігає назви типів відділів для таблиці “products”. Відношення “departments” наведено у таблиці 2.2.

Таблиця 2.2 – Відношення “departments”

Атрибут	Тип даних	Допустимі значення	Значення за умовчанням	Обов'язковість	Примітки
id_department	Лічильник	Натуральні числа	-	Так	Первинний ключ
name_of_department	Текстовий	Кирилиця	-	Так	-

Таблиця orders зберігає записи зроблених замовлень, що були отримані шляхом заповнення форми і даних кошику, а саме дані форми: ім'я клієнта, номер телефону, тип доставки, адрес доставки, адрес відділення для самовивозу, коментар та час доставки; дані кошику: продукт, його кількість та ціна замовлення. Відношення “orders” наведено у таблиці 2.3.

Таблиця 2.3 – Відношення “orders”

Атрибут	Тип даних	Допустимі значення	Значення за умовчанням	Обов'язковість	Примітки
id_order	Лічильник	Натуральні числа	-	Так	Первинний ключ
id_product	Числовий	Натуральні числа	-	Так	Зовнішній ключ
quantity	Числовий	Натуральні числа	-	Так	-
name_of_client	Текстовий	Кирилиця	-	Так	-
number_of_phone	Текстовий	Натуральні числа/ Латиниця	-	Так	-

Продовження таблиці 2.3.

id_type_delivery	Числовий	Натуральні числа	-	Так	Зовнішній ключ
address_delivery	Текстовий	Латиниця/ Натуральні числа	Null	Ні	
id_address_selfPickup	Числовий	Натуральні числа	Null	Ні	Зовнішній ключ
comment	Текстовий	Кирилиця/ Латиниця	Null	Ні	
time_delivery	Текстовий	Натуральні числа/ Латиниця	0	Ні	
order_price	Числовий	Подвійне з плаваючою точкою	-	Так	

Таблиця “type_delivery” зберігає назви типів доставок продуктів для таблиці “orders”. Відношення “type_delivery” наведено у таблиці 2.4.

Таблиця 2.4 – Відношення “type_delivery”

Атрибут	Тип даних	Допустимі значення	Значення за умовчанням	Обов'язковість	Примітки
id_type_delivery	Лічильник	Натуральні числа	-	Так	Первинний ключ
type_delivery	Текстовий	Кирилиця	-	Так	-

Таблиця “address_selfpickup” зберігає адреси ділянок, у яких можна забрати замовлення. Ця таблиця є додатковою до таблиці “orders” і наведена у таблиці 2.5.

Таблиця 2.5 – Відношення “address_selfpickup”

Атрибут	Тип даних	Допустимі значення	Значення за умовчанням	Обов'язковість	Примітки
id_address_selfPickup	Лічильник	Натуральні числа	-	Так	Первинний ключ
address_selfPickup	Текстовий	Кирилиця/ Натуральні числа	-	Так	-

Об'єднавши всі таблиці можна отримати базу даних продуктового магазину. ER-модель бази даних зображена на рисунку 2.1.

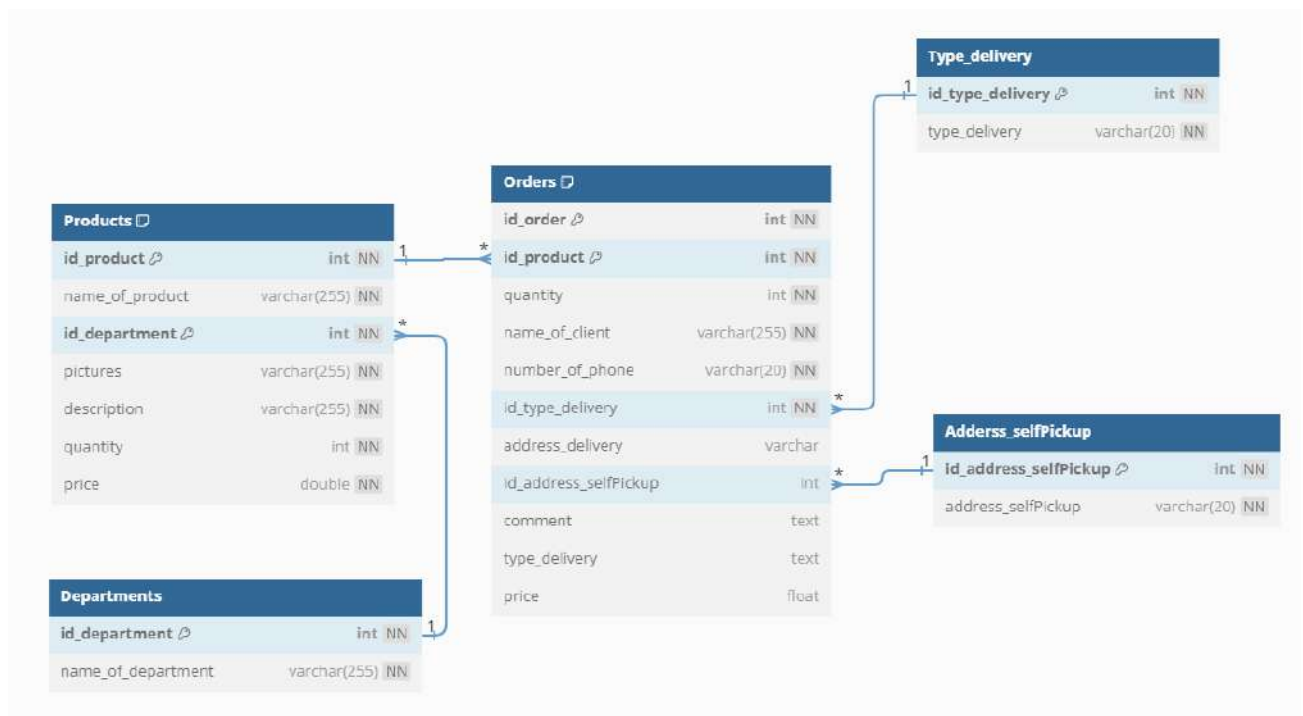


Рисунок 2.1 –ER- модель бази даних продуктового магазину

Після створення схеми бази даних продуктового магазину, її було перенесено у середовище HeidiSQL для подальшої роботи над комп'ютерною системою.

2.4 Діаграми послідовностей

Діаграма послідовностей відображає взаємодію між структурними об'єктами будь-якої системи чи проєкту, а саме відображає порядок виконуваних дій, які відбуваються під час виконання певних функцій. У нашому випадку, діаграма послідовностей відображає взаємодію між складовими комп'ютерної системи продуктового магазину з клієнт-серверною архітектурою. Серед об'єктів діаграми є клієнт або адміністратор, користувацький інтерфейс у вигляді браузера, сервер з використанням фреймворку Node.js і база даних, що створена за допомогою СКБД MySQL.

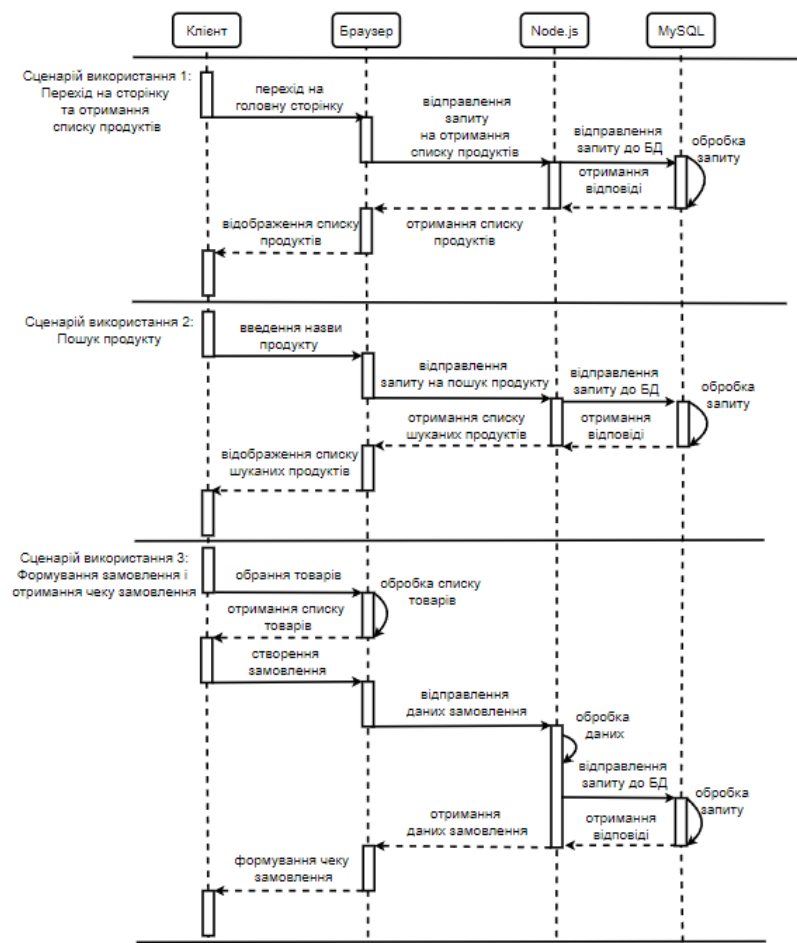


Рисунок 2.2 – Діаграма послідовностей для клієнта

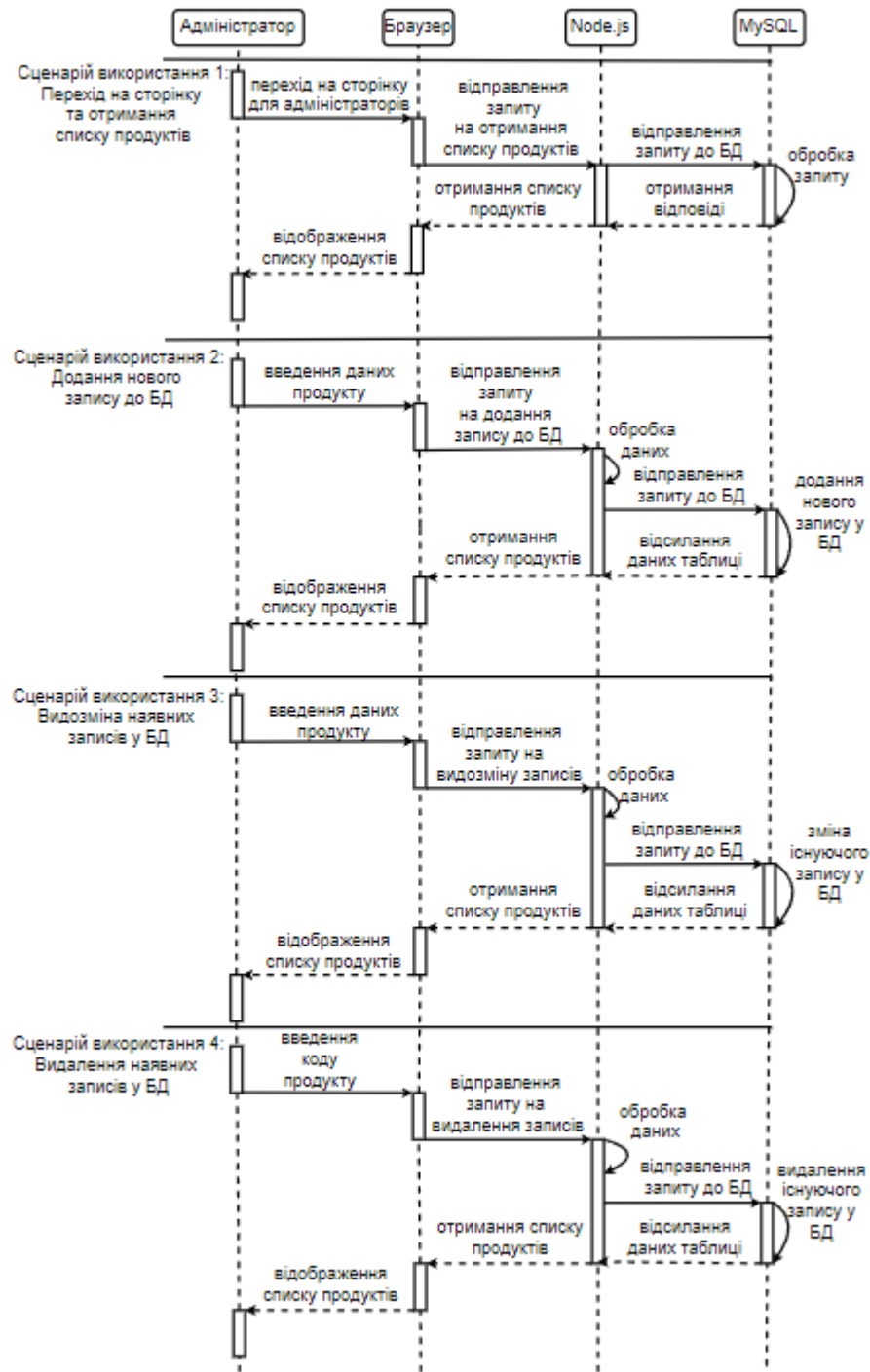


Рисунок 2.3 – Діаграма послідовностей для адміністратора

На рисунках 2.2 і 2.3 зображені діаграми послідовностей для обох користувачів: клієнта і адміністратора відповідно. Дані діаграми послідовностей поділені на сценарії використання користувачами. Діаграма, що знаходиться на рисунку 2.2 поділена на три сценарії: перехід на сторінку і отримання списку

продуктів, пошук продуктів й формування замовлення та отримання чека замовлення. Діаграма, що знаходиться на рисунку 2.3, має чотири сценарії використання: перехід на сторінку та отримання списку продуктів, додання нового запису, видозмінення та видалення наявних записів у базі даних.

Таким чином, представлені на рисунках 2.2 і 2.3 діаграми послідовностей формують уявлення про взаємодію між об'єктами комп'ютерної системи продуктового магазину та порядок дій при виконаннях основних функцій.

3 ПРОГРАМНА РОЗРОБКА СИСТЕМИ

3.1 Розробка серверної частини

Як було сказано раніше, серверна частина розроблюється за допомогою javascript з використанням фреймворків “Node.js” і “Express.js”. Для того, щоб використовувати фреймворки потрібно виконати такі дії:

1. завантажити “Node.js”;
2. створити директорію та ініціювати новий проєкт “Node.js”. Це можна зробити через термінал Windows, вказавши команди, що вказані у лістингу 3.1;
3. завантажити бібліотеку “Express.js”, ввівши цю команду:
“npm install express”.

Після встановлення та налаштування необхідних фреймворків можна переходити до написання серверної частини.

Основним є створення вебсервера на основі “Express”, який підключається до бази даних MySQL і надання дозволу на CORS запити, тобто надання доступу певним доменам надсилати запити до інтерфейсу програмного застосунку (API). Дані дії наведені у лістингу 3.2.

Лістинг 3.1 – Створення директорії та ініціювання “Node.js”

```
mk-dir new_project
cd new_project
npm init
```

Лістинг 3.2 – Створення вебсервера

```
import express, { response } from "express";
import mysql from "mysql2/promise";
import cors from "cors";

const app = express();
const port = 1598;

var connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'root',
  database: 'mylittleshop'
});

app.use(cors());
```

Наступним є створення обробників для запитів на отримання результатів SQL запитів з бази даних на відповідні дії користувачів. Дані обробники запитів створюються за допомогою функції “use()” з бібліотеки “express”.

Для того, щоб задовольнити потреби комп’ютерної системи, вона має містити такі обробники подій:

- отримання списку продуктів, відсортованих за типом відділу;
- пошук продуктів за його ім’ям;
- додавання продукту до кошика за його ідентифікатором;
- отримання ідентифікатора останнього запису замовлення;
- отримання інформації запису замовлення за його ідентифікатором;
- додавання нового запису замовлення до бази даних;
- отримання списку продуктів, відсортованих за ідентифікатором;
- додавання запису нового продукту до бази даних;

- отримання запису продукту за його ідентифікатором;
- редагування запису наявного продукту у базі даних;
- видалення запису наявного продукту у базі даних.

Приклад функції “use()” зображений у лістингу 3.3.

Лістинг 3.3 – Шаблон функції “use()”

```
const app = express();
app.use(
  '/Шлях за яким буде доступний маршрут',
  async (req, res) => {
    // обробник запиту,
    // де req-це об'єкт запиту,
    // а res – обробник відповіді
    const {список аргументів, що надсилаються серверу} =
req.query;
    (await connection).query(`SQL запит до бази даних`)
    .then (async (response) =>{
      res.send(response[0]); // надсилання відповіді
      (результату запиту)
    })
    .catch(async error => {console.log(error);} // обробник помилки
  )
);
```

Більшість обробників запитів подібні до шаблону і відрізняються шляхом маршруту, списком аргументів, що передаються та SQL запитом. Однак, серед них визначається обробник подій для додавання нового запису замовлення до бази даних (Лістинг 3.4 і 3.5).

Лістинг 3.4 – Код обробник подій для додавання нового запису замовлення до бази даних, при умові, що товар один

```
app.use(
  '/addOrder',
  async (req, res) => {
    var {name, phone_number, delivery_type, address,
id_selfPickup, comment, content, time, price} = req.query;
    if(delivery_type == 1){
      id_selfPickup = "NULL";
    } else if (delivery_type == 2){
      address = "NULL";
    }
    const parsedContent = JSON.parse(content);
```

```

const productIds = parsedContent.map(item => item.id);
const amounts = parsedContent.map(item => item.quantity);
var orderId;
(await connection).beginTransaction();

try {
  if(productIds.length == 1){
    console.log("add one");
    (await connection).query(`INSERT INTO orders
(id_product, quantity, name_of_client, number_of_phone,
id_type_delivery, address_delivery, id_address_selfPickup, comment,
time_delivery, order_price)
VALUES (${productIds[0]}, ${amounts[0]},
'${name}', '${phone_number}', ${delivery_type}, '${address}',
${id_selfPickup}, '${comment}', '${time}', '${price}')

```

У лістингу 3.4 наведено порядок виконання дій для додання нового запису замовлення, якщо клієнт замовив один тип товару, а саме реалізує обробку запиту на маршрут `"/addOrder"`.

Після отримання запиту код витягує дані з параметрів запиту, такі як ім'я клієнта, номер телефону, тип доставки, адреса тощо. В залежності від типу доставки (вказаного в параметрі `"delivery_type"`), встановлюються значення для ідентифікатора самовивозу або адреси доставки.

Далі, вміст товарів (content) розпаковується з формату JSON, після чого отримуються ідентифікатори товарів та їх кількість. Потім, розпочинається транзакції в базі даних. Це потрібно для того, щоб усі операції, що пов'язані із додаванням замовлення до бази даних, були виконані як єдине ціле. Якщо, в ході виконання, будь-яка з цих операцій не виконається, всі попередні операції в межах цієї транзакції будуть скасовані, щоб база даних залишалась у консистентному стані. У даному випадку, код спробує виконати один SQL-запит до бази даних, щоб зберегти інформацію про замовлення.

У разі успішного виконання операцій з базою даних транзакція комітується, тобто завершує транзакцію і зберігає всі дії, що були зроблені у даній транзакції, до бази даних. У випадку виникнення помилки транзакція відкачується, і відповідь містить інформацію про помилку.

Лістинг 3.5 – Обробник подій для додавання нового запису замовлення до бази даних, при умові, що товарів більше одного

```

...
    } else {
        console.log("add more");
        (await connection).query(`INSERT INTO orders
(id_product, quantity, name_of_client, number_of_phone,
id_type_delivery, address_delivery, id_address_selfPickup, comment,
time_delivery, order_price)
VALUES (${productIds[0]}, ${amounts[0]},
'${name}', '${phone_number}', ${delivery_type}, '${address}',
${id_selfPickup}, '${comment}', '${time}', '${price}')`)
        .then( value => {
            orderId = value[0].insertId;
        })
        .then(async () => {
            for(let i=1; i<productIds.length; i++){
                (await connection).query(`INSERT INTO orders
(id_order, id_product, quantity, name_of_client, number_of_phone,
id_type_delivery, address_delivery, id_address_selfPickup, comment,
time_delivery, order_price)
VALUES (${orderId}, ${productIds[i]},
${amounts[i]}, '${name}', '${phone_number}', ${delivery_type},
'${address}', ${id_selfPickup}, '${comment}', '${time}',
'${price}')`)
            }
        })
    }
...

```

Лістинг 3.5 доповнює код, що наведений у лістингу 3.6. Наведений код додає алгоритм альтернативного виду замовлення, де клієнт замовив декілька різних продуктів. У цьому випадку спершу виконується запит для першого товару, після чого отриманий ідентифікатор замовлення використовується для додавання решти товарів до цього замовлення виконуючи потрібну кількість запитів.

3.2 Розробка клієнтської частини

Клієнтська частина являє собою користувацький інтерфейс, за допомогою якого користувач може отримувати доступ до функції комп'ютерної системи. Інтерфейс матиме вигляд вебсторінки. Це означає, що користувачі повинні володіти базовими навичками користування комп'ютером і браузером для можливості користування інтерфейсом.

Користувацький інтерфейс або User interface (UI) – це засіб для візуалізації та взаємодії з різними програмними системами. До UI відносяться різні варіанти: текстові, графічні, тактильні, жестові, голосові, матеріальні. Оскільки, користувацький інтерфейс є вебсторінкою, то його створення виконується у такі основні етапи:

- формування задачі та функціонала. Даний етап було пройдено раніше, а саме у підрозділі 2.1;
- проектування і дизайн. Цей етап містить у собі розробку ескізів та макету користувацького інтерфейсу і створення візуального дизайну з визначенням кольорової палітри та структури інтерфейсу користувача;
- розробка. Етап передбачає верстку сторінок (фронтенд) і розробка функціонала для них (бекенд).

- випробування. Випробування передбачає проведення аналізу працездатності функцій сторінок шляхом проведення тестів комп'ютерної системи;
- розгортання і обслуговування. Наведений етап містить у собі обрання хостингу і доменного імені та своєчасне оновлення системи у разі виникнення помилок чи введення нового, чи покращення наявного функціонала.

3.3 Проєктування і дизайн користувацького інтерфейсу

Оскільки, етап з формуванням задачі та функціонала був пройдений, то наступним є створення макета та дизайну. Для цього було використано графічний редактор “figma”.

В процесі розроблення макета та дизайну було дотримано принципу KISS – “keep it simple, stupid” або “keep it short and simple”. Даний принцип означає, що компоненти системи чи іншого проєкту мають залишатися простими та зрозумілими аби не зіпсувати користувацький досвід і, тим самим, не відштовхнути потенційного клієнта. Також, було визначено з кольоровою палітрою, назвою і логотипом.

Важливим етапом є забезпечення адаптивності інтерфейсу, що передбачає створення різних варіантів візуалізації, але з однаковою зручністю використання, на різних пристроях із різними розмірами екранів. Адаптивність необхідна для розширення кола потенційних користувачів, дозволяючи їм взаємодіяти з комп'ютерною системою на різноманітних пристроях без втрати зручності та функціональності.

Результатом проєктування є отримання дизайну користувацького інтерфейсу для клієнта і для адміністратора для різних версій: десктопної (монітори комп'ютерів) і мобільної (екрани мобільних телефонів).

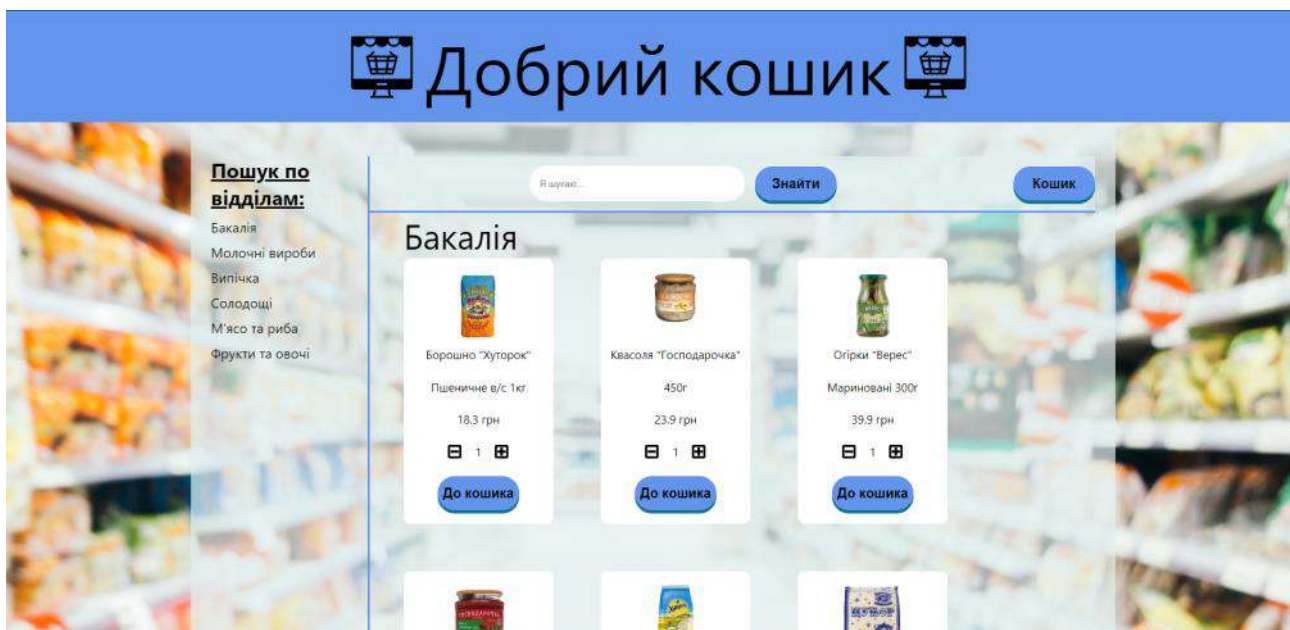


Рисунок 3.1 – Головна сторінка десктопної версії

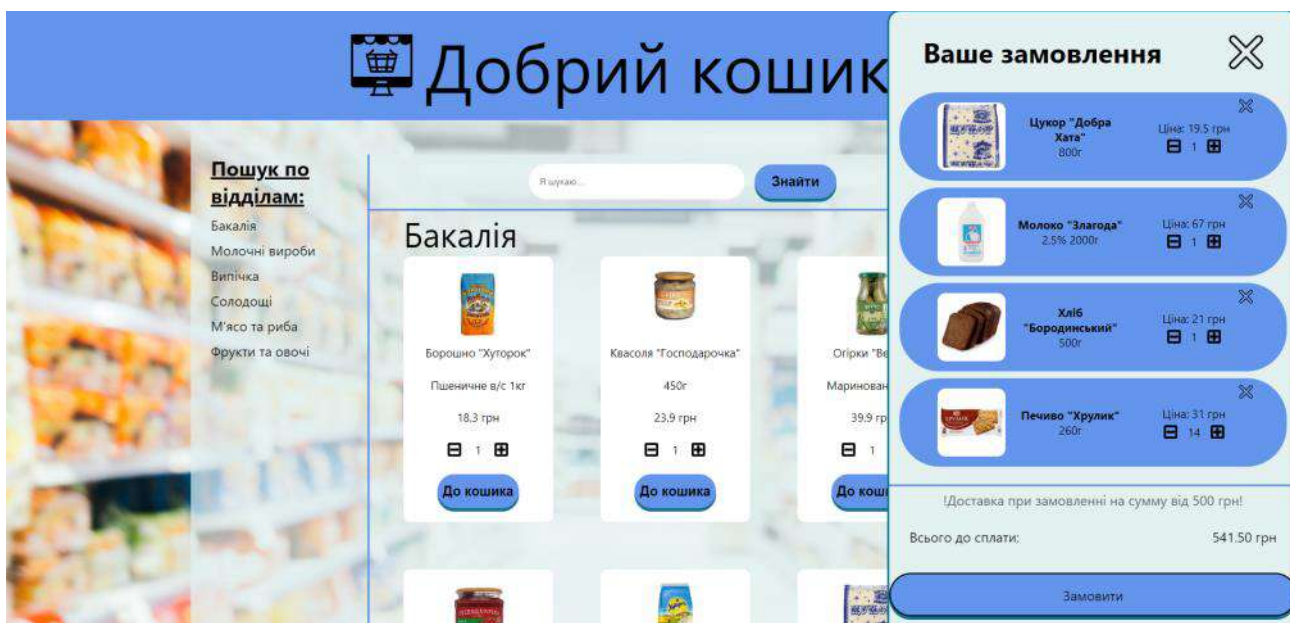
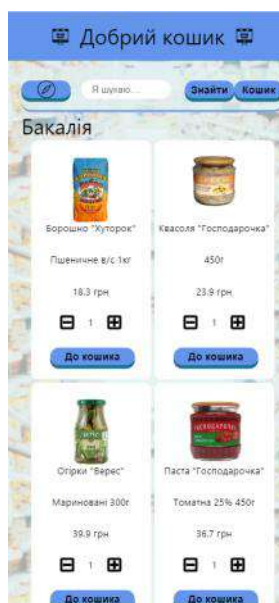
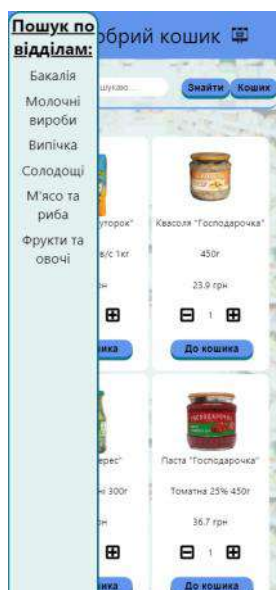


Рисунок 3.2 - Головна сторінка з відкритим кошиком десктопної версії



(A)



(B)



(B)

Рисунок 3.3 – Головна сторінка (A), головна сторінка з відкритим навігаційним меню (B) і головна сторінка з відкритим кошиком (B) мобільної версії

Рисунок 3.4 – Варіації форми для оформлення замовлення, що залежать від типу доставки

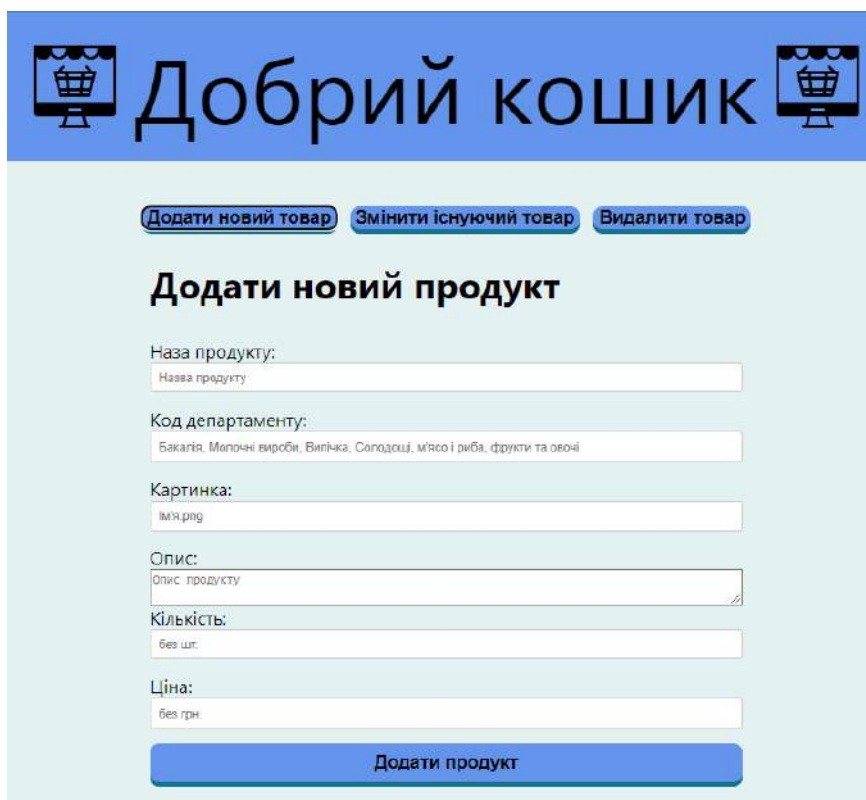


Рисунок 3.5 – Відображення чека замовлення

Таким чином, на рисунках 3.1, 3.2, 3.4 і 3.5 представлені основні функції користувацького інтерфейсу клієнта та адаптивність головної сторінки. Форма для оформлення замовлення та чек оформленого замовлення не потребують демонстрації їх адаптивності, оскільки вони не змінюються візуально, а лише пропорційно зменшуються відповідно до зменшення розміру екрана.

Добрий кошик						
Поточний список продуктів						
Код продукту	Назва продукту	Код департаменту	Картинка	Опис	Кількість	ціна
1	Борошно "Хуторок"	1	flour.png	Пшеничне в/с 1кг	100	18.3
2	Кавсиля "Господарочка"	1	bean.png	450г	100	23.9
3	Огірки "Верес"	1	cucumberVeres.png	Мариновані 300г	100	39.9
4	Паста "Господарочка"	1	paste.png	Томатна 25% 450г	100	35.7
5	Рис "Хуторок"	1	rice.png	длинный шлифованный 800г	100	7.9
6	Цукор "Добра Хата"	1	sugar.png	800г	100	19.5
7	Кефір "Селянський"	2	kefirSelsansky.png	2.5% 900г	100	21.5
8	Масло "Людське"	2	butterLustdorf.png	Солодковий/кислий Селянський 63% 200г	100	4.6
9	Молоко "Злагода"	2	milk.png	2.5% 2000г	100	6.7
10	Сир Плавлений "Звени Горі" Приби	2	meltedcheeseZveniGora.png	Стакан 90г	100	19.9
11	Сметана "Селянська"	2	sourCreamPasant.png	Стакан 20% 350г	100	33.5
12	Хліб "Бородняський"	3	bread.png	500г	100	21
13	Хліб "Гостиний"	3	breadGostiny.png	Пакований Різаний в/с 450г	100	19.7
14	Печиво з цукором	3	bunPyuzhka.png	250г	100	10
15	Сирна паличка	3	cheeseStick.png	140г	100	22
16	Пампушки з часником	3	donutWithGarlic.png	200г	100	16.49
17	Хліб "Пісочний"	3	loafMustard.png	400г	100	18.5
18	Молоко	4	cream.png	100г	100	3.35

Рисунок 3.6 – Головна сторінка адміністрації



Добрий кошик

Додати новий товар Змінити існуючий товар Видалити товар

Додати новий продукт

Назва продукту:

Код департаменту:

Картинка:

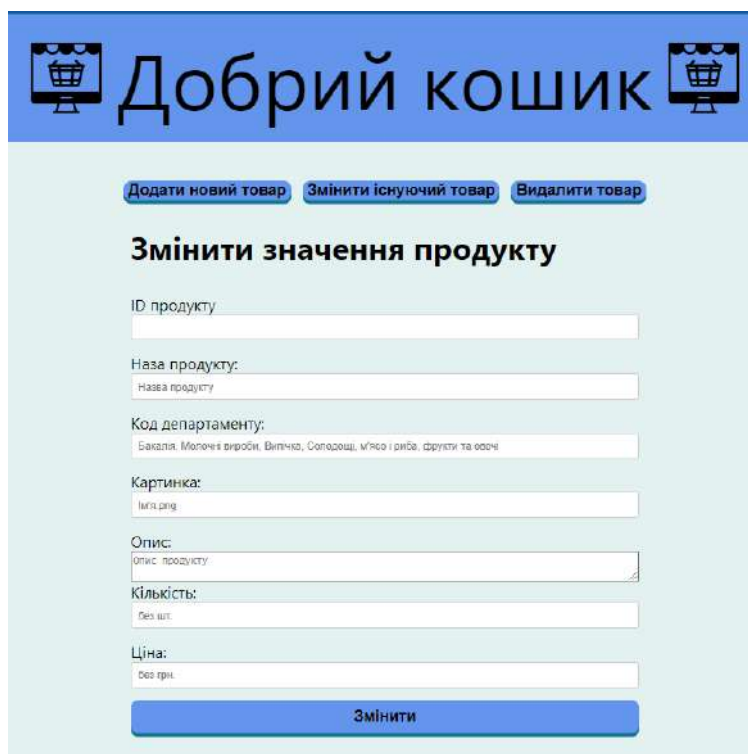
Опис:

Кількість:

Ціна:

Додати продукт

Рисунок 3.7 – Форма для додання нового запису до бази даних



Добрий кошик

Додати новий товар Змінити існуючий товар Видалити товар

Змінити значення продукту

ID продукту

Назва продукту:

Код департаменту:

Картинка:

Опис:

Кількість:

Ціна:

Змінити

Рисунок 3.8 – Форма для видозміна існуючого запису у базі даних

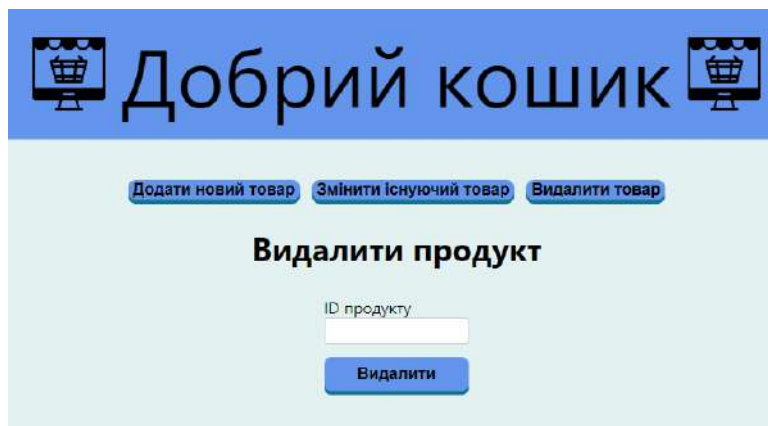


Рисунок 3.9 – Форма для видалення існуючого запису у бази даних

На рисунках 3.6, 3.7, 3.8 і 3.9 представлені основні функції користувацького інтерфейсу адміністратора. В адаптивності даної сторінки потреби немає, оскільки користування основними функціями адміністратора не передбачається на мобільних приладах.

3.4 Верстка та розробка функціонала сторінок

Результатом цього етапу є отримання повноцінного вебсайту з усім потрібним функціоналом як для клієнта, так і для адміністратора.

Процес верстки включає перетворення макетів у формат HTML та CSS. В результаті верстки вебсайту було отримано три вебсторінки: “index.html”, “shopbag.html” і “administrator.html”.

Сторінка “index.html” є головною сторінкою, де відображаються усі або шукані товари та кошик. “shopbag.html” є сторінкою, де клієнт може оформити замовлення за допомогою форми та отримати чек замовлення. На сторінці “administrator.html” адміністратор може виконувати власні основні функції.

Розробка функціонала для кожної з наявних вебсторінок представлена в додатках А, Б і В у лістингах А.1, А.2, А.3, Б.1, Б.2, Б.3, В.1, В.2, В.3 і В.4 у вигляді програмного коду.

У лістингу А.1 наведено код, за допомогою якого на головну сторінку завантажуються наявні записи продуктів з бази даних. Для цього він використовує метод “`fetch()`”, він надсилає запит до сервера за адресою “`http://localhost:1598/getProducts`” і отримує відповідь у форматі JSON. Потім дані обробляються та відображаються в різних категоріях товарів (бакалія, молочні продукти, випічка, солодощі, м'ясо, фрукти), що відповідають ідентифікаторам відділів. Для кожного товару створюється HTML-код, який включає зображення, назву, опис і ціну товару, а також кнопки для додавання до кошика і зміни кількості. Після відображення товарів на сторінці виконується ініціалізація функцій для керування кошиком: завантаження даних з локального сховища, додавання обробників подій для кнопок зміни кількості товару і функції додавання товарів до кошика. Останні функції потрібні для оновлення даних і ввімкнення можливості виконання інших функцій аби запобігти їх завчасного використання.

Лістинг А.2 реалізує функцію пошуку товарів на вебсторінці. При натисканні кнопки пошуку викликається функція “`addSearch()`”, яка додає обробник подій на кнопку пошуку. Після введення пошукового запиту у відповідне поле та натискання кнопки відправляється запит на сервер з введеним текстом. Отримавши відповідь у форматі JSON, код видаляє попередні результати пошуку та навігаційні елементи зі сторінки. Потім створюється HTML-код для відображення результатів пошуку, ідентичний до HTML-коду у лістингу А.1. і додається до відповідного контейнера на сторінці.

Лістинг А.3 представляє функцію додавання товарів до кошика на вебсторінці. Функція “`AddToCart()`” додає обробники подій до всіх кнопок "До кошика". Коли кнопка натиснута, перевіряється наявність даних у локальному сховищі браузера і, якщо потрібно, ініціалізується порожній масив для кошика.

Ідентифікатор товару та його кількість зчитуються з HTML-елементів сторінки. Далі, поточний вміст кошика зчитується з локального сховища та оновлюється: якщо товар вже є в кошику, його кількість збільшується, інакше товар додається як новий елемент. Оновлені дані зберігаються в локальному сховищі.

Після цього, виконується запит до сервера з переліком ідентифікаторів товарів у кошику, і отримані дані використовуються для оновлення відображення кошика на сторінці. HTML-код для кожного товару включає зображення, назву, опис, ціну та елементи управління кількістю. Отриманий HTML-код вставляється в елемент з ідентифікатором “shopList”. Потім, викликаються додаткові функції для видалення товарів з кошика, додавання обробників подій для кнопок зміни кількості товару та оновлення загальної суми кошика.

У лістингу Б.1 наведено функцію “addToBD”, яка здійснює передачу даних замовлення до бази даних. Вона спочатку отримує значення з полів форми: ім'я, номер телефону, прапорець доставки, адресу, магазин для самовивозу, час доставки, коментар та вміст кошика. Адреса доставки формується з кількох полів та зберігається як рядок. Якщо обрано “самовивіз”, адреса доставки встановлюється як “Null”, а “дільниця для самовивозу” – як значення, отримане з форми.

Вміст кошика отримується з локального сховища браузера і перетворюється у формат JSON. Ідентифікатори товарів і їх кількість конвертуються у числовий формат і зберігаються у вигляді рядка JSON. Дані замовлення відправляються на сервер за допомогою запиту “fetch”, який включає всі зібрані дані.

Після успішного завершення запиту виконується функція “createPayList()”, яка створює чек замовлення. Дана функція наведена у лістингу В.2.

У лістингу Б.2 наведена функція для створення чека замовлення. Для цього виконуються два “fetch” запити: перший – для отримання останнього ідентифікатору замовлення, другий – для отримання даних запису замовлення за ідентифікатором. Використовуючи ці дані, створюється чек замовлення у вигляді HTML-коду, де описані введені дані клієнта і розписано вартість кожного товару

у списку покупок. Також створюється кнопка, натиснувши на яку користувач повернеться до головної сторінки і дані локального сховища зітруться.

Для форми оформлення замовлення створена перевірка правильності введення даних у поля форми. Валідація наведена у лістингу В.3. Дана валідація перевіряє кожне поле вводу за допомогою допоміжних функцій, що є обробниками допустимих символів. Якщо даний обробник помічає недопустимий символ або пусте важливе для заповнення поле, то над полем для вводу створюється динамічний об'єкт, який повідомляє про зроблену помилку.

Лістинг В.1 відображає код створення списку наявних продуктів з їх даними для сторінки адміністратора. Даний код отримує список продуктів за допомогою “fetch” запиту у форматі JSON, і для кожного елемента даних створюється новий рядок у таблиці, і для кожного рядка додаються клітинки, що заповнюються відповідними значеннями полів. Ці значення відображаються в однойменних клітинках таблиці.

У лістингу В.2, В.3 і В.4 вказано ввімкнення кожної з форм для додання, видозмінення та видалення записів з бази даних. Їх конструкція схожа: зчитують дані з полів для вводу і відсилають їх на сервер, викликаючи “fetch” запит, де виконується один з трьох запитів на видозміну бази даних. І, заключним, код вимикає форми і формує таблицю зі списком наявних продуктів. Однак, у лістингу В.2 наведена функція “CheckID()”, яка доповнює конструкцію коду, додаючи функціонал для форми. Ця функція, при введенні ідентифікатора у відповідне поле для вводу, перевіряє його наявність у таблиці і, якщо він є, наповнює інші поля для вводу даними запису за відповідним ідентифікатором з бази даних, що спрощує процес редагування запису.

3.6 Випробування комп'ютерної системи

Випробування комп'ютерної системи є важливим етапом для релізу проекту. В процесі виконання етапу були проведені випробовування (тести) головних функцій комп'ютерної системи. Випробування клієнтського функціоналу зображено на рисунках 3.10 - 3.20.

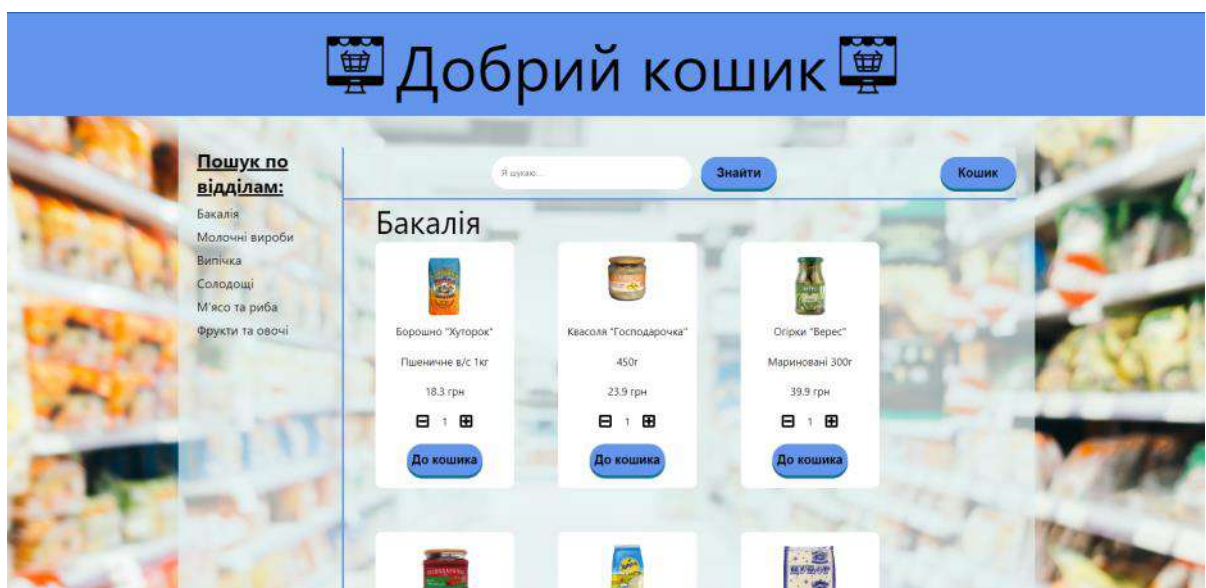


Рисунок 3.10 – Виконання функції відображення списку товарів клієнту

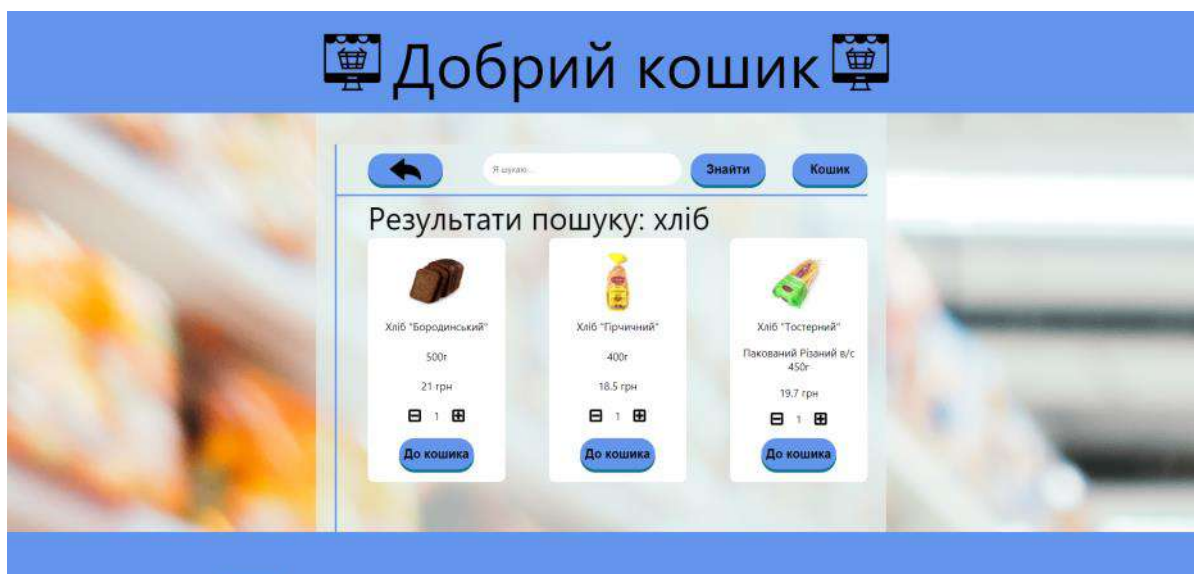


Рисунок 3.11 – Виконання функції з пошуком продуктів за їх ім'ям



Рисунок 3.12 – Виконання функції зміни кількості товару у кошику, де початковий вигляд зображений зліва, а виконана функція – у правій стороні



Рисунок 3.13 – Виконання функції видалення товару з кошика, де початковий вигляд зображений зліва, а виконана функція – у правій стороні

Добрий кошик

Контакти
Повинні бути букви Українського алфавіту

12345

Телефон
Неправильно введено дані (Введіть 14 символів)

+380(1111111111111111)

Оберіть тип доставки

☐ Самовивіз

☒ Доставка на дім

Адреса

Вулиця замовника 9

1 2 6

15:00

Коментар
Коментар замовлення (опціонально)

Всі поля повинні бути заповненими

Замовити

Рисунок 3.14 – Робота валідації форми оформлення замовлення

Добрий кошик

Контакти

Ім'я замовника

Телефон

+380(987456321)

Оберіть тип доставки

☐ Самовивіз

☒ Доставка на дім

Адреса

Вулиця замовника 9

1 2 6

15:00

Коментар
Коментар замовлення (опціонально)

Замовити

Рисунок 3.15 – Правильно заповнена форма

Ваше замовлення прийнято

Код вашого замовлення: 10

69.7X1 = 69 грн

Печиво "MajorCocoa":

25X1 = 25 грн

Печиво "Хрулик":

31X1 = 31 грн

Шоколад молочний "Milka":

52X1 = 52 грн

Рулет з маком:

29X1 = 29 грн

Замовлення на ім'я: Ім'я замовника

На номер телефону: 380(987456321

З вибраним типом доставки: доставка на дім

Сума до сплати: 734.64 грн

Закрити

Рисунок 3.16 – Виконання функції отримання чека замовлення

Поточний список продуктів						
Код продукту	Назва продукту	Код департаменту	Картинка	Опис	Кількість	ціна
1	Борошно "Хуторок"	1	flour.png	Пшеничне в/с 1кг	100	18.3
2	Квасоля "Господарочка"	1	bean.png	450г	100	23.9
3	Огірки "Верес"	1	cucumberVeres.png	Мариновані 300г	100	39.9
4	Паста "Господарочка"	1	paste.png	Томатна 25% 450г	100	36.7
5	Рис "Хуторок"	1	rice.png	довгий шліфований 800г	100	79
6	Цукор "Добра Хата"	1	sugar.png	800г	100	19.5
7	Кефір "Селянський"	2	kefirSelianskiy.png	2.5% 900г	100	21.5
8	Масло "Люстдорф"	2	butterLustdorf.png	Солодковершкове Селянське 63% 200г	100	46
9	Молоко "Злагода"	2	milk.png	2.5% 2000г	100	67
10	Сир Плавлений "Звени Гора" Гриби	2	meltedcheeseZveniGora.png	Стакан 90г	100	19.9
11	Сметана "Селянська"	2	sourCreamPeasant.png	Стакан 20% 350г	100	33.5
12	Хліб "Бородинський"	3	bread.png	500г	100	21
13	Хліб "Тостерний"	3	breadToaster.png	Пакований Різаний в/с 450г	100	19.7
14	Плюшка з цукром	3	bunPlyushka.png	250г	100	10
15	Сирна палочка	3	cheeseStick.png	140г	100	22
16	Пампушки з часником	3	donutWithGarlic.png	200г	100	16.49
17	Хліб "Гірничний"	3	loafMustard.png	400г	100	18.5
18	Морква	6	carrot.png	100г	100	1.55

Рисунок 3.17 – Виконання функції створення таблиці записів наявних товарів з бази даних

24	Рулет з маком	4	rollWithPoppy.png	200г	100	29	
25	Риба	5	fish.png	100 г	100	10	
26	Риба	8	fish.png	100 г fish in river	1000	11	

Рисунок 3.18 – Результат додавання нового запису до бази даних: форма для додавання (вище) і результат у таблиці (нижче)

25	Риба	5	fish.png	100 г	100	10	
26	Інша назва	5	fish.png	інший опис	100	5	

Рисунок 3.19 – Результат змінення наявного запису у базі даних: форма для зміни (вище) і результат у таблиці (нижче)

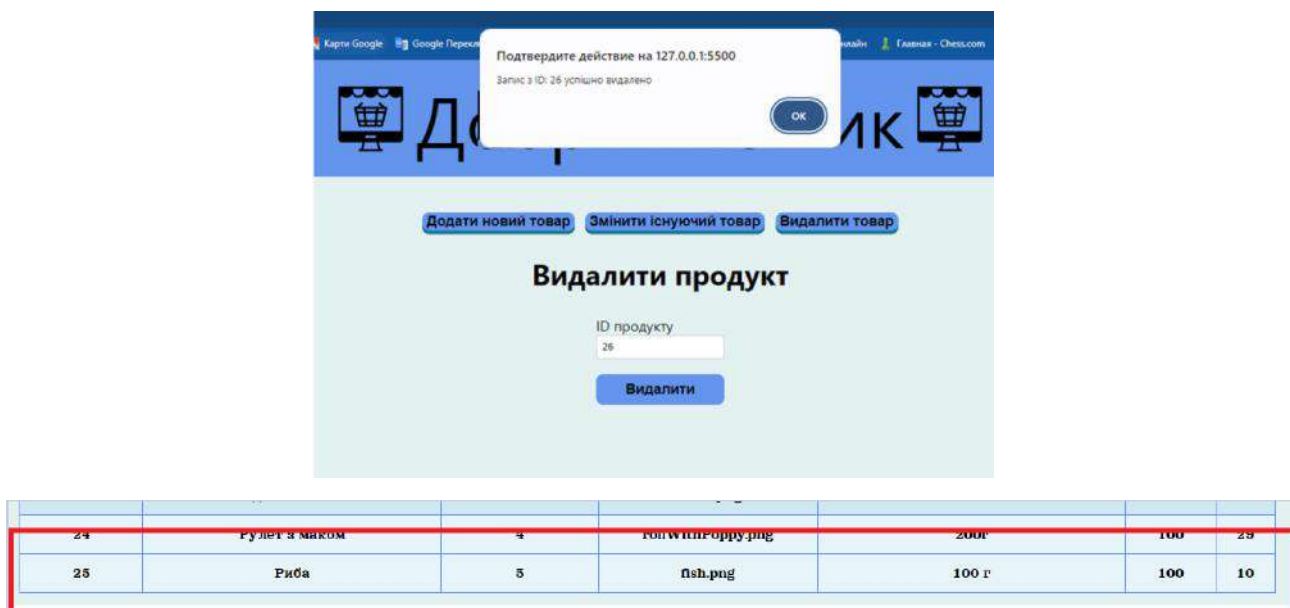


Рисунок 3.20 – Результат видалення наявного запису у базі даних: форма для видалення (вище) і результат у таблиці (нижче)

Таким чином, у результаті проведення випробування виявлено, що дана комп'ютерна система здатна виконувати задачу, яка була їй поставлена, виконуючи її головні функції.

3.7 Розгортання і обслуговування комп'ютерної системи

У даному етапі відбувається хостинг готової комп'ютерної системи у мережі Інтернет та слідкування за його працездатністю. Це означає моніторинг системи і виправлення помилок чи додання нового функціоналу. Роблячи це, зменшуються шанси на відсторонення клієнта та його втрата.

На даний момент проєкт хоститься на приватній адресі. Резервування доменного імені у мережі Інтернет не розглядається у найближчий час.

ВИСНОВКИ

Завдяки розробці комп'ютерної системи для продуктового магазину було значно розширено спектр послуг, доступних через мережу Інтернет. Створені сервер, користувацький інтерфейс та база даних сприяли покращенню взаємодії з клієнтами, забезпечили надання нових послуг та підвищили конкурентоспроможність магазину порівняно з іншими підприємств зі схожим сектором послуг.

В процесі виконання були виконані такі етапи розробки комп'ютерної системи:

- вибору технологій для реалізації комп'ютерної системи, де було проаналізовані наявні архітектури програмного забезпечення, технології для реалізації клієнтської та серверної частин, фреймворки для мов програмування, СКБД та зроблено висновки в обранні технологій;
- моделювання предметної області, де було визначено загальні вимоги до комп'ютерної системи, описано відношення і отримання ER-модель бази даних та створення діаграми послідовностей для візуалізації процесів у комп'ютерній системі;
- розробка серверної частини, де описано можливості серверу та способи їх досягнення;
- розробка клієнтської частини, де описані етапи створення користувацького інтерфейсу у вигляді веб-сайту. Результатом проходження цих етапів є формування мети та задачі, отримання дизайну користувацького інтерфейсу, зверстано і наповнено функціоналом та протестовано.

У результаті проєкту була створена комп'ютерна система, яка містить увесь необхідний функціонал для забезпечення зручного та ефективного оформлення замовлень продуктів харчування через Інтернет.

Таким чином, розроблена комп'ютерна система задовольняє потреби сучасного клієнта, збільшує власний прибуток, знаходить нових клієнтів та фіксує себе як магазин з онлайн продажу продуктів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Архітектура програмного забезпечення: все що треба знати.
URL: <https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya> (дата звернення: 29.04.2024);
2. Клієнт-серверна архітектура.
URL: <https://training.qatestlab.com/blog/technical-articles/client-server-architecture/> (дата звернення: 29.04.2024);
3. Exploring the Benefits and Limitations of Component-Based Software Architecture. URL: <https://moldstud.com/articles/p-exploring-the-benefits-and-limitations-of-component-based-software-architecture>; (дата звернення: 29.04.2024);
4. Монолітна архітектура ПЗ. URL: <https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/> (дата звернення: 29.04.2024);
5. Порівняння мікросервісної та монолітної архітектур.
URL: <https://www.atlassian.com/ru/microservices/microservices-architecture/microservices-vs-monolith> (дата звернення: 29.04.2024);
6. Мікросервісна архітектура.
URL: <https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d> (дата звернення: 29.04.2024);
7. Jon Duckett HTML and CSS: Design and Build Websites: Вид-во John Wiley & Sons, 2011. 512 с; URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (дата звернення: 30.04.2024);
8. What is Python? Executive Summary.
URL: <https://www.python.org/doc/essays/blurb/> (дата звернення: 30.04.2024);
9. ЦО TAKE PHP? URL: <https://freehost.com.ua/ukr/faq/wiki/chto-takoe-php/> (дата звернення: 30.04.2024);
10. Ruby для початківців: чим цікава ця мова і як її ефективно вивчати.
URL: <https://dou.ua/lenta/articles/ruby-for-beginners/> (дата звернення: 30.04.2024);

11. Ruby on Rails vs Javascript- What's the Difference (Pros and Cons). URL: <https://cloudinfrastructureservices.co.uk/ruby-on-rails-vs-javascript-whats-the-difference> (дата звернення: 30.04.2024);
12. Що таке фреймворк: пояснюємо простими словами. URL: <https://brainlab.com.ua/uk/blog-uk/shho-take-frejmwork-poyasnyuyemo-prostymy-slovamy> (дата звернення: 30.04.2024);
13. Next.js. Технологія сучасної веброзробки. URL: <https://habr.com/ru/companies/auriga/articles/786912/> (дата звернення: 01.05.2024);
14. Start building with Next.js. URL: <https://nextjs.org/learn> (дата звернення: 01.05.2024);
15. Next JS vs React :Which Framework to choose for Front end in 2024? URL: radixweb.com/blog/nextjs-vs-react (дата звернення: 01.05.2024);
16. What is Node.js (Node)?
URL: <https://www.techtarget.com/whatis/definition/Nodejs> (дата звернення: 01.05.2024);
17. What is Angular? URL: <https://angular.io/guide/what-is-angular> (дата звернення: 01.05.2024);
18. Веб-фреймворк Express (Node.js/JavaScript).
URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs (дата звернення: 01.05.2024);
19. What Does Oracle Database Mean?
URL: <https://www.techopedia.com/definition/8711/oracle-database> (дата звернення: 01.05.2024);
20. MySQL HeatWave Database Service.
URL: <https://www.oracle.com/mysql/> (дата звернення: 01.05.2024);
21. What is PostgreSQL? URL: <https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/> (дата звернення: 01.05.2024);
22. Основні відомості про бази даних

URL: <https://support.microsoft.com/uk-ua/topic/основні-відомості-про-бази-даних>
(дата звернення: 01.05.2024);

23. HeidiSQL: What's this? URL: <https://www.heidisql.com/> (дата
звернення: 01.05.2024).

ДОДАТОК А

КОДИ ДЛЯ РЕАЛІЗАЦІЇ ФУНКЦІОНАЛУ НА ГОЛОВНІЙ СТОРІНЦІ

Лістинг А.1 – Код додання відсортованого списку продуктів на сторінку

```
//LOAD from BD to PAGE
fetch ('http://localhost:1598/getProducts')
.then ((response) => {return response.json()})
.then ((data) => {
    let grocery =
document.getElementById("grocery").querySelector('.container_viddil_listOfProduct');
    let milk =
document.getElementById("milk").querySelector('.container_viddil_listOfProduct');
    let baking =
document.getElementById("baking").querySelector('.container_viddil_listOfProduct');
    let sweets =
document.getElementById("sweets").querySelector('.container_viddil_listOfProduct');
    let meat =
document.getElementById("meat").querySelector('.container_viddil_listOfProduct');
    let fruit =
document.getElementById("fruit").querySelector('.container_viddil_listOfProduct');
    let activeDepartment = null;

    for (let i=0; i<data.length; i++){
        switch (parseInt(data[i].id_department)){
            case 1:
                activeDepartment = grocery; break;
            case 2:
                activeDepartment = milk; break;
            case 3:
                activeDepartment = baking; break;
            case 4:
                activeDepartment = sweets; break;
            case 5:
                activeDepartment = meat; break;
            case 6:
                activeDepartment = fruit; break;
        }
        activeDepartment.innerHTML+=
        `
        <div id =
"${data[i].id_product}"class="container_department_listOfProduct_war
e">
```

```

        <div
class="container_department_listOfProduct_ware_picture">
        </img>
        </div>
        <div
class="container_department_listOfProduct_ware_other">
            <span>${data[i].name_of_product}</span>
            <span>${data[i].description}</span>
            <span>${data[i].price} грн</span>
            <div
class="container_department_listOfProduct_ware_other_price">
                <div class="price_quantity">
                    
                    <div
class="chopbag__block__price__number__epta">1</div>
                    
                    </div>
                </div>
                <button class="price_button">
                    <span class="price_button_text">До
кошика</span>
                </button>
            </div>
        </div>
    }
})
.then(() => {
    //cart.clearCart();
    cart.addFromlocalStorage();
    cart.addListener();
    cart.addToCart();
    NotefCart();
});

```

Лістинг А.2 – Функція для пошуку товарів

```

function addSearch() {
    document.querySelector('.container_searchAndShopbag_buttonFind')
        .addEventListener('click', function() {
        let InputSearch =
document.querySelector('.container_searchAndShopbag_inputText');
        let searchText= InputSearch.value;

fetch(`http://localhost:1598/searchProduct?name=${searchText}`)
    .then ((response) => {return response.json()})
    .then ((data) => {

```

```

        try{
            const elements =
document.querySelectorAll('.container_viddil');
            elements.forEach(element => {
                element.remove();
            });
            const elements2 =
document.querySelectorAll('.container_department_listOfProduct_ware'
);
            elements2.forEach(element => {
                element.remove();
            });
            const elements3 =
document.querySelectorAll('.navigation');
            elements3.forEach(element => {
                element.remove();
            });
        }
        catch(error){}
        let html = '';
        let wrapper =
document.querySelector('.container_body');
        html+=
        `<div class="container_viddil">
        <span class="container_viddil_name">Результати
пошуку: ${searchText}</span>
        <div class="container_viddil_listOfProduct">
        `;
        for (let i=0; i<data.length; i++){
            html+=
            `
            <div id =
"${data[i].id_product}"class="container_department_listOfProduct_war
e">

                <div
class="container_department_listOfProduct_ware_picture">
                    </img>
                </div>
                <div
class="container_department_listOfProduct_ware_other">
                    <span>${data[i].name_of_product}</span>
                    <span>${data[i].description}</span>
                    <span>${data[i].price} грн</span>
                </div>
                <div
class="container_department_listOfProduct_ware_other_price">
                    <div class="price_quantity">
                        

```



```

        NotefCart()
    })
})
}

```

Лістинг А.3 – Функція додавання товарів до кошика

```

function AddToCart() {
    btnToCart = document.querySelectorAll('.price_button');
    for (let btn of btnToCart){
        btn.addEventListener('click', function(){
            if (localStorage.getItem("cart") == null)
            {localStorage.setItem("cart", "[]");}

            let id =
this.closest('.container_department_listOfProduct_ware').id;
            let quantity =
this.closest('.container_department_listOfProduct_ware').querySelector(
'.chopbag__block__price__number__epta').innerHTML;

            let massive=[];
            let obj = JSON.parse(localStorage.getItem("cart"));
            for (let i=0; i<obj.length; i++){
                massive.push({
                    id : obj[i].id,
                    quantity: obj[i].quantity
                })
            }
            if (massive.find(item => item.id === id)){
                massive.find(item => item.id ===
id).quantity=parseInt(massive.find(item => item.id ===
id).quantity)+parseInt(quantity);
            } else{
                massive.push({
                    id:id,
                    quantity: quantity
                });
            }
            localStorage.setItem("cart", JSON.stringify(massive));
            obj = JSON.parse(localStorage.getItem("cart"));
            fetch(`http://localhost:1598/toCart?ids=${obj.map(item =>
item.id)}`)
                .then(response => response.json())
                .then(data => {
                    let html = '';
                    let cart = localStorage.getItem("cart");
                    let quan = JSON.parse(cart);
                    for (let i=0; i<data.length; i++)
                    {
                        let bag = quan.find(element => element.id ==
data[i].id_product).quantity;

```

```

        html+=
        `
        <li id = "${data[i].id_product}" class="shopbag_block">
        
        <div class="shopbag_block_withImage">
            </img>
        </div>
        <div class="shopbag_block_withOutImage">
            <span
class="shopbag_block_withOutImage_name">${data[i].name_of_product}</
span>
            <span
class="shopbag_block_withOutImage_desc">${data[i].description}</span
>
            </div>
            <div class="shopbag_block_withPrice">
                <span class="shopbag_block_withPrice_price">Цена:
<span class="priceForOne">${data[i].price}</span> ррн</span>
                <div
class="shopbag_block_withPrice_price_quantity">
                    
                    <span
class="shopbag_block_withPrice_price_quantity_num">${bag}</span>
                    
                    </div>
                </div>
            </li>
        `
    }
    document.getElementById("shopList").innerHTML=html;
    deleteStorage();
    addListernerShop();
    updatePrice();
    });
});
}
}

```

ДОДАТОК Б

КОДИ ДЛЯ РЕАЛІЗАЦІЇ ОФОРМЛЕННЯ ЗАМОВЛЕННЯ

Лістинг Б.1 – Функція додання замовлення до бази даних

```
function addToBD() {
    let NAME = document.querySelector('._name').value;
    let MOBILE = document.querySelector('._tel').value;
    let CHECKBOX =
document.querySelector('._checkbox').checked;
    let ADRES = document.querySelector('._street');
    let ADRES_DATA =
document.querySelectorAll('._address_data');
    let ADRES_STRING = 'вул.' + ADRES.value + ' під`їзд: ' +
ADRES_DATA[0].value + ' Поверх: ' + ADRES_DATA[1].value + '
Квартира: ' + ADRES_DATA[2].value;
    let MAGAZ_ADRES =
document.querySelector('._your').value;
    let CHECKBOX_NUM=0;
    if (CHECKBOX == true){
        CHECKBOX_NUM =2;
        ADRES_STRING = "Null";

    } else {
        CHECKBOX_NUM=1;
        MAGAZ_ADRES = "Null";
    }
    let TIME = document.querySelector('._time').value;
    let COMMENT =
document.querySelector('._comments__textarea').value;
    let CART = JSON.parse(localStorage.getItem('cart'));
    let PRICE =
JSON.stringify(localStorage.getItem('price')).replace(/["]/g, '');
    let obj = JSON.parse(localStorage.getItem("cart"));

    let convertedArray = CART.map(obj => {
        return {
            id: parseInt(obj.id),
            quantity: parseInt(obj.quantity)
        };
    });
    let outputStream = JSON.stringify(convertedArray);

    fetch(`http://localhost:1598/addOrder?name=${NAME}&phone_number=${MOBILE}&delivery_type=${CHECKBOX_NUM}&address=${ADRES_STRING}&id_selfPickup=${MAGAZ_ADRES}&comment=${COMMENT}&content=${outputStream}&time=${TIME}&price=${localStorage.getItem('price')}`)
        .then((response) => {return response.json();})
        .then ((data) => {
```

```

        console.log("sucksesful");
        createPayList();
    })
}

```

Лістинг Б.2 – Функція створення чека замовлення

```

function createPayList(){
    let id_order = 0;
    fetch('http://localhost:1598/getLastIdOrder')
    .then((response) => response.json())
    .then((data) => {
        let id_order = data.map(item => item.id_order);
        let id = document.querySelector('.id');
        id.innerHTML = "Код вашого замовлення: " + id_order[0];

        return
        fetch(`http://localhost:1598/getInfoOrder?id=${parseInt(id_order)}`)
        ;
    })
    .then((response) => response.json())
    .then((data) => {
        let html='';
        let wrapperList
=document.querySelector('.payList_body_item_list');

        for (i=0; i<data.length; i++){
            html +=
            `
            <p
class="payList_body_item">${data[i].name_of_product}: </p>
            <p
class="payList_body_item">${data[i].price}X${data[i].quantity} =
            ${parseInt(data[i].price)*parseInt(data[i].quantity)} грн</p>
            `
        }
        wrapperList.innerHTML=html;
        let html2='';
        let wrapperForm =
document.querySelector('.payList_body_formData_list');
        html2+=
        `
        <p class="payList_body_formData">Замовлення на ім'я:
        ${data[0].name_of_client}</p>
        <p class="payList_body_formData">На номер телефону:
        ${data[0].number_of_phone}</p>
        <p class="payList_body_formData">З вибраним типом
        доставки: ${data[0].type_delivery}</p>
        `
        if (data[0].type_delivery == "Доставка на дім"){
            html2=
            `

```

```

        ${html2}
        <p class="payList_body_formData">За адресом:
    ${data[0].address_delivery}</p>
        <p class="payList_body_formData">За часом:
    ${data[0].time_delivery}</p>
        `;
    } else if (data[0].type_delivery == "Самовивіз"){
        let adres;
        if (data[0].id_address_selfPickup == 1){
            adres = "вул. Шевченко 133Б";
        } else if (data[0].id_address_selfPickup == 2){
            adres = "пров. Лісний 15А";
        }
        html2=
        `
        ${html2}
        <p class="payList_body_formData">За адресом
діляниці:  ${adres}</p>
        `;
    }
    wrapperForm.innerHTML = html2;

    let priceElement =
document.querySelector('.payList_footer');
    let price = priceElement.innerHTML;
    price = `Сума до сплати: ${data[0].order_price} грн`;
    priceElement.innerHTML = price;

    let btnClose =
document.querySelector('.payList_footer_btnClose_close');
    btnClose.addEventListener('click', function(){
        let payList = document.querySelector('.payList ');
        payList.classList.remove('_active');
        localStorage.setItem('cart', "[]");
        localStorage.setItem('price', 0.0);
        window.location.href = "../index.html";
    })
    payList();
});
}

```

Лістинг Б.3 – код валідації форми для оформлення замовлення

```

const FORM = document.getElementById('form');
let formReq = document.querySelectorAll('._req');
FORM.addEventListener('submit', formSend);

async function formSend(e){
    e.preventDefault();
    let error = formValidate(FORM);
    if (error == 0){
        console.log("OK");
    }
}

```

```

        await addToBD();
    } else{
        console.log(" not OK");
        //alert("Виділені поля мають бути заповнені");
    }
}
function formValidate(e){
    let error = 0;
    let CHECKBOX =
document.querySelector('._checkbox').checked;
    let errorText = document.querySelector('.button__make-
order');
    DestroyError();
    let formReq = document.querySelectorAll('._req');
    for (let i=0; i<formReq.length; i++){
        const input = formReq[i];
        formRemoveError (input);
        if (input.classList.contains('_name')){
            if (input.value.length != 0 &&
validationText(input.value) == 0)
            {
                formRemoveError(input);
            }else{
                formAddError (input);
                error++;
            }

            //error++;
        } else if (input.classList.contains('_tel')){
            if (input.value.length != 0 &&
validationTel(input.value) == 0)
            {
                validationTel (input.value);
                formRemoveError(input);
            }else{
                formAddError (input);
                error++;
            }
        }
        } else if (CHECKBOX == false &&
input.classList.contains('_street')){
            if ( input.value != 0)
            {
                formRemoveError(input);
            } else{
                formAddError (input);
                error++;
            }
        }
        } else if (CHECKBOX == false &&
input.classList.contains('_street-data')){
            if (input.value.length != 0 &&
validationNum(input.value) == 0)
            {

```

```

        validationNum(input.value);
        formRemoveError(input);
    } else{
        formAddError (input);
        error++;
    }
    } else if (CHECKBOX == true &&
input.classList.contains('_your')){
    if (input.value != 0)
    {
        formRemoveError(input);
    } else{
        formAddError (input);
        error++;
    }
    }
    }

    if (error != 0 ){
        CreateError(errorText, "Всі поля повинні бути
заповненим");
    }
    return error;
}
function validationText (input){
    let index = 0;
    let len=input.length;
    let
digits="йцукенгшщзхїєждлорпавіфячсмитьбюйцукенгшщзхїєждлорпавіфячсми
ТЬБЮ' ";
    if (len > 50)
    {
        CreateError(document.getElementById('name'), "Забагато
символів (Введіть кількість символів < 50)");
        index++;
    }
    for(i=0; i<10; i++)
    {if (digits.indexOf(input.charAt(i))<0)
    {CreateError(document.getElementById('name'), "Повинні бути
букви Українського алфавіту");
    index++;
    break}
    }
    return index;
}
function validationTel (input)
{
    let index = 0;
    let len=input.length;
    let digits="+ (0123456789";
    if (len != 14)
    {

```

```

        CreateError(document.getElementById('tel'),
"Неправильно введені дані ( Введіть 14 символів)");
        index++;
    }
    for(i=0; i<10; i++)
    {if (digits.indexOf(input.charAt(i))<0)
    {CreateError(document.getElementById('tel'), "У номері
телефону повинні бути числа");
        index++;
        break}
    }
    return index;
}
function validationNum (input){
    let index = 0;
    let len=input.length;
    let digits="0123456789";
    if (len > 3)
    {
        CreateError(document.getElementById('num'),
"Забагато символів ( Введіть кількість символів < 3)");
        index++;
    }
    for(i=0; i<10; i++)
    {if (digits.indexOf(input.charAt(i))<0)
    {CreateError(document.getElementById('num'),
"Неправильно введені дані (Повинні бути цифри)");
        index++;
        break}
    }
    return index;
}
function CreateError(element, text){
    let span1 = document.createElement("p");
    span1.className='textError';
    span1.style.color = 'cornflowerblue';
    span1.innerHTML = text;
    var parentDiv =element.parentNode;
    parentDiv.insertBefore(span1, element);
}
function DestroyError(){
    let span = document.querySelectorAll('.textError');
    for (let i=0; i<span.length; i++)
    {
        span[i].remove();
    }
}
function formAddError (input){
    input.parentElement.classList.add('error');
    input.classList.add('_error');}
function formRemoveError (input){
    input.parentElement.classList.remove('error');
    input.classList.remove('_error');}

```

ДОДАТОК В

КОДИ ДЛЯ РЕАЛІЗАЦІЇ ФУНКЦІОНАЛУ АДМІНІСТРАТОРА

Лістинг В.1 – Функція створення таблиці записів наявних продуктів

```
const TABLE = document.getElementById('data-table');
function createTable() {
    fetch('http://localhost:1598/getAllProd')
    .then(response => response.json())
    .then(data => {
        maxIdProduct = data.length;
        const tableBody = document.getElementById('data-
table').getElementsByTagName('tbody')[0];
        data.forEach(item => {
            const row = tableBody.insertRow();
            const cellId = row.insertCell(0);
            const cellName = row.insertCell(1);
            const cellDepart = row.insertCell(2);
            const cellPicture = row.insertCell(3);
            const cellDescription = row.insertCell(4);
            const cellQuantity = row.insertCell(5);
            const cellPrice = row.insertCell(6);

            cellId.textContent = item.id_product;
            cellName.textContent = item.name_of_product;
            cellDepart.textContent = item.id_department;
            cellPicture.textContent = item.pictures;
            cellDescription.textContent = item.description;
            cellQuantity.textContent = item.quantity;
            cellPrice.textContent = item.price;
        });
    })
    .catch(error => console.error('Помилка отримання даних:',
error));
}
```

Лістинг В.2 – Код додання нового запису до бази даних

```
const BtnAddInDown = document.getElementById('BtnAddInDown');
BtnAddInDown.addEventListener('click', (event)=>{
    event.preventDefault();
    let NewProduct = {
        name:document.getElementById('name').value,
        id_department:document.getElementById('department-
code').value,
        picture:document.getElementById('image').value,
```

```

description:document.getElementById('description').value,
    quantity:document.getElementById('quantity').value,
    price:document.getElementById('price').value
    }
    //console.log(NewProduct);
    if(NewProduct.name == ''|| NewProduct.id_department== '' ||
NewProduct.description == ''|| NewProduct.picture == ''||
NewProduct.quantity == ''|| NewProduct.price == ''){
        alert("Всі поля мають бути заповнені");
        return;
    }

    fetch(`http://localhost:1598/addNewProduct?name=${NewProduct.name}&i
d_department=${NewProduct.id_department}&pictures=${NewProduct.pictu
re}
&description=${NewProduct.description}&quantity=${NewProduct.quantit
y}&price=${NewProduct.price}`)
        .then((response) => {return response.json();})
        alert("Запис спішно зроблено");
        let formAdd =document.getElementById('formAdd');
        formAdd.reset();
        HideTable ();
        deleteTable();
        createTable();
        h2_list.classList.remove('_hidden');
        TABLE.classList.remove('_hidden');
    })

```

Лістинг В.3 – Код зміна існуючого запису наявного продукту у базі даних

```

function checkId() {
    let id =
document.getElementById('id_products_change').value;
    if (id > 0 && id <= maxIdProduct) {
        fetch(`http://localhost:1598/getProductById?id=${id}`)
            .then((response) => {
                return response.json();
            })
            .then((data) => {
                document.getElementById('name_change').value =
data.name_of_product;
                document.getElementById('department-
code_change').value = data.id_department;
                document.getElementById('image_change').value =
data.pictures;

                document.getElementById('description_change').value =
data.description;

```

```

document.getElementById('quantity_change').value = data.quantity;
document.getElementById('price_change').value =
data.price;
old=document.getElementById('department-
code_change').value;
    })
    .catch((error) => {
        console.error('Fetch error:', error);
    });
}
else{
    document.getElementById('name_change').value = null;
    document.getElementById('department-code_change').value
= null;
    document.getElementById('image_change').value = null;
    document.getElementById('description_change').value =
null;
    document.getElementById('quantity_change').value =
null;
    document.getElementById('price_change').value = null;
}
}
// function to change form
const BtnChangeInDown =
document.getElementById('BtnChangeInDown');
let label = document.getElementById('id_products_change');
let id = 0;
let old=0;
label.addEventListener('input', checkId);
BtnChangeInDown.addEventListener('click', (event)=>{
    event.preventDefault();
    id = document.getElementById('id_products_change').value;
    let NewProduct = {
        name:document.getElementById('name_change').value,
        id_department:document.getElementById('department-
code_change').value,
        picture:document.getElementById('image_change').value,
description:document.getElementById('description_change').value,
quantity:document.getElementById('quantity_change').value,
        price:document.getElementById('price_change').value
    }

    http://localhost:1598/changeData?id=26&name=%D0%A0%D0%B8%D0%B1%D0%B0
&id_departmentNew=5&id_departmentOld=5&pictures=123&description=10&q
uantity=100&price=100

    fetch(`http://localhost:1598/changeData?id=${id}&name=${NewProduct.n
ame}&id_departmentNew=${NewProduct.id_department}&id_departmentOld=${
old}&pictures=${NewProduct.picture}

```

```

&description=${NewProduct.description}&quantity=${NewProduct.quantit
y}&price=${NewProduct.price}`);
    alert("Запис з ID: "+id + " успішно видозмінено");
    let formformChange =document.getElementById('formChange');
    formformChange.reset();
    HideTable();
    deleteTable();
    createTable();
    h2_list.classList.remove('_hidden');
    TABLE.classList.remove('_hidden');
  })

```

Лістинг В.4 – Код видалення записів наявних продуктів у базі даних

```

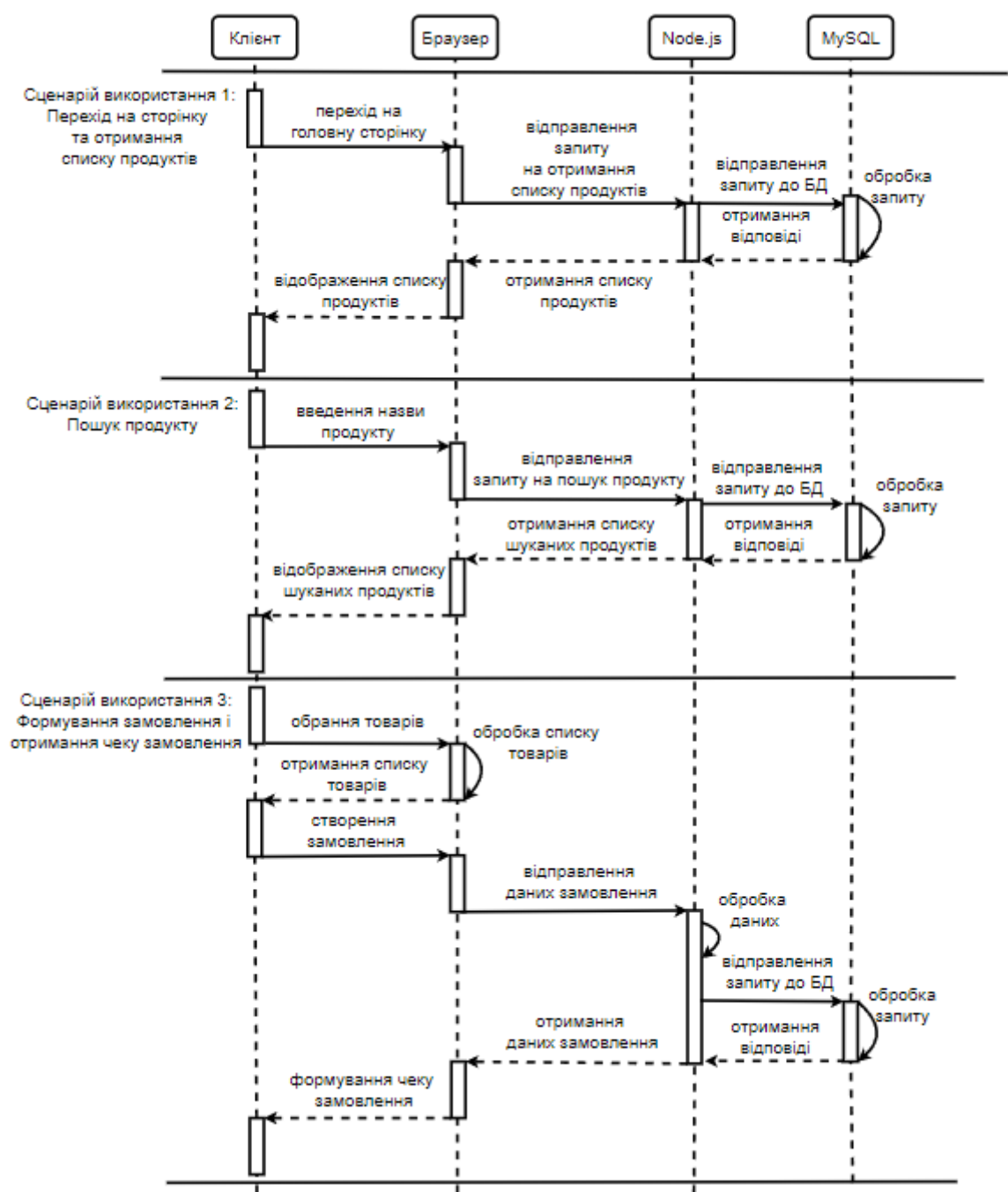
const BtnRemoveInDown =
document.getElementById('BtnRemoveInDown');
BtnRemoveInDown.addEventListener('click', ()=>{
  let removeLabel =
document.getElementById('id_products_remove').value;
  console.log(removeLabel);
  console.log(maxIdProduct);
  if (removeLabel > 0 && removeLabel<= maxIdProduct){

fetch(`http://localhost:1598/deleteProduct?id=${removeLabel}`);
    alert("Запис з ID: "+ removeLabel + " успішно видалено");
    let formRemove =document.getElementById('formRemove');
    formRemove.reset();
    HideTable();
    deleteTable();
    createTable();
    h2_list.classList.remove('_hidden');
    TABLE.classList.remove('_hidden');
  }
  else {alert("Запис з ID: "+ removeLabel + " не існує")};

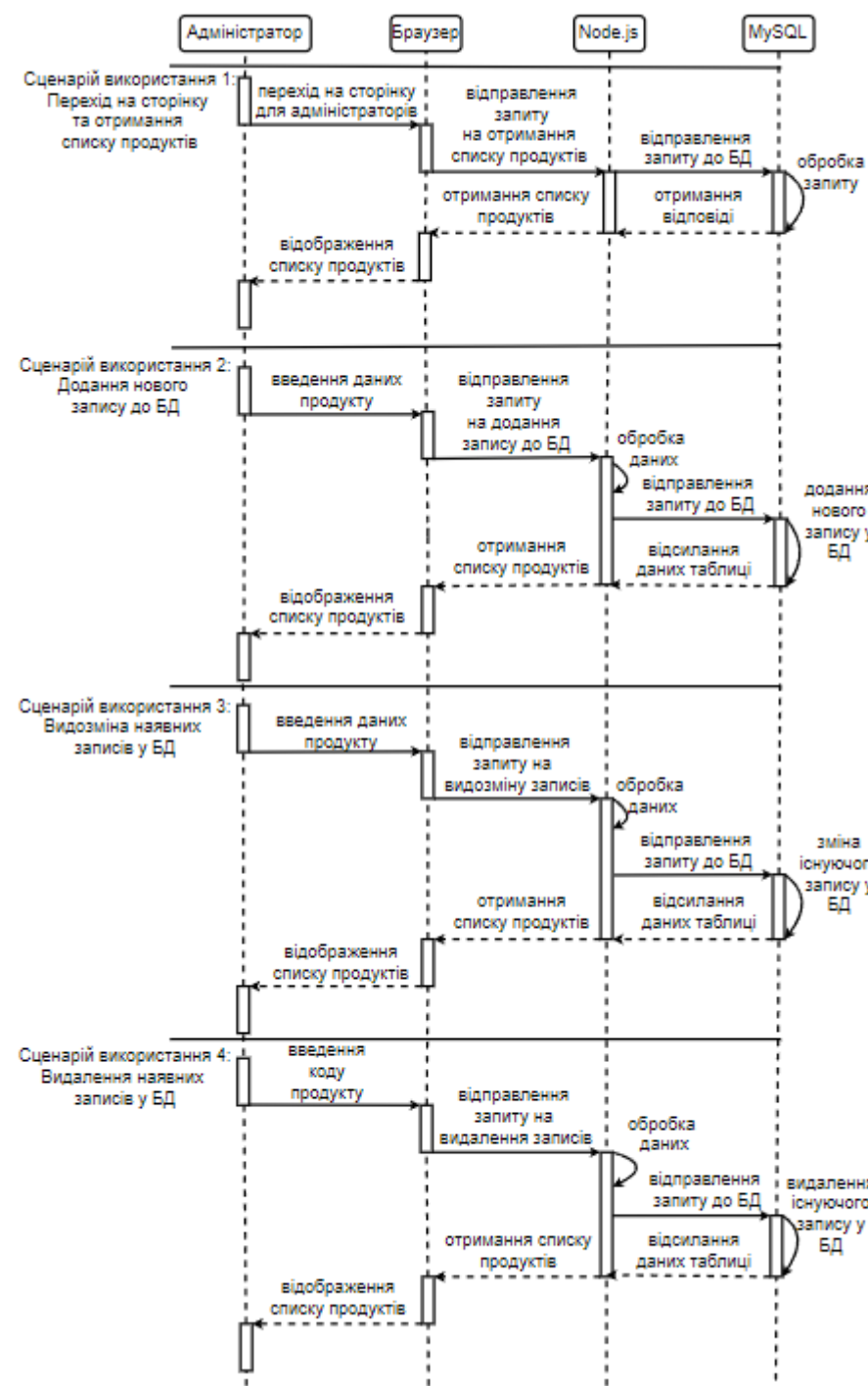
})

```

UML діаграми послідовностей



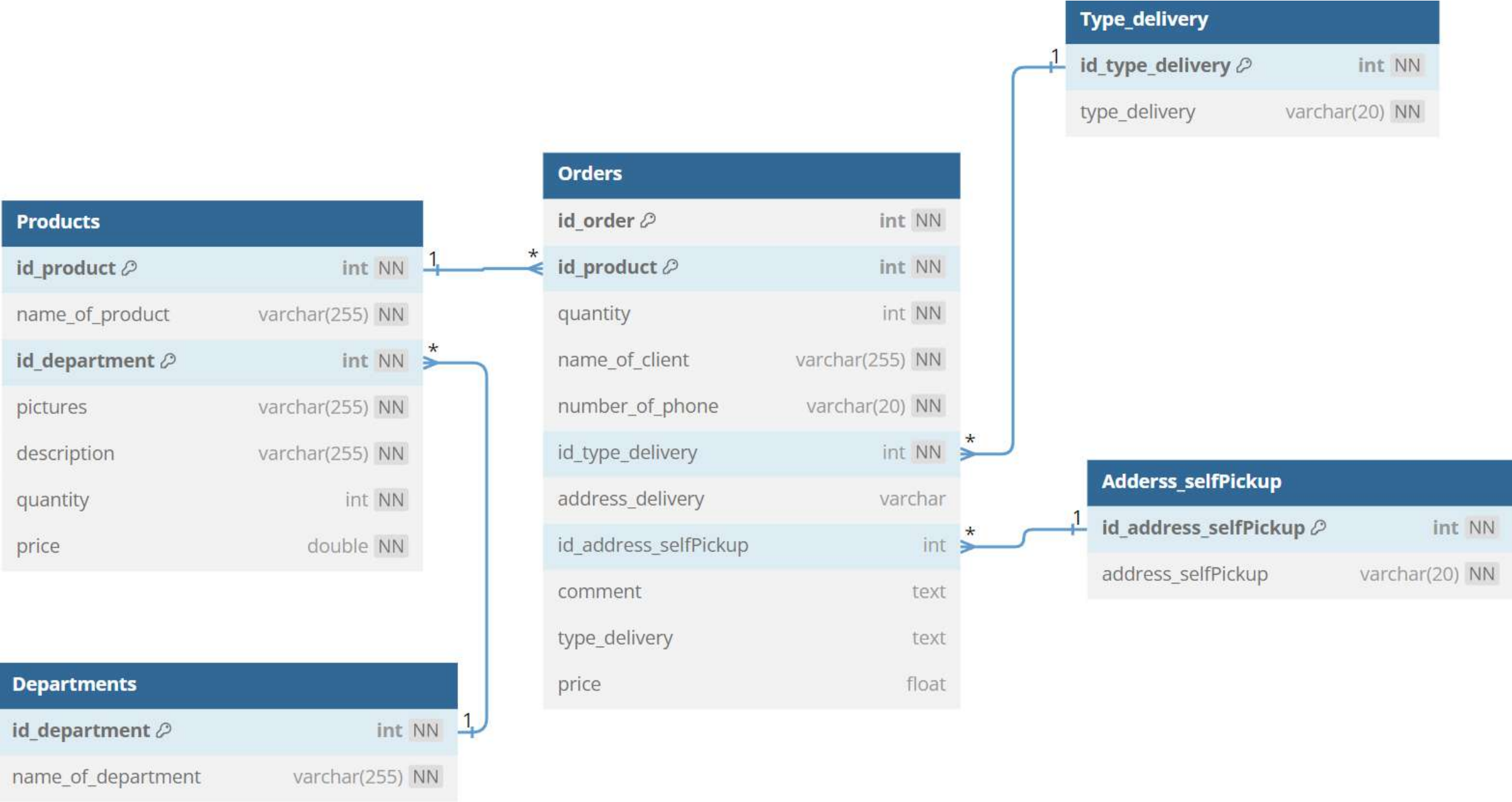
Діаграма послідовностей для клієнта



Діаграма послідовностей для адміністратора

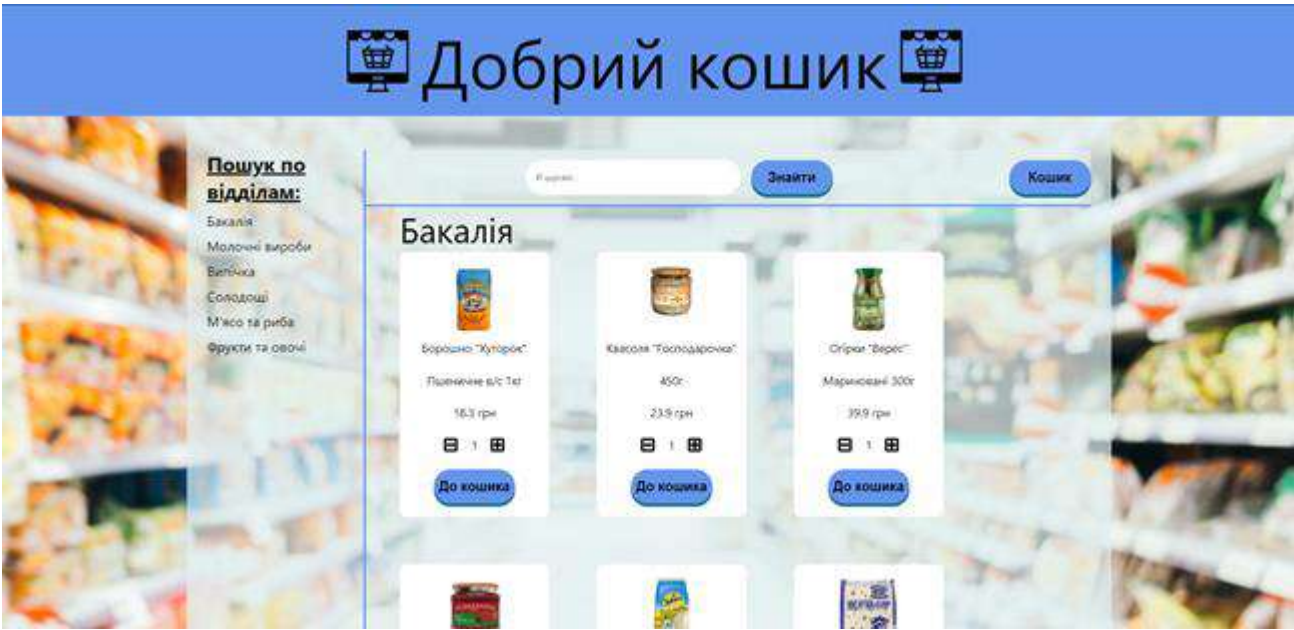
					13.02070849.00048 ПЛ1					
					Розробка комп'ютерної системи продуктового магазину	Лист.		Маса	Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата						
Розроб.		Рюмін О. О.								
Перев.		Скрупський С.Ю								
Т.контр.										
Н.контр.		Польська О.В.			UML діаграми послідовностей	НУ «Запорізька політехніка» КНТ-520				
Затв.		Кудерметов Р.К.								

ER-модель бази даних комп'ютерної системи

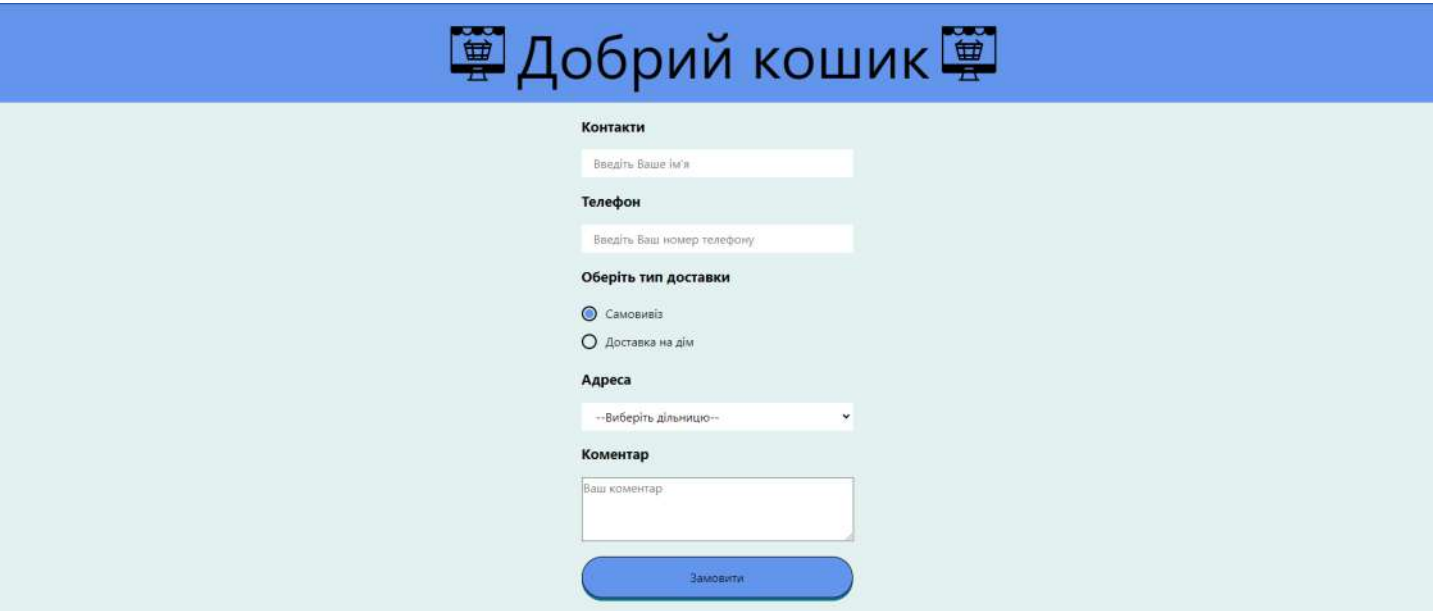


					13.02070849.00048 ПЛ2							
					Розробка комп'ютерної системи продуктового магазину				Лист.		Маса	Масшт.
Зам.	Лист.	№ докум.	Підп.	Дата								
Розроб.		Рюмін О. О	Р									
Перев.		Скрупський С.Ю	СКР									
Т.контр.					ЕР-модель бази даних комп'ютерної системи				Лист 1		Листів 1	
									НУ «Запорізька політехніка» КНТ-520			
Н.контр.		Польська О.В.	П									
Затв.		Кудерметов Р.К.	К									

Сторінки веб-сайту комп’ютерної системи продуктового магазину



Головна сторінка



Сторінка оформлення замовлення

Добрий кошик						
Додати новий товар Зберегти існуючий товар Вилучити товар						
Поточний список продуктів						
Ідентифікатор	Назва продукту	Код департаменту	Категорія	Опис	Кількість	Ціна
1	Хліб борошняний "Хлібчик"	1	хліб.rpt	Пшеничний в/с 1кг	100	18.3
2	Масло "Богородиця"	1	масло.rpt	450г	100	23.8
3	Сирок "Варені"	1	сирок/варени.rpt	Маринований 300г	100	39.9
4	Паста "Богородиця"	1	паста.rpt	Томатна 20% 450г	100	36.7
5	Рис "Хлібчик"	1	рис.rpt	Довгий широким 800г	100	79
6	Цукор "Добра Хата"	1	цукор.rpt	300г	100	19.5
7	Молоко "Світличка"	2	молоко/сметана.rpt	2.5% 300г	100	21.5
8	Масло "Богородиця"	2	масло/сливки.rpt	Сметановий 10% 300г	100	48
9	Молоко "Світличка"	2	молоко.rpt	2.5% 200г	100	87
10	Сир "Світличка" "Дівочий" 1кг	2	сир/сметана/сливки.rpt	Сметана 90%	100	19.9
11	Сметана "Світличка"	2	сметана/сливки.rpt	Сметана 20% 200г	100	23.5
12	Хліб "Богородиця"	3	хліб.rpt	300г	100	21
13	Хліб "Світличка"	3	хліб/сливки.rpt	Пшеничний 1кг в/с 450г	100	18.7
14	Пиво "Світличка"	3	пиво/сливки.rpt	250г	100	18
15	Сирок "Світличка"	3	сирок/сливки.rpt	140г	100	22
16	Пиво "Світличка"	3	пиво/сливки.rpt	300г	100	18.48
17	Хліб "Світличка"	3	хліб/сливки.rpt	400г	100	18.5
18	Молоко "Світличка"	3	молоко.rpt	100г	100	1.55

Сторінка адміністрації

					13.02070849.00048 ПЛЗ								
					Розробка комп'ютерної системи продуктового магазину				Лист.		Маса	Масшт.	
									Лист 1		Листів 1		
Зам.	Лист.	№ докум.	Підп.	Дата	Сторінки веб-сайту комп'ютерної системи продуктового магазину				НУ «Запорізька політехніка» КНТ-520				
Розроб.		Рюмін О. О											
Перев.		Скруський С.Ю											
Т.контр.													
Н.контр.		Польська О.В.											
Затв.		Кудерметов Р.К.											