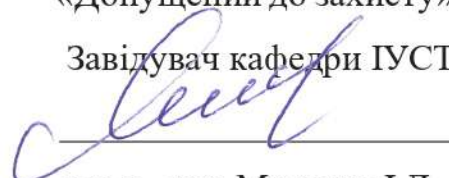


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами Кафедра
інформаційних управляючих систем та технологій

«Допущений до захисту»

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

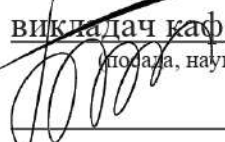
на тему:

Розроблення інформаційної системи інтернет-магазину страв японської кухні

Виконав: студент групи 4147ст

 Лебеденко С.С.
(підпис)

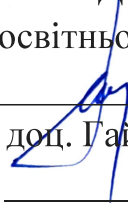
Керівник роботи:

викладач кафедри ІУСТ
 (посада, науковий ступінь вчене звання)
Бобровський О.С.
(підпис)

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Лебеденку Станіславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Розроблення інформаційної системи інтернет-магазину страв японської кухні
- Керівник роботи викладач кафедри ІУСТ Бобровський Олексій Сергійович
Затверджені наказом ректора № 295-уч від «27» березня 2025 року.
2. Термін подання роботи: 20 червня 2025 р.
3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали переддипломної практики
4. Перелік питань, що належать до розробки (найменування розділів) Аналіз предметної області та постановка задачі, розробка проєктних рішень, реалізація проєкту, охорона праці, висновки
5. Перелік презентаційних матеріалів: титульна сторінка, мета та актуальність роботи, контекстна діаграма, діаграма варіантів використання, концептуальна модель даних, логічна модель даних, фізична модель даних, діаграми взаємодії, діаграми діяльності, діаграма класів, діаграма розгортання системи, результати розробки, висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Бобровський О.С.	03.02.2025	22.02.2025
2	Бобровський О.С.	22.02.2025	21.03.2025
3	Бобровський О.С.	21.03.2025	25.04.2025
4	Бобровський О.С.	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


Лебеденко С.С.

(ПБ)

Керівник роботи



(підпис)

Бобровський О.С.

(ПБ)

АНОТАЦІЯ

В даній кваліфікаційній роботі представлено розроблення інформаційної системи для інтернет-магазину страв японської кухні «MykoSushi».

У першому розділі проведено аналіз предметної області з метою краще зрозуміти те, що з себе представляє інтернет-магазин та сформулювати представлення принципів роботи майбутньої інформаційної системи. Був проведений аналіз готових рішень, який допоміг розібратися в перевагах та недоліках вже створених платформ для замовлення їжі онлайн. Це також допомогло зрозуміти, що саме потрібно реалізувати і які функції важливіше. Також були сформульовані основні проблеми з якими можуть стикнутися клієнти, сформовані вимоги і складено технічне завдання.

У другому розділі вся увага приділена створенню проектних рішень, які включають в себе інформаційне, програмне й технічне забезпечення інформаційної системи.

У третьому розділі описано як було створено інформаційну систему та підготовлено основну документацію, необхідну для впровадження та подальшу її експлуатацію.

У четвертому розділі описані питання з охорони праці які виникають під час роботи з комп'ютером та програмним забезпеченням, а також заходам, що допомагають зробити умови праці безпечними й зручними.

Кваліфікаційна робота виконана на 106 сторінках машинописного тексту, який містить 25 рисунків, 16 таблиць, 2 додатки, список використаних джерел з 21 найменувань.

Робота виконана українською мовою.

Ключові слова: інформаційна система, інтернет-магазин, японська кухня, веб-розробка, технічна документація.

ABSTRACT

This qualification work presents the development of an information system for an online store selling Japanese cuisine dishes called «Japanese Quarter».

In the first chapter, an analysis of the subject area was conducted in order to better understand the nature of an online store and to form a vision of the principles of operation for the future information system. An analysis of existing solutions was carried out, which helped to identify the advantages and disadvantages of already created platforms for ordering food online. This also made it possible to determine what exactly needs to be implemented and which functions are the most important. In addition, the main problems that customers may face were identified, requirements were formulated, and a technical specification was compiled.

The second chapter is entirely devoted to the development of design solutions, which include the information, software, and technical support of the information system.

The third chapter describes how the information system was created and how the main documentation required for implementation and further operation was prepared.

The fourth chapter covers occupational safety issues that arise when working with computers and software, as well as measures that help make working conditions safer and more comfortable.

The qualification work is executed on 106 pages of typewritten text, which contains 25 figures, 16 tables, 2 appendices, a list of references from 21 titles.

The work was done in Ukrainian.

Keywords: information system, online store, Japanese cuisine, web development, technical documentation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ.....	9
1.1. Аналіз предметної області.....	9
1.2. Аналіз існуючих рішень.....	12
1.3. Формулювання проблеми	14
1.4. Визначення вимог до інформаційної системи	17
1.5. Постановка задачі	18
1.6 Висновок до першого розділу	20
РОЗДІЛ 2. РОЗРОБКА ПРОЄКТНИХ РІШЕНЬ.....	22
2.1. Загальносистемні рішення	22
2.2. Рішення з інформаційного забезпечення	29
2.3. Рішення з технічного забезпечення	34
2.4. Рішення з програмного забезпечення	35
2.5. Висновок до другого розділу.....	41
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЄКТУ.....	43
3.1. Аргументація вибору засобів розробки.....	43
3.2. Розробка структури програмного забезпечення.....	46
3.3. Результати розробки програмного забезпечення	47
3.4. Висновок до третього розділу	48
РОЗДІЛ 4. ОХОРОНА ПРАЦІ	50
4.1. Загальні вимоги	50
4.2. Вимоги до організації робочого місця.....	50
4.3 Вимоги до освітлення.....	53
4.4 Вимоги електробезпеки	54
4.5 Вимоги до рівнів шуму та вібрації	58
4.6 Висновок до четвертого розділу	59
ВИСНОВКИ	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТОК А.....	64
ДОДАТОК Б.....	67

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

UML (Unified Modeling Language) – уніфікована мова моделювання;

БД – база даних.

ПЗ – програмне забезпечення;

ДсанПіН – державні санітарні норми і правила;

ДСТУ - стандарти, розроблені відповідно до чинного законодавства України, що встановлюють для загального і багаторазового застосування правила, загальні принципи або характеристики, які стосуються діяльності чи її результатів, з метою досягнення оптимального ступеня впорядкованості, розроблені на основі консенсусу та затверджені уповноваженим органом;

ПК – персональний комп'ютер;

ДНАОП – державні нормативні акти про охорону праці;

ERD (entity-relationship diagram, ER-діаграма) – діаграма сутність-зв'язок.

Web-server – це програма, яка працює на інтернет-сайті та видає веб-сторінки за запитом відвідувачам цього сайту.

Server - це будь-який єдиний комп'ютер в інтернеті, що дозволяє іншим машинам, грубо кажучи, використовувати себе в якості «посередника» при передачі даних.

Стек (Technical Stack), також відомий як стек технологій - є набором програмних та апаратних засобів, що використовуються для розробки та функціонування веб-додатків, програмного забезпечення або інших інформаційних систем.

Frontend – це публічна частина web-додатків (веб-сайтів), з якою користувач може взаємодіяти і контактувати напряму.

Backend - це частина програмного забезпечення або веб-додатку, яка відповідає за логіку, обробку даних, взаємодію з базою даних, авторизацію користувачів та інші внутрішні процеси, які не видно користувачу.

ВСТУП

Зі стрімким розвитком цифрових технологій та масовим переходом користувачів в онлайн, інтернет змінив свою роль. Тепер це не просто джерело даних, а цілий канал для спілкування, маркетингу товарів та надання послуг. Все більше компаній прагнуть представити свою продукцію в онлайн середовищі. Сучасна онлайн-презентація вже давно вийшла за рамки традиційних рекламних банерів чи публікацій у ЗМІ. Веб-простір активно використовується для розробки інтерактивних платформ, які дозволяють користувачам не лише дізнаватись інформацію, а й здійснювати повноцінні покупки.

Інтернет-магазини сьогодні постачають великий вибір товарів – від звичних речей до спеціалізованих високотехнологічних виробів. Значної ваги онлайн-торгівля набуває серед людей із динамічним ритмом життя, які більшість часу проводять за комп'ютером та не мають можливості витратити час на походи до звичайних магазинів. Для таких клієнтів онлайн-замовлення – це зручне, швидке й ефективне рішення. Особливо, коли мова йде про готові страви, що дозволяє заощадити час і отримати їжу прямо до дверей.

У зв'язку з цим розробка зручної та ефективної інформаційної системи для інтернет-магазину страв японської кухні є надзвичайно важливою. Це дозволяє не тільки спростити процес замовлення для клієнта, а й покращити внутрішні бізнес-процеси.

Актуальність даної теми полягає у збільшенні потреб малого та середнього бізнесу в дієвих засобах онлайн-торгівлі, здатних забезпечити швидке обслуговування клієнтів та управління замовленнями. Особливо це стосується ресторанного бізнесу, де швидкість обробки замовлення та якість представлення продукту відіграють ключову роль. У розробці інформаційної системи для такого інтернет-магазину важливо врахувати специфіку продукту,

простоту оформлення замовлення, адаптивність дизайну та технічне забезпечення.

Об’єкт дослідження – процес автоматизації онлайн-замовлень страв у сфері громадського харчування.

Предмет дослідження – методи, засоби та технології проектування і реалізації інформаційної системи інтернет-магазину страв японської кухні.

Аналіз опублікованих матеріалів та доступних даних свідчить про широку увагу науковців до веб-розробки, покращення інтерфейсів користувача, електронної торгівлі та інформаційних систем у ресторанному бізнесі. У роботах сучасних учених, зокрема в областях веб-інженерії та UX-дизайну, зосереджуються на ефективних методах розробки інтерфейсів, створенні архітектури клієнт-сервер та адаптивному дизайну веб-сторінок, які знаходять застосування у подібних проєктах.

Мета дослідження – розробити інформаційну систему інтернет-магазину страв японської кухні, яка забезпечить зручне замовлення продукції, її ефективне адміністрування та врахує особливості предметної області.

Для досягнення мети необхідно вирішити такі завдання:

- провести аналіз предметної області, дослідити особливості ринку онлайн-замовлень страв;
- визначити функціональні та нефункціональні вимоги до інформаційної системи інтернет-магазину;
- розробити структуру та архітектуру веб-сторінки з урахуванням принципів зручності та ефективності користування;
- реалізувати прототип інтернет-магазину з базовими можливостями для замовлення страв;
- дослідити питання охорони праці при роботі з комп’ютерною технікою у контексті розробки інформаційних систем.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1.Аналіз предметної області

1.1.1 Загальна характеристика предметної області

Торгівля є процесом обміну товарів, послуг, цінностей і грошей. В широкому розумінні, вона представляє собою форму підприємницької діяльності, яка включає купівлю та продаж товарів.

Магазин електронної комерції (інтернет-магазин) - це прикладна система, побудована з використанням технології системи електронної комерції. Як і звичайний магазин, електронний магазин реалізує такі основні функції: презентація товару (послуги) покупцеві, обробка замовлень, продаж і доставка товару. [1]

Інтернет-магазин поєднує в собі елементи прямого маркетингу з образом відвідування традиційного магазину.

Відмінною рисою інтернет-магазинів, якщо порівнювати зі звичними формами торгівлі, є їхня здатність представляти набагато ширший асортимент товарів та послуг. Вони також надають споживачам суттєво більше інформації, потрібної для ухвалення рішення про купівлю. До того ж, комп'ютерні технології дозволяють персоналізувати взаємодію з кожним клієнтом. Врахування історії відвідувань та попередніх покупок дає змогу сформувати індивідуальний підхід. [1]

З розвитком цифрових технологій електронна комерція стала невід'ємною частиною сучасного життя. Все більше людей віддають перевагу онлайн-покупкам завдяки їхній зручності, економії часу та можливості обирати з широкого асортименту товарів, не виходячи з дому. Інтернет-магазини стали не лише платформою для купівлі, але й важливим інструментом для просування брендів, формування лояльності клієнтів і розвитку малого та середнього бізнесу.

Сфера діяльності, що розглядається в даній роботі, стосується онлайн-продажу готових страв японської кухні через інтернет-магазин. Такий тип

комерції поєднує елементи ресторанного сервісу та електронної торгівлі. Основне завдання таких сервісів – забезпечення швидкого доступу до меню, оформлення замовлення, прийому оплати та доставки продукції до клієнта.

Доставка - це процес перевезення товарів, вантажів або листів з одного місця в інше. Вона може здійснюватися різними способами, включаючи автомобільну, залізничну, морську, повітряну або кур'єрську доставку. [1]

Доставка їжі - це процес доставки готової їжі з ресторану або кухні до місця, вказаного замовником. Це зручна послуга, яка дозволяє людям насолоджуватися їжею без необхідності виходити з дому або офісу. [1]

В сучасних умовах коли життя набирає шалених обертів, служби доставки стають надзвичайно актуальними. Вони дозволяють економити час, зменшувати фізичні навантаження на клієнтів та забезпечувати комфорт навіть у найбільш завантажені будні дні. Саме тому доставка готової їжі стала популярною послугою серед офісних працівників, студентів та сімей.

Японська кухня має стабільний попит серед користувачів, особливо у великих містах. Через зростання популярності здорового харчування та доставки додому, значна частина клієнтів обирає саме онлайн-формат замовлення страв.

Японська кухня включає різноманітні страви, серед яких особливе місце займають суші та роли. Суші – це традиційна страва, що складається з рису, приправленого оцтом, у поєднанні з морепродуктами, рибою та овочами. Роли – популярний різновид суші, де начинка обгортається листом норі. Сети – це зібрані у набори комбінації різних суші та ролів, що дозволяють скуштувати одразу декілька смаків у межах одного замовлення. Японська кухня відома балансом смаку, естетикою подачі та натуральністю інгредієнтів.

1.1.2 Характеристика діяльності інтернет-магазину «MykoSushi»

Інтернет-магазин «MykoSushi» має спеціалізуватися на онлайн-продажі страв японської кухні з доставкою безпосередньо до дверей клієнта. Основна концепція даної платформи полягає у наданні користувачам максимально зручного та швидкого способу замовлення улюблених страв. Завдяки інтернет-

магазину кожен бажаючий зможе всього за декілька кліків оформити замовлення на свіжі та якісні суші, роли, сети, а також інші популярні японські страви, отримавши їх у максимально короткі терміни.

«MykoSushi» буде пропонувати широкий асортимент продукції, яка готуватиметься безпосередньо після оформлення замовлення, що гарантує свіжість та високу якість кожної страви. Для приготування використовуватимуться виключно ретельно відібрані інгредієнти преміум-якості, що відповідатимуть сучасним гастрономічним стандартам та автентичним традиціям японської кухні. Особлива увага приділятиметься не лише смаковим властивостям, а й естетичному вигляду страв, адже оформлення є важливою складовою японської гастрономічної культури.

Асортимент інтернет-магазину буде включати як класичні варіації суші з такими популярними інгредієнтами, як лосось, вугор чи тунець, так і оригінальні авторські позиції від шеф-кухаря. Останні створюватимуться з урахуванням локальних гастрономічних вподобань клієнтів, дозволяючи кожному знайти страву за власним смаком. Таким чином, «MykoSushi» прагнучиме поєднати традиційний японський підхід із сучасними кулінарними тенденціями.

Сервіс інтернет-магазину передбачатиме гнучкий функціонал та буде розроблений таким чином, щоб забезпечити комфортне використання для кожного клієнта, навіть якщо він не має великого досвіду користування онлайн-сервісами. Інтерфейс буде простим, зрозумілим та інтуїтивним, що значно спростить навігацію по сторінці.

Гнучкий функціонал системи передбачає наявність зручного кошика для замовлень, де користувачі зможуть легко додавати чи редагувати обрані страви. Також буде передбачена можливість авторизації, що дозволить зберігати історію покупок, спрощуючи подальші замовлення. Крім того, клієнти матимуть змогу залишати коментарі, до страв, ділитися враженнями та допомагати іншим користувачам з вибором. Такий підхід створюватиме більш

довірливу атмосферу навколо сервісу та сприятиме постійному вдосконаленню якості обслуговування.

1.2. Аналіз існуючих рішень

Ринок доставки їжі, особливо японської, стрімко набирає обертів в електронній комерції. Зі зростанням популярності здорового харчування і зручності онлайн-замовлень, все більше людей обирають доставку додому замість відвідування класичних ресторанів. Це сприяє появу великої кількості веб-сервісів та мобільних додатків, що дозволяють замовити їжу з доставкою прямо до дверей, забезпечуючи зручний і швидкий доступ до улюблених страв. Такі сервіси зазвичай застосовують ефективні методи електронної комерції та маркетингу, щоб задовольнити потреби сучасних споживачів.

Одним з яскравих прикладів такого сервісу є інтернет-магазин «Insushi.mk.su», головна сторінка якого зображена на рисунку 1.1. Ця веб-сторінка є представником мережі ресторанів, що спеціалізуються на приготуванні страв японської кухні, і активно працює в Україні, зокрема в Миколаєві.

Інтерфейс цієї сторінки має сучасний адаптивний дизайн, що дозволяє користувачам зручно взаємодіяти з платформою. Зокрема, користувачі можуть:

- ознайомлюватися з меню, згрупованим за різними категоріям (суші, роли, сети, напої тощо);
- додавати страви до кошика для подальшого оформлення замовлення;
- вказувати адресу доставки, щоб страви доставлялися без затримок.

У той час як інтерфейс сторінки має деякі переваги, є кілька аспектів, які потребують покращення. Система має обмежений функціонал, зокрема:

- відсутня корзина для покупок, що ускладнює процес додавання кількох страв до замовлення перед оформленням покупки;
- не передбачено збереження історії замовлень для користувачів, що обмежує можливість повторного замовлення;
- відсутність персональних рекомендацій для користувачів на основі їхніх попередніх замовлень;

- відсутня повноцінна адміністративна панель, що ускладнює управління замовленнями;
- відсутні додаткові можливості для інтерактивної взаємодії з користувачами, такі як коментарі до замовлення чи рейтинг страв.

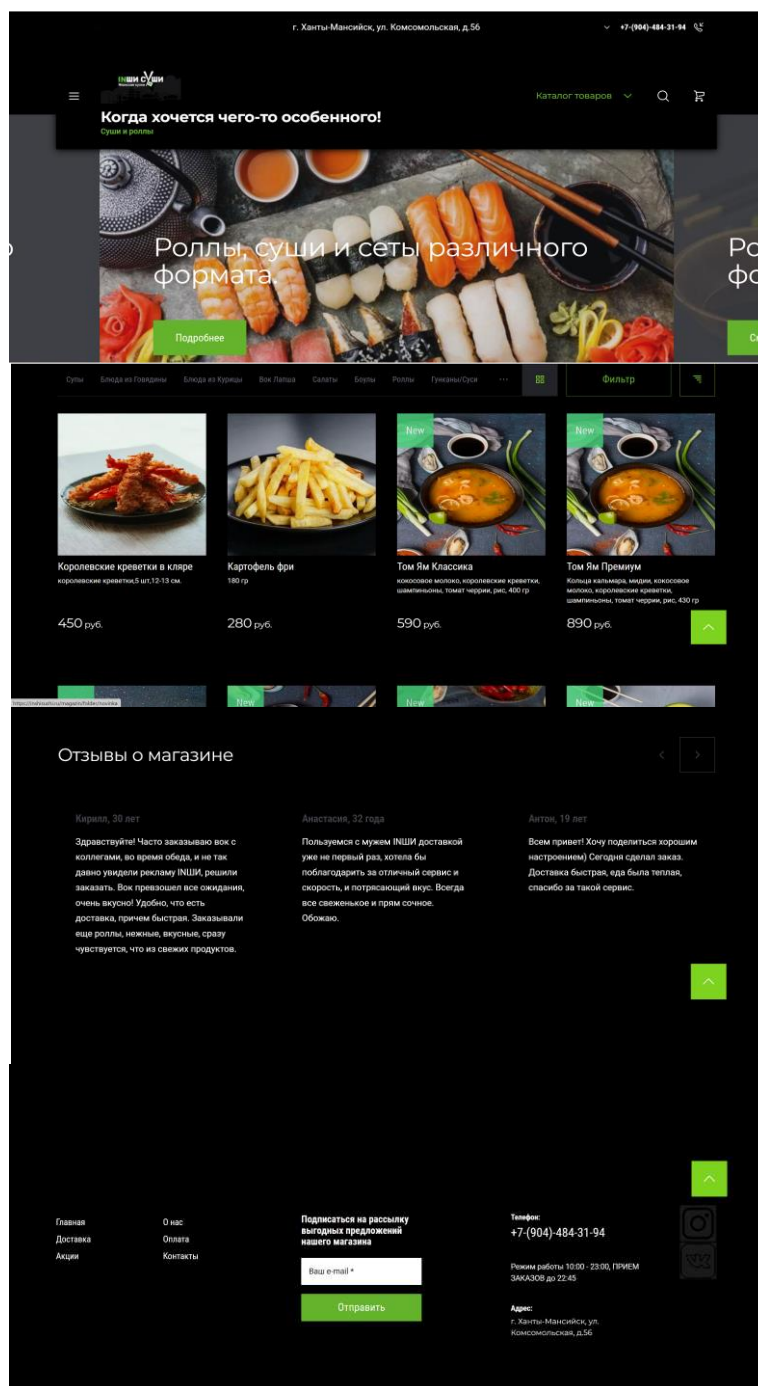


Рисунок 1.1 – Головна сторінка інтернет-магазину «Insushi.mk.su»

Інтернет-магазин «Sushisea.su» є ще одним популярним сервісом

доставки страв японської кухні, який активно працює на ринку України. Це одна з найбільших платформ для замовлення суші і ролів з доставкою додому. Ось деякі з його позитивних сторін:

- магазин регулярно пропонує акції та знижки, що робить його привабливим для бюджетних покупців;
- постійне оновлення асортименту;
- сторінка має сучасний та приємний дизайн, що робить його зручним для перегляду.

Також у цього інтернет-магазину є і негативні сторони. Наприклад як:

- інтерфейс сторінки, хоч і дуже привабливий, але не завжди інтуїтивно зрозумілий;
- відсутність функції кошика ускладнює процес вибору кількох страв для замовлення одночасно, що може призвести до необхідності повторно додавати страви до замовлення.

Інтерфейс головної сторінки інтернет-магазину «Sushisea.su» можна побачити на рисунку 1.2.

1.3.Формулювання проблеми

У процесі аналізу існуючих рішень у сфері онлайн-замовлення страв японської кухні виявлено низку проблем, які суттєво впливають на якість обслуговування та результативність електронної комерції у цій сфері.

По-перше, більшість існуючих інтернет-магазинів мають обмежений набір інструментів для комунікації з користувачами. Наприклад, на багатьох сторінках відсутні функції оцінювання страв або написання відгуків, що значно ускладнює потенційним клієнтам вибір. Сучасний клієнт, згідно з останніми тенденціями розвитку електронної комерції, прагне бачити інструменти соціального підтвердження якості товарів у вигляді рейтингів та відгуків інших споживачів. Таким чином, відсутність такої функціональності створює перешкоди для формування довіри до продукту та знижує ймовірність замовлення.

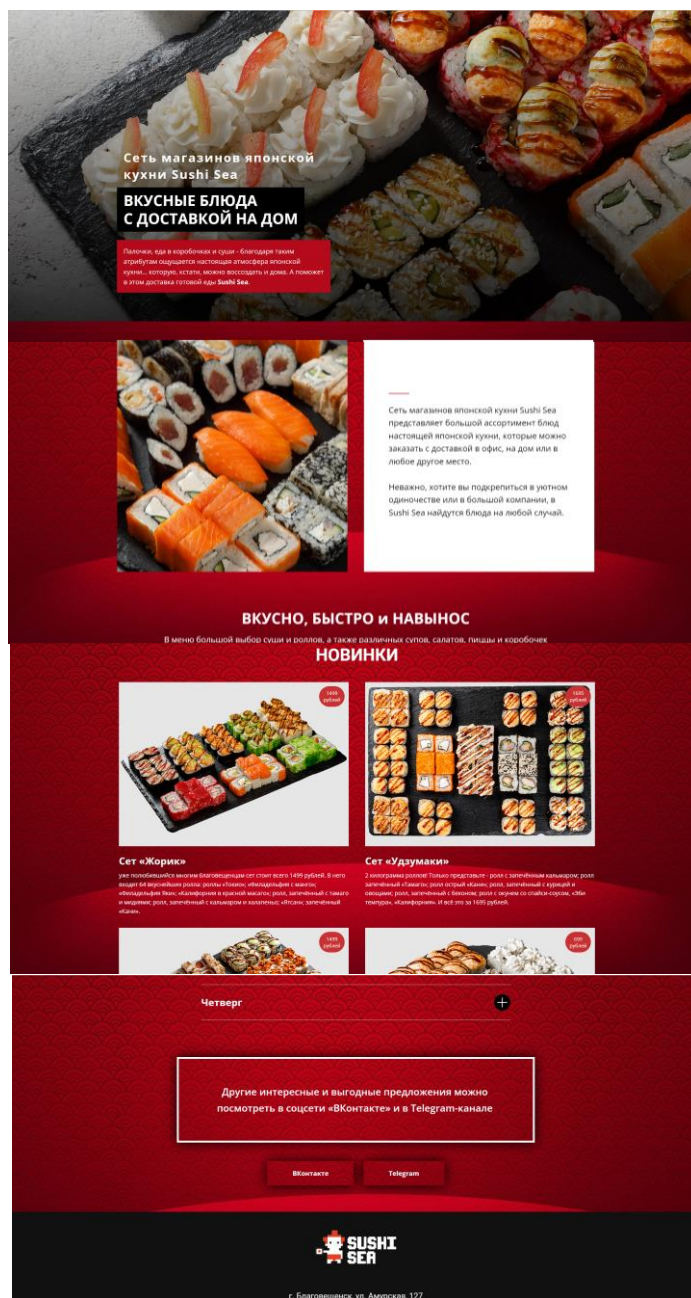


Рисунок 1.2 – Головна сторінка інтернет-магазину «Sushisea.su»

По-друге, інтерфейси багатьох веб-сторінок є перевантаженими або недостатньо інтуїтивно зрозумілі. Це ускладнює навігацію, особливо для нових користувачів або тих, хто не має великого досвіду користування онлайн-сервісами. З огляду на те, що цільовою аудиторією сервісів доставки їжі є максимально широка група споживачів, важливим є забезпечення простоти та логічної зрозумілості роботи з інтернет-магазином.

Третьою значною проблемою є відсутність базових елементів зручності при оформленні замовлення. Зокрема, деякі сервіси не реалізують повноцінного

кошика товарів чи списку обраного, що змушує клієнтів одразу переходити до оформлення, без можливості сформувати набір страв наперед. Це знижує зручність користування веб-сторінкою і може призвести до втрати ймовірних замовлень.

Виходячи з наведеного аналізу, можна сформулювати основні проблеми, які мають бути вирішені при розробці нового інтернет-магазину страв японської кухні «MykoSushi»:

- забезпечення можливості залишення відгуків та оцінок до кожної страви, що сприятиме підвищенню довіри клієнтів та полегшить процес вибору продуктів;
- реалізація простого та інтуїтивно зрозумілого інтерфейсу, що дозволить користувачам швидко орієнтуватися на сторінці, формувати замовлення та отримувати необхідну інформацію;
- впровадження повноцінного функціоналу кошика, який дозволить зберігати обрані страви перед оформленням замовлення;
- можливість зручного внесення оцінок та коментарів через адміністративну панель сторінки, що дасть змогу відображати рейтинги страв на основі реальних відгуків клієнтів.

Таким чином, створення інтернет-магазину «MykoSushi» є необхідним етапом у розвитку існуючого бізнесу доставки страв японської кухні. Нова веб-сторінка дозволить спростити процеси прийому замовлень, поліпшити обслуговування клієнтів, забезпечити зручний інтерфейс для користувачів та підтримку актуального асортименту страв, що є важливим для подальшого зростання компанії та підвищення її ефективності. А також сприятиме підвищенню впізнаваності бренду через активне представлення бізнесу в інтернеті.

1.4.Визначення вимог до інформаційної системи

1.4.1 Загальна характеристика

На основі аналізу предметної області та виявлених проблем існуючих рішень сформульовано вимоги до інформаційної системи інтернет-магазину страв японської кухні «MykoSushi». Вимоги розподіляються на дві основні групи: функціональні та нефункціональні.

1.4.2 Функціональні вимоги

Функціональні вимоги визначають основні можливості, які повинна надавати інформаційна система для забезпечення якісного та ефективного обслуговування користувачів: [2]

- реєстрація та авторизація користувачів з можливістю перегляду власних даних;
- перегляд меню, що структуроване за категоріями (суші, роли, сети тощо);
- пошук страв за назвою або за обраними параметрами;
- можливість додавання страв до кошика для подальшого формування замовлення;
- оформлення замовлення із зазначенням адреси доставки та контактної інформації;
- залишення відгуків та оцінок для кожної страви після оформлення замовлення;
- відображення середнього рейтингу страв на основі отриманих оцінок;
- управління замовленнями через адміністративну панель;
- можливість адміністратором редагувати контент сторінки: додавання нових страв, оновлення меню, управління знижками.

1.4.3 Нефункціональні вимоги

- простий, інтуїтивно зрозумілий інтерфейс для кінцевого користувача;
- адаптивний дизайн сторінки для коректного відображення на різних пристроях (комп'ютери, планшети, смартфони);
- швидке завантаження сторінок навіть при великій кількості товарів та

користувачів;

- забезпечення безпеки персональних даних клієнтів;
- легкість у підтримці та оновленні контенту через адміністративну панель;
- масштабованість системи для подальшого розширення функціоналу в майбутньому.

1.5. Постановка задачі

1.5.1 Загальні відомості

Необхідно розробити інтернет-магазин страв японської кухні «MykoSushi» із гнучким функціоналом та зрозумілим інтерфейсом. Проект призначено для підприємства, яке вже веде діяльність у сфері харчування та потребує ефективного веб-сервісу для підвищення впізнаваності бізнесу в інтернеті та зручного представлення свого асортименту.

1.5.2 Призначення і мета створення системи

Призначенням системи є організація онлайн-продажу страв японської кухні для забезпечення зручного способу оформлення замовлень клієнтами та підвищення впізнаваності бренду за допомогою інтернету.

Метою створення системи є розширення можливостей бізнесу шляхом виходу в онлайн-простір, збільшення кількості замовлень, забезпечення сучасного користувацького досвіду при перегляді та виборі страв японської кухні, а також створення гнучкої платформи для подальшого розвитку.

1.5.3 Вимоги до інтернет-магазину

Інтернет-магазин має бути розроблений як одностороння клієнтська система, орієнтована на простоту взаємодії з користувачем. Сторінка має бути доступна з комп'ютерів різної роздільної здатності і забезпечувати швидке завантаження контенту. Структура веб-сторінки повинна включати головну сторінку, каталог страв із категоріями, сторінки з описами страв, кошик, а також блоки для відображення рейтингів та відгуків від користувачів.

1.5.4 Вимоги до функціональних можливостей інтернет-магазину

Система повинна забезпечувати:

- реєстрацію та авторизацію користувачів;
- каталогізацію страв за категоріями (роли, суші, сети, салати);
- пошук страв за назвою;
- перегляд детальної інформації про страву (фото, опис, ціна, склад);
- додавання страв до кошика;
- перегляд рейтингу страв на основі оцінок клієнтів;
- адміністрування каталогу товарів, категорій та відгуків через адміністративну панель.

1.5.5 Вимоги до інформаційного забезпечення інтернет-магазину

Система повинна працювати з наступними вхідними даними:

- інформація про страви (назва, опис, зображення, ціна, категорія);
- інформація про користувачів;
- інформація про категорії (назва, опис);
- інформація про рейтинг (оцінка).

У вихідних даних ми отримуємо:

- дані про страви;
- дані про користувачів;
- список рейтинговий (оцінка);
- список категорій страв;
- дані про замовлення;
- дані про рейтинг.

1.5.6 Вимоги до лінгвістичного та програмного забезпечення

Для реалізації інтернет-магазину буде використано PERN стек технологій, який забезпечує гнучкість, масштабованість і зручність розробки:

- Frontend: React – бібліотека JavaScript для створення динамічного інтерфейсу користувача, що дозволяє реалізувати швидкий та адаптивний клієнтський інтерфейс;
- Backend: Node.js – серверне середовище виконання JavaScript, що дозволяє створити продуктивний API для взаємодії клієнта з сервером;
- Серверний фреймворк: Express – легкий і гнучкий фреймворк для

Node.js, який забезпечує ефективне керування маршрутами;

- База даних: PostgreSQL – потужна об'єктно-реляційна система керування базами даних, що забезпечує надійне зберігання й обробку даних;
- Системи контролю версій: Git – для відстеження змін у коді та спільної роботи над проєктом;
- Інструменти розробника: npm для керування залежностями та інше.

1.5.7 Вимоги до технічного забезпечення

Для стабільної роботи інтернет-магазину необхідний сервер із наступними характеристиками:

- процесор не нижче Intel Core i5 або AMD Ryzen 5;
- оперативна пам'ять – від 8 ГБ;
- жорсткий диск SSD обсягом не менше 20 ГБ;
- підключення до інтернету зі швидкістю не менше 10 Мбіт/с;
- клієнтські пристрої: сучасні браузері на ПК.

1.5.8 Вимоги до документації

До складу проєктної документації мають входити:

- загальний опис системи;
- керівництво користувача;
- інструкція з наповнення та супроводу бази даних;
- інструкція з розгортання проєкту;
- тест-план системи.

1.5.9 Порядок контролю та приймання системи

Перевірка функціональності системи виконується розробником шляхом модульного і комплексного тестування з залученням тестових сценаріїв. Після завершення тестування системи передається замовнику для фінального приймального тестування. У разі відповідності системи заявленим вимогам підписується акт приймання проєкту. [2]

1.6 Висновок до першого розділу

У першому розділі було проведено аналіз предметної області інтернет-магазину страв японської кухні. Визначено ключові проблеми, зокрема

відсутність зручної системи для обробки замовлень, управління товарами та взаємодії з клієнтами. Було сформульовано основні вимоги до майбутньої інформаційної системи, як функціональні, так і нефункціональні. Також запропоновано загальну ідею автоматизованого рішення, яке допоможе покращити роботу магазину та зробить процес обслуговування більш зручним та ефективним.

РОЗДІЛ 2. РОЗРОБКА ПРОЄКТНИХ РІШЕНЬ

2.1. Загальносистемні рішення

2.1.1 Загальна характеристика інформаційної системи

Веб-сторінка інтернет-магазину страв японської кухні реалізована як інформаційно-функціональний ресурс, що дозволяє користувачам переглядати наявний асортимент, додавати обрані страви до кошика, а також проходити процедуру реєстрації або авторизації.

Головною метою цієї сторінки є забезпечення зручної подачі інформації про продукцію, полегшення процесу ознайомлення з меню, а також формування замовлення за допомогою кошика. Усі дії на сторінці здійснюються в рамках простого та інтуїтивно зрозумілого інтерфейсу, розрахованого як на нових, так і на постійних користувачів.

Ключовий функціонал включає:

- каталог продукції із поділом на категорії (суші, роли, сети тощо);
- авторизація та реєстрація, необхідні для створення індивідуального кошика;
- додавання страв до кошика, який зберігається під конкретним користувачем;
- можливість редагування вмісту кошика перед оформленням замовлення;
- адміністративна частина, що дозволяє змінювати асортимент, оновлювати ціни, редагувати опис або фото;
- видалення або оновлення даних про страви в разі зміни меню чи складу.

Веб-сторінка забезпечує достатній рівень взаємодії через просту логіку користування: від перегляду до замовлення. Така структура дозволяє ефективно просувати у мережі продукти японської кухні.

2.1.2 Вибір засобів моделювання

Проектування програмного забезпечення є ключовим етапом у процесі створення інформаційної системи, що передбачає формалізацію вимог, розробку архітектури та вибір оптимальних рішень для реалізації функціоналу майбутнього продукту. У межах цього проєкту для моделювання системи було використано об'єктно-орієнтовану парадигму проектування, яка дозволяє представити логіку роботи системи через взаємодію об'єктів, що відповідають реальним сутностям предметної області. [2]

Для візуалізації структури та динаміки роботи системи застосовувалась UML (Unified Modeling Language) – уніфікована мова моделювання, яка дозволяє створювати чіткі та зрозумілі діаграми для розробників і замовників. Створення моделей здійснювалось за допомогою спеціалізованого програмного забезпечення Rational Rose, яке забезпечує побудову діаграм класів, випадків використання, послідовностей та інших структурних елементів, необхідних для повного опису майбутньої системи.

2.1.3 Розробка контекстної моделі системи

Проектування програмного забезпечення традиційно розпочинається з побудови контекстної діаграми, яка дозволяє окреслити загальні межі системи, її основні функції та взаємодії з зовнішнім середовищем. Такий підхід дає змогу на ранньому етапі візуалізувати ролі учасників і визначити загальну структуру комунікацій між системою та її користувачами. [2]

У рамках даного проєкту були виділені дві ключові діючі особи:

- Користувач – особа, яка переглядає каталог, формує замовлення через кошик;
- Адміністратор – особа, відповідальна за оновлення інформації на веб-сторінці, керування асортиментом.

Контекстну діаграму, яка ілюструє ці взаємозв'язки, наведено на рисунку 2.1.

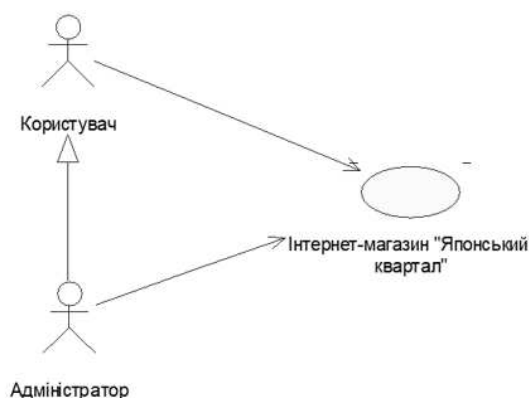


Рисунок 2.1 – Контекстна діаграма для веб-сторінки інтернет-магазину страв японської кухні

2.1.4 Розробка діаграми варіантів використання

Для формалізації функцій, які буде мати майбутня система, було побудовано діаграму варіантів використання. Завдяки цій діаграмі можна чітко зрозуміти, що може робити кожен тип користувача та логіку взаємодії із системою. Це полегшує глибокий аналіз майбутніх задач під час розробки та тестування веб-сторінки. [3]

Створена діаграма варіантів використання має такі зв'язки між акторами та варіантами використання:

- include (включення) – вказує на обов'язкову дію, що є складовою іншої (наприклад, авторизація перед управлінням контентом);
- extend (розширення) – позначає необов'язкові, але доступні за певних умов дії (наприклад, застосування фільтрів до перегляду страв).

Діаграми варіантів використання в контексті проєкту, крім загального візуального подання функціональності, також дають змогу:

- визначити ключові дії, які маж реалізовувати веб-сторінка для взаємодії з користувачами;
- зрозуміти, які ролі виконують різні типи користувачів (наприклад, відвідувач, зареєстрований клієнт, адміністратор) та як саме вони взаємодіють із системою;
- зобразити можливі варіанти роботи зі сторінкою (від перегляду страв

до додавання їх до кошика), встановити їх порядок та залежності;

– визначити ключові функції сторінки, що є найважливішими для ефективного користування системою з боку цільової аудиторії.

Діаграму варіантів використання наведено на рисунку 2.2.

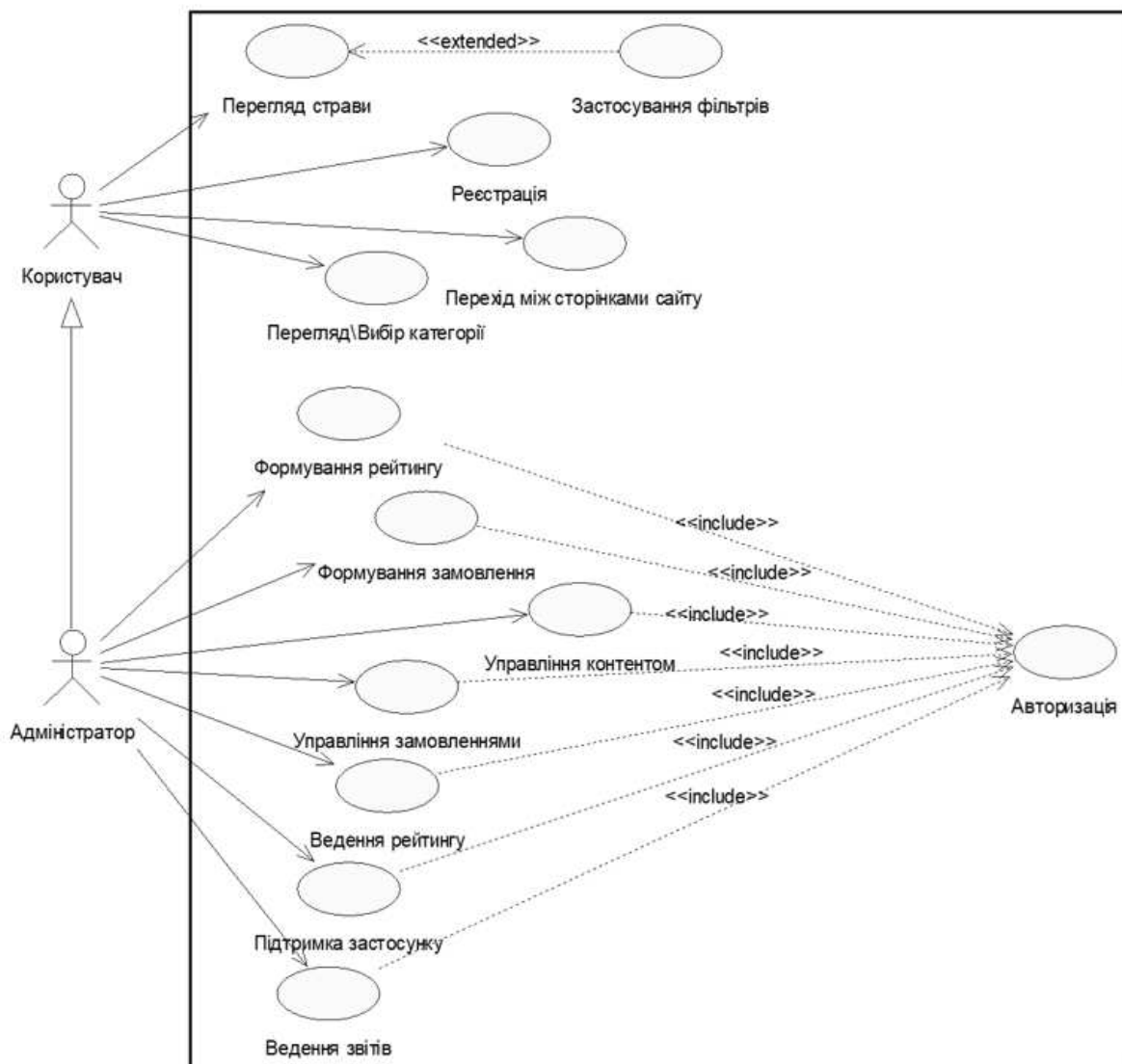


Рисунок 2.2 – Діаграма варіантів використання інтернет-магазину страв японської кухні

Специфікації варіантів використання представлені в таблицях 2.1-2.10.

Таблиця 2.1 – Специфікація прецеденту «Авторизація»

Введення задач	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу авторизуватися у системі.
Передумови	Користувач повинен ввести адресу електронної пошти, яка вже

	була зареєстрована
Основний потік подій	Користувач ініціює заповнення форми авторизації, система перевіряє валідність введених даних і входить у все існуючий акаунт.
Альтернативний потік подій	Система пропонує можливість повторити введення або завершити варіант використання.
Постумови	Не визначені.

Таблиця 2.2 – Специфікація прецеденту «Перегляд інформації про страву»

Налаштування	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу переглядати інформацію про товар на сторінці
Передумови	Не визначені
Основний потік подій	Користувач ініціює перехід по гіперпосиланню деякого товару
Альтернативний потік подій	Не визначено
Постумови	Не визначені.

Таблиця 2.3 – Специфікація прецеденту «Перехід між сторінками сторінки»

Налаштування	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу переглядати сторінки сторінки
Передумови	Не визначені
Основний потік подій	Користувач ініціює перехід, натискає гіперпосилання на інші сторінки сторінки, система оброблює запит
Альтернативний потік подій	Не визначено
Постумови	Не визначені.

Таблиця 2.4 – Специфікація прецеденту «Перегляд\Вибір категорії»

Налаштування	
Складова	Опис

1	2
Короткий опис	Даний варіант використання дозволяє користувачу переглядати категорії страв
Передумови	Не визначені
Основний потік подій	Користувач ініціює перехід, натискає гіперпосилання у полі категорії, система оброблює запит
Альтернативний потік подій	Не визначено
Постумови	Не визначені.

Таблиця 2.6 – Специфікація прецеденту «Формування рейтингу»

Налаштування	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу переглядати й надавати рейтинг товару
Передумови	Користувач повинен бути авторизованим у системі
Основний потік подій	Користувач ініціює перехід до опису товару, система виводить результат
Альтернативний потік подій	Не визначено
Постумови	Не визначені.

Таблиця 2.7 – Специфікація прецеденту «Формування замовлення»

Налаштування	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу сформувати своє замовлення
Передумови	Користувач повинен бути авторизованим у системі
Основний потік подій	Користувач ініціює додання товарів до замовлення, система оброблює запит
Альтернативний потік подій	Не визначено
Постумови	Не визначені.

Таблиця 2.8 – Специфікація прецеденту «Управління замовленнями»

Налаштування

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє адміністратору управляти замовленням користувача
Передумови	Адміністратор повинен бути авторизованим у системі
Основний потік подій	Адміністратор ініціює перехід до БД, змінює дані про замовлення, система обробляє запит
Альтернативний потік подій	Не визначено
Постумови	Не визначені.

Таблиця 2.9 – Специфікація прецеденту «Управління контентом»

Налаштування	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє адміністратору управляти контентом на сторінці
Передумови	Адміністратор повинен бути авторизованим у системі
Основний потік подій	Адміністратор ініціює перехід до гіперпосилання «Адмін-панель», у якому змінює, додає, оновлює, видаляє контент у інтернет-магазині, система оброблює запит
Альтернативний потік подій	Не визначено
Постумови	Не визначені.

Таблиця 2.10 – Специфікація прецеденту «Ведення рейтингу»

Налаштування	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє адміністратору вести й додавати рейтинг товару, залежно від оцінки користувачами
Передумови	Адміністратор повинен бути авторизованим у системі
Основний потік подій	Адміністратор ініціює перехід до бази даних, змінює дані про рейтингову оцінку, система оброблює запит
Альтернативний потік подій	Не визначено
Постумови	Не визначені.

2.2. Рішення з інформаційного забезпечення

2.2.1 Визначення сутностей інформаційної підсистеми

Для коректного функціонування веб-ресурсу необхідно чітко визначити ключові елементи системи, які відображають логіку її роботи та дозволяють реалізувати всі необхідні процеси з обробки, зберігання та відображення даних. Визначено такі основні сутності:

- користувач – представляє особу, яка взаємодіє із системою. Зберігає основні дані, такі як ім'я, пошта, пароль, роль та адреса доставки;
- страва – ключова одиниця асортименту. Містить такі характеристики, як назва, ціна, вага, склад. Одна страва може належати до певної категорії та входити до декількох замовлень;
- категорія – визначає приналежність страв до певної групи (наприклад, роли, суші, сети). Категорії допомагають структурувати каталог для зручного перегляду;
- замовлення – сутність, що відображає сформовану покупку. Містить дату, час, статус, а також інформацію про страви, що входять до нього, та користувача, який його оформив;
- рейтинг – дозволяє користувачам оцінювати якість страв. Містить числову оцінку та текстовий коментар, прив'язаний до певного користувача;
- статус – використовується для опису етапу обробки замовлення;
- адреса доставки – інформація, яку надає користувач при оформленні замовлення. Пов'язана з його обліковим записом і використовується для виконання доставки.

2.2.2 Розробка концептуальної моделі даних

Для більш чіткого уявлення про структуру предметної області та взаємозв'язки між її основними об'єктами було побудовано концептуальну модель. Вона відображає ключові сутності, які беруть участь у функціонуванні інтернет-магазину, та їхні зв'язки. [4] Такий підхід дозволяє сформулювати узагальнене уявлення про логіку зберігання та обробки даних у майбутній системі.

Концептуальну модель інтернет-магазину страв японської кухні наведено на рисунку 2.3.

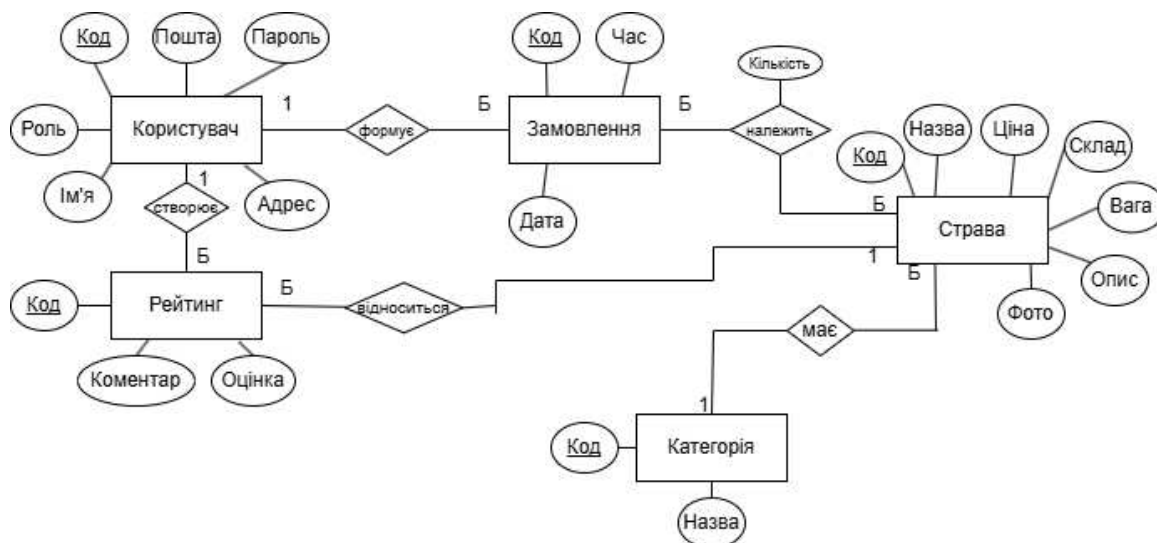


Рисунок 2.3 – Концептуальна модель інтернет-магазину страв японської кухні

2.2.3 Розробка логічної моделі даних

Логічна модель даних – це детальний опис того, як інформація буде зберігатися в базі даних. На цьому етапі визначаються таблиці, поля в них, типи даних (наприклад, число, текст, дата), а також зв'язки між таблицями. [4]

На основі концептуальної моделі були створені таблиці для кожної сутності: користувачів, замовлень, страв, категорій, рейтингів тощо. Для кожної таблиці зазначено унікальний ідентифікатор (первинний ключ), а також зовнішні ключі для зв'язків з іншими таблицями. Також враховано, які дані обов'язкові для заповнення, а які – ні.

Логічна модель допомагає побудувати зручну та зрозумілу структуру бази даних, щоб система працювала правильно і без помилок. У майбутньому саме за цією схемою буде створена фізична база даних. Логічна модель даних представлена на рисунку 2.4.

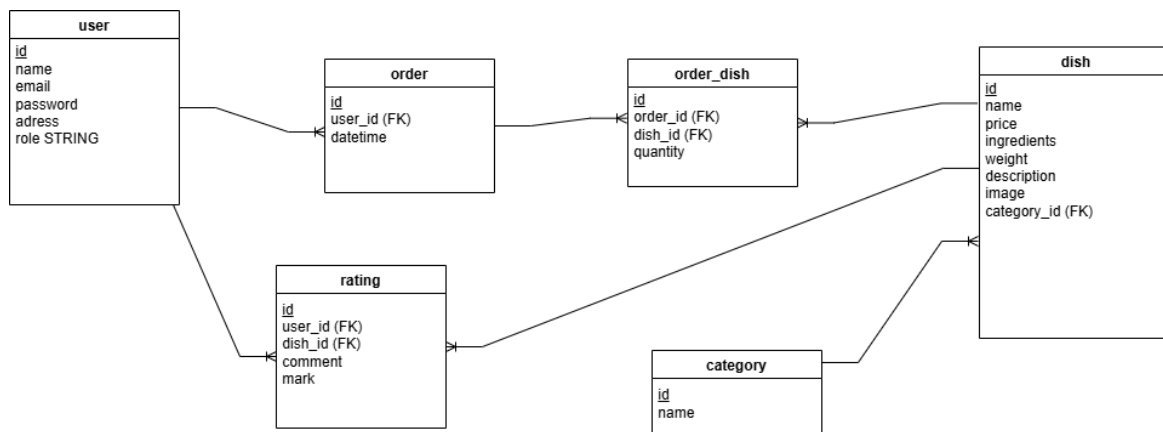


Рисунок 2.4 – Логічна модель інтернет-магазину страв японської кухні

2.2.4 Розробка фізичної моделі даних

Фізична модель даних – це схема, яка описує, як саме дані зберігатися в базі даних на практиці. У цій моделі враховані всі технічні деталі: назви таблиць і полів, типи даних, ключі, зв'язки між таблицями та інші обмеження. [4]

PostgreSQL відноситься до реляційних систем керування базами даних.

Реляційна база даних – це база даних, яка базується на реляційній моделі даних. Використання реляційних баз даних було запропоновано доктором Коддом з IBM в 1970 році. Реляційні СКБД використовуються для роботи з реляційними базами даних. [4]

Основною ідеєю реляційної моделі є використання таблиць зі структурованими рядками та стовпцями для представлення даних.

Для кожної сутності з логічної моделі створено таблицю з конкретними полями. Наприклад:

- у таблиці Користувач зберігаються ім'я, пошта, пароль, адреса доставки, роль тощо;
- у таблиці Страва є назва, ціна, вага, склад і посилання на категорію;
- таблиця Замовлення містить дату, час, статус і зв'язок із користувачем;

Рейтинг включає оцінку та коментар до конкретної страви від певного користувача.

Усі поля мають чітко визначені типи даних, наприклад:

- VARCHAR – для текстових значень (імена, назви, пошта);
- INT – для числових полів (коди, оцінки, ролі);
- DECIMAL – для зберігання цін;
- DATETIME – для запису дати й часу.

Також встановлені первинні ключі для ідентифікації записів та зовнішні ключі для зв'язків між таблицями. Це допомагає зберігати логіку між даними та не допустити помилок при їх використанні.

Фізична модель даних представлена у вигляді сутностей з атрибутами на рисунку 2.5.

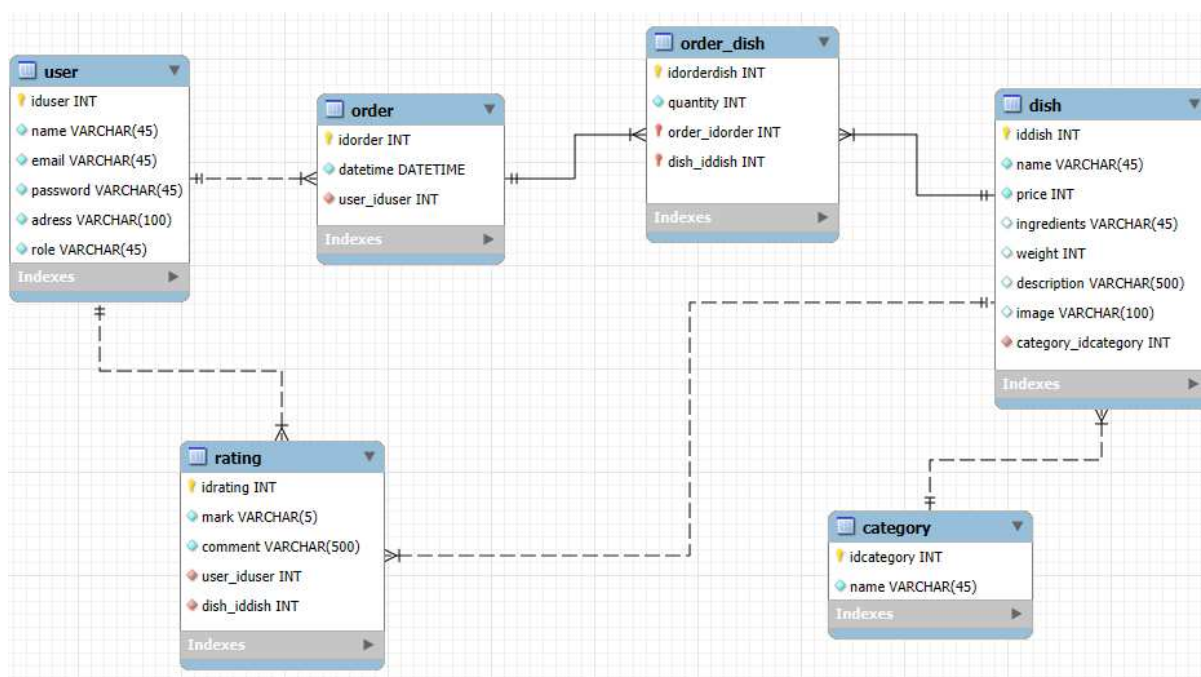


Рисунок 2.5 – Фізична модель бази даних до інтернет-магазину страв японської кухні

У таблицях 2.11 – 2.16 наведено стовпці для всіх таблиць системи відповідно з вказанням типу даних та виду ключа.

Таблиця 2.11 Специфікація до сутності «Користувач»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
iduser	PK	INT	NOT NULL
name		VARCHAR(45)	NOT NULL
email		VARCHAR(45)	NOT NULL
password		VARCHAR(45)	NOT NULL
adress		VARCHAR(100)	NOT NULL
role		VARCHAR(45)	NOT NULL

Таблиця 2.12 Специфікація до сутності «Замовлення»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
idorder	PK	INT	NOT NULL
datetime		DATETIME	NOT NULL
user_iduser	FK	INT	NOT NULL

Таблиця 2.13 Специфікація до сутності «Замовлення_Страва»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
idorderdish	PK	INT	NOT NULL
quantity		INT	NOT NULL
order_idorder	FK	INT	NOT NULL
dish_iddish	FK	INT	NOT NULL

Таблиця 2.14 Специфікація до сутності «Страва»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
iddish	PK	INT	NOT NULL
name		VARCHAR(45)	NOT NULL
price		INT	NOT NULL
ingredients		VARCHAR(45)	
weight		INT	
description		VARCHAR(500)	
image		VARCHAR(100)	
category_idcategory	FK	INT	NOT NULL

Таблиця 2.15 Специфікація до сутності «Рейтинг»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
idrating	PK	INT	NOT NULL
mark		VARCHAR(5)	NOT NULL
comment		VARCHAR(500)	NOT NULL
user_iduser	FK	INT	NOT NULL
dish_iddish	FK	INT	NOT NULL

Таблиця 2.16 Специфікація до сутності «Категорія»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
idcategory	PK	INT	NOT NULL
name		VARCHAR(45)	NOT NULL

2.3. Рішення з технічного забезпечення

2.3.1 Вимоги до технічного забезпечення

Вимоги до технічного забезпечення, узагальнені у підрозділі 1.5.7, детальніше розглядаються нижче з урахуванням як серверної, так і клієнтської частин інформаційної системи.

Щоб забезпечити стабільну роботу інформаційної системи, яка реалізована у вигляді веб-сторінки страв японської кухні, комп'ютер користувача повинен відповідати базовим технічним вимогам. Це дозволить швидко завантажувати веб-сторінки, переглядати каталог товарів, додавати страви до кошика та взаємодіяти із системою без затримок.

Рекомендовані параметри пристрою:

- операційна система: macOS, Microsoft Windows або Linux;
- процесор Intel Core i3 або AMD Ryzen 3;
- об'єм оперативної пам'яті: від 4 ГБ;
- вільне місце на жорсткому диску: не менше 1 ГБ.

2.3.2 Склад технічного забезпечення

Користувачеві потрібно мати пристрій із доступом до мережі Інтернет – це може бути персональний комп'ютер або ноутбук. Обов'язкова наявність сучасного веб-браузера, що підтримує HTML5, CSS3 та JavaScript (наприклад, Google Chrome, Mozilla Firefox або Microsoft Edge та інші). На пристрої також має бути встановлена підтримка cookies, яка потрібна для коректної роботи з кошиком та авторизацією.

Крім того, система потребує серверного середовища для розміщення веб-сторінки та обробки запитів користувача. Сервер має підтримувати роботу з базами даних, сучасні мови програмування, а також мати достатню продуктивність:

- процесор не нижче Intel Core i5 або AMD Ryzen 5;
- оперативна пам'ять – від 8 ГБ;
- твердотільний накопичувач (SSD) обсягом від 20 ГБ;
- стабільне інтернет-з'єднання зі швидкістю не менше 10 Мбіт/с.

Саме серверна частина відповідає за обробку замовлень, збереження інформації про користувачів, рейтинги, категорії та інші сутності системи.

2.4. Рішення з програмного забезпечення

2.4.1 Проєктування програмного забезпечення

У процесі розробки інтернет-магазину страв японської кухні особлива увага приділялася проєктуванню програмного забезпечення, яке забезпечує зручну, безпечну та ефективну взаємодію користувача з системою. Щоб досягти цього, було використано сучасні методи об'єктно-орієнтованого аналізу та проєктування, які дозволяють краще зрозуміти структуру та логіку роботи системи.

Зокрема, було побудовано UML-діаграми, що стали основною для реалізації функціональних та інформаційних процесів.

Загалом проєктування програмної частини базується на принципах зручності, надійності та гнучкості, що дозволяє масштабувати систему в майбутньому без суттєвих змін у базовій архітектурі. [2]

2.4.2 Діаграми взаємодії

Для моделювання поведінки користувачів інтернет-магазину страв японської кухні були використані діаграми взаємодії, що дозволяють наочно представити процеси обміну повідомленнями між об'єктами системи у відповідь на певні дії. Основну увагу зосереджено на діаграмах послідовності оскільки вони дають змогу відобразити логіку виконання ключових сценаріїв у чітко визначеній часовій послідовності. [3]

Крім того, для уточнення структурних зв'язків між об'єктами в межах цих сценаріїв застосовано діаграму кооперації. Вона дає змогу не лише простежити послідовність виклику методів, але й проаналізувати топологію зв'язків між об'єктами, що бере участь у взаємодії. Це забезпечує краще розуміння взаємозалежностей між елементами системи, що важливо для побудови якісної архітектури програмного забезпечення.

Загалом, діаграми взаємодії відіграють важливу роль у плануванні логіки функціонування інтернет-магазину, дозволяючи своєчасно виявити можливі

помилки в логіці виконання процесів та оптимізувати структуру взаємодії між компонентами. [3]

Діаграма послідовності для суб'єкта «Користувач» при виконанні варіанта використання «Реєстрація» представлена на рисунку 2.6.



Рисунок 2.6 – Діаграма послідовності для суб'єкта «Користувач» при виконанні варіанта використання «Реєстрація»

Діаграма послідовності для суб'єкта «Користувач» при виконанні варіанта використання «Профіль» представлена на рисунку 2.7.

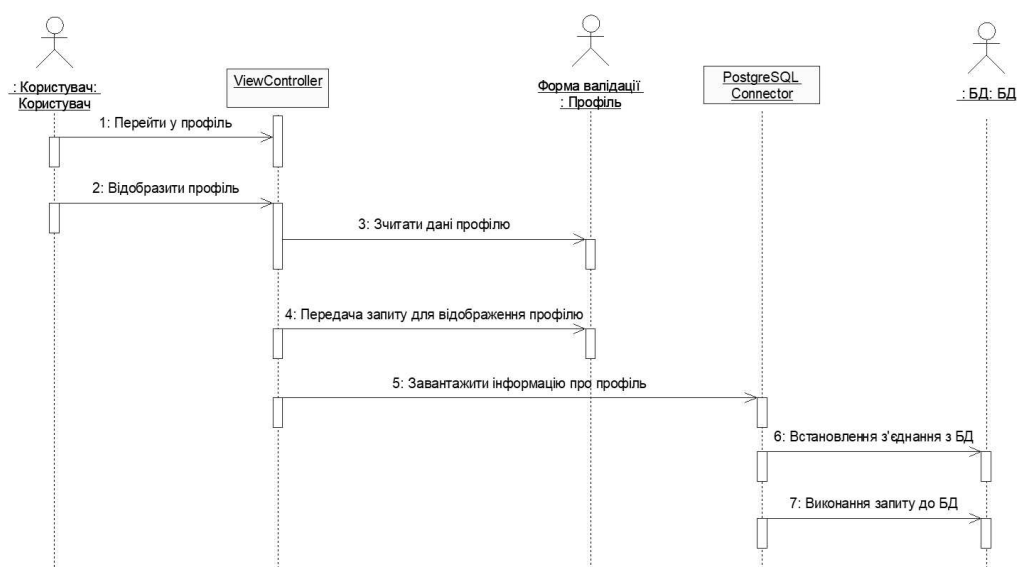


Рисунок 2.7 – Діаграма послідовності для суб'єкта «Користувач» при виконанні варіанта використання «Профіль»

Діаграма послідовності для суб'єкта «Користувач» при виконанні варіанта використання «Формування замовлення» представлена на рисунку 2.8.

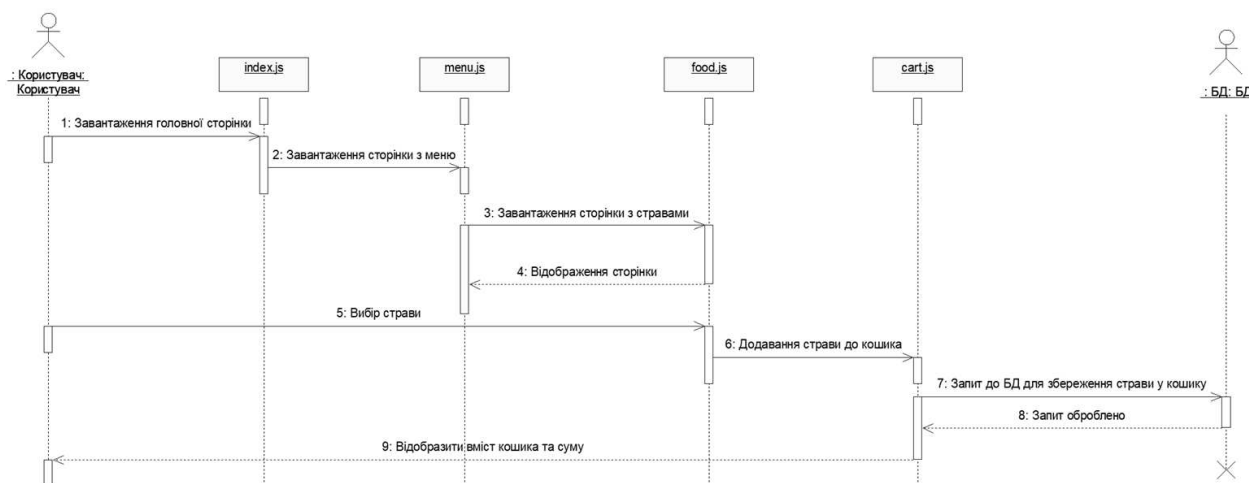


Рисунок 2.8 – Діаграма послідовності для суб'єкта «Користувач» при виконанні варіанта використання «Формування замовлення»

2.4.3 Діаграми діяльності

Діаграма діяльності використовується для моделювання логіки виконання окремих бізнес-процесів у системі. Вона дозволяє представити послідовність дій, які виконуються користувачами та системою при реалізації певного функціоналу, що особливо корисно для аналізу та оптимізації процесів. [3]

У межах розробки інтернет-магазину страв японської кухні були побудовані діаграми діяльності для таких варіантів використання:

- Авторизація адміністратора – демонструє процес входу до системи адміністративного користувача. Починається з введення логіна й пароля, далі система перевіряє правильність введених даних, і в разі успіху надає доступ до панелі керування (рисунок 2.9).

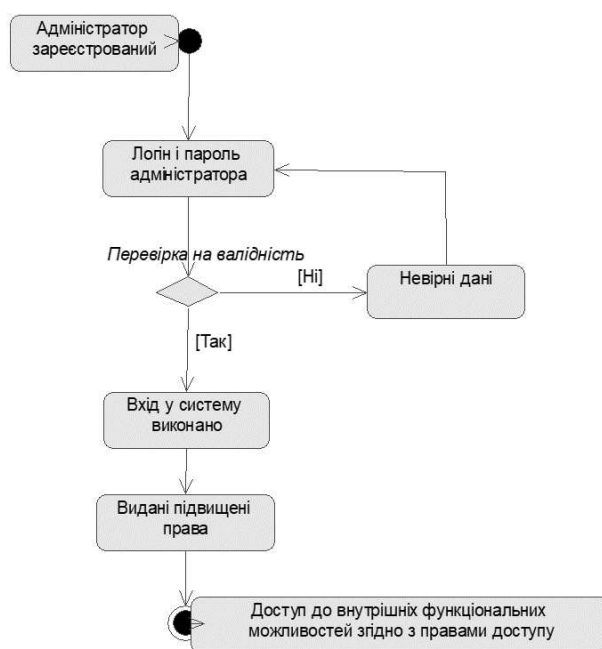


Рисунок 2.9 – Діаграма діяльності для варіанта використання «Авторизація»

– Введення рейтингу – відображає, як користувач після оформлення замовлення може оцінити страву. Після вибору страви для оцінювання користувач виставляє рейтинг, який фіксується у системі, що дозволяє надалі впливати на загальну оцінку продукту. Дану діаграму діяльності можна побачити на рисунку 2.10.

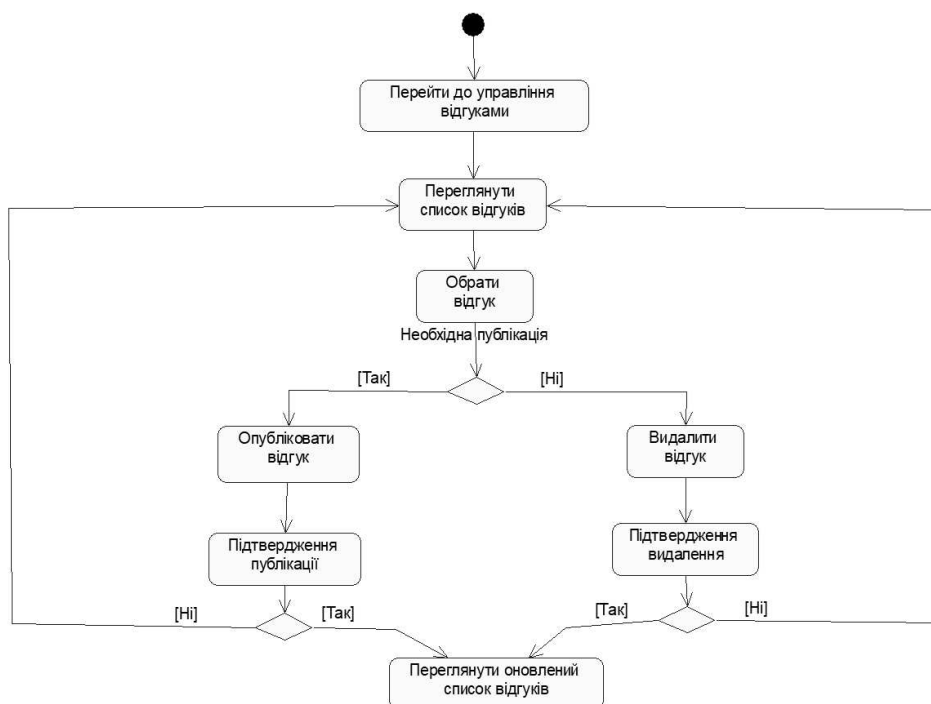


Рисунок 2.10 – Діаграма діяльності для варіанта використання «Введення рейтингу»

2.4.4 Розробка діаграми класів

Для представлення взаємозв'язків між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описання їх внутрішньої структури і типи відношень, було побудовано діаграму класів.

Діаграма класів для інтернет-магазину страв японської кухні представлена на рисунку 2.11.

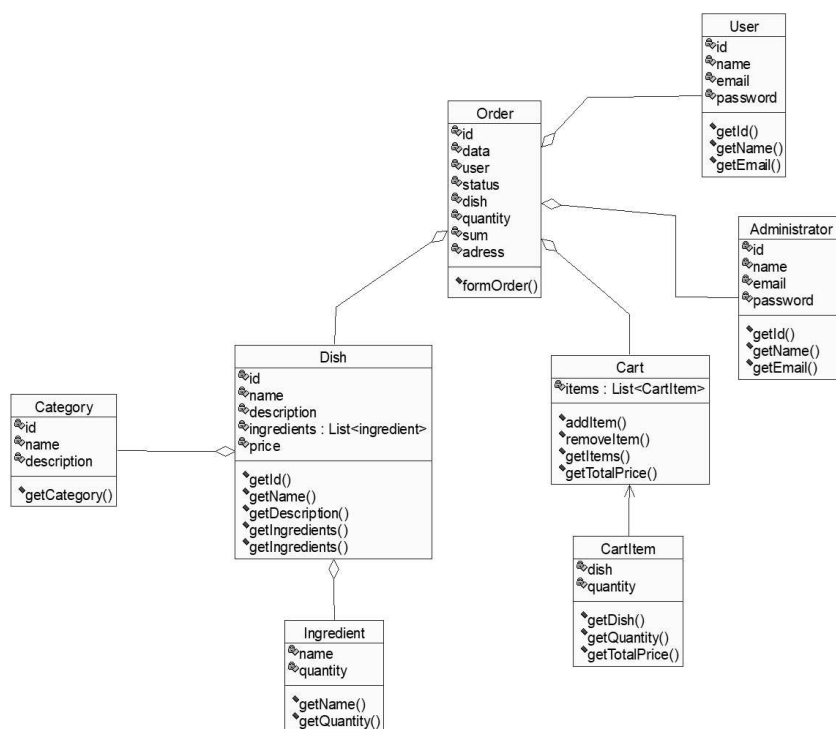


Рисунок 2.11 – Діаграма класів для інтернет-магазину з продажі страв японської кухні

У цій діаграмі класів ми маємо основний клас Order. Методи цього класу дозволяють сформувати замовлення.

Клас Dish представляє окрему страву, зі своїм ідентифікатором (id), назвою (name), описом (description), списком інгредієнтів (ingredients) та ціною (price). Методи цього класу надають доступ до властивостей страви.

Клас Cart представляє кошик зі списком елементів (items). Методи цього класу дозволяють додавати та видаляти елементи з кошика, отримувати список елементів та загальну вартість кошика.

Клас `CartItem` представляє окремий елемент у кошику, який складається з об'єкту страви (`dish`) та кількості (`quantity`). Цей клас має методи для отримання даних про елемент кошика.

Клас `Ingredient` представляє окремий інгредієнт і має властивості `id` (ідентифікатор) та `name` (назва). Цей клас дозволяє представляти інгредієнти, які використовуються у стравах.

Клас `User` представляє користувача магазину і має властивості та методи, пов'язані з користувачем, такі як `id` (ідентифікатор), `name` (ім'я), `email` (електронна пошта) і т.д. Цей клас надає можливості для реєстрації, авторизації та керування обліковими записами користувачів.

Клас `Administrator` представляє адміністратора магазину і має властивості та методи, специфічні для адміністратора, такі як `id` (ідентифікатор), `name` (ім'я), `email` (електронна пошта) і т.д. Цей клас надає можливості для керування та адміністрування магазином, включаючи додавання та видалення страв, управління замовленнями та інше.

2.4.5 Розробка діаграми розгортання системи

Діаграма розгортання використовується для візуалізації фізичної архітектури інформаційної системи, зокрема для демонстрації того, як програмні компоненти розміщені на різних апаратних вузлах. Така діаграма є важливою частиною проєктування, оскільки дозволяє краще зрозуміти, яким чином реалізується зв'язок між серверами, клієнтськими пристроями та іншими технічними ресурсами. [3]

У межах створення інтернет-магазину страв японської кухні побудована діаграма розгортання, яка ілюструє ключові елементи системи: клієнтський пристрій (браузер користувача), веб-сервер, сервер додатків і сервер бази даних. Кожен з компонентів виконує свою роль у забезпеченні стабільної та швидкої роботи веб-сторінки.

Дана діаграма відображає наступні важливі аспекти:

- пристрої, за допомогою яких здійснюється доступ до сторінки;
- з'єднання між компонентами системи;

- розташування різних компонентів системи;
- канали передачі даних між користувачем і сервером (HTTP/HTTPS-з'єднання).

Такий підхід дає змогу заздалегідь оцінити навантаження на різні частини системи, передбачити можливі проблемні зони та спланувати інфраструктурні ресурси для розширення у майбутньому. Діаграма також слугує корисним інструментом для адміністраторів та розробників при налаштуванні серверного середовища та впровадженні нових функцій.

Діаграма розгортання інтернет-магазину зображена на рисунку 2.12.

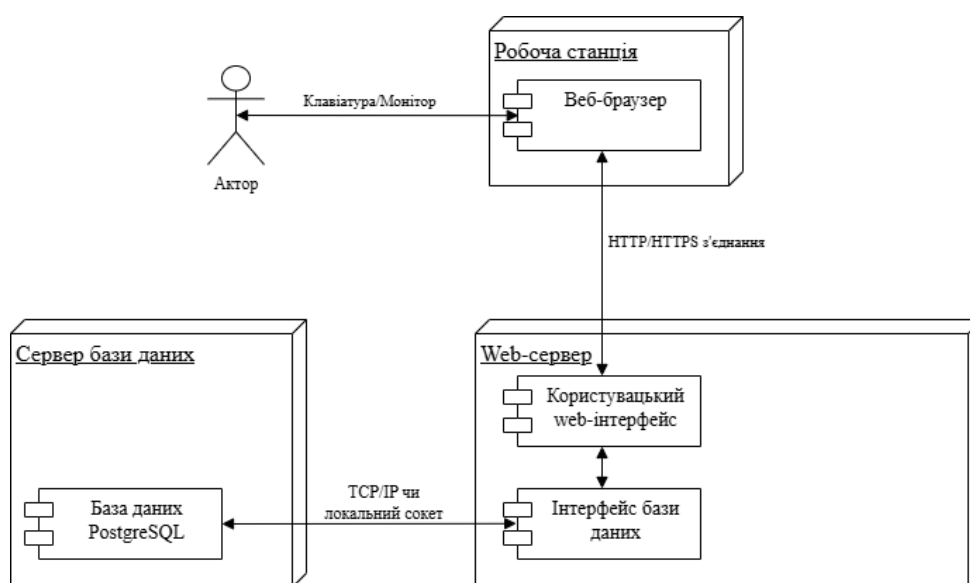


Рисунок 2.12 – Діаграма розгортання системи інтернет-магазину страв японської кухні

2.5. Висновок до другого розділу

У цьому розділі було сформовано основні проєктні рішення для розробки веб-сторінки інтернет-магазину страв японської кухні. Визначено загальну характеристику інформаційної системи, обґрунтовано вибір засобів моделювання та побудовано діаграми: контекстну, варіантів використання, класів, діяльності та розгортання системи.

Було розроблено концептуальну, логічну та фізичну моделі даних, що забезпечують структуроване зберігання інформації про користувачів, замовлення та асортимент продукції. Визначено вимоги до інформаційного, технічного й програмного забезпечення, а також склад апаратного середовища.

Обрані технології та архітектурні рішення забезпечують масштабованість, безпечну обробку даних, ефективну взаємодію з користувачем. Результати цього етапу створили надійну основу для реалізації функціональної і зручної веб-сторінки.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1. Аргументація вибору засобів розробки

Для реалізації інформаційної системи інтернет-магазину страв японської кухні було обрано сучасний стек технологій, який забезпечує високу продуктивність, масштабованість та простоту підтримки системи. Архітектура програмного забезпечення базується на поділі функціоналу між клієнтською та серверною частинами, з використанням необхідних фреймворків, бібліотек та СКБД.

В якості засобу розробки інформаційної системи було обрано середовище Visual Studio Code (рисунок 3.1).

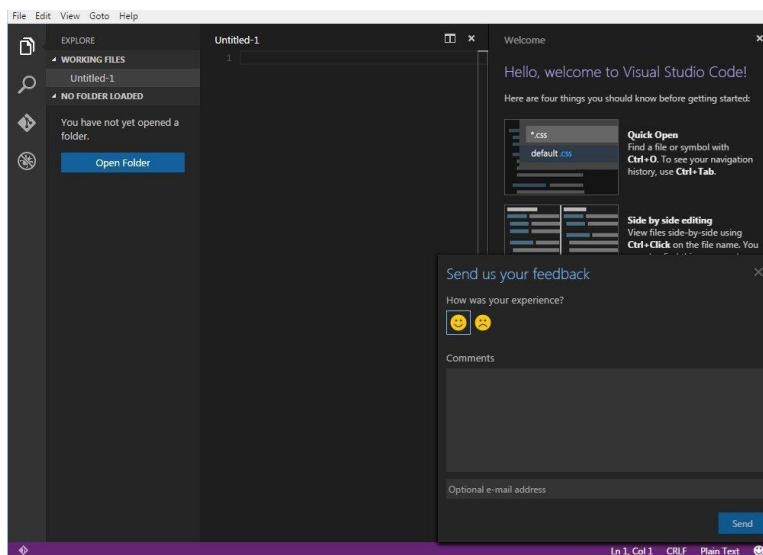


Рисунок 3.1 – Головне вікно середовища Visual Studio Code

Visual Studio Code — це редактор вихідного коду, який можна використовувати з різними мовами програмування, включаючи C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. [5] Він базується на структурі Electron, яка використовується для розробки вебдодатків Node.js, які працюють на механізмі компонування Blink. Visual Studio Code використовує той самий компонент редактора (під кодовою назвою «Monaco»), який використовується в Azure DevOps (раніше називався Visual Studio Online і Visual Studio Team Services). З коробки Visual Studio Code містить базову підтримку для більшості поширених мов програмування. Ця базова підтримка включає підсвічування

синтаксису, зіставлення дужок, згортання коду та настроювані фрагменти. [6]

Стек технологій, що застосовується для розробки серверної частини (Backend):

- Node or Node.js (рисунок 3.2) — програмна платформа, заснована на движку V8 (який компілює JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованої мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями вводу-виводу через свій API, написаний на C++, підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з коду JavaScript. Node.js використовується в основному на сервері, виступаючи в ролі веб-сервера, але є можливість розробки вузлових.js і настільних віконних додатків (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмних мікроконтролерів (наприклад, tessel, low.js і espruino). Перевага використання Node.js полягає в його високій швидкодії та здатності обробляти велику кількість одночасних запитів. Це можливо завдяки тому, що він не чекає завершення однієї операції, щоб почати іншу. Тому Node.js добре підходить для сайтів і додатків, які працюють у реальному часі, наприклад, чати або інтернет-магазини. [8]



Рисунок 3.2 – Логотип Node.js

- Express — це мінімалістичний і гнучкий веб-фреймворк для додатків Node.js, що надає великий набір функцій для мобільних і веб-додатків. [9] Маючи у своєму розпорядженні безліч методів утиліт HTTP і проміжного програмного забезпечення, можна швидко і легко створити надійний API. Express надає тонкий шар основних функцій веб-додатків, які не заважають працювати з вузлом.js функціями. [10]

– PostgreSQL (рисунок 3.3) — об'єктно-реляційна система керування базами даних (СКБД). Є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite). Порівняно з іншими проєктами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення. Сервер PostgreSQL написаний на мові С. Зазвичай розповсюджується у вигляді набору текстових файлів із сирцевим кодом. Перевага використання полягає у його надійності, відкритому вихідному коді та високій відповідності стандартам SQL. Ця система керування базами даних підтримує складні запити, транзакції та забезпечує стабільну роботу навіть з великими обсягами даних. Завдяки активній спільноті розробників PostgreSQL постійно оновлюється та розширюється, що робить її ефективним і безпечним рішенням для зберігання та обробки інформації. [11]



Рисунок 3.3 – Логотип СКБД PostgreSQL

- Sequelize — це бібліотека ORM для програм Node.js, яка відображає таблиці в базі даних та зв'язки між ними з класами. [10]
- Технологічний стек, що використовується для реалізації клієнтської частини (Frontend):
 - React.JS (рисунок 3.4) - це декларативна, ефективна та гнучка бібліотека JavaScript для створення інтерфейсів користувача. Він дозволяє збирати складний інтерфейс користувача з невеликих, ізольованих фрагментів коду, які називаються «компонентами». Перевага використання React полягає у його компонентній структурі, що забезпечує повторне використання коду, підвищену продуктивність завдяки віртуальному DOM, активну спільноту

розробників, а також простоту інтеграції з іншими бібліотеками та фреймворками. Це робить React оптимальним вибором для створення масштабованих, динамічних веб-застосунків. [12]



Рисунок 3.4 – Логотип бібліотеки React.js

– React Bootstrap - це безкоштовний набір інструментів з відкритим кодом, призначений для створення вебсайтів та вебдодатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних вебсайтів і вебдодатків. [12]

– Axios - це простий HTTP-клієнт для браузера та node.js на основі Promise. Axios надає просту у використанні бібліотеку в невеликому пакеті з дуже розширюваним інтерфейсом. [9]

– React-router-dom - надає специфічні для DOM методи, що можуть використовуватись на верхньому рівні вашого додатку, а також як запасний спосіб аби вийти з React моделі, якщо це необхідно. У більшості ваших компонентів вам не потрібно використовувати цей модуль. [12]

– MobX - це бібліотека управління станом, яку можна використовувати з будь-яким фреймворком JavaScript. React і MobX потужні разом і працюють як цілісна структура. MobX надає механізм для зберігання та оновлення стану програми. [13]

3.2. Розробка структури програмного забезпечення

Головна сторінка інтернет-магазину має різний вигляд в залежності від ролі користувача. Наприклад, якщо це адміністратор то доступні додаткові елементи інтерфейсу, зокрема з'являється кнопка на перехід до адмін-панелі, де можна додавати, редагувати чи видаляти страви, категорії, оновлювати ціни чи зображення. Адміністратор має повний доступ до структури сайту та може

керувати вмістом каталогу.

Зареєстровані користувачі можуть переглядати доступні страви в меню, додавати страви до кошика, а також здійснювати попереднє замовлення через дзвінок оператору. Незареєстрований користувач має змогу ознайомитися з асортиментом, переглядати опис страв, але не зможе додати їх до кошика, йому буде запропоновано пройти реєстрацію або авторизацію.

Інтерфейс головної сторінки є простим та інтуїтивно зрозумілим. Усі елементи чітко підписані відповідно до свого призначення. Це дозволяє швидко знайти потрібну страву. Для покращення зручності користування використано сучасну колірну гаму з перевагою білих та червоних відтінків, що асоціюються з японською тематикою. Усі шрифти вибрано для комфортного прочитання, а компоненти сторінки мають логічне розміщення та візуальну структуру.

Фрагменти тексту програми представлено в додатку Б.

3.3. Результати розробки програмного забезпечення

В результаті розробки було реалізовано увесь задуманий функціонал.

Додаток включає такі вікна як:

- головне вікно;
- реєстрація;
- вікно інформації про страву;
- адміністративна панель.

Головне вікно веб-сторінки представлено на рисунку 3.5.

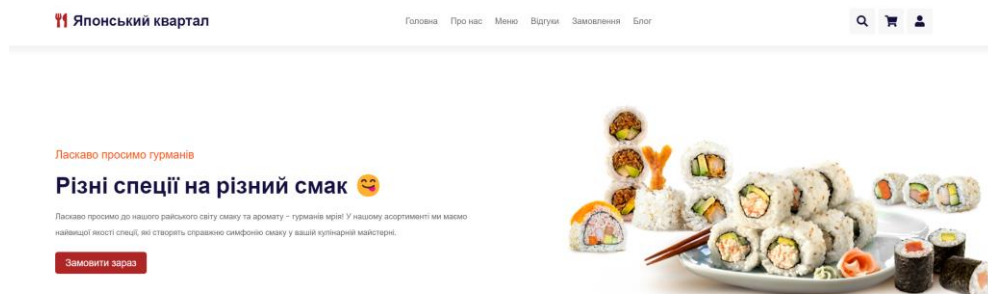


Рисунок 3.5 – Головне вікно веб-сторінки інтернет-магазину страв японської кухні

Система надає можливість обрати функцію авторизації або реєстрації. Форма авторизації й реєстрації представлена на рисунках 3.5 – 3.6.

Рисунки 3.5 – 3.6 – Вікно авторизації та реєстрації

Після натискання на кнопку «Увійти» відкривається форма авторизації. Система надає можливість ввести адресу електронної пошти та пароль, або відмінити вхід. Це показано на рисунках 3.7 та 3.8.

Рисунок 3.7 – Адміністративна панель

Рисунок 3.8 – Вікно з детальною інформацією про страву

Інструкцію користувача представлено в додатку А.

3.4. Висновок до третього розділу

У розділі було реалізовано усі ключові етапи створення програмного забезпечення для інтернет-магазину страв японської кухні. Аргументовано вибір стеку технологій. Такий підхід забезпечив високу продуктивність,

надійність і зручність у розробці та супроводі.

Розроблено структуру веб-сторінки з урахуванням ролей користувачів – адміністраторів, зареєстрованих і незареєстрованих відвідувачів. Інтерфейс головної сторінки створено з орієнтацією на інтуїтивну взаємодію та логічну навігацію.

Функціонал системи реалізований із застосуванням сучасних технологій, що забезпечує прозорість, масштабованість та підвищення ефективності бізнес-процесів. Проведено загальне тестування з метою виявлення і усунення можливих помилок. Виявлені недоліки були виправлені. Результати реалізації повністю відповідають вимогам, сформульованим на етапах проектування, це підтверджує обґрунтованість обраних технічних рішень.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1. Загальні вимоги

Згідно ст. 1 Закону України «Про охорону праці» під охороною праці розуміється система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів спрямованих на збереження здоров'я і працездатності людини в процесі праці. На основі вимог КЗОТ конкретні заходи по створенню здорових та безпечних умов праці, попередженню нещасних випадків та професійних захворювань регламентуються спеціальними правилами та нормами. Згідно з Законом України «Про охорону праці» нормативні акти можуть бути як єдиними для всіх галузей народного господарства, так і спеціальними для даної галузі. Державні міжгалузеві й галузеві нормативні акти про охорону праці - це правила, стандарти, норми, положення, інструкції та інші документи, яким надана сила правових норм, обов'язкових для виконання. [14]

Даний розділ кваліфікаційної роботи присвячений розгляду питань охорони праці при роботі за комп'ютером.

4.2. Вимоги до організації робочого місця

Конструкція робочого місця користувача ПК має забезпечувати підтримання оптимальної робочої пози з такими ергономічними характеристиками: ступні ніг - на підлозі або на підставці для ніг; стегна - в горизонтальній площині; передпліччя - вертикально; лікті - під кутом 70 - 90 град. до вертикальної площини; зап'ястя зігнуті під кутом не більше 20 град. відносно горизонтальної площини, нахил голови - 15 - 20 град. відносно вертикальної площини. [14]

Якщо користування ПК є основним видом діяльності, то ПК і його периферійні пристрої (принтер, сканер тощо) розміщується на основному робочому столі, як правило, з лівого боку. [14]

Якщо використання ПК є періодичним, то він, як правило, розміщується

на приставному столі, переважно з лівого боку від основного робочого столу. [14]

Кут між поздовжніми осями основного та приставного столів має бути 90 - 140 град. [14]

Висота робочої поверхні столу для ПК має бути в межах 680 - 800 мм, а ширина - забезпечувати можливість виконання операцій в зоні досяжності моторного поля. [14]

Рекомендовані розміри столу: висота - 725 мм, ширина - 600 -1400 мм, глибина - 800 - 1000 мм. [14]

Робочий стіл для ПК повинен мати простір для ніг висотою не менше 600 мм, шириною не менше 500 мм, глибиною на рівні колін не менше 450 мм, на рівні витягнутої ноги – не менше 650 мм. [14]

Робочий стіл для ПК, як правило, має бути обладнаним підставкою для ніг шириною не менше 300 мм та глибиною не менше 400 мм, з можливістю регулювання по висоті в межах 150 мм та кута нахилу опорної поверхні - в межах 20 град. Підставка повинна мати рифлену поверхню та бортик на передньому краї заввишки 10 мм. Застосування підставки для ніг тими, у кого ноги не дістають до підлоги, є обов'язковим. [14]

Робоче сидіння (сидіння, стілець, крісло) користувача ПК повинно мати такі основні елементи: сидіння, спинку, стаціонарні або знімні підлокітники. [14]

У конструкцію сидіння можуть бути введені додаткові елементи, що не є обов'язковими: підголовник та підставка для ніг. Робоче сидіння користувача ПК повинно бути підйомно-поворотним, таким, що регулюється за висотою, кутом нахилу сидіння та спинки, за відстанню спинки до переднього краю сидіння, висотою підлокітників. Регулювання кожного параметра має бути незалежним, плавним або ступінчатим, мати надійну фіксацію. [14]

Хід ступінчатого регулювання елементів сидіння має становити для лінійних розмірів 15 - 20 мм, для кутових - 2 - 5 град. Зусилля під час регулювання не повинні перевищувати 20 Н. Ширина та глибина сидіння

повинні бути не меншими за 400 мм. Висота поверхні сидіння має регулюватися в межах 400 - 500 мм, а кут нахилу поверхні - від 15 град. вперед до 5 град. назад. Поверхня сидіння має бути плоскою, передній край - заокругленим. Висота спинки сидіння має становити 300+ -20 мм, ширина - не менше 380 мм, радіус кривизни в горизонтальній площині - 400 мм. Кут нахилу спинки повинен регулюватися в межах 0 - 30 град. відносно вертикального положення. Відстань від спинки до переднього краю сидіння повинна регулюватись у межах 260 - 400 мм. [14]

Для зниження статичного напруження м'язів рук необхідно застосовувати стаціонарні або знімні підлокітники довжиною не менше 250 мм, шириною - 50 - 70 мм, що регулюються по висоті над сидінням у межах 230+/-30 мм та по відстані між підлокітниками в межах 350 - 500 мм. [14]

Поверхня сидіння, спинки та підлокітників має бути напівм'якою, з неслизьким, ненаелектризуючим, повітронепроникним покриттям та забезпечувати можливість чищення від бруду. [14]

Монітор та клавіатура мають розташовуватися на оптимальній відстані від очей користувача, але не ближче 600 мм, з урахуванням розміру алфавітно-цифрових знаків та символів. [14]

Розташування монітору має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +/-30 град. від лінії зору працівника. [14]

Клавіатуру слід розміщувати на поверхні столу або на спеціальній, регульованій за висотою, робочій поверхні окремо від столу на відстані 100 - 300 мм від краю. Кут нахилу клавіатури має бути в межах 5 - 15 град. [14]

Робоче місце з ПК слід оснащувати пюпітром (тримачем) для документів, що легко переміщується. Пюпітр (тримач) для документів повинен бути рухомим та встановлюватись вертикально (або з нахилом) на тому ж рівні та відстані від очей користувача ПК, що і монітор. [14]

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість монітору,

зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля: по висоті 900 - 1300 мм, по глибині 400 - 500 мм. [14]

При потребі високої концентрації уваги під час виконання робіт з високим рівнем напруженості суміжні робочі місця з ПК необхідно відділяти одне від одного перегородками висотою 1,5 - 2 м. [14]

4.3 Вимоги до освітлення

Відносно вікон робоче місце повинно бути розміщено так, щоб природне світло було збоку, переважно з лівого та забезпечувати коефіцієнт природної освітленості не нижче 1,5 %. Робоче місце, обладнане ПК повинно бути розташоване так, щоб уникнути попадання в очі прямого світла. Джерела штучного світла рекомендується розташувати з обох сторін від екрану паралельно напрямку зору. Щоб уникнути світових блисків від екрану, клавіатури, освітлювальних пристроїв, сонця в напрямку очей необхідно застосовувати антиблискові сітки, спеціальні фільтри для екрану, захисні козирки, жалюзі на вікнах. Вікна приміщень повинні мати регульовальні пристрої для відкривання. [14]

Штучне освітлення приміщення має бути обладнане системою загального рівномірного освітлення. У приміщеннях, де переважають роботи з документами, допускається вживати систему комбінованого освітлення (додатково до загального освітлення встановлюються світильники місцевого освітлення). Загальне освітлення має бути виконане у вигляді суцільних або переривчатих ліній світильників, що розміщуються збоку від робочих місць (переважно зліва) паралельно лінії зору працівників. Застосування світильників без розсіювачів та екранних сіток забороняється. [14]

Як джерело світла при штучному освітленні повинні застосовуватися, як правило, люмінесцентні лампи типу ЛБ. Допускається у світильниках місцевого освітлення застосовувати лампи розжарювання. Яскравість світильників загального освітлення в зоні кутів випромінювання від 50 град. до 90 град. відносно вертикалі в подовжній і поперечній площинах повинна складати не

більше 200 кд/кв. м , а захисний кут світильників повинен бути не більшим за 40 град. Коефіцієнт запасу для освітлювальної установки загального освітлення слід приймати рівним 1,4. Коефіцієнт пульсації повинен не перевищувати 5 %. Рівень освітленості на робочому столі в зоні розташування документів має бути в межах 300 – 500 лк. [14]

У разі неможливості забезпечити даний рівень освітленості системою загального освітлення допускається застосування світильників місцевого освітлення, але при цьому не повинно бути відблисків на поверхні екрану та збільшення освітленості екрану більше ніж до 300 лк. Світильники місцевого освітлення повинні мати напівпрозорий відбивач світла з захисним кутом не меншим за 40 град. Необхідно передбачити обмеження прямої близькості від джерела природного та штучного освітлення, при цьому яскравість поверхонь, що світяться (вікна, джерела штучного світла) і перебувають у полі зору, повинна бути не більшою за 200 кд/кв. м. Необхідно обмежувати відбиту близькість шляхом правильного вибору типів світильників та розміщенням робочих місць відносно джерел природного та штучного освітлення. При цьому яскравість відблисків на моніторі не повинна перевищувати 40 кд/кв. м, яскравість стелі при застосуванні системи відбивного освітлення не повинна перевищувати 200 кд/кв. м. Необхідно обмежувати нерівномірність розподілу яскравості в полі зору осіб, що працюють з монітором, при цьому відношення значень яскравості робочих поверхонь не повинно перевищувати 3:1, а робочих поверхонь і навколишніх предметів (стіни, обладнання) – 5:1. [14]

Необхідно використовувати систему вимикачів, що дозволяє регулювати інтенсивність штучного освітлення залежно від інтенсивності природного, а також дозволяє освітлювати тільки потрібні для роботи зони приміщення. Для забезпечення нормованих значень освітлення необхідно очищати віконне скло та світильники не рідше ніж 2 рази на рік, та своєчасно проводити заміну ламп, що перегоріли. [14]

4.4 Вимоги електробезпеки

Лінія електромережі для живлення персональних комп'ютерів, їх

периферійних пристроїв (принтер, сканер тощо) виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Використання нульового робочого провідника як нульового захисного провідника забороняється. Нульовий захисний провід прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення. Не допускається підключення на щиті до одного контактного затискача нульового робочого та нульового захисного провідників. [14]

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі повинна бути не менше площі перерізу фазового провідника. Усі провідники повинні відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму тощо. У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних комп'ютерів, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. [14]

Персональний комп'ютери, периферійні пристрої повинні підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Необхідно унеможливити з'єднання контактів фазових провідників з контактами нульового захисного провідника. Неприпустимим є підключення комп'ютерів, периферійних пристроїв до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв. [14]

Електромережі штепсельних з'єднань та електророзеток для живлення персональних комп'ютерів, периферійних пристроїв слід виконувати за магістральною схемою, по 3 – 6 з'єднань або електророзеток в одному колі. [14]

Штепсельні з'єднання та електророзетки для напруги 12 В та 36 В за своєю конструкцією повинні відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. [14]

Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 36 В, мають бути пофарбовані в колір, який візуально значно відрізняється від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В. [14]

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах. Електромережу штепсельних розеток для живлення персональних комп'ютерів, периферійних пристроїв при розташуванні їх уздовж стін приміщення прокладають по підлозі поряд зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. При розташуванні в приміщенні за його периметром до 5 персональних комп'ютерів, використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволяється прокладання їх без металевих труб та гнучких металевих рукавів. Електромережу штепсельних розеток при розташуванні комп'ютерів та периферійних пристроїв у центрі приміщення, прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не дозволяється застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, що містять сірку. Відкрита прокладка кабелів під підлогою забороняється. Металеві труби та гнучкі металеві рукави повинні бути заземлені. [14]

Конструкція знімної підлоги повинна бути такою, щоб забезпечувались: вільний доступ до кабельних комунікацій під час обслуговування; стійкість до горизонтальних зусиль при частково знятих плитах; вирівнювання поверхні підлоги за допомогою регулювальних опорних елементів; взаємозамінюваність

плит. Плити знімної підлоги повинні бути важкогорючими, з межею вогнестійкості не менше 0,5 год., або негорючими. Покриття плит виконують з матеріалів, які під час горіння не виділяють шкідливих токсичних речовин та газів, що сприяють корозії. Опори та стояки знімної підлоги повинні бути негорючими. Покриття плит підлоги повинно бути гладеньким, міцним, антистатичним, таким, яке легко чистити пілососом або прибирати вологим способом. Необхідно забезпечити відведення з покриття підлоги статичних зарядів. [14]

Отвори в плитах для прокладання кабелів електроживлення виконуються безпосередньо в місцях встановлення устаткування. Простір під знімною підлогою розділяють негорючими діафрагмами на відсіки площею не більше 250 кв. м. Межа вогнестійкості діафрагми повинна бути не меншою за 0,75 год. Комунікації прокладають крізь діафрагми в спеціальних обоймах з застосуванням негорючих ущільнювачів для запобігання проникненню вогню з одного відсіку в інший, а також з підпільного простору в приміщення. Підпільний простір під знімною підлогою має бути оснащений системою автоматичної пожежної сигналізації та засобами пожежогасіння. Для протирання підлоги застосовують рідини, пара яких не утворює вибухопожежонебезпечних сумішей з повітрям та не викликає корозії контактів електричних з'єднань. [14]

Для підключення переносної електроапаратури застосовують гнучкі проводи в надійній ізоляції. Тимчасова електропроводка від переносних приладів до джерел живлення виконується найкоротшим шляхом без заплутування проводів у конструкціях машин, приладів та меблях. [14]

Доточувати проводи можна тільки шляхом паяння з наступним старанним ізолюванням місць з'єднання. [14]

Є неприпустимими:

- експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізольованими провідниками;

– користування пошкодженими розетками, розгалужувальними та з'єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;

– підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканиною та іншими горючими матеріалами, експлуатація їх зі знятими ковпаками (розсіювачами);

– використання електроапаратури та приладів в умовах, що не відповідають вказівкам (рекомендаціям) підприємств-виготовлювачів. [14]

4.5 Вимоги до рівнів шуму та вібрації

Рівні шуму та вібрації на робочих місцях осіб, що працюють з ПК, визначаються відповідно до ДсанПіН 3.3.2-007-98. [14]

Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот представлені в таблиці 4.1. [14]

Таблиця 4.1 - Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот [14]

Вид трудової діяльності, робочі місця	Рівні звукового тиску в дБ в октавних смугах із середньгеометричними частотами, Гц									Рівні звуку, еквівалентні рівні звуку, дБА/дБАекв.
	31,5	63	125	250	500	1000	2000	4000	8000	
Програмісти	86	71	61	54	49	45	42	40	38	50
Оператори в залах обробки інформації на ПК та оператори комп'ютерного набору	96	83	74	68	63	60	57	55	54	65
В приміщеннях для розташування шумних агрегатів	103	91	83	77	73	70	68	66	64	75

Для забезпечення нормованих рівнів шуму у виробничих приміщеннях та на робочих місцях застосовуються шумопоглинальні засоби, вибір яких обґрунтовується спеціальними інженерно-акустичними розрахунками. Як

засоби шумопоглинання повинні застосовуватися негорючі або важкогорючі спеціальні перфоровані плити, панелі, мінеральна вата з максимальним коефіцієнтом звукопоглинання в межах частот 31,5 – 8000 Гц, або інші матеріали аналогічного призначення, дозволені для оздоблення приміщень органами державного санітарно-епідеміологічного нагляду. Крім того, необхідно застосовувати підвісні стелі з аналогічними властивостями. [14]

4.6 Висновок до четвертого розділу

В даному розділі було проаналізовано основні проблеми охорони праці, що можуть виникнути під час роботи працівника. Було виділено основні вимоги до приміщення, мікроклімату в приміщенні, освітлення та основних ергономічних характеристик.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи була розроблена інформаційна система інтернет-магазину страв японської кухні «MykoSushi».

Метою роботи стало підвищення ефективності діяльності підприємства через створення інтернет-магазину. Розроблена веб-сторінка дає можливість зручно переглядати актуальне меню страв підприємства та формувати замовлення через кошик. Особливо було приділено увагу до інтуїтивно зрозумілого інтерфейсу з гарним дизайном та до подачі інформації про страви й розширено про кожну страву з урахуванням назви, складу, ваги, категорії та її ціни.

Під час розробки проведений аналіз існуючих у мережі рішень у сфері онлайн-торгівлі готовими стравами, були підкреслені їх сильні та слабкі сторони. Це дозволило визначити вимоги до функціоналу майбутньої веб-сторінки, наприклад, необхідність зручної навігації на сторінці чи групування страв за категоріями.

На етапі проектування створено архітектуру сторінки та розроблено базу даних для зберігання інформації та метаданих. Це забезпечило структурованість даних і основу для реалізування усього функціоналу інформаційної системи.

Інтернет-магазин включає у себе реалізовані основні функції, зокрема перегляд каталогу страв, фільтрація за категоріями (суші, роли, сети, гарячі страви і т.д.), що дозволяє користувачам швидко знаходити потрібну їм страву. Кожна страва має свою сторінку та представлена з фото, описом, інгредієнтами, вагою та вартістю. Також розроблений функціонал кошику, що дозволяє зберігати вибрані страви для їх подальшої обробки. Крім користувацької частини веб-сторінки, також розроблено окрема адміністративна панель, яка доступна користувачам з правами доступу адміністратора, який забезпечує керування контентом сторінки. До керування контентом сторінки входить – редагування страв, оновлення цін, додавання нових страв і категорій.

Веб-сторінка була створена з використанням таких мов програмування як JavaScript та сучасного технологічного стеку. Для серверної частини використано платформу Node.js, яка забезпечує обробку запитів і логіку взаємодії з базою даних. Клієнтська частина розроблена за допомогою бібліотеки React, це дозволило створити швидкий та адаптивний інтерфейс. Для зберігання і обробки інформації використовується реляційна система керування базами даних PostgreSQL, яка є сучасною та гарантує надійність, якість та гнучкість у роботі з даними. Інтерфейс веб-сторінки створений за допомогою мови розмітки HTML5 та таблиць стилю CSS3.

Результатом виконаної кваліфікаційної роботи, є веб-сторінка, що виступає повнофункціональним інтернет-магазином страв японської кухні. Вона дає змогу користувачам зручно ознайомлюватися з меню, швидко знаходити страви завдяки фільтрації по категоріям, а також переглядати більш детальну інформацію про самі страви.

Розроблена веб-сторінка може покращити взаємодію з клієнтами та спростити процес взаємодії з платформою. Подальший розвиток проєкту може включати в себе розширення функціоналу, інтеграцію з іншими сервісами, покращення інтерфейсу користувача, збільшення можливостей взаємодії з сторінкою для адміністраторів та масштабування платформи для потреб більшої аудиторії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Юдін О. М., Макарова М. В., Лавренюк Р. М. Системи електронної комерції: створення, просування і розвиток. – Київ: КНЕУ, 2021. – 232 с.
2. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. – Електронні текстові дані (1 файл: 2,02 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 192с.
3. ДСТУ ISO/IEC 2382:2017 Інформаційні технології. Словник термінів
4. Погромська Г. С., Махровська Н. А. Бази даних: проектування та реалізація: навчально-методичний посібник для студентів спеціальності 122 «Комп'ютерні науки» / Міністерство освіти і науки України, Миколаївський національний університет імені В.О. Сухомлинського. – Миколаїв, 2021. – 108 с.
5. Глушаков І. С., Мірошніченко С. В. Розробка програмних продуктів у Visual Studio Code: навчальний посібник. – Харків: ХНУРЕ, 2020. – 148 с.
6. Visual Studio Code [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Visual_Studio_Code
7. JavaScript – довідка та документація по мові програмування від Mozilla Developer Network. – [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/uk/docs/Web/JavaScript>
8. Node.js — офіційна документація середовища виконання JavaScript. – [Електронний ресурс]. – Режим доступу: <https://nodejs.org/>
9. Express.js [Електронний ресурс] – Режим доступу: <https://expressjs.com/>
10. Ethan Brown. Web Development with Node and Express: Leveraging the JavaScript Stack. – O'Reilly Media, 2019. – 334 p.
11. PostgreSQL – офіційна документація системи керування базами даних. – [Електронний ресурс]. – Режим доступу: <https://www.postgresql.org/docs/>

12. React – JavaScript-бібліотека для створення інтерфейсів користувача. – [Електронний ресурс]. – Режим доступу: <https://reactjs.org/>
13. MobX [Електронний ресурс] – Режим доступу: <https://mobx.js.org/getting-started.html>
14. Безпечні умови праці державних службовців на робочих місцях, обладнаних комп'ютерною технікою. – [Електронний ресурс]. – Режим доступу: <https://studfile.net/preview/11044139/>
15. ДСТУ 2226-93 Автоматизовані системи. Терміни та визначення.
16. ДСТУ 5034:2008 Інформація і документація. Науково-інформаційна діяльність. Терміни та визначення понять.
17. ДСТУ ISO/IEC TR 10032:2012 Інформаційні технології. Еталонна модель керування даними.
18. ДСТУ ISO/IEC 14764:2014 Інженерія програмного забезпечення. Процеси життєвого циклу програмного забезпечення. Технічне обслуговування.
19. Методичні вказівки до виконання кваліфікаційної роботи за спеціальністю 122 – «Системний аналіз» галузі знань 12 – «Інформаційні технології» першого (бакалаврського) рівня вищої освіти / І. Л. Михелєв, А.Ю. Гайда, О. В. Гайдаєнко, Н. Р. Книрік, О. І. Маршак, Є. Ю. Беркунський, Г. С. Морозова, – Миколаїв : НУК, 2018, – 12 с.

ДОДАТОК А

Керівництво користувача

А.1 Модуль користувача

А.1.1 Вхід у додаток

Щоб почати роботу з інтернет-магазином страв японської кухні «MykoSushi», не треба реєструватися одразу.

Після відкриття посилання, буде перенаправлено на сторінку вибору виду входу. Вхід у вже існуючий обліковий запис або реєстрація нового аккаунту (див. рисунок А.1.1.1).

Рисунок А.1.1.1 – Вікно з вибором виду входу

А.1.2 Головна сторінка

Після входу у систему користувач потрапляє на головне вікно. Головне вікно додатку представлено на рисунку А.1.2.1

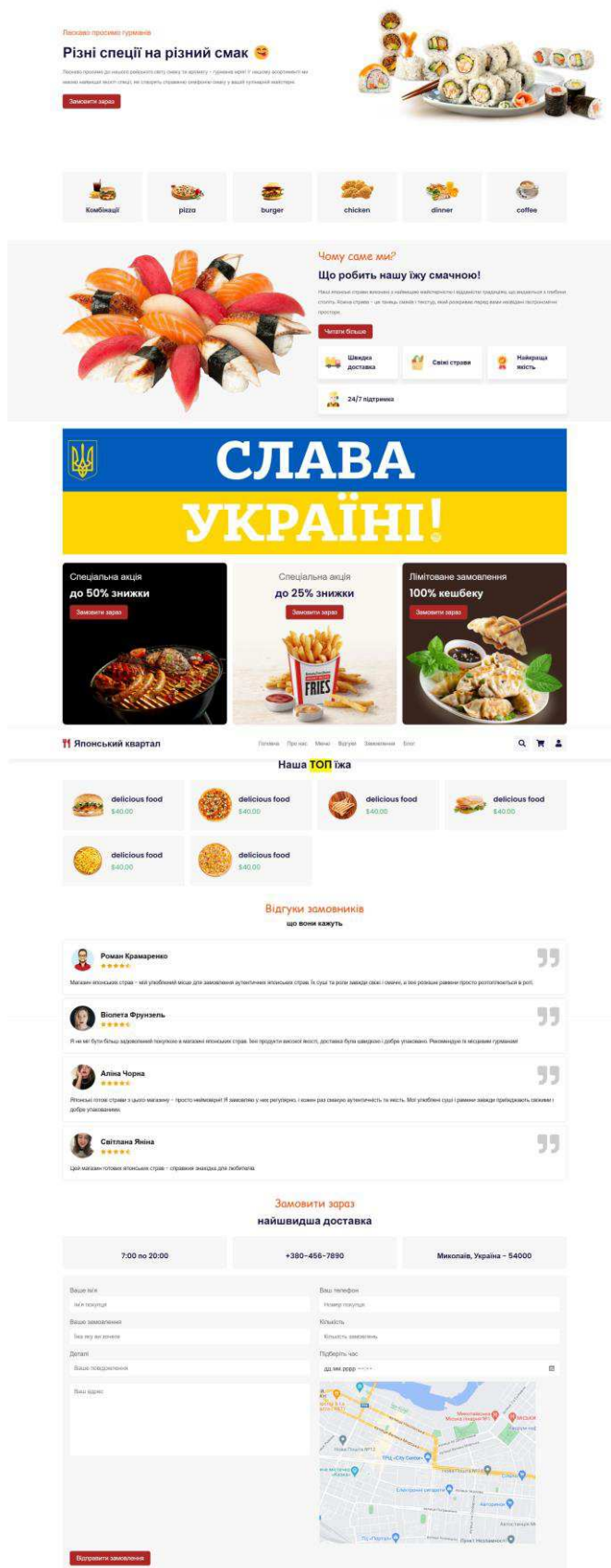


Рисунок А.1.2.1 – Головне вікно інтернет-магазину страв японської кухні

Головна сторінка включає навібар для навігації між модулями веб-сторінки, а саме: пошуком, кошиком та логіном (див. рисунок А.1.2.2).



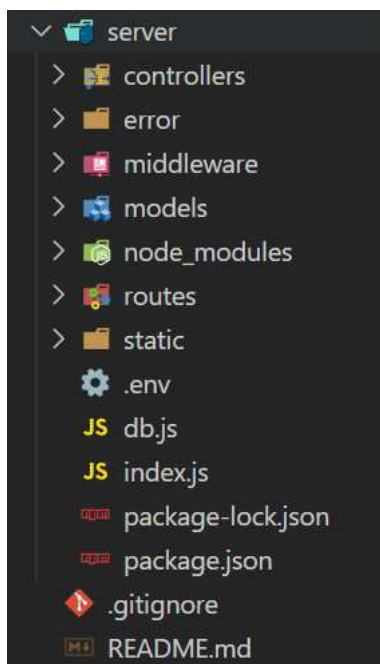
Рисунок А.1.2.2 – Слайдер на головній сторінці

ДОДАТОК Б

Фрагменти лістингу

В Visual Studio Code весь код складається з декількох файлів сценаріїв, файлів каскадного стилю, файлів розмітки гіпертексту, які розташовані в певному місці проекту.

Back-end stack (сервер) складається з:



Лістинг Б.1 – «index.js»

```
require('dotenv').config()
const express = require('express')
const sequelize = require('./db')
const models = require('./models/models')
const cors = require('cors')
const fileUpload = require('express-fileupload')
const router = require('./routes/index')
const errorHandler = require('./middleware/ErrorHandlerMiddleware')
const path = require('path')

const PORT = process.env.PORT || 5000

const app = express()
app.use(cors())
app.use(express.json())
app.use(express.static(path.resolve(__dirname, 'static')))
app.use(fileUpload({}))
app.use('/api', router)
```

```
// Обработка ошибок, последний MiddleWare
app.use(errorHandler)

const start = async() => {
  try {
    await sequelize.authenticate()
    await sequelize.sync()
    app.listen(PORT, () => console.log(`Server started on port ${PORT}`))
  } catch (e) {
    console.log(e)
  }
}

start()
```

Лістинг Б.2 – «db.js»

```
const {Sequelize} = require('sequelize') //деструктуризация

module.exports = new Sequelize( //конструктор
  process.env.DB_NAME, //указываем имя БД
  process.env.DB_USER, //пользователь
  process.env.DB_PASSWORD, //пароль
  {
    //Указываем диалект PostgreSQL
    dialect: 'postgres',
    host: process.env.DB_HOST,
    port: process.env.DB_PORT
  }
)
```

Лістинг Б.3 – «.env»

```
PORT=5000
DB_NAME=online_store
DB_USER=postgres
DB_PASSWORD=2003
DB_HOST=localhost
DB_PORT=5432
SECRET_KEY=random_secret_key123
```

Лістинг Б.4 – «routes»

```
const Router = require('express')
const router = new Router()
const brandController = require('../controllers/brandController')
const checkRole = require('../middleware/checkRoleMiddleware')
router.post('/', checkRole('ADMIN'), brandController.create)
router.get('/', brandController.getAll)
module.exports = router

const Router = require('express')
const router = new Router()
const categoryController = require('../controllers/categoryController')
const checkRole = require('../middleware/checkRoleMiddleware')
router.post('/', checkRole('ADMIN'), categoryController.create)
router.get('/', categoryController.getAll)
module.exports = router

const Router = require('express')
const router = new Router()
const foodController = require('../controllers/foodController')
const checkRole = require('../middleware/checkRoleMiddleware')
router.post('/', checkRole('ADMIN'), foodController.create)
router.get('/', foodController.getAll)
router.get('/:id', foodController.getOne)
module.exports = router

const Router = require('express')
const router = new Router()
const foodRouter = require('./foodRouter')
const userRouter = require('./userRouter')
const brandRouter = require('./brandRouter')
const categoryRouter = require('./categoryRouter')
router.use('/user', userRouter)
router.use('/category', categoryRouter)
router.use('/brand', brandRouter)
router.use('/food', foodRouter)
module.exports = router

const Router = require('express')
const router = new Router()
```

```

const userController = require('../controllers/userController')
const authMiddleware = require('../middleware/authMiddleware')

router.post('/registration', userController.registration)
router.post('/login', userController.login)
router.get('/auth', authMiddleware, userController.check)

module.exports = router

```

Лістинг Б.5 – «models»

```

const sequelize = require('../db')
const {DataTypes} = require('sequelize') //импорт класса DataTypes
const { _attributes } = require('../db')

const User = sequelize.define('user', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  email: {type: DataTypes.STRING, unique: true,},
  password: {type: DataTypes.STRING},
  role: {type: DataTypes.STRING, defaultValue: "USER"},
})

const Basket = sequelize.define('basket', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
})

const BasketFood = sequelize.define('basket_food', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
})

const Food = sequelize.define('food', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  name: {type: DataTypes.STRING, unique: true, allowNull: false},
  price: {type: DataTypes.INTEGER, allowNull: false},
  rating: {type: DataTypes.INTEGER, defaultValue: 0},
  img: {type: DataTypes.STRING, allowNull: false},
})

const Category = sequelize.define('category', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  name: {type: DataTypes.STRING, unique: true, allowNull: false},
})

```

```

const Brand = sequelize.define('brand', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  name: {type: DataTypes.STRING, unique: true, allowNull: false},
})

const Rating = sequelize.define('rating', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  rate: {type: DataTypes.INTEGER, allowNull: false},
})

const FoodInfo = sequelize.define('food_info', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  title: {type: DataTypes.STRING, allowNull: false},
  description: {type: DataTypes.STRING, allowNull: false},
})

const CategoryBrand = sequelize.define('category_brand', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
})

User.hasOne(Basket) //User включает Basket
Basket.belongsTo(User) //Basket принадлежит пользователю

User.hasMany(Rating)
Rating.belongsTo(User)

Basket.hasMany(BasketFood)
BasketFood.belongsTo(Basket)

Category.hasMany(Food)
Food.belongsTo(Category)

Brand.hasMany(Food)
Food.belongsTo(Brand)

Food.hasMany(Rating)
Rating.belongsTo(Food)

Food.hasMany(BasketFood)
BasketFood.belongsTo(Food)

Food.hasMany(FoodInfo, {as: 'info'});
FoodInfo.belongsTo(Food)

```

```

Category.belongsToMany(Brand, {through: CategoryBrand })
Brand.belongsToMany(Category, {through: CategoryBrand })
module.exports = {

  User,
  Basket,
  BasketFood,
  Food,
  Category,
  Brand,
  Rating,
  CategoryBrand,
  FoodInfo
}

```

Лістинг Б.6 – «middleware»

```

const jwt = require('jsonwebtoken')

module.exports = function (req, res, next) {
  if (req.method === "OPTIONS") {
    next()
  }
  try {
    const token = req.headers.authorization.split(' ')[1] // Bearer
    asfasnfkajsfjnk
    if (!token) {
      return res.status(401).json({message: "Не авторизован"})
    }
    const decoded = jwt.verify(token, process.env.SECRET_KEY)
    req.user = decoded
    next()
  } catch (e) {
    res.status(401).json({message: "Не авторизован"})
  }
};

const jwt = require('jsonwebtoken')

module.exports = function(role) {
  return function (req, res, next) {

```



```

    if (req.method === "OPTIONS") {
        next()
    }
    try {
        const token = req.headers.authorization.split(' ')[1] // Bearer
asfasnfkajsfnj
        if (!token) {
            return res.status(401).json({message: "Не авторизован"})
        }
        const decoded = jwt.verify(token, process.env.SECRET_KEY)
        if (decoded.role !== role) {
            return res.status(403).json({message: "Нет доступа"})
        }
        req.user = decoded;
        next()
    } catch (e) {
        res.status(401).json({message: "Не авторизован"})
    }
};
}

const ApiError = require('../error/ApiError');

module.exports = function (err, req, res, next) {
    if(err instanceof ApiError) {
        return res.status(err.status).json({message: err.message})
    }
    return res.status(500).json({message: "Непредвиденная ошибка!"})
}

```

Лістинг Б.7 – «error»

```

class ApiError extends Error{
    constructor(status, message) {
        super();
        this.status = status
        this.message = message
    }
    static badRequest(message) {
        return new ApiError(404, message)
    }

    static internal(message) {

```

```

        return new ApiError(500, message)
    }

    static forbidden(message) {
        return new ApiError(403, message)
    }
}
module.exports = ApiError

```

Лістинг Б.8 – «controllers»

```

const {Brand} = require('../models/models')
class brandController {
    async create(req, res) {
        const {name} = req.body
        const brand = await Brand.create({name})
        return res.json(brand)
    }

    async getAll(req, res) {
        const brands = await Brand.findAll()
        return res.json(brands)
    }
}
module.exports = new brandController()

const ApiError = require('../error/ApiError');
const {Category} = require('../models/models')

class categoryController {
    async create(req, res) {
        const {name} = req.body
        const category = await Category.create({name})
        return res.json(category)
    }

    async getAll(req, res) {
        const categorys = await Category.findAll()
        return res.json(categorys)
    }
}
module.exports = new categoryController()

```

```

const uuid = require('uuid')
const path = require('path')
const {Food, FoodInfo} = require('../models/models')
const ApiError = require('../error/ApiError')

class foodController {
  async create(req, res, next) {
    try {
      let {name, price, brandId, categoryId, info} = req.body
      const {img} = req.files
      let fileName = uuid.v4() + ".jpg"
      img.mv(path.resolve(__dirname, '..', 'static', fileName))
      const food = await Food.create({name, price, brandId, categoryId, img:
fileName})

      if (info) {
        info = JSON.parse(info)
        info.forEach(i =>
          FoodInfo.create({
            title: i.title,
            description: i.description,
            deviceId: food.id
          })
        )
      }

      return res.json(food)
    } catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }

  async getAll(req, res) {
    let {brandId, categoryId, limit, page} = req.query
    page = page || 1
    limit = limit || 9
    let offset = page * limit - limit
    let foods;
    if(!brandId && !categoryId) {
      foods = await Food.findAndCountAll({limit, offset})
    }
    if(brandId && !categoryId) {
      foods = await Food.findAndCountAll({where:{brandId}, limit, offset})
    }
  }
}

```

```

    }
    if(!brandId && categoryId) {
        foods = await Food.findAndCountAll({where:{categoryId}, limit,
offset}))
    }
    if(brandId && categoryId) {
        foods = await Food.findAndCountAll({where:{categoryId, brandId},
limit, offset}})
    }
    return res.json(foods)
}
async getOne(req, res) {

    const {id} = req.params
    const food = await Food.findOne(
        {
            where: {id},
            include: [{model: FoodInfo, as: 'info'}]
        },
    )
    return res.json(food)
}
}
module.exports = new foodController()

const ApiError = require('../error/ApiError');
const bcrypt = require('bcrypt') //хранить данные в хэшированном виде
const jwt = require('jsonwebtoken')
const {User, Basket} = require('../models/models')

const generateJwt = (id, email, role) => {
    return jwt.sign(
        {id, email, role},
        process.env.SECRET_KEY,
        {expiresIn: '24h'}
    )
}
}
class userController {
    //Авторизация происходит по JWT токenu. Токен доступа - разделенный точками
на 3 части
    //Токен прикрепляется к запросам, где необходима авторизация
    async registration(req, res, next) {
        const {email, password, role} = req.body
        if (!email || !password) {

```

```

        return next(ApiError.badRequest('Некоректний email чи password'))
    }
    const candidate = await User.findOne({where: {email}})
    if (candidate) {
        return next(ApiError.badRequest('Користувач з email вже існує'))
    }
    const hashPassword = await bcrypt.hash(password, 5)
    const user = await User.create({email, role, password: hashPassword})
    const basket = await Basket.create({userId: user.id})
    const token = generateJwt(user.id, user.email, user.role)

    return res.json({token})
}

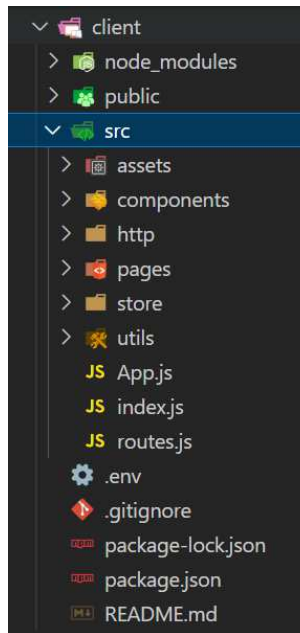
async login(req, res, next) {
    const {email, password} = req.body
    const user = await User.findOne({where: {email}})
    if(!user) {
        return next(ApiError.internal('Користувач не знайдено'))
    }
    let comparePassword = bcrypt.compareSync(password, user.password)
    if (!comparePassword) {
        return next(ApiError.internal('Користувач не знайдено'))
    }
    const token = generateJwt(user.id, user.email, user.role)
    return res.json({token})
}

async check(req, res, next) {
    const token = generateJwt(req.user.id, req.user.email, req.user.role)
    return res.json({token})
}
}

module.exports = new userController()

```

Front-end stack (клієнт) складається з:



Лістинг Б.9 – «.env»

```
REACT_APP_API_URL='http://localhost:5000/'
```

Лістинг Б.10 – «routes.js»

```
import Admin from "../pages/Admin";
import FoodPage from "../pages/FoodPage";
import Basket from "../pages/Basket";
import Shop from "../pages/Shop";
import Auth from "../pages/Auth";
import {ADMIN_ROUTE, BASKET_ROUTE, FOOD_ROUTE, LOGIN_ROUTE, REG_ROUTE,
SHOP_ROUTE} from "../utils/consts";

export const authRoutes = [
  {
    path: ADMIN_ROUTE,
    Component: Admin
  },
  {
    path: BASKET_ROUTE,
    Component: Basket
  },
]

export const publicRoutes = [
  {
    path: SHOP_ROUTE,
    Component: Shop
  },
]
```

```

    {
      path: LOGIN_ROUTE,
      Component: Auth
    },
    {
      path: REG_ROUTE,
      Component: Auth
    },
    {
      path: FOOD_ROUTE + '/:id',
      Component: FoodPage
    },
  ],
]

```

Лістинг Б.11 – «index.js»

```

import React, {createContext} from 'react';
import App from './App';
import { createRoot } from 'react-dom/client';
import UserStore from './store/UserStore'
import FoodStore from './store/FoodStore';

export const Context = createContext(null)

const rootElement = document.getElementById('root');
const root = createRoot(rootElement); // createRoot(container!) if you use
TypeScript
root.render(
  <Context.Provider value={{
    user: new UserStore(),
    food: new FoodStore()
  }}>
    <App />
  </Context.Provider>,
);

```

Лістинг Б.12 – «App.js»

```

import React, {useContext, useEffect, useState} from 'react';
import {BrowserRouter} from "react-router-dom";
import AppRouter from "./components/AppRouter";
import NavBar from "./components/NavBar";
import {observer} from "mobx-react-lite";
import {Context} from "../index";
import {check} from "../http/userAPI";

```

```

import {Spinner} from "react-bootstrap";

const App = observer(() => {
  const {user} = useContext(Context)
  const [loading, setLoading] = useState(true)

  useEffect(() => {
    check().then(data => {
      user.setUser(true)
      user.setIsAuth(true)
    }).finally(() => setLoading(false))
  }, [])

  if (loading) {
    return <Spinner animation={"grow"}/>
  }

  return (
    <BrowserRouter>
      <NavBar/>
      <AppRouter />
    </BrowserRouter>
  );
});

export default App;

```

Лістинг Б.13 – «utils/consts.js»

```

export const ADMIN_ROUTE = '/admin';
export const LOGIN_ROUTE = '/login';
export const REG_ROUTE = '/registration';
export const SHOP_ROUTE = '/';
export const BASKET_ROUTE = '/basket';
export const FOOD_ROUTE = '/food';

```

Лістинг Б.14 – «store»

```

import {makeAutoObservable} from "mobx";

export default class FoodStore {
  constructor() {
    this._categories = []
  }

```



```

        this._brands = []
        this._foods = []
        this._selectedCategory = {}
        this._selectedBrand = {}
        this._page = 1
        this._totalCount = 0
        this._limit = 3
        makeAutoObservable(this)
    }

    setCategories(categories) {
        this._categories = categories
    }
    setBrands(brands) {
        this._brands = brands
    }
    setFoods(foods) {
        this._foods = foods
    }
    setSelectedCategory(category) {
        this.setPage(1)
        this._selectedCategory = category
    }
    setSelectedBrand(brand) {
        this.setPage(1)
        this._selectedBrand = brand
    }
    setPage(page) {
        this._page = page
    }
    setTotalCount(count) {
        this._totalCount = count
    }

    get categories() {
        return this._categories
    }
    get brands() {
        return this._brands
    }
    get foods() {
        return this._foods
    }

```

```

    get selectedCategory() {
        return this._selectedCategory
    }
    get selectedBrand() {
        return this._selectedBrand
    }
    get totalCount() {
        return this._totalCount
    }
    get page() {
        return this._page
    }
    get limit() {
        return this._limit
    }
}
import {makeAutoObservable} from "mobx";
export default class UserStore {
    constructor() {
        this._isAuth = false
        this._user = {}
        makeAutoObservable(this)
    }
    setIsAuth(bool) {

        this._isAuth = bool
    }
    setUser(user) {
        this._user = user
    }

    get isAuth() {
        return this._isAuth
    }
    get user() {
        return this._user
    }
}

```

Лістинг Б.15 – «pages»

```

import React, {useState} from 'react';
import { Button, Container } from 'react-bootstrap';

```

```

import CreateCategory from '../components/modals/CreateCategory';
import CreateBrand from '../components/modals/CreateBrand';
import CreateFood from '../components/modals/CreateFood';

const Admin = () => {
  const [brandVisible, setBrandVisible] = useState(false)
  const [categoryVisible, setCategoryVisible] = useState(false)
  const [foodVisible, setFoodVisible] = useState(false)

  return (
    <Container className='d-flex flex-column'>
      <Button variant={"outline-dark"}
        className="mt-4 p-2" onClick={() =>
setCategoryVisible(true)}>Додати категорію
      </Button>
      <Button variant={"outline-dark"}
        className="mt-4 p-2" onClick={() =>
setBrandVisible(true)}>Додати вид
      </Button>
      <Button variant={"outline-dark"}
        className="mt-4 p-2" onClick={() => setFoodVisible(true)}>Додати
страву
      </Button>
      <CreateCategory show={categoryVisible} onHide={() =>
setCategoryVisible(false)} />
      <CreateBrand show={brandVisible} onHide={() => setBrandVisible(false)} />
      <CreateFood show={foodVisible} onHide={() => setFoodVisible(false)} />
    </Container>
  );
};

export default Admin;

import React, {useContext, useState} from 'react';
import {Button, Card, Container, Form} from "react-bootstrap";
import {useLocation, NavLink, useNavigate} from "react-router-dom";
import {LOGIN_ROUTE, REG_ROUTE, SHOP_ROUTE} from "../utils/consts";
import {observer} from "mobx-react-lite";
import {login, registration} from "../http/userAPI";
import {Context} from "../index";
import Row from "react-bootstrap/Row";
import {$authHost} from "../http";

```

```

const Auth = observer(() => {
  const {user} = useContext(Context)
  const location = useLocation()
  const history = useNavigate()
  const isLogin = location.pathname === LOGIN_ROUTE
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')

  const click = async () => {
    try {
      let data;
      if (isLogin) {
        data = await login(email, password);
      } else {
        data = await registration(email, password);
      }
      user.setUser(user)

      user.setIsAuth(true)
      history(SHOP_ROUTE)
    } catch (e) {
      alert(e.response.data.message)
    }
  }

  return (
    <Container
      className="d-flex justify-content-center align-items-center"
      style={{height: window.innerHeight - 54}}
    >
      <Card style={{width: 600}} className="p-5">
        <h2 className="m-auto">{isLogin ? 'Авторизація' :
"Регистрація"}</h2>
        <Form className="d-flex flex-column">
          <Form.Control
            className="mt-3"
            placeholder="Уведіть ваш email..."
            value={email}
            onChange={e => setEmail(e.target.value)}
          />

```

```

        <Form.Control
          className="mt-3"
          placeholder="Уведіть ваш пароль..."
          value={password}
          onChange={e => setPassword(e.target.value)}
          type="password"
        />
        <Row className="justify-content-end mt-3 pl-3 pr-3">
          {isLogin ?
            <div>
              Відсутній          акаунт?          <NavLink
to={REG_ROUTE}>Зареєструйтесь!</NavLink>
            </div>
            :
            <div>
              Вже          є          акаунт?          <NavLink
to={LOGIN_ROUTE}>Увійдіть!</NavLink>
            </div>
          }
          <Button
            variant={"outline-success"}
            onClick={click}
            style={{width: 150}}
          >
            {isLogin ? 'Вхід' : 'Реєстрація'}
          </Button>
        </Row>
      </Form>

    </Card>
  </Container>

  )))

export default Auth;

import React from 'react';

const Basket = () => {
  return (
    <div>
      basket
    </div>
  )
}

```

```

    </div>
  );
};

export default Basket;

import React, {useEffect, useState} from 'react';
import {Button, Card, Col, Container, Form, Image, Row} from "react-bootstrap";
import bigStar from '../assets/bigStar.png';
import {useParams} from 'react-router-dom';
import {observer} from "mobx-react-lite";
import {useContext} from "react";
import {Context} from "../index";
import { fetchOneFood } from '../http/foodAPI';

const FoodPage = () => {
  const {user} = useContext(Context)
  const [food, setFood] = useState({info: []})
  const {id} = useParams()
  const [foodVisible, setFoodVisible] = useState(false)

  useEffect(() => {
    fetchOneFood(id).then(data => setFood(data))
  }, [])

  const [value, setValue] = useState('')

  const description = []
  return (
    <Container className="mt-3">

      <Row>
        <Col md={4}>
          <Image width={300} height={300}
src={process.env.REACT_APP_API_URL + food.img}/>
        </Col>
        <Col md={4}>
          <Row className="d-flex flex-column align-items-center">
            <h2>{food.name}</h2>
            <div
              className="d-flex align-items-center justify-content-
center"

```

```

        style={{background: `url(${bigStar}) no-repeat center
center`, width:240, height: 240, backgroundSize: 'cover', fontSize:64}}
      >
        {food.rating}
      </div>
    </Row>
  </Col>
  <Col md={4}>
    <Card
      className='d-flex flex-column align-items-center justify-content-
around'
      style={{width:300, height:300, fontSize: 32, border: '5px solid
lightgray'}}
    >
      <h3>{food.price} грн.</h3>
      <Button variant={'outline-dark'} className="bg-danger text-
light">ОБИПАЮ!</Button>
    </Card>
  </Col>
</Row>
<Row className='d-flex flex-column m-3'>
  <h1>Характеристики</h1>
  {food.info.map((info, index) =>
    <Row key={info.id} style={{background: index % 2 === 0 ? 'lightgray'
: 'transparent', padding: 10}}>
      {info.title}: {info.description}
    </Row>
  )}
</Row>
</Container>
);
};

export default FoodPage;

import React, { useContext, useEffect } from 'react';
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import {Container} from "react-bootstrap";
import CategoryBar from "../components/CategoryBar";
import BrandBar from "../components/BrandBar";
import FoodList from "../components/FoodList";
import {observer} from "mobx-react-lite";

```

```

import {Context} from "../index";
import {fetchBrands, fetchCategories, fetchFoods} from "../http/foodAPI"
import Pages from '../components/Pages';

const Shop = observer(() => {
  const {food} = useContext(Context)
  const {user} = useContext(Context)

  useEffect(() => {
    fetchCategories().then(data => food.setCategories(data))
    fetchBrands().then(data => food.setBrands(data))
    fetchFoods().then(data => food.setFoods(data.rows))
  }, [food.page, food.selectedType, food.selectedBrand,])

  return (
    <Container>
      <Row className="mt-2">
        <Col md={3}>
          <CategoryBar/>
        </Col>
        <Col md={9}>
          <BrandBar/>
          <FoodList/>
          <Pages/>
        </Col>
      </Row>
    </Container>
  );
});

export default Shop;

```

Лістинг Б.16 – «http»

```

import {$authHost, $host} from "../index";

export const createCategory = async (category) => {
  const {data} = await $authHost.post('api/category', category)
  return data
}

```



```

export const fetchCategories = async () => {
  const {data} = await $host.get('api/category')
  return data
}

export const createBrand = async (brand) => {
  const {data} = await $authHost.post('api/brand', brand)
  return data
}

export const fetchBrands = async () => {
  const {data} = await $host.get('api/brand', )
  return data
}

export const createFood = async (food) => {
  const {data} = await $authHost.post('api/food', food)
  return data
}

export const fetchFoods = async (categoryId, brandId, page, limit) => {
  const {data} = await $host.get('api/food', {params: {
    categoryId, brandId, page, limit
  }})
  return data
}

export const fetchOneFood = async (id) => {
  const {data} = await $host.get('api/food' + '/' + id)
  return data
}

import axios from "axios";

const $host = axios.create({
  baseURL: process.env.REACT_APP_API_URL
})

const $authHost = axios.create({
  baseURL: process.env.REACT_APP_API_URL
})

```

```

const authInterceptor = config => {
  config.headers.authorization = `Bearer ${localStorage.getItem('token')}`
  return config
}

$authHost.interceptors.request.use(authInterceptor)

export {
  $host,
  $authHost
}

import {$authHost, $host} from "../index";
import jwt_decode from "jwt-decode";

export const registration = async (email, password) => {
  const {data} = await $host.post('api/user/registration', {email: email,
password: password, role: 'USER'})
  localStorage.setItem('token', data.token)
  return jwt_decode(data.token)
}

export const login = async (email, password) => {
  const {data} = await $host.post('api/user/login', {email: email, password:
password})
  localStorage.setItem('token', data.token)
  return jwt_decode(data.token)
}

export const check = async () => {
  const {data} = await $authHost.get('api/user/auth')
  localStorage.setItem('token', data.token)
  return jwt_decode(data.token)
}

export const checkRole = async () => {

  const {data} = await $authHost.get('api/user/auth')
  localStorage.setItem('token', data.token)
  return jwt_decode(data.token)
}

```

Лістинг Б.17 – «components»

```

import React, {useState} from 'react';
import Modal from "react-bootstrap/Modal";
import {Button, Form} from "react-bootstrap";
import { createBrand } from '../http/foodAPI';

const CreateBrand = ({show, onHide}) => {
  const [value, setValue] = useState('')

  const addBrand = () => {
    createBrand({name: value}).then(data => {
      setValue('')
      onHide()
    })
  }

  return (
    <Modal
      show={show}
      onHide={onHide}
      centered
    >
      <Modal.Header closeButton>
        <Modal.Title id="contained-modal-title-vcenter">
          Додати вид
        </Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form>
          <Form.Control
            value={value}
            onChange={e => setValue(e.target.value)}
            placeholder="Введіть назву виду"
          />
        </Form>
      </Modal.Body>
      <Modal.Footer>
        <Button variant="outline-danger"
          onClick={onHide}>Закрити</Button>
        <Button variant="outline-success"
          onClick={addBrand}>Додати</Button>
      </Modal.Footer>
    </Modal>
  )
}

```

```

    );
  };

export default CreateBrand;

import React, {useState} from 'react';
import Modal from "react-bootstrap/Modal";
import {Form, Button} from "react-bootstrap";
import { createCategory } from '../http/foodAPI';

const CreateCategory = ({show, onHide}) => {
  const [value, setValue] = useState('')

  const addCategory = () => {
    createCategory({name: value}).then(data => {
      setValue('')
      onHide()
    })
  }

  return (
    <Modal
      show={show}
      onHide={onHide}
      centered
    >
      <Modal.Header closeButton>
        <Modal.Title id="contained-modal-title-vcenter">
          Додати категорію
        </Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form>
          <Form.Control
            value={value}
            onChange={e => setValue(e.target.value)}
            placeholder="Уведіть назву категорії"
          />
        </Form>

        </Modal.Body>
        <Modal.Footer>

```

```

        <Button                                variant="outline-danger"
onClick={onHide}>Закрити</Button>

        <Button                                variant="outline-success"
onClick={addCategory}>Додати</Button>

        </Modal.Footer>
    </Modal>

    );
};

export default CreateCategory;

import React, {useContext, useEffect, useState} from 'react';
import Modal from "react-bootstrap/Modal";
import {Button, Dropdown, Form, Row, Col} from "react-bootstrap";
import {Context} from "../../index";
import {observer} from "mobx-react-lite";
import DropdownMenu from 'react-bootstrap/esm/DropdownMenu';
import {createFood, fetchBrands, fetchCategories, fetchFoods} from
"../../http/foodAPI"

const CreateFood = observer(({show, onHide}) => {
    const {food} = useContext(Context)
    const [name, setName] = useState('')
    const [price, setPrice] = useState(0)
    const [file, setFile] = useState(null)
    const [brand, setBrand] = useState([])
    const [category, setCategory] = useState([])
    const [info, setInfo] = useState([])

    useEffect(() => {
        fetchCategories().then(data => food.setCategories(data))
        fetchBrands().then(data => food.setBrands(data))
    }, [])

    const addInfo = () => {
        setInfo([...info, {title: '', description: '', number: Date.now()}])
    }

    const removeInfo = (number) => {
        setInfo(info.filter(i => i.number !== number))
    }
}
```

```

const selectFile = e => {
  setFile(e.target.files[0])
}

const changeInfo = (key, value, number) => {
  setInfo(info.map(i => i.number === number ? {...i, [key]: value} : i))
}

const addFood = () => {
  const formData = new FormData()
  try{
    formData.append('name', name)
    formData.append('price', `${price}`)
    formData.append('img', file)
    formData.append('brandId', food.selectedBrand.id)
    formData.append('categoryId', food.selectedCategory.id)
    formData.append('info', JSON.stringify(info))
    createFood(formData).then(data => onHide())
  } catch(e){
    alert(e)
  }
}

return (
  <Modal
    show={show}
    onHide={onHide}
    centered
  >
    <Modal.Header closeButton>
      <Modal.Title id="contained-modal-title-vcenter">
        Додати пристрій
      </Modal.Title>
    </Modal.Header>
    <Modal.Body>
      <Form>
        <Dropdown className="mt-2 mb-2">
          <Dropdown.Toggle>{food.selectedCategory.name || "Оберіть
катеропію"}</Dropdown.Toggle>
          <Dropdown.Menu>
            {food.categories.map(category =>
              <Dropdown.Item
                onClick={() =>

```

```

food.setSelectedCategory(category) }

        key={category.id}
      >
        {category.name}
      </Dropdown.Item>
    ) }
  </Dropdown.Menu>
</Dropdown>
<Dropdown className="mt-2 mb-2">
  <Dropdown.Toggle>{food.selectedBrand.name || "Оберіть
бренд"}</Dropdown.Toggle>
  <Dropdown.Menu>
    {food.brands.map(brand =>
      <Dropdown.Item
        onClick={() => food.setSelectedBrand(brand)}
        key={brand.id}
      >
        {brand.name}
      </Dropdown.Item>
    ) }
  </Dropdown.Menu>
</Dropdown>

<Form.Control
  value={name}
  onChange={e => setName(e.target.value)}
  className="mt-3"
  placeholder="Введіть назву"
/>
<Form.Control
  value={price}
  onChange={e => setPrice(Number(e.target.value))}
  className="mt-3"
  placeholder="Введіть вартість"
  type="number"
/>

<Form.Control
  className="mt-3 ua"
  type="file"
  onChange={selectFile}
/>
<hr/>

```

```

<Button
  variant={"outline-dark"}

  onClick={addInfo}
>
  Додати нову характеристику
</Button>
{info.map(i =>
  <Row className="mt-4" key={i.number}>
    <Col md={4}>
      <Form.Control
        value={i.title}
        onChange={(e) => changeInfo('title',
e.target.value, i.number)}
        placeholder="Введіть назву характеристики"
      />
    </Col>
    <Col md={4}>
      <Form.Control
        value={i.description}
        onChange={(e) => changeInfo('description',
e.target.value, i.number)}
        placeholder="Введіть опис характеристики"
      />
    </Col>
    <Col md={4}>
      <Button
        onClick={() => removeInfo(i.number)}
        variant={"outline-danger"}
      >
        Видалити
      </Button>
    </Col>
  </Row>
)}
</Form>
</Modal.Body>
<Modal.Footer>
  <Button
    variant="outline-danger"
    onClick={onHide}>Закрити</Button>
  <Button
    variant="outline-success"
    onClick={addFood}>Додати</Button>

```



```

        </Modal.Footer>
      </Modal>
    );
  }
)
export default CreateFood;

import React, {useContext} from 'react';
import {Context} from "../index"
import {
  Routes,
  Route,
  Navigate
} from "react-router-dom";
import {authRoutes, publicRoutes} from "../routes";
import {SHOP_ROUTE} from "../utils/consts";
import {observer} from "mobx-react-lite";

const AppRouter = observer(() => {
  const {food, user} = useContext(Context)
  console.log(user)
  console.log(food)
  return(
    <Routes>
      {user._isAuth && authRoutes.map(({path, Component}) =>
        <Route key = {path} path={path} element={<Component/>}
exact/>
      )}
      {publicRoutes.map(({path , Component}) =>
        <Route key = {path} path = {path} element={<Component/>}
exact/>
      )}
    </Routes>

  );
});

export default AppRouter;

import React, {useContext} from 'react';

```

```

import {observer} from "mobx-react-lite";
import {Context} from "../index";

import {Card, Row} from "react-bootstrap";

const BrandBar = observer(() => {
  const {food} = useContext(Context)
  return (

    <Row className="d-flex">
      {food.brands.map(brand =>
        <Card
          style={{cursor:'pointer', width: 'min-content'}}
          key={brand.id}
          className="p-3"
          onClick={() => food.setSelectedBrand(brand)}
          border={brand.id === food.selectedBrand.id ? 'danger' :
'light'}
        >
          {brand.name}
        </Card>
      )}
    </Row>
  );
});

export default BrandBar;

import React, {useContext} from 'react';
import {observer} from "mobx-react-lite";
import {Context} from "../index";
//import Col from "react-bootstrap/Col";
import ListGroup from "react-bootstrap/ListGroup";

const CategoryBar = observer(() => {
  const {food} = useContext(Context)
  return (
    <ListGroup>
      {food.categories.map(category =>
        <ListGroup.Item
          style={{cursor: 'pointer'}}
          active={category.id === food.selectedCategory.id}
          onClick={() => food.setSelectedCategory(category)}

```

```

        key={category.id}
      >
        {category.name}
      </ListGroup.Item>

    )}
  </ListGroup>
);
});
export default CategoryBar;

import React, {useContext} from 'react';
import {Card, Col} from "react-bootstrap";
import {useNavigate} from "react-router-dom";
import {FOOD_ROUTE} from "../utils/consts";
import Image from "react-bootstrap/Image";
import star from "../assets/star.png";
import {Context} from "../index";

const FoodItem = ({food}) => {
  const history = useNavigate()
  const {user} = useContext(Context)
  return (
    <Col md={3} className="mt-2" onClick={() => history(FOOD_ROUTE + '/' +
food.id)}>
      <Card style={{border: '2px solid lightgray', width: 150, height: 230,
cursor: 'pointer'}}>
        <Image width={147} height={150} src={process.env.REACT_APP_API_URL +
food.img}/>

        <div className="text-black-50 mt-1 d-flex justify-content-between align-
items-center">
          <div>Cymwi</div>
          <div className="d-flex align-items-center">
            <div>{food.rating}</div>
            <Image width={18} height={18} src={star}/>
          </div>
        </div>
        <div>{food.name}</div>
      </Card>
    </Col>
  );
};

```

```

};

export default FoodItem;

import React, {useContext} from 'react';
import {observer} from "mobx-react-lite";

import {Context} from "../index";
import {Row} from "react-bootstrap";
import FoodItem from "../FoodItem";

const FoodList = observer(() => {
  const {food} = useContext(Context)

  return (
    <Row className="mt-1 d-flex">
      {food.foods.map(food =>
        <FoodItem key={food.id} food={food}/>
      )}
    </Row>
  );
});

export default FoodList;

import React, {useContext} from 'react';
import {Context} from "../index";
import Navbar from "react-bootstrap/Navbar";
import Nav from "react-bootstrap/Nav";
import {NavLink} from "react-router-dom";
import {ADMIN_ROUTE, LOGIN_ROUTE, REG_ROUTE, SHOP_ROUTE} from "../utils/consts";
import {Button} from "react-bootstrap";
import {observer} from "mobx-react-lite";
import Container from "react-bootstrap/Container";
import {useNavigate} from 'react-router-dom'

const NavBar = observer(() => {
  const {user} = useContext(Context)
  const history = useNavigate()

  const logOut = () => {

```

```

        user.setUser({})
        user.setIsAuth(false)
    }

    return (
        <Navbar bg="dark" variant="dark">
            <Container>
                <label style={{color: 'white'}} onClick={() =>
history(SHOP_ROUTE)}>MykoSushi</label>
                {user.isAuth ?
                    <Nav className="ms-auto" style={{color: 'white'}}>
                        <Button
                            variant={"outline-light"}
                            onClick={() => history(ADMIN_ROUTE)}
                            className="ms-3"
                        >
                            Адмін панель
                        </Button>
                        <Button
                            variant={"outline-light"}
                            onClick={() => logOut()}
                            className="ms-3"
                        >
                            Вийти
                        </Button>
                    </Nav>
                    :
                    <Nav className="ms-auto" style={{color: 'white'}}>
                        <Button variant={"outline-light"} onClick={() =>
history(LOGIN_ROUTE)}>Авторизація</Button>
                    </Nav>
                }
            </Container>
        </Navbar>

    );
});

export default NavBar;

import React, {useContext} from 'react';
import {observer} from "mobx-react-lite";

```

```
import {Context} from "../index";
import {Pagination} from "react-bootstrap";

const Pages = observer(() => {
  const {food} = useContext(Context)
  const pageCount = Math.ceil(food.totalCount / food.limit)
  const pages = [1,2,3,4,5]

  for (let i = 0; i < pageCount; i++) {
    pages.push(i+1)
  }

  return (
    <Pagination className="mt-3">
      {pages.map(page =>
        <Pagination.Item
          key={page}
          active={food.page === page}
          onClick={() => food.setPage(page)}
        >
          {page}
        </Pagination.Item>
      )}
    </Pagination>
  );
});

export default Pages;
```

Лістинг Б.18 – «index.html»

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta
      name="description"
      content="Web site created using create-react-app"
```

```

/>
<link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
<!--
  manifest.json provides metadata used when your web app is installed on a
  user's mobile device or desktop. See
  https://developers.google.com/web/fundamentals/web-app-manifest/
-->
<link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
<!--
  Notice the use of %PUBLIC_URL% in the tags above.
  It will be replaced with the URL of the `public` folder during the build.
  Only files inside the `public` folder can be referenced from the HTML.
  Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
  work correctly both with client-side routing and a non-root public URL.
  Learn how to configure a non-root public URL by running `npm run build`.
-->
<link
    rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
    integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
    crossorigin="anonymous"

/>
<title>React App</title>
</head>
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>
</html>

```