

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБСАЙТУ
ДЛЯ ЗАМОВЛЕННЯ ПРОДУКТІВ ХАРЧУВАННЯ
З ДОСТАВКОЮ
SOFTWARE DEVELOPMENT FOR A FOOD ORDERING AND
DELIVERY WEBSITE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-132

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ТКАЧ Р.Ю.

(ПРИЗВИЩЕ та ініціали)

Керівник СУББОТІН С.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

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

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ТКАЧА Романа Юрійовича
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення програмного забезпечення вебсайту для замовлення продуктів харчування з доставкою. Software Development for a Food Ordering and Delivery Website

керівник проєкту (роботи) д.т.н., професор, Субботін Сергій Олександрович,
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., зав.каф.		
Нормоконтроль	Калініна М.В., асистент		

7. Дата видачі завдання « 20 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Роман ТКАЧ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій СУББОТІН
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 71 с., 4 табл., 34 рис., 3 дод., 28 джерел.

ВЕБЗАСТОСУНОК, ДОСТАВКА ЇЖІ, DJANGO, АСИНХРОННА ОБРОБКА, CELERY.

Об'єкт дослідження – процес інженерії програмного забезпечення для доставки продуктів харчування.

Предмет дослідження – програмне забезпечення вебсайту для автоматизації процесу замовлення та доставки їжі.

Мета роботи – підвищення рівня автоматизації та ефективності роботи закладу громадського харчування шляхом розроблення програмного забезпечення вебсайту для зручного замовлення страв, безпечної онлайн-оплати та оптимізації аналітичної роботи адміністративного персоналу.

Матеріали, методи та технічні засоби: мова програмування Python, бекенд-фреймворк Django, система керування реляційними базами даних PostgreSQL, брокер повідомлень Redis та система фонові обробки Celery. Клієнтську частину реалізовано з використанням HTML5, CSS3 (Bootstrap 5), JavaScript (AJAX) та Chart.js. Інтегровано API платіжної системи Stripe, Telegram API та бібліотеку WeasyPrint для генерації PDF.

Результати. Розроблено та підготовлено до розгортання комплексну інформаційну систему замовлення та доставки їжі. Реалізовано клієнтський інтерфейс (каталог страв, кошик на базі сесій, застосування промокодів, чекаут) та закриту адміністративну панель (робоче місце персоналу для зміни статусів, дашборд аналітики з генерацією PDF-звітів, автоматизовані сповіщення).

Висновки. Створена програмна система автоматизує ключові бізнес-процеси служби доставки, знижує залежність від сторонніх агрегаторів (і їхніх

комісій), гарантує цілісність фінансових даних та забезпечує повний контроль над замовленнями і статистикою.

Галузь використання – заклади громадського харчування та служби доставки їжі.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 71 pages, 4 tables, 34 figures, 3 appendixes, 28 sources.

WEB APPLICATION, FOOD DELIVERY, DJANGO, ASYNCHRONOUS PROCESSING, CELERY.

Object of study is the process of software engineering for food ordering and delivery.

Subject of study is the website software for automating the food ordering and delivery process.

The purpose of the work is to increase the level of automation and efficiency of a food service establishment by developing website software for convenient dish ordering, secure online payment, and optimizing the analytical work of the administrative staff.

Materials, methods, and tools: Python programming language, Django backend framework, PostgreSQL RDBMS, Docker containerization technologies, Redis message broker, and Celery asynchronous task queue. The client side is implemented using HTML5, CSS3 (Bootstrap 5), JavaScript (AJAX), and Chart.js. Integrations include Stripe API, Telegram API, and the WeasyPrint PDF generation library.

Results. A comprehensive food ordering and delivery information system has been developed and prepared for deployment. Both a client interface (catalog, session-based cart, promo codes, checkout) and an administrative panel (staff workspace for status updates, analytics dashboard with PDF report generation, automated notifications) were implemented.

Conclusions. The created software system automates key business processes of a delivery service, reduces dependence on third-party aggregators, guarantees the integrity of financial data, and provides full control over orders and statistics.

The field of use is catering establishments (restaurants, cafes) and independent food delivery services.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Аналіз проблеми та постановка завдань дослідження	12
1.1 Огляд предметної області.....	12
1.2 Аналіз існуючих аналогів вебзастосунків підтримки роботи служби доставки.....	14
1.3 Постановка завдання.....	19
2 Матеріали і методи.....	20
2.1 Вибір мови програмування та серверного фреймворку.....	20
2.2 Задача автоматизації життєвого циклу замовлення та забезпечення цілісності фінансових даних	22
2.3 Алгоритм обробки замовлень та асинхронної генерації звітної документації.....	23
2.4 Вибір системи керування базами даних	26
2.5 Висновки за розділом 2	27
3 Опис програми.....	28
3.1 Опис програми.....	28
3.2 Моделювання роботи з програмою за функціональними вимогами	28
3.3 Опис бази даних	31
3.4 Висновки за розділом 3	34
4 Експлуатація, тестування та експериментальне дослідження програми	35
4.1 Експлуатація програми.....	35
4.2 Експериментальне дослідження швидкодії обробки замовлень при використанні асинхронних черг завдань	43
Висновки	47
Перелік джерел посилання	49
Додаток А Технічне завдання	52

Додаток Б Фрагмент тексту програми	57
Додаток В Слайди презентації.....	64

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface;
PDF – Portable Document Format;
БД – база даних;
ПЗ – програмне забезпечення.

ВСТУП

Розвиток інформаційних технологій та повсюдне використання мобільних пристроїв докорінно змінили підходи до надання послуг у сфері громадського харчування. В умовах високої конкуренції та стрімкого зростання ринку доставки їжі наявність власного вебзастосунку стає критичною необхідністю для ресторанів, кафе та спеціалізованих сервісів. Застосування готових агрегаторів накладає суттєві фінансові обмеження у вигляді високих комісій та нівелює можливості розвитку власного бренду. Таким чином, розробка незалежної, масштабованої та автоматизованої програмної системи, яка б охоплювала весь життєвий цикл замовлення – від вибору страв клієнтом до генерації фінансової звітності для адміністрації – є актуальним науково-практичним завданням інженерії програмного забезпечення. Об'єктом дослідження в роботі є процес замовлення та доставки продуктів харчування. Предметом дослідження виступає ПЗ вебсайту для автоматизації процесу замовлення та доставки їжі.

Метою роботи є розроблення програмного забезпечення вебсайту для замовлення продуктів харчування з доставкою, що дозволить підвищити рівень автоматизації та ефективності роботи закладу громадського харчування, забезпечити зручне замовлення страв, безпечну онлайн-оплату та оптимізацію аналітичної роботи адміністративного персоналу.

Для досягнення поставленої мети необхідно виконати ряд завдань, серед яких: аналіз предметної області та існуючих аналогів систем замовлення їжі, проєктування архітектури реляційної бази даних та логіки взаємодії компонентів вебзастосунку, реалізація серверної та клієнтської частин застосунку із забезпеченням адаптивності інтерфейсу, інтеграція зовнішніх програмних інтерфейсів для обробки онлайн-оплат та надсилання автоматизованих сповіщень, забезпечення фонові обробки ресурсомістких завдань за допомогою брокера повідомлень. Практичне значення отриманих результатів полягає у створенні готового до впровадження вебзастосунку,

який скорочує час обробки замовлень та надає адміністративному персоналу потужний інструментарій для аналітики й прийняття управлінських рішень.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд предметної області

Ринок доставки їжі в Україні демонструє стабільне зростання, адаптуючись до сучасних соціально-економічних викликів [1]. Задоволення потреб клієнтів вимагає від закладів громадського харчування не лише забезпечення високої якості продукції, але й надання швидких, інтуїтивно зрозумілих онлайн-сервісів для її замовлення. У рамках предметної області можна виділити дві ключові категорії користувачів, взаємодію яких має підтримувати розроблювана система:

- клієнти: авторизовані або анонімні користувачі, які переглядають каталог продукції, формують кошик замовлення, застосовують маркетингові купони та здійснюють оплату зручним для них способом;
- адміністративний персонал: співробітники закладу, які потребують оперативного доступу до списку нових замовлень, інструментів швидкої зміни їх статусів та засобів аналітики для відстеження фінансових потоків і генерації звітності.

З інженерної точки зору, проектування сучасних інформаційних систем у сфері сучасних вебзастосунків вимагає переходу від традиційних монолітних та синхронних моделей обробки даних до гнучких асинхронних архітектурних патернів. Зростання обсягів транзакцій у пікові години навантаження ставить жорсткі вимоги до відмовостійкості, масштабованості бекенд-інфраструктури та пропускної здатності сервера [2]-[3].

Ефективне управління чергами завдань, ізоляція ресурсомістких процесів та мінімізація часу відгуку інтерфейсу стають критичними факторами, що безпосередньо впливають на конкурентоспроможність цифрового продукту на ринку.

Окрім оптимізації швидкодії, особливої уваги потребує забезпечення логічної та транзакційної цілісності фінансових даних. У динамічному бізнес-середовищі, де ціноутворення, наявність страв та умови маркетингових акцій

можуть змінюватися в режимі реального часу, архітектура бази даних повинна гарантувати сувору фіксацію стану замовлення на момент його оформлення [4]. Поєднання цих суворих бекенд-вимог із необхідністю побудови користувацького інтерфейсу, що здатен утримувати увагу клієнта, формує комплексне науково-практичне завдання. Вирішення цього завдання потребує залучення сучасних вебфреймворків, надійних СУБД та передових методів інженерії програмного забезпечення.

Важливим аспектом інженерії програмного забезпечення такого типу є врахування сучасних тенденцій розвитку інтерфейсів користувача та забезпечення кросплатформової адаптивності системи. Сучасний споживач висуває високі вимоги до швидкості завантаження та візуального відгуку контенту на різних типах пристроїв, що зумовлює широке використання методології Mobile First та залучення гнучких CSS-фреймворків для побудови інтерфейсних рішень [5].

Застосування технологій асинхронного обміну даними на клієнтській стороні дозволяє мінімізувати обсяги мережевого трафіку, забезпечуючи безшовне оновлення елементів сторінки без повного перезавантаження вікна браузера [6]. Це суттєво підвищує показники зрозумілості інтерфейсу й оптимізує взаємодію користувача з інформаційним середовищем застосунку.

Таким чином, системний аналіз предметної області та виявлених інженерних викликів свідчить про необхідність детального вивчення існуючих на ринку аналогів. Це дозволить чітко диференціювати розроблюване ПЗ, виявити архітектурні недоліки готових систем і обґрунтувати доцільність створення інженерного продукту на базі обраного стеку технологій [7].

1.2 Аналіз існуючих аналогів вебзастосунків підтримки роботи служби доставки

Для визначення основних вимог до розроблюваного програмного забезпечення було проведено аналіз існуючих на ринку рішень. Сучасні системи замовлення їжі можна умовно поділити на дві основні категорії: глобальні платформи-агрегатори та індивідуальні вебсайти закладів громадського харчування [8].

Одним із найпопулярніших представників першої категорії є міжнародний сервіс доставки Glovo. Інтерфейс вибору страв у цьому застосунку представлено на рис. 1.1 [9].

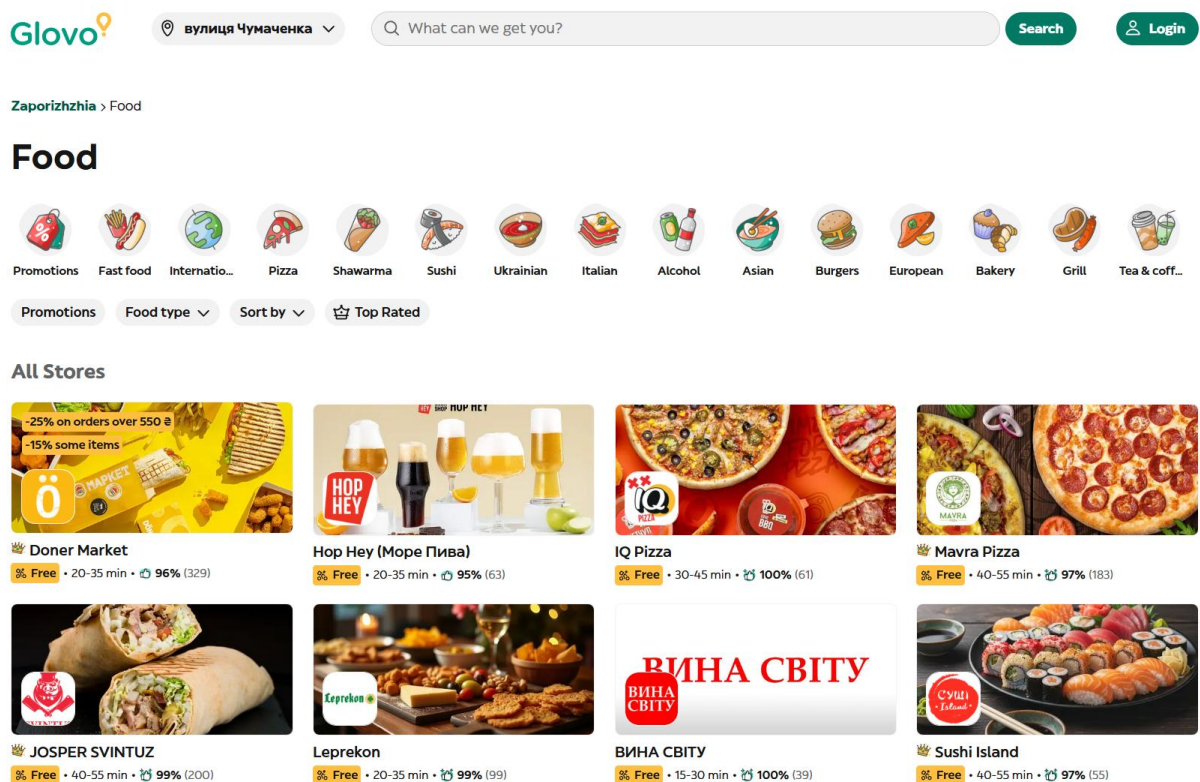


Рисунок 1.1 – Інтерфейс сторінки вибору ресторану у сервісі Glovo [9]

Перевагами платформ-агрегаторів є велика клієнтська база та власна логістична інфраструктура. Проте для власників ресторанного бізнесу співпраця з такими сервісами має суттєві недоліки. По-перше, це високі

комісійні збори (які можуть сягати до 30% від суми замовлення). По-друге, використання агрегатора унеможливлює повноцінне просування власного бренду, оскільки всі заклади представлені у стандартизованому інтерфейсі без можливості кастомізації. Крім того, заклад позбавлений можливості збирати детальну аналітику щодо своєї клієнтської бази.

До другої категорії належать власні вебсайти локальних ресторанів. Як приклад такого рішення розглянемо вебсайт місцевого закладу «Sir Lancelot», головну сторінку якого зображено на рис. 1.2 [10].

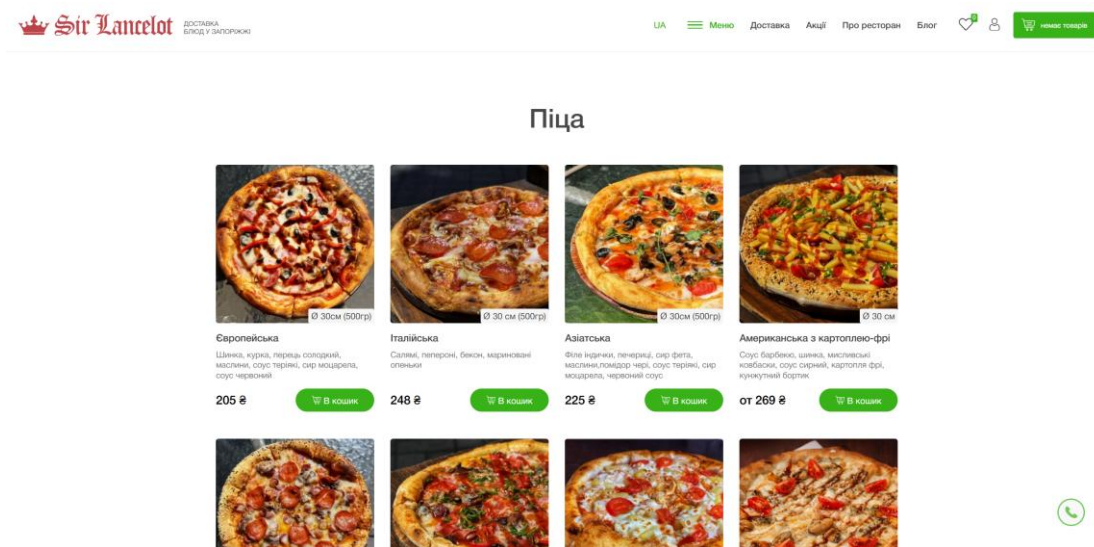


Рисунок 1.2 – Інтерфейс сторінки вибору страв у закладі Sir Lancelot [10]

Наявність власного сайту дозволяє закладу уникати комісій сторонніх платформ та вибудовувати пряму комунікацію з клієнтами [11]. Проте аналіз подібних локальних ресурсів часто виявляє ряд суттєвих технічних недоліків. До них належать: застарілий дизайн, відсутність адаптивної верстки для мобільних пристроїв, незручна реалізація кошика, що потребує повного перезавантаження сторінки при кожній дії, а також відсутність системи відстеження актуального статусу замовлення та інтеграції з сучасними платіжними шлюзами.

З архітектурної точки зору, більшість локальних рішень побудовані на базі застарілих універсальних систем управління контентом (CMS). Для таких

систем характерний високий ступінь зв'язності між логікою доступу до даних та візуальним представленням, що унеможливорює швидке масштабування та гнучку кастомізацію [12]. Саме тому при проєктуванні розроблюваної системи було прийнято рішення відмовитися від монолітних CMS на користь архітектури з розділеними рівнями даних та представлення, що забезпечить необхідну гнучкість і готовність до зростання навантажень.

На противагу монолітним CMS, сучасні стандарти веб-інженерії вимагають застосування спеціалізованих фреймворків, які використовують патерн MVC (Model-View-Controller) або його похідні. Це дозволяє чітко розмежувати бізнес-логіку, маршрутизацію та роботу з базою даних, забезпечуючи високу модульність коду [13].

Іншим критичним недоліком проаналізованих платформ є відсутність механізмів асинхронної обробки фонових завдань. У типових синхронних системах виконання ресурсомістких операцій наприклад, генерація розрахункових документів або маршрутизація сповіщень персоналу на зовнішні сервери – блокує основний потік виконання вебсервера. Це неминуче призводить до деградації продуктивності при збільшенні кількості запитів [14]. Проєктування відмовостійкої архітектури вимагає інтеграції брокерів повідомлень та ізольованих робочих процесів для делегування I/O-залежних задач, що гарантує високу доступність системи під час пікових навантажень. Перехід до асинхронної моделі обробки даних дозволить мінімізувати час відгуку сервера, оптимізувати використання апаратних ресурсів та забезпечити безперебійну роботу платформи навіть за умови значного масштабування.

На сучасному українському ринку електронної комерції у сфері харчування домінують великі платформи-агрегатори, такі як Glovo, Bolt Food [15]. Для прийняття рішень щодо архітектури розроблюваної системи було проведено порівняльний аналіз цих рішень за ключовими критеріями (табл. 1.1).

Таблиця 1.1 – Порівняння архітектури та бізнес-можливостей аналогів

Критерій порівняння	Платформи-агрегатори (на прикладі Glovo)	Локальні сайти ресторанів (на прикладі Sir Lancelot)	Розроблювана система
1	2	3	4
Брендування та впізнаваність	Просування бренду агрегатором. Заклад губиться серед конкурентів.	Власний бренд, проте загальна презентація часто страждає від застарілого дизайну.	Повний контроль над брендом, сучасний дизайн на базі Bootstrap 5.
Фінансові умови	Висока комісія (аж до 30% з кожного замовлення).	Відсутність комісій посередникам.	Відсутність комісій. Оплата лише фіксованого відсотка еквайрингу за онлайн-платежі.
Оформлення та Оплата	Сучасні онлайн-платежі (вбудовані в застосунок агрегатора).	Часто обмежені виключно готівкою або переказом на картку.	Безпечна інтеграція зі Stripe API для миттєвих безготівкових розрахунків.

Кінець таблиці 1.1

1	2	3	4
Аналітика	Базові статистичні дані, що надаються з затримкою.	Аналітика зазвичай відсутня або потребує додаткового ПЗ.	Інтерактивний дашборд (Chart.js).
Гнучкість бізнес-логіки та кастомізація	Низька: заклад обмежений жорсткими шаблонами та правилами агрегатора	Середня: обмежена можливостями базової CMS та доступними плагінами	Максимальна: повний архітектурний контроль над моделями, процесами
Доступ до клієнтських даних (маркетинг)	Відсутній: клієнтська база повністю належить агрегатору	Повний: дані зберігаються у власній базі, але аналітичні інструменти зазвичай базові	Повний: можливість будувати складні вибірки, дашборди та персоналізовані розсилки
Формування електронної звітності	Автоматизоване надання стандартних чеків	Потребує встановлення важких, часто платних сторонніх плагінів	Нативна генерація PDF-документів (бібліотека WeasyPrint)

Як видно з таблиці 1.1, розробка власного сучасного програмного забезпечення є оптимальним вибором. Це дозволяє усунути високі фінансові

витрати на агрегатори та водночас забезпечити клієнтам технічний рівень взаємодії, якого позбавлені типові локальні вебсайти старих зразків.

1.3 Постановка завдання

На основі проведеного аналізу предметної області та недоліків існуючих аналогів, головним завданням даної дипломної роботи є проєктування та розробка вебзастосунку з наступними ключовими функціональними та архітектурними вимогами:

- каталог продукції: розробка системи меню з категоризацією та оптимізованим відображенням деталей страв через модальні вікна;
- кошик користувача: реалізація механізму кошика на базі сесій, що дозволяє зберігати інформацію як для авторизованих, так і для анонімних користувачів, та синхронізувати його при реєстрації.;
- система замовлень та оплат: створення модуля оформлення замовлення з підтримкою двох методів: готівкою при отриманні та онлайн-оплатою банківською картою через інтеграцію з платіжним шлюзом Stripe API;
- маркетинговий модуль: впровадження системи промокодів з перевіркою валідності та фіксацією знижки у записі замовлення в БД;
- асинхронна архітектура: інтеграція брокера повідомлень Redis та системи Celery для винесення ресурсомістких процесів (відправка сповіщень у Telegram, генерація PDF-документів через WeasyPrint [16] та надсилання електронних листів) у фоновий режим;
- адміністративна панель: розробка закритого дашборду для персоналу, який включає інтерактивну таблицю замовлень з можливістю зміни статусу, а також блок візуальної аналітики, що базується на SQL-агрегаціях у СУБД PostgreSQL [17].

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування та серверного фреймворку

Для розробки серверної частини системи замовлення та доставки їжі найдоцільнішим вибором виступає мова програмування Python. Ця мова відрізняється потужною екосистемою, яка ідеально підходить для реалізації складних бекенд-архітектур, безпечної інтеграції платіжних шлюзів та керуванням асинхронних черг через брокери повідомлень. Визначившись із базовою технологією, ключове інженерне завдання звелось до порівняння двох домінуючих вебфреймворків – Django та Flask [18]-[19] (табл. 2.1).

Таблиця 2.1 – Порівняння вебфреймворків

Показник	Django	Flask
Придатність для реалізації складної бізнес-логіки (e-commerce)	Розробка ведеться мовою Python, доступна широка екосистема для фінансових розрахунків та фонових задач	Розробка ведеться мовою Python, підтримується інтеграція будь-яких пакетів
Наявні готові засоби розробки	Автентифікація, механізм шаблонізації, адміністративна панель, об'єктно-реляційне відображення (ORM)	Потребує самостійного встановлення та налаштування засобів розробки
Рівень вбудованої безпеки	Високий (захист від CSRF, XSS, SQL-ін'єкцій інтегровано в ядро)	Базовий (безпека реалізується розробником вручну)
Час розробки	Швидкий	Порівняно більший

Система Django пропонує монолітну архітектуру з підходом «усе включено», що є критичним фактором для проєктів електронної комерції. Наявність потужної вбудованої системи об'єктно-реляційного відображення (ORM), готового механізму автентифікації користувачів та автоматично згенерованої адміністративної панелі дозволяє зосередити зусилля безпосередньо на проєктуванні бізнес-логіки ресторану, а не на написанні рутинного інфраструктурного коду. Працюючи з Flask, розробник змушений самотійно підбирати, інтегрувати та підтримувати десятки сторонніх розширень для досягнення аналогічного базового функціоналу. Це природним чином збільшує тривалість розробки та підвищує ризики виникнення вразливостей через можливу несумісність сторонніх компонентів. Враховуючи гостру потребу в суворому контролі фінансових транзакцій, захисті від веб-вразливостей та зручному управлінні життєвим циклом замовлень персоналом, саме Django було визначено як фундаментальний фреймворк для програмної реалізації.

Обґрунтування вибору фреймворку Django також спирається на принципи проєктування надійних програмних систем. Використання патерну архітектури Model-View-Template забезпечує слабку зв'язність між бізнес-логікою, моделлю даних та інтерфейсом користувача. Такий підхід значно спрощує процеси рефакторингу, розширення функціоналу та модульного тестування, що є критично важливим при інженерії систем із високою ціною помилки [20]-[21].

Крім того, вбудована система ORM не лише прискорює ітеративну розробку, але й виступає потужним рівнем абстракції, що ізолює бекенд від специфіки конкретної СУБД [22]. Це дозволяє уникнути прямого написання низькорівневих SQL-запитів, що повністю нівелює ризик вразливостей типу SQL-ін'єкцій. Загалом, інформаційна безпека є однією з найсильніших сторін обраного рішення: фреймворк містить нативні механізми захисту від найбільш поширених вебзагроз, включаючи міжсайтовий скриптинг, підробку міжсайтових запитів та клікджекінг [23].

Враховуючи гостру потребу в суворому контролі фінансових транзакцій, захисті від вебвразливостей та зручному управлінні життєвим циклом замовлень персоналом, саме Django було визначено як фундаментальний фреймворк для програмної реалізації. На відміну від мікрофреймворків, розробка на яких часто зводиться до інтеграції різномірних бібліотек із різним рівнем підтримки, ядро Django гарантує високу стабільність коду та відповідає сучасним інженерним стандартам побудови корпоративних рішень [24].

2.2 Задача автоматизації життєвого циклу замовлення та забезпечення цілісності фінансових даних

Задача автоматизації процесу оформлення та обробки замовлень у розробленій системі зводиться до контролю потоку даних від моменту формування кошика користувачем до фінальної доставки страв та генерації звітної документації. Головна технічна складність полягає не лише у збереженні інформації в БД, а й у гарантуванні абсолютної фінансової стабільності транзакцій. Ресторанний бізнес характеризується високою динамікою: ціни на страви можуть змінюватися щодня залежно від вартості інгредієнтів, меню регулярно оновлюється, а маркетингові акції мають чіткі часові обмеження. ПЗ повинно ізолювати історичні дані замовлень від будь-яких майбутніх змін у каталозі.

Механізм роботи з клієнтом базується на використанні тимчасових сесій. Анонімний або авторизований відвідувач взаємодіє з каталогом, додаючи страви до кошика через асинхронні AJAX-запити. Система ідентифікує цей кошик, зберігає кількість обраних порцій та відстежує потенційно застосований промокод. Перехід до стадії оформлення ініціює створення жорсткого запису в СУБД PostgreSQL. Саме на цьому етапі відбувається денормалізація фінансових метрик: поточна ціна кожної страви копіюється в позицію замовлення, а відсоток активованої знижки фіксується у

профілі самого замовлення. Якщо через місяць адміністратор видалить застосований купон або підвищить ціну на конкретну піцу, архітектура БД гарантує, що старий чек залишиться незмінним, посилаючись на страви через механізм збереження нульових вказівників, але утримуючи власні розрахункові значення.

Після успішної фіксації транзакції, залежно від обраного методу оплати, система скеровує потік виконання. Вибір безготівкового розрахунку перенаправляє клієнта на зовнішній платіжний шлюз Stripe. Оформлення готівкою миттєво надсилає дані брокеру повідомлень Redis та фоновому воркеру Celery. Такий архітектурний підхід критично необхідний для того, щоб ресурсомісткі процеси - виклик зовнішніх API Telegram для сповіщення кухні, конвертація HTML-шаблонів у бінарні PDF-файли за допомогою WeasyPrint та надсилання листів через SMTP-сервер - не блокували основний HTTP-потік, забезпечуючи миттєвий відгук інтерфейсу [25]-[26].

2.3 Алгоритм обробки замовлень та асинхронної генерації звітної документації

Метод наскрізної обробки замовлень у розробленому вебзастосунку передбачає реалізацію суворої послідовності етапів транзакції. Процес охоплює шлях від фіксації наміру клієнта до інтеграції з фінансовими шлюзами та паралельної диспетчеризації сповіщень. Архітектура алгоритму спроектована таким чином, щоб гарантувати ізоляцію тривалих I/O операцій від основного потоку обробки HTTP-запитів.

На першому етапі робочі дані вивантажуються з тимчасового сховища. Таким сховищем виступає механізм сесій вебсервера, куди користувач попередньо додавав обрані страви. Кожен екземпляр даних у сесії перевіряється на відповідність структурі каталогу: ідентифікатор страви звіряється з БД PostgreSQL для підтвердження її фактичної наявності. Одночасно встановлюється перелік застосованих маркетингових

інструментів, зокрема перевіряється валідність промокоду, якщо його ідентифікатор присутній у контексті поточної сесії.

Далі формується транзакційне ядро замовлення. За кожним ініційованим замовленням створюється первинний запис у моделі, куди інтегруються контактні дані клієнта. Система виконує жорстку денормалізацію, поточні ціни страв копіюються у зв'язані таблиці позицій замовлення. Це дозволяє зафіксувати вартість кожної одиниці товару на момент покупки. Усі замовлення, для яких було ідентифіковано застосування купона, отримують прямий запис відсотка знижки у профіль замовлення, що блокує можливість зміни фінансової історії у випадку редагування умов акції адміністратором. Такий підхід цілком відповідає сучасним стандартам проєктування стійких БД [27].

При формуванні фінальних даних числові показники, які містять фінансові значення, проходять специфічну трансформацію залежно від обраного способу оплати. Для ініціації платіжної сесії Stripe загальна сума перетворюється на мінімальні розрахункові одиниці. Це кількість копійок, яка розраховується шляхом множення фінальної суми з урахуванням усіх знижок на сто та приведення результату до цілочисельного формату, що є обов'язковою вимогою API платіжного шлюзу.

Для успішно сформованих транзакцій на наступному етапі запускається конвеєр асинхронної обробки за допомогою системи Celery, що представляє собою ізольовані фонові процеси, роботу яких координує брокер повідомлень Redis. Загальну логіку та послідовність проходження даних через етапи алгоритму обробки відображено у вигляді блок-схеми (рис. 2.1).

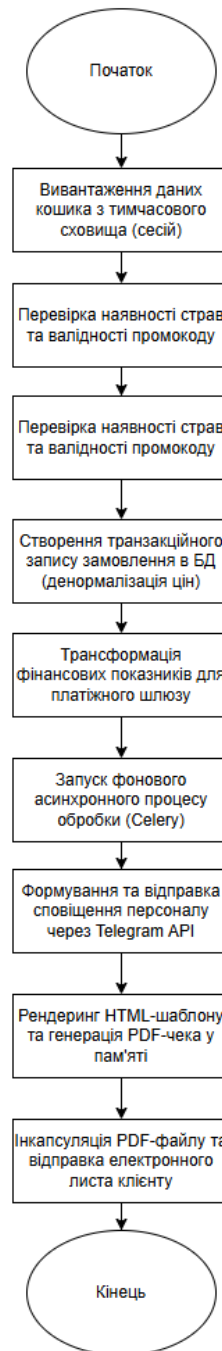


Рисунок 2.1 – Алгоритм обробки замовлення та генерації звітної документації

Делегування ресурсомістких завдань у фон дозволяє миттєво повернути користувачу сторінку успішного оформлення, не змушуючи його чекати на виконання мережевих запитів до сторонніх серверів [28].

У межах ізольованого завдання формується товарний чек. Для цього використовується бібліотека WeasyPrint. Система витягує дані збереженого замовлення, компілює їх через HTML-шаблон із відповідними правилами каскадних таблиць стилів (CSS) та рендерить у бінарний потік формату PDF в оперативній пам'яті. Це виключає необхідність виконання повільних операцій запису та читання з фізичного диска сервера.

Паралельно з генерацією документа на основі словника замовлення виконується структурування текстового повідомлення для адміністративного персоналу. Значення сумарної вартості, перелік обраних позицій страв та дані клієнта об'єднуються з використанням HTML-розмітки. Підготовлений пакет даних передається через протокол HTTPS методом POST до відкритого інтерфейсу Telegram API, що забезпечує миттєве сповіщення кухонного персоналу про нове надходження.

Останній етап алгоритму реалізується вже для повністю сформованих файлів та метаданих. На даному етапі об'єкт електронного листа інкапсулює тіло повідомлення, адресу отримувача та згенерований раніше бінарний потік PDF-файлу, який прикріплюється як вкладення. Тоді SMTP-клієнт виконує відправку пакета на сервер електронної пошти. Даний багатоступеневий етап фонові обробки повторюється для кожного нового успішно зафіксованого замовлення.

2.4 Вибір системи керування базами даних

На етапі початкового проєктування та локальної розробки допускалося використання легкої СУБД SQLite, проте для фінальної експлуатації та забезпечення відмовостійкості було обрано PostgreSQL. PostgreSQL є потужною об'єктно-реляційною системою керування БД, яка забезпечує високу продуктивність при паралельному доступі до даних, що є характерним для вебсайтів з великою кількістю одночасних замовлень.

Крім того, PostgreSQL суворо дотримується стандартів SQL та забезпечує надійну підтримку транзакцій (ACID), що гарантує логічну цілісність інформації при оформленні фінансових операцій, застосуванні промокодів та обробці позицій кошика.

2.5 Висновки за розділом 2

Ключові архітектурні рішення, запропоновані за результатами виконання розділу, полягають у наступному:

- метод забезпечення цілісності фінансових даних, який базується на денормалізації критичних параметрів транзакції (цін страв та відсотків знижок) у момент оформлення замовлення. Це гарантує незмінність історичної звітності незалежно від подальших модифікацій каталогу чи маркетингових акцій;
- архітектура асинхронної обробки замовлень, реалізована за допомогою черг Celery та брокера Redis. Винесення ресурсомістких операцій (генерація PDF-чеків через WeasyPrint, взаємодія з Telegram API та SMTP-серверами) у фоновий режим дозволило мінімізувати час очікування відповіді для кінцевого користувача та підвищити відмовостійкість платформи;
- механізм наскрізного управління станом кошика, що використовує сесії для синхронізації вибору товарів між анонімними та авторизованими відвідувачами, забезпечуючи безшовний користувацький досвід на базі технології AJAX.

Визначені матеріали та розроблені алгоритми взаємодії компонентів формують повний набір вимог для подальшої програмної реалізації та впровадження системи.

3 ОПИС ПРОГРАМИ

3.1 Опис програми

ПЗ розроблено для комплексної автоматизації бізнес-процесів закладу громадського харчування у сфері онлайн-замовлень та доставки їжі. Архітектурне рішення цілісно об'єднує клієнтський веб-інтерфейс та закриту адміністративну екосистему. З боку кінцевого споживача система надає інструментарій для інтуїтивної навігації каталогом, перегляду деталізованої інформації про інгредієнти страв, динамічного управління вмістом сесійного кошика із застосуванням маркетингових промокодів та безпечного проведення фінансових транзакцій. Для управлінського персоналу комплекс реалізує глибокий безперервний контроль над життєвим циклом транзакцій: від моменту первинної фіксації замовлення до зміни статусів приготування і доставки, що супроводжується асинхронною генерацією звітної PDF-документації, керуванням сповіщень та об'єднанням статистичних показників рентабельності в панелі аналітики.

3.2 Моделювання роботи з програмою за функціональними вимогами

Функціональні вимоги до програмного забезпечення вебсайту доставки продуктів харчування чітко розмежовано між двома основними ролями клієнтом, який ініціює замовлення, та менеджером (адміністративним персоналом), що забезпечує їх обробку та моніторинг.

Для базового користувача вебзастосунку (клієнта) визначено наступний перелік сценаріїв використання:

- виконання авторизації, що включає альтернативний сценарій — реєстрацію, необхідну для нових відвідувачів, які бажають зберегти історію замовлень;
- навігація каталогом страв;

- перегляд детальної інформації про страву (інгредієнти, ціна, фотографія) через виклик модального вікна;
- додавання страви до кошика, що супроводжується динамічним оновленням лічильника товарів за допомогою AJAX-запитів;
- керування вмістом кошика;
- застосування промокоду, що ініціює перерахунок фінальної суми з фіксацією відсотка знижки;
- оформлення замовлення, що розгалужується на два підсценарії залежно від обраного методу розрахунку, а саме оплатою готівкою при отриманні або онлайн-оплатою карткою через Stripe;
- відтворення особистого кабінету, де користувач може переглядати історію своїх успішних транзакцій.

Візуалізацію взаємодії Клієнта із системою наведено на діаграмі варіантів використання (рис. 3.1).

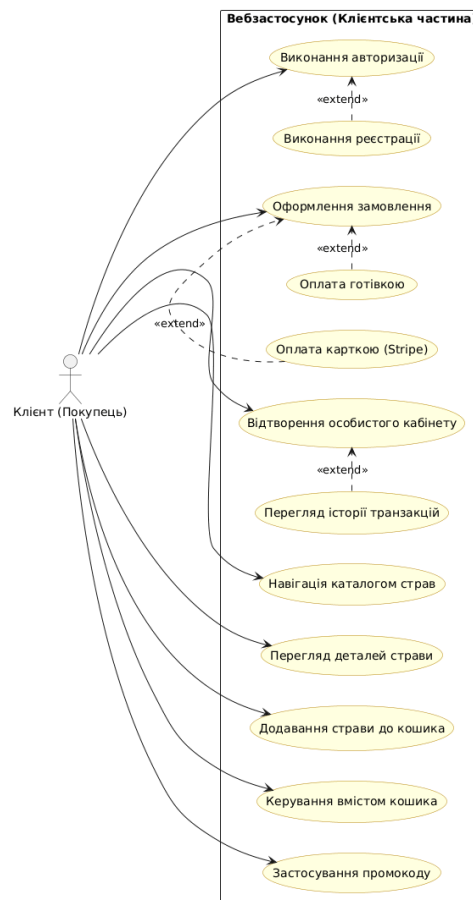


Рисунок 3.1 – Діаграма варіантів використання програми клієнтом

Для менеджера закладу громадського харчування передбачено розширений адміністративний інструментарій, що охоплює наступні сценарії використання:

- автентифікація в закритій панелі керування із перевіркою рівня доступу;
- моніторинг списку активних замовлень у режимі реального часу, реалізований через таблицю з пагінацією для оптимізації навантаження;
- оперативна зміна статусу замовлення (наприклад, з «Готується» на «В дорозі»), що автоматично запускає синхронну відправку Email-сповіщення клієнту про зміну стану його замовлення;
- відтворення аналітичного дашборда, що збирає фінансові показники та дозволяє виявляти пікові години навантаження на кухню за допомогою візуальних графіків Chart.js;
- генерація електронних документів, зокрема конвертація обраного графіку у PDF-формат для потреб внутрішньої звітності.

Архітектура адміністративної взаємодії змодельована на відповідній діаграмі варіантів використання для Менеджера (рис. 3.2).

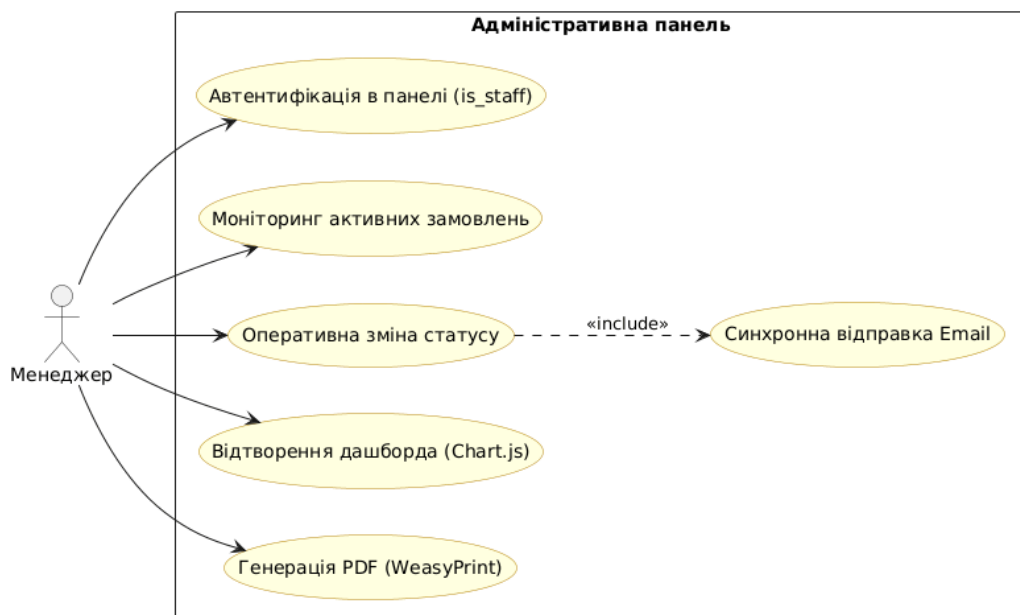


Рисунок 3.2 – Діаграма варіантів використання програми менеджером

3.3 Опис бази даних

БД спроектовано на основі реляційної СУБД PostgreSQL. Структура побудована з дотриманням принципів нормалізації для усунення надмірності даних та забезпечення їх логічної цілісності за допомогою зовнішніх ключів. Опис створеної структури подано нижче.

Для групування асортименту та структурування меню закладу громадського харчування вирішено виконати збереження категорій у БД таблицею Category з наступними полями:

- id – унікальний номер ідентифікації категорії страв;
- name – комерційна назва категорії;
- slug – унікальний ідентифікатор, необхідний для побудови SEO-оптимізованих URL-адрес та фронтенд-фільтрації;
- is_active – логічний прапорець, що дозволяє адміністратору тимчасово приховувати категорію з клієнтського інтерфейсу без необхідності її фізичного видалення.

Збереження детальної інформації про кожну позицію меню реалізовано таблицею FoodItem з полями:

- id – унікальний номер ідентифікації страви;
- category_id – категорія, до якої належить ця страва (зовнішній ключ до таблиці Category). Використовується стратегія каскадного видалення, що автоматично видаляє страву при ліквідації її батьківської категорії;
- name – назва страви, яка відображається в каталозі;
- slug – унікальний URL-ідентифікатор конкретної позиції;
- description – текстове поле для детального опису інгредієнтів та особливостей приготування;
- price – базова вартість страви, що гарантує відсутність похибок округлення при фінансових транзакціях;
- image – шлях до графічного файлу із зображенням страви;

– `is_available` – індикатор фактичної наявності товару на кухні для можливості його оперативного приховування з меню.

Керування системою знижок та маркетинговими акціями забезпечується через збереження параметрів промокодів у таблиці `Coupon`:

- `id` – унікальний номер ідентифікації купона;
- `code` – символічний код, який клієнт вводить для активації знижки;
- `valid_from` – дата та час початку дії маркетингової акції;
- `valid_to` – часова межа, після якої застосування промокоду блокується;

- `discount` – цілочисельний розмір знижки у відсотках (від 0 до 100);
- `active` – логічна позначка для можливості дострокової ручної деактивації купона менеджером.

Центральною сутністю системи, що об'єднує контактну інформацію клієнта, фінансові підсумки та статус виконання, є таблиця `Order`, яка містить наступні поля:

- `id` – унікальний номер ідентифікації замовлення;
- `user` – власник замовлення, якщо клієнт є авторизованим;
- `first_name` – ім'я клієнта для персоналізації комунікації;
- `last_name` – прізвище клієнта;
- `email` – електронна пошта для фонового надсилання PDF-чека за допомогою Celery;

- `phone_number` – контактний телефон для зв'язку;
- `address` – адреса доставки сформованого кошика;
- `created_at` – дата та час ініціалізації транзакції в системі;
- `updated_at` – дата та час останньої модифікації статусу замовлення;
- `status` – поточний стан виконання замовлення;
- `paid` – логічний індикатор успішного підтвердження оплати;
- `stripe_id` – унікальний ідентифікатор транзакції платіжної системи Stripe;

- `payment_method` – обраний спосіб розрахунку (готівка або банківська картка);
- `coupon_id` – застосований промокод (зовнішній ключ до таблиці `Coupon`);
- `discount` – денормалізоване значення відсотка знижки на момент оформлення, яке фіксується для захисту фінансової історії.

Збереження складу замовлень, оскільки один чек може містити декілька різних страв, реалізовано через розв'язну таблицю `OrderItem`:

- `id` – номер ідентифікації позиції у чеку;
- `order_id` – зовнішній ключ до таблиці `Order`;
- `food_item_id` – зовнішній ключ до таблиці `FoodItem`;
- `price` – зафіксована фактична вартість одиниці страви на момент покупки;
- `quantity` – замовлена кількість порцій даної страви.

Таблиця `OrderItem` виступає реалізацією для реляційної БД зв'язку типу «багато-до-багатьох» між таблицями `Order` та `FoodItem` із додаванням додаткових транзакційних атрибутів.

Діаграму бази даних представлено на рис. 3.3.

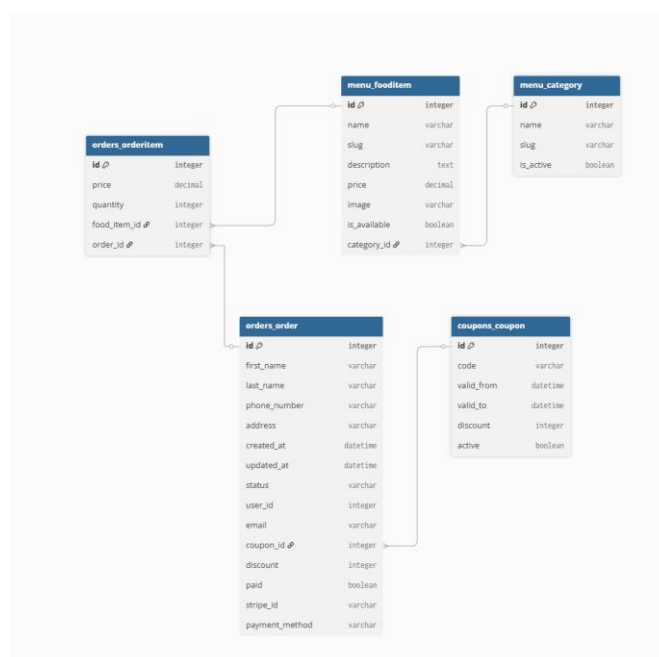


Рисунок 3.3 – Діаграма бази даних

3.4 Висновки за розділом 3

За результатами даного розділу успішно завершено етап проєктування та отримано повністю готову архітектурну й логічну модель програмного забезпечення вебсайту доставки продуктів харчування. Основні здобутки полягають у наступному:

- повністю сформовано та задокументовано логіку роботи системи: Чітко розмежовано готовий функціонал для двох ключових акторів (Клієнта та Менеджера). Завдяки побудованим діаграмам варіантів використання (Use Case) остаточно закріплено всі робочі сценарії взаємодії: від навігації каталогом, динамічного управління кошиком і чекауту для покупців до диспетчеризації замовлень, генерації PDF-звітності та моніторингу аналітики для адміністративного персоналу;
- створено готову до розгортання структуру бази даних: Розроблено та фіналізовано логічну ER-схему реляційної бази даних. Усі системні сутності (категорії, страви, замовлення, деталі замовлень, маркетингові купони), їх атрибути та зв'язки спроектовані з урахуванням суворих вимог нормалізації.

Отримані специфікації поведінки акторів і схема бази даних є повністю готовим фундаментом, який забезпечив швидкий перехід до етапу безпосереднього написання коду та впровадження системи.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ
ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Експлуатація програми

Клієнтський шлях розпочинається з головної сторінки, де представлено інтерактивний каталог продукції (рис. 4.1). Користувач має змогу миттєво перейти до потрібної категорії (наприклад, роли, піца, напої) за допомогою якірних посилань. Взаємодія з конкретною позицією меню ініціює виклик спливаючого модального вікна (рис. 4.2), яке збирає вичерпну інформацію: фотографію, перелік інгредієнтів, точну вартість та елементи керування порціями.

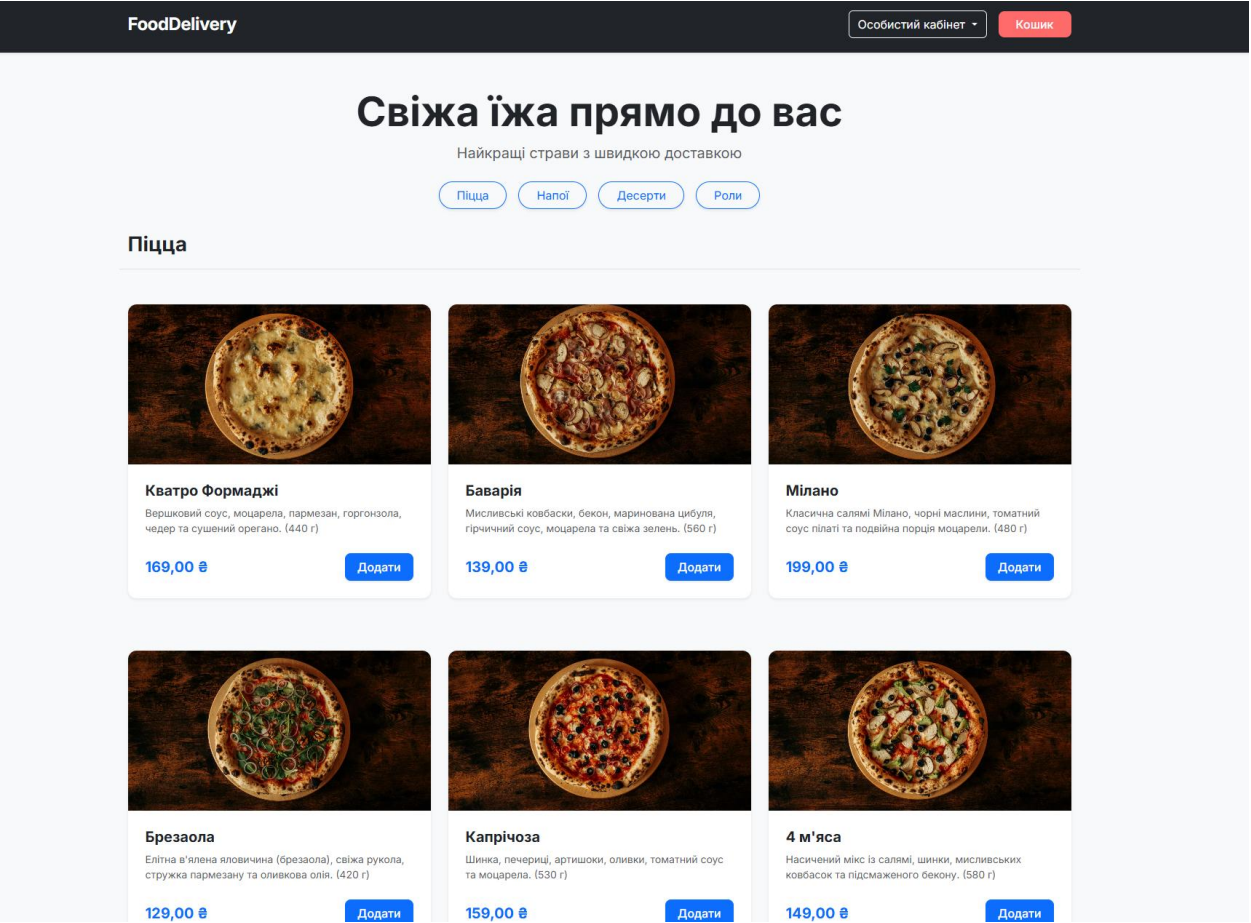


Рисунок 4.1 – Відтворення каталогу страв

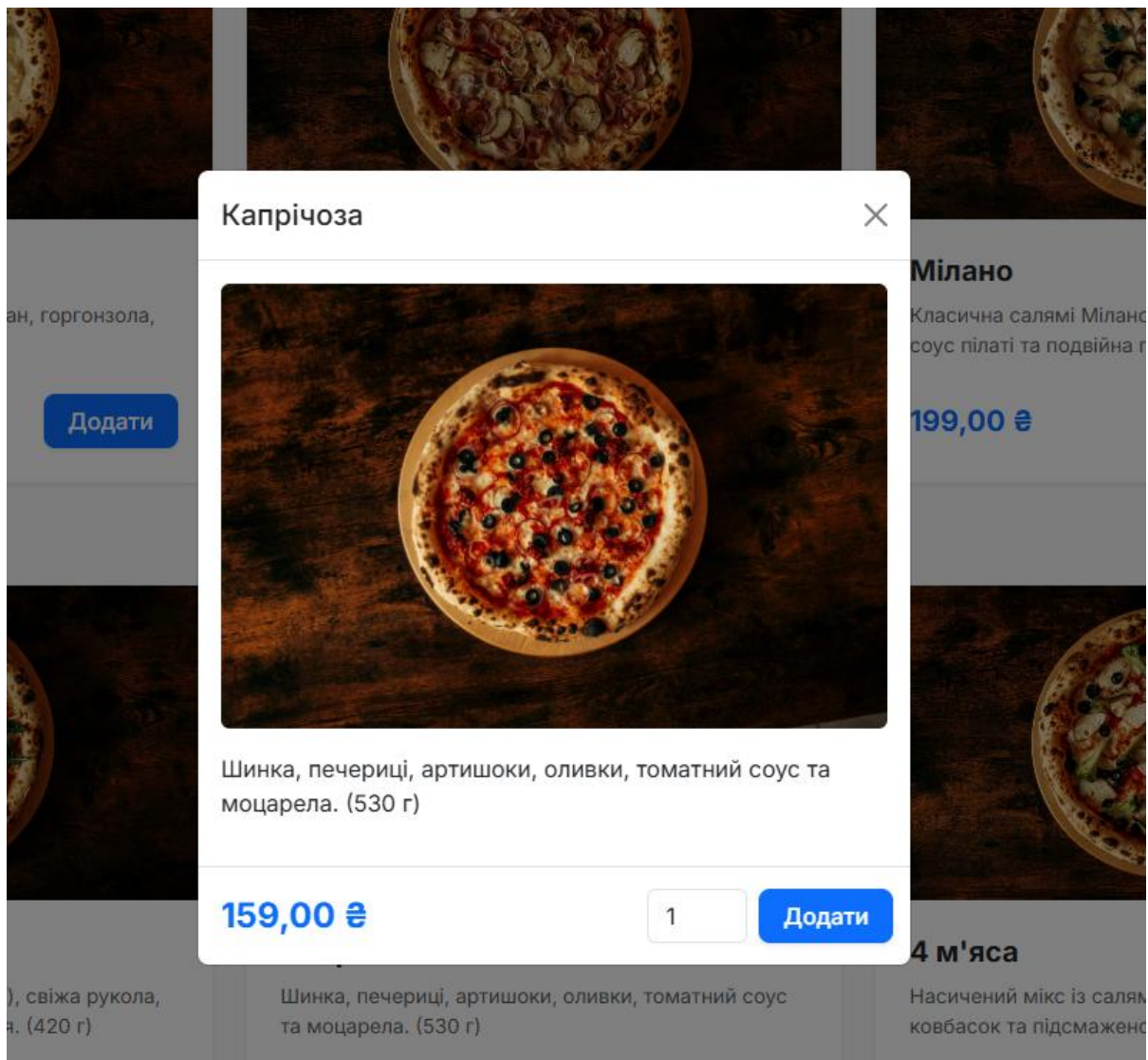


Рисунок 4.2 – Детальної інформація у модальному вікні

Розроблена програма має інтуїтивний та зрозумілий інтерфейс як для кінцевого споживача, так і для управлінського персоналу. Дизайн інтерфейсу побудований з урахуванням принципів зручності використання та сучасних стандартів веброзробки. інтерфейс коректно відображається на різних типах пристроїв і екранах, забезпечуючи комфортну роботу незалежно від платформи використання. Навігацію реалізовано у вигляді адаптивної верхньої панелі, яка автоматично стискається у компактне меню на мобільних пристроях, забезпечуючи безперешкодний доступ до ключових розділів системи (рис. 4.3).

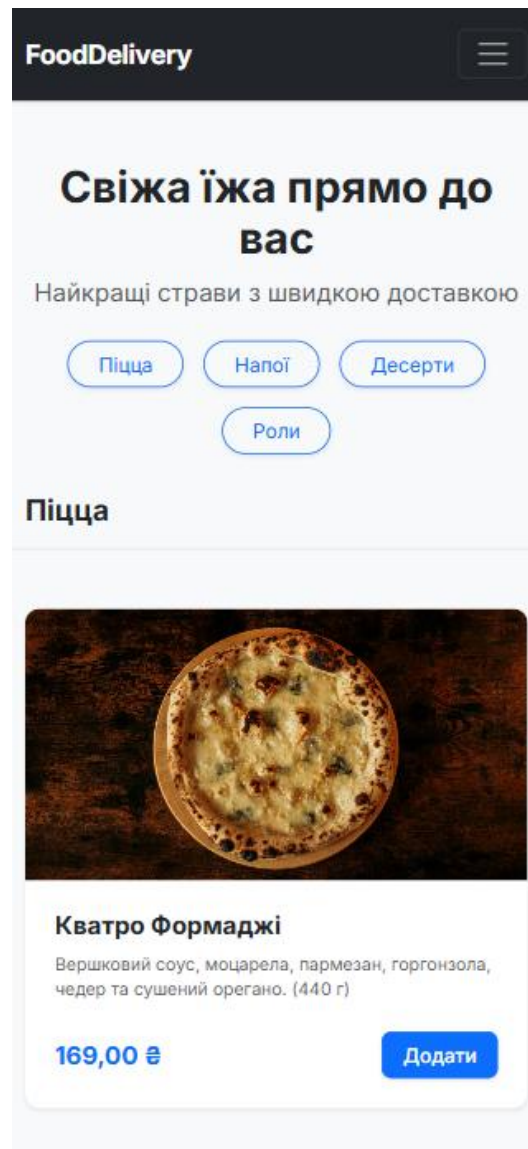


Рисунок 4.3 – Адаптивна панель навігації

Додавання обраної страви ініціює фоновий AJAX-запит, який непомітно для користувача оновлює стан сесійного кошика, динамічно змінюючи цифрові індикатори на навігаційній панелі. Перехід до самого кошика відкриває доступ до управління сформованим замовленням (рис. 4.4): тут клієнт може переглянути кількість позицій, видаляти непотрібні страви та активувати маркетингові промокоди. Успішна валідація купона миттєво перераховує фінальну суму. На етапі оформлення система пропонує гнучкий вибір платіжного вектора (рис. 4.5): класичний розрахунок готівкою при доставці або ж ініціалізацію безпечної транзакційної сесії через інтегрований шлюз Stripe.

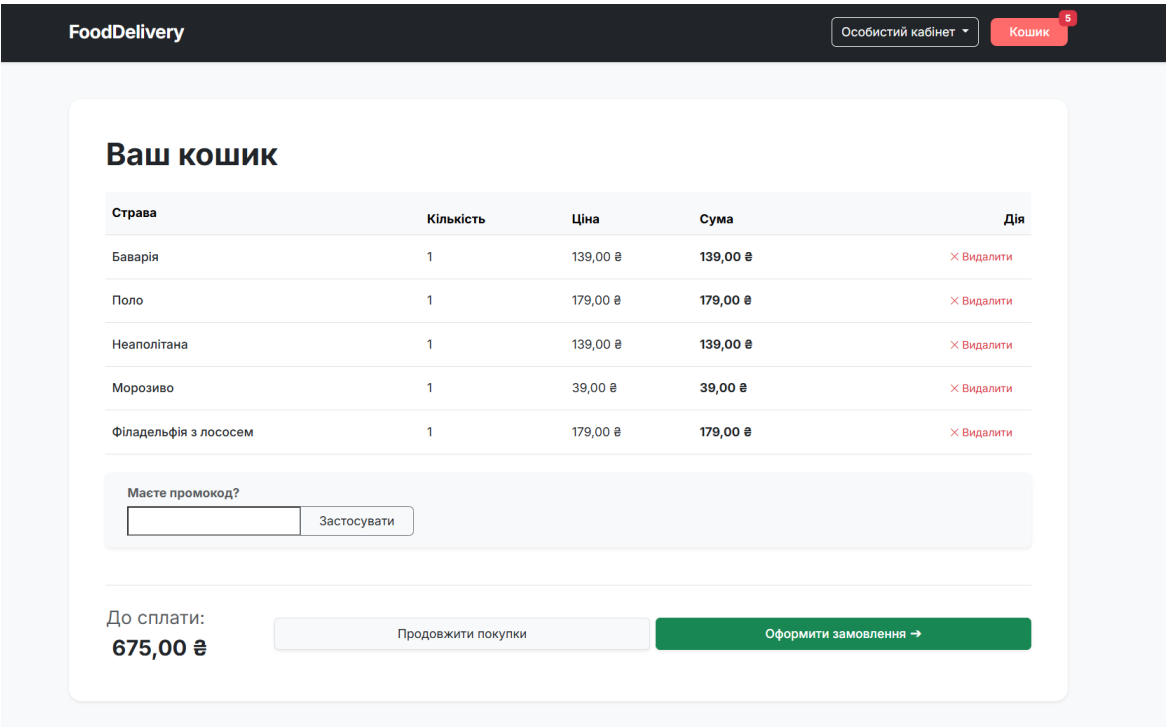


Рисунок 4.4 - Управління кошиком

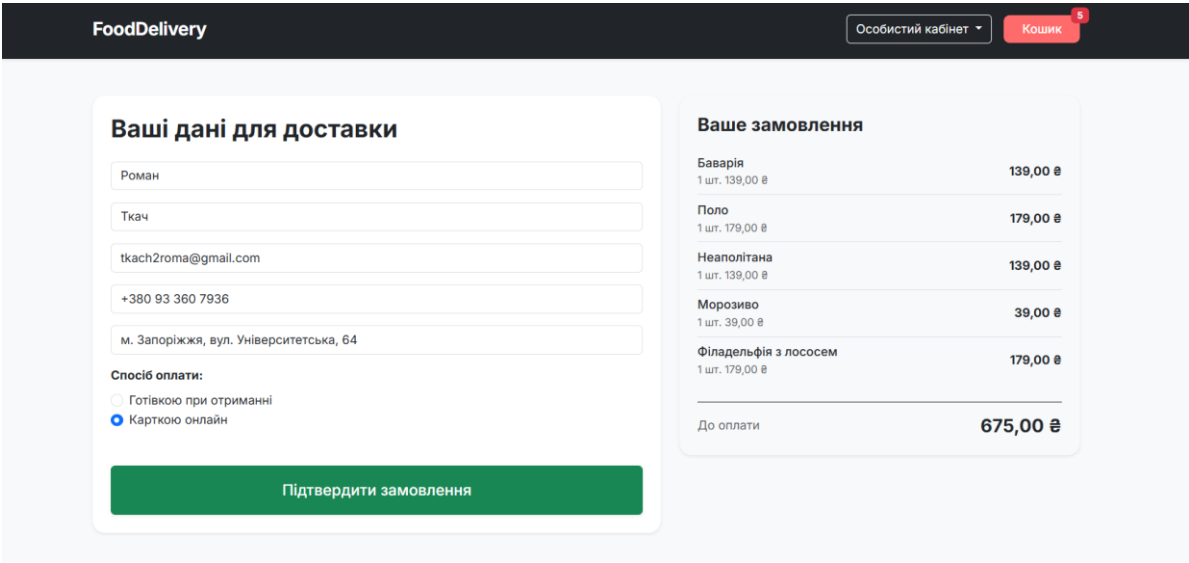


Рисунок 4.5 – Заповнення форми та вибір методу оплати

Одразу після успішної фіксації транзакції, система активує асинхронний алгоритм сповіщення. За допомогою фонового воркера Celery генерується структуроване текстове повідомлення, яке через POST-запит до відкритого інтерфейсу Telegram API миттєво надсилається у робочий чат-бот персоналу закладу (рис. 4.6). Це повідомлення містить унікальний

ідентифікатор замовлення, перелік обраних страв, контактні дані клієнта та підсумкову суму. Такий підхід дозволяє кухарям негайно розпочати приготування без необхідності безперервно оновлювати сторінку панелі адміністратора.

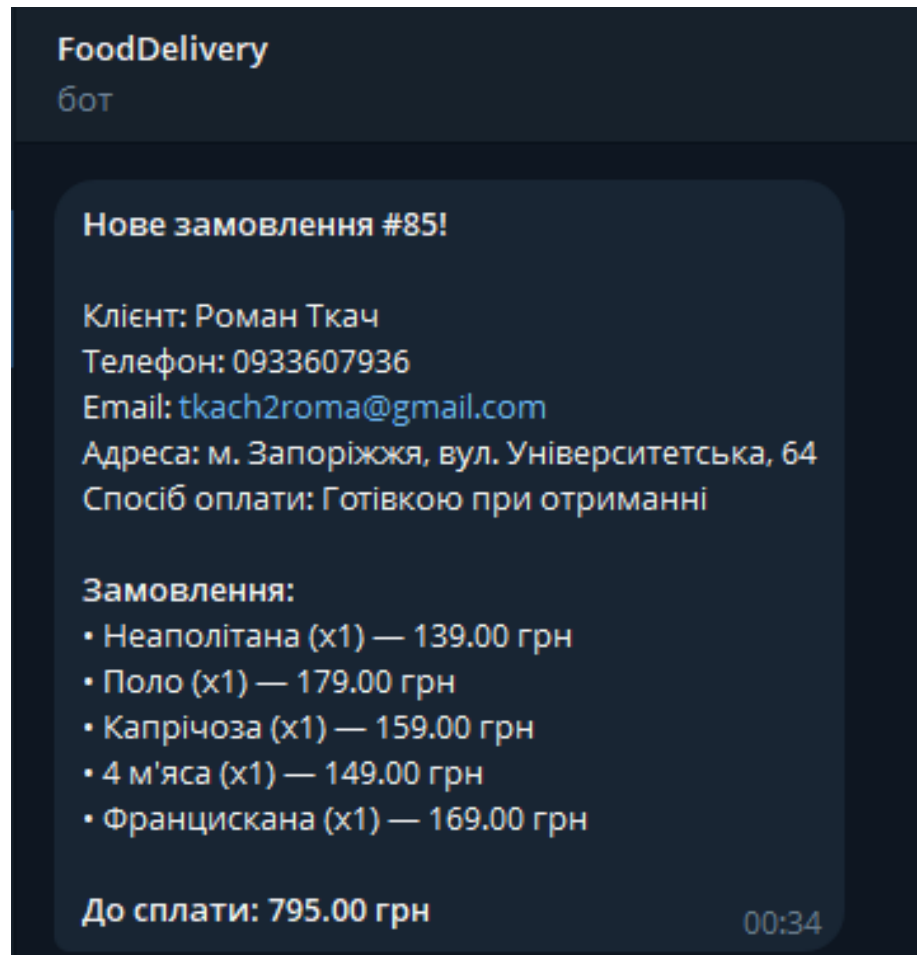


Рисунок 4.6 - Сповіщення про нове замовлення у Telegram-боті

ПЗ самостійно генерує PDF-файл чека за допомогою бінарної компіляції HTML-шаблонів сервісом WeasyPrint. Сама візуальна структура згенерованого документа (рис. 4.7) відповідає стандартам електронної комерції: вона містить повну номенклатуру страв, ціни на момент покупки, застосований відсоток знижки та реквізити закладу. Окрім ручного завантаження менеджером для друку, цей же PDF-файл автоматично прикріплюється як вкладення до електронного листа, який надсилається клієнту одразу після створення замовлення.

Чек до замовлення #89

Дата: 13.05.2026 22:12
Клієнт: Роман Ткач
Телефон: 093 360 7936
Email: tkach2roma@gmail.com
Адреса: м. Запоріжжя, вул. Університетська, 64
Спосіб оплати: Готівкою при отриманні

Товар	Кількість	Ціна за од.	Сума
Баварія	1	139,00 грн	139,00 грн
Неаполітана	1	139,00 грн	139,00 грн
Францискана	1	169,00 грн	169,00 грн
Чай	1	19,00 грн	19,00 грн
Проміжна сума:			466,00 грн
Знижка:			20%
До сплати:			372,80 грн

Рисунок 4.7 – Структура згенерованого PDF-чека замовлення

Авторизовані користувачі отримують доступ до особистого кабінету (рис. 4.8), де центральне місце займає модуль історії взаємодій. Цей розділ виводить хронологічно структурований список усіх минулих замовлень. Клієнт може розгорнути будь-яке замовлення для детального перегляду позицій та цін (рис. 4.9).

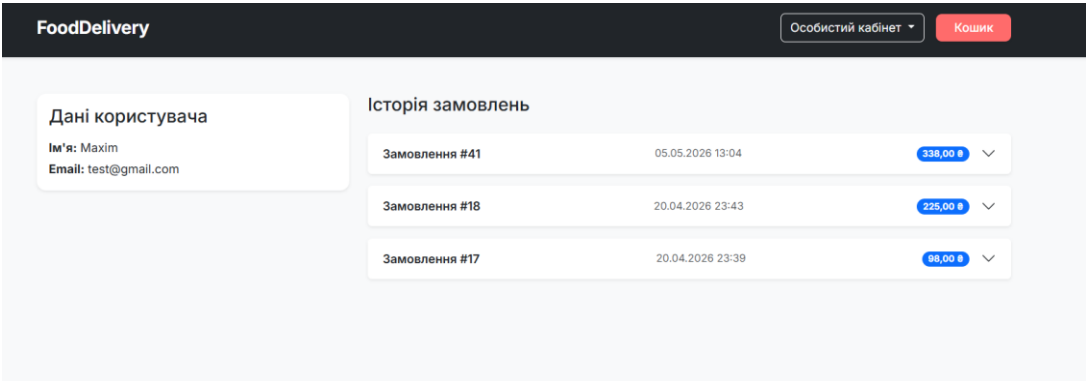


Рисунок 4.8 – Особистий кабінет клієнта

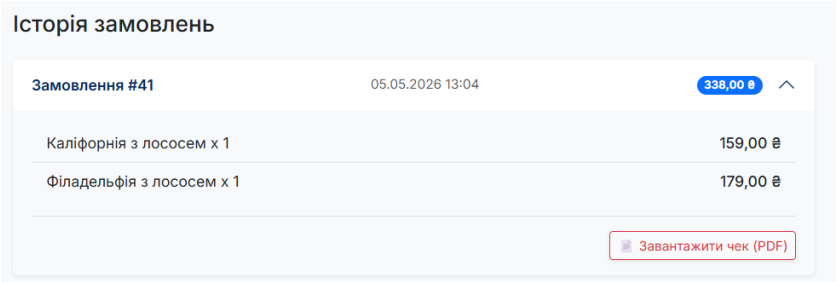


Рисунок 4.9 – Деталі історії минулого замовлення

Експлуатація системи адміністративним персоналом вимагає попередньої авторизації з підтвердженим рівнем доступу. Робочий простір менеджера відрізняється від клієнта. Головна панель містить пагіновану таблицю активних замовлень (рис. 4.10). Диспетчер може в один клік змінити статус приготування через випадаючий список, ця дія підкріплена скриптом Fetch API, що надсилає JSON-пакет на сервер, фіксує зміни у базі даних та візуально підсвічує успішне оновлення рядка, паралельно запускаючи механізм Email-сповіщення замовника (рис. 4.11).

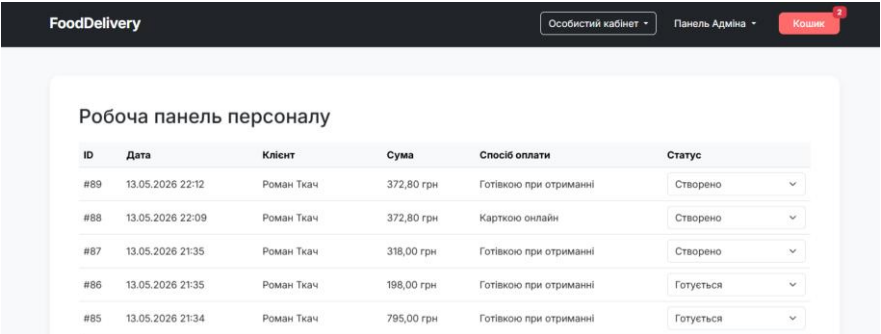


Рисунок 4.10 – Панель активних замовлень

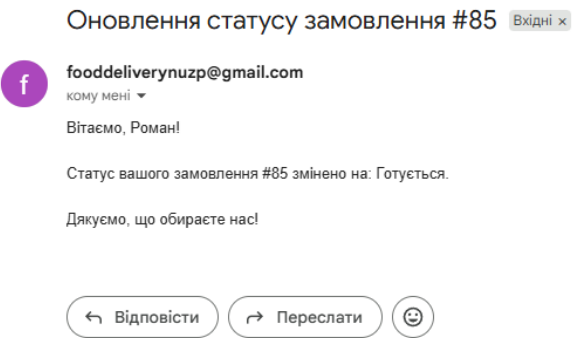


Рисунок 4.11 – Сповіщення клієнта про зміну статусу на електронну пошту

Для глибокого моніторингу бізнес-показників передбачено окрему вкладку аналітичного дашборду (рис. 4.12). Інтерфейс збирає фінансову статистику за обрані періоди, виводячи загальну суму доходу та перелік найбільш популярних страв. Візуальна складова забезпечується динамічними графіками Chart.js, які наочно ілюструють пікові години навантаження на кухню. З цієї ж панелі менеджер має можливість згенерувати PDF-документ звіту для будь-якої дати, що формується сервісом WeasyPrint та завантажується на пристрій для подальшого друку чи звітності (рис. 4.13).

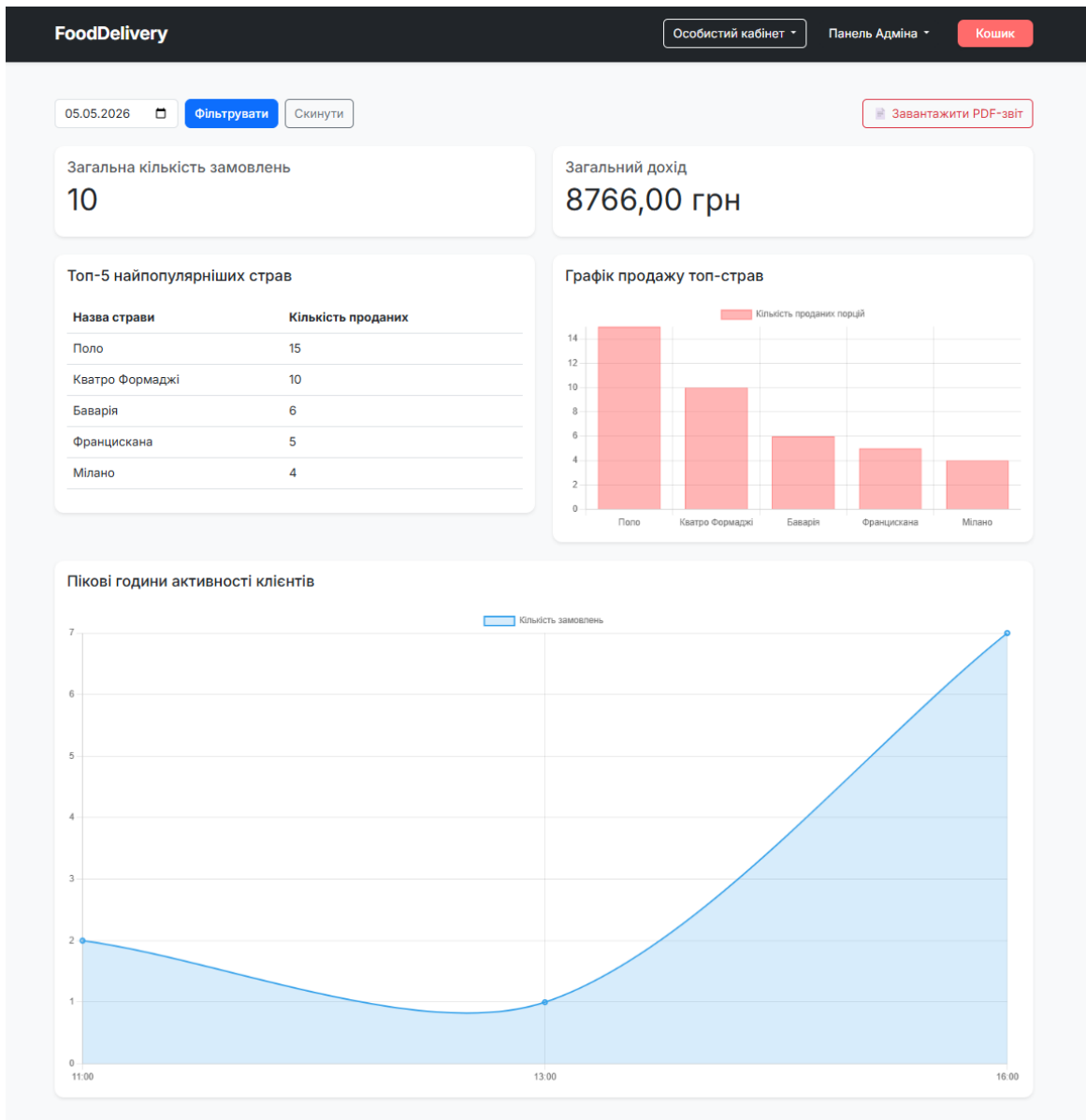


Рисунок 4.12 – Аналітичний дашборд з графіками Chart.js

Звіт зі статистики замовлень	
Звіт за дату: 2026-05-05	
ЗАГАЛЬНА КІЛЬКІСТЬ ЗАМОВЛЕНЬ	10
ЗАГАЛЬНИЙ ДОХІД	8766,00 грн
Топ найпопулярніших страв	
Назва страви	Кількість продажів
Поло	15
Квартро Формаджі	10
Баварія	6
Францискана	5
Мілан	4
Пікові години активності клієнтів	
Година	Кількість замовлень
11:00	2
13:00	1
16:00	7

Рисунок 4.13 – PDF-документ звіту

4.2 Експериментальне дослідження швидкодії обробки замовлень при використанні асинхронних черг завдань

Метою даного етапу дослідження стала кількісна оцінка ефективності впровадженої асинхронної архітектури на базі Celery та Redis у порівнянні з класичним синхронним підходом до обробки транзакцій. У сучасних вебсистемах доставки їжі час відгуку сервера впливає на рівень задоволеності клієнтів та коефіцієнт конверсії. Оскільки життєвий цикл замовлення у розробленому ПЗ включає низку ресурсомістких операцій - таких як генерація складних PDF-документів через WeasyPrint, мережеві запити до Telegram API та взаємодія з SMTP-серверами, то виникла необхідність експериментально підтвердити доцільність винесення цих процесів за межі основного HTTP-циклу.

Тестовий стенд експерименту включав образ вебсервера Django, БД PostgreSQL 15, брокер повідомлень Redis та воркер Celery. Для збору метрик було змодельовано стандартний сценарій оформлення замовлення. Вимірювання проводилися для двох конфігурацій:

- синхронна конфігурація: усі завдання (генерація чека, відправка Telegram-нотифікації менеджера, надсилання листа клієнту) виконуються послідовно в межах одного запиту контролера `order_create`;
- асинхронна конфігурація: контролер лише фіксує дані в БД та делегує обробку черзі Celery через метод `.delay()`, миттєво повертаючи відповідь користувачу.

Для забезпечення достовірності результатів було виконано 10 ітерацій для кожного сценарію (рис 4.14), результати яких усереднювалися. Основні часові витрати на виконання окремих підзадач у синхронному режимі представлені в таблиці 4.1.

```

=== Результати замірів часу ===
БД: 58.68 мс
PDF: 239.63 мс
Telegram: 385.24 мс
Email: 1785.59 мс
=====

```

Рисунок 4.14 – Одна з ітерацій експерименту

Таблиця 4.1 – Розподіл часових витрат при синхронній обробці замовлення

Етап обробки (підзадача)	Середній час виконання, мс	Відсоток від загального часу, %
Валідація даних та запис у PostgreSQL	57,2	2,15
Генерація PDF-чека (WeasyPrint)	194,6	7,32
Запит до Telegram API (Bot сповіщення)	435,6	16,37
Відправка Email з вкладенням (SMTP)	1972,6	74,16
Загальний час блокування сервера	2660,0	100

Аналіз даних таблиці 4.1 свідчить, що при синхронному підході клієнт змушений очікувати на відповідь сервера понад 2,5 секунди. Основне навантаження припадає на зовнішні I/O операції (SMTP та API) та процес рендерингу документа, який є критично залежним від ресурсів процесора. У реальних умовах експлуатації за наявності нестабільного мережевого з'єднання із зовнішніми сервісами цей час може зростати до 5–7 секунд, що є неприпустимим для комерційних систем.

Результати порівняльного дослідження швидкодії обох конфігурацій за показником загального часу очікування відповіді наведено у таблиці 4.2.

Таблиця 4.2 – Порівняння швидкодії синхронного та асинхронного методів

Тип архітектурного рішення	Середній час очікування, мс	Пропускна здатність запитів/сек
Синхронна обробка	2660,0	~0,38
Асинхронна обробка (Celery + Redis)	115,0	~8,69
Покращення показника	у ~23 рази	

Впровадження асинхронного методу дозволило скоротити час блокування основного потоку виконання до 115 мс, що фактично дорівнює часу на валідацію та один SQL-транзакційний запис. Решта ресурсомістких операцій виконується паралельно в ізольованому воркері, не впливаючи на сприйняття швидкодії системи користувачем.

Отримані результати дозволяють наочно побачити різницю в швидкості виконання процесу замовлення. Це підтверджує, що обрана архітектура з використанням брокера Redis забезпечує високу масштабованість: вебсервер здатний обробляти значно більшу кількість одночасних замовлень, оскільки він звільнений від необхідності чекати завершення фонових процесів. Таким чином, експериментально доведено, що

асинхронна модель є оптимальною для реалізації вебсайту доставки їжі, оскільки вона гарантує мінімальну затримку інтерфейсу та стабільність системи при зростанні інтенсивності вхідного трафіку.

ВИСНОВКИ

У роботі спроектовано та повністю реалізовано програмне забезпечення вебсайту для замовлення продуктів харчування з доставкою, що дозволило комплексно автоматизувати ключові бізнес-процеси закладу громадського харчування та успішно досягти поставленої мети. Створена інформаційна система усуває необхідність залучення високовартісних платформ-агрегаторів, надаючи підприємству незалежний канал комунікації зі споживачами та повний контроль над власними фінансовими потоками.

Практичним результатом виконання дипломного проєкту є розгорнутий вебзастосунок, розділений на клієнтський та закритий адміністративний простір. Для покупців розроблено безшовний механізм навігації інтерактивним меню, управління сесійним кошиком без обов'язкової попередньої авторизації, систему валідації маркетингових промокодів та модуль безпечного безготівкового розрахунку. Персонал закладу отримав інструментарій для миттєвої зміни статусів замовлень і моніторингу показників рентабельності через візуалізований аналітичний дашборд. Технологічним фундаментом серверної частини виступила мова Python з вебфреймворком Django та системою керування реляційними базами даних PostgreSQL. Клієнтську взаємодію побудовано на основі фреймворку Bootstrap 5, асинхронних AJAX-запитів та JavaScript-бібліотеки Chart.js.

Головна інженерна цінність роботи полягає у розробці стійкої архітектури, яка гарантує стабільність навіть під час високих навантажень. Проведене експериментальне дослідження швидкодії підтвердило критичну необхідність впровадження розподілених черг завдань. Перенесення ресурсомістких I/O операцій - генерації PDF-звітності за допомогою бібліотеки WeasyPrint, взаємодії з Telegram API та відправки електронних листів через SMTP – до фонового воркєру Celery під управлінням брокера повідомлень Redis кардинально зменшило час блокування основного потоку виконання. Отримані результати тестування довели, що такий підхід

забезпечує високу пропускну здатність сервера та миттєвий відгук інтерфейсу, що робить розроблену систему повністю придатною для комерційної експлуатації в умовах пікових навантажень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Доставка їжі в Україні: ключові тренди та цифри на 2025 рік. Star Lunch – Доставка обідів у Києві з щоденним меню. URL: <https://www.starlunch.com.ua/food-delivery-in-ukraine-key-trends-and-figures-for-2025/> (дата звернення: 12.05.2026).
2. Sommerville I. Software Engineering / I. Sommerville. – 10th ed. – Boston : Pearson, 2015. – 816 p.
3. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. – Boston : Addison-Wesley Professional, 2012. – 560 p.
4. Date C. J. An Introduction to Database Systems / C. J. Date. – 8th ed. – Boston : Pearson, 2003. – 1024 p.
5. Круг С. Не змушуйте мене думати. / С. Круг. – Київ : ArtHuss, 2024. – 198 с.
6. Фленаган Д. JavaScript: Детальний посібник / Д. Фленаган. – 7-ме вид. – Київ. : Науковий світ, 2021. – 720 с.
7. Мартін Р. Чиста архітектура. Мистецтво розроблення програмного забезпечення / Р. Мартін. – Харків : Фабула, 2019. – 416 с.
8. ТОП-застосунків для доставки їжі. CASES. URL: <https://cases.media/en/article/top-zastosunkiv-dlya-dostavki-yizhi> (дата звернення: 12.05.2026).
9. Food delivery in Zaporizhzhia. Glovo. URL: https://glovoapp.com/en/ua/zaporizhzhia/categories/food_1 (date of access: 14.05.2026).
10. Доставка піци у Запоріжжі. Sir Lancelot. URL: <https://lancelot.zp.ua/> (дата звернення: 14.05.2026).
11. Устимович Ю. Glovo, Rocket та Bolt Food: топ-5 застосунків для доставки їжі в Україні. *ThePage*. URL: <https://thepage.ua/ua/business/servisi-dostavki-yizhi-v-ukrayini-top-5-najpopulyarnishih-sluzhb> (дата звернення: 14.05.2026).

12. Richards M. Fundamentals of Software Architecture: An Engineering Approach / M. Richards, N. Ford. – O'Reilly Media, 2020. – 432 p.
13. Pressman R. Web Engineering: A Practitioner's Approach / R. Pressman, D. Lowe. – New York : McGraw-Hill, 2009. – 458 p..
14. Клеппман М. Проєктування систем, що працюють з великими обсягами даних / М. Клеппман. – Харків : Фабула, 2021. – 688 с.
15. Чому власний сайт доставки кращий за агрегатори для вашого закладу. *Інструменти для зростання бізнесу. Автоматизації бізнесу в Україні. CRM-система, мобільний додаток, web-сайт.* URL: <https://qbtools.com.ua/blog/pochemu-sobstvennyy-sayt-dostavki-luchshe-agregatorov-dlya-vashego-zavedeniya> (дата звернення: 12.05.2026).
16. WeasyPrint 68.1 documentation. *WeasyPrint 68.1 documentation.* URL: <https://doc.courtbouillon.org/weasyprint/stable/> (date of access: 12.05.2026).
17. PostgreSQL: Documentation. *PostgreSQL: The world's most advanced open source database.* URL: <https://www.postgresql.org/docs/> (date of access: 12.05.2026)
18. Django documentation | Django documentation. *Django Project.* URL: <https://docs.djangoproject.com/en/6.0/> (date of access: 12.05.2026).
19. Welcome to Flask – Flask Documentation (3.1.x). *Welcome to Flask – Flask Documentation (3.1.x).* URL: <https://flask.palletsprojects.com/en/stable/> (date of access: 12.05.2026).
20. Percival H. Test-Driven Development with Python: Obey the Testing Goat: Using Django, Selenium, and JavaScript / H. Percival. – 2nd ed. – O'Reilly Media, 2017. – 622 p.
21. Pressman R. S. Software Engineering: A Practitioner's Approach / R. S. Pressman, B. R. Maxim. – 9th ed. – McGraw-Hill Education, 2019. – 704 p.
22. Obe R. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database / R. Obe, L. Hsu. – 3rd ed. – O'Reilly Media, 2017. – 314 p.

23. Stuttard D. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws / D. Stuttard, M. Pinto. – 2nd ed. – Wiley, 2011. – 912 p.
24. Rubio A. Django 3 By Example: Build powerful and reliable Python web applications from scratch / A. Rubio. – 3rd ed. – Packt Publishing, 2020. – 754 p.
25. Telegram APIs. *Telegram APIs*. URL: <https://core.telegram.org/> (date of access: 12.05.2026).
26. Fowler M. Refactoring: Improving the Design of Existing Code / M. Fowler. – 2nd ed. – Boston : Addison-Wesley Professional, 2019. – 448 p.
27. Coronel C. Database Systems: Design, Implementation, & Management / C. Coronel, S. Morris. – 13th ed. – Cengage Learning, 2019. – 816 p.
28. Brahmi K. Mastering Celery: A Guide to Background Tasks, Workers, and Parallel Processing in Python. URL: <https://khairi-brahmi.medium.com/mastering-celery-a-guide-to-background-tasks-workers-and-parallel-processing-in-python-eea575928c52> (date of access: 12.05.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для організації та управління процесами прийому онлайн-замовлень і доставки їжі.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розроблення програмного забезпечення вебсайту для замовлення продуктів харчування з доставкою», затверджене наказом Національного університету «Запорізька політехніка» №139 від 7 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для автоматизації бізнес-процесів закладу громадського харчування у сфері доставки їжі.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для реалізації комплексної роботи вебзастосунку замовлення та доставки їжі як щодо обслуговування клієнтів, так і щодо управління бізнес-процесами закладу, необхідно забезпечити такі функціональні можливості:

- виконання реєстрації та авторизації покупців та адміністративного персоналу;
- відтворення особистого кабінету клієнта;
- відтворення каталогу страв із можливістю навігації за категоріями;
- відтворення детальної інформації про обрану страву;
- керування сесійним кошиком;
- застосування маркетингового промокоду з автоматичним перерахунком фінальної вартості замовлення;

- оформлення замовлення з розгалуженням сценаріїв оплати;
- відтворення історії замовлень в особистому кабінеті клієнта;
- відтворення деталізованого вмісту збереженого замовлення;
- моніторинг списку активних замовлень у режимі реального часу менеджером системи;
- оперативна зміна статусу виконання замовлення адміністративним персоналом;
- автоматична генерація звітної документації;
- автоматичне асинхронне надсилання сповіщень при оформленні та зміні статусу замовлення;
- відтворення аналітичного дашборда з візуалізацією фінансових показників для менеджера;
- керування каталогом страв, категоріями та маркетинговими купонами через закриту адміністративну панель.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення вебсайту замовлення та доставки їжі повинен бути інтуїтивно зрозумілим, зручним та адаптивним. Повинна бути забезпечена можливість комфортної роботи користувачів через сучасні веббраузери.

А.3.3 Вимоги до надійності

Програмне забезпечення вебсайту для замовлення та доставки їжі повинно забезпечувати надійне та безперебійне функціонування. Критично важливі процеси (оформлення транзакцій, генерація PDF-чеків, відправка Email та Telegram сповіщень) повинні бути стійкими до збоїв і виконуватися асинхронно для запобігання блокуванню основного потоку виконання

сервера. Фінансові дані замовлень мають бути захищені механізмами цілісності реляційної бази даних.

A.3.4 Умови експлуатації

Експлуатація ПЗ вебсайту замовлення та доставки їжі має забезпечуватися виконанням ряду вимог до серверної частини:

- наявність вебсервера з підтримкою інтерпретатора мови програмування Python 3;
- налаштування Django-застосунку для виконання на сервері із забезпеченням бази даних під керуванням СУБД PostgreSQL;
- наявність розгорнутого брокера повідомлень Redis та запущеного фонового процесу Celery для обробки асинхронних завдань;
- інсталювані пакети та системні залежності для роботи генератора PDF-документів (WeasyPrint).

A.3.5 Вимоги до складу та параметрів технічних засобів

Для апаратного забезпечення сервера доцільним є використання віртуального або фізичного виділеного сервера з процесором, обчислювальної потужності якого достатньо для оброблення запитів у режимі реального часу. Обсяг оперативної пам'яті сервера має забезпечувати одночасну стабільну роботу вебфреймворку, СУБД PostgreSQL, брокера Redis та процесів Celery без зниження продуктивності системи. Для зберігання кодової бази, бази даних і медіафайлів потрібен накопичувач із вільним дисковим простором, що забезпечує резерв для подальшого масштабування.

Програмне забезпечення серверної частини передбачає наявність операційної системи, що підтримує роботу сучасних вебтехнологій. На клієнтській стороні достатньо використання будь-якого сучасного пристрою (персонального комп'ютера, смартфона, планшета) із встановленим

веббраузером, який підтримує стандарти HTML5, CSS3 та JavaScript для коректного відображення користувацького інтерфейсу. Зазначені вимоги створюють передумови для стабільної роботи системи.

А.3.6 Вимоги до маркування і пакування

Програмне забезпечення вебсайту може бути записане на будь-якому носії інформації. На пакуванні повинна бути назва «Програмне забезпечення вебсайту замовлення та доставки їжі».

А.4 Вхідні та вихідні дані системи

Вхідними даними до ПЗ вебсайту замовлення та доставки їжі є інформація, що вводиться користувачами в процесі взаємодії з вебінтерфейсом та передається на серверну частину для опрацювання. До таких даних належать: облікові відомості для реєстрації та автентифікації, параметри взаємодії з кошиком, маркетингові промокоди, а також контактні відомості й адреси при оформленні замовлення. Для адміністративного персоналу вхідними даними є інформація для наповнення каталогу та статуси опрацювання замовлень. Окрім цього, вхідними даними виступають сигнали від зовнішніх сервісів про статус проведення транзакцій.

Вихідними даними системи є результати оброблення запитів користувачів. До них належать: сформований вміст кошика, розрахована фінальна вартість замовлення з урахуванням застосованих знижок, згенеровані офіційні електронні документи, сформовані та відправлені Email і Telegram-сповіщення, а також статистичні дані, що відображаються у вигляді візуальних графіків в аналітичному дашборді адміністратора.

ДОДАТОК Б
Текст програми

Б.1 Текст файла cart.py

```

class Cart:
    def __init__(self, request):
        self.session = request.session
        cart = self.session.get(settings.CART_SESSION_ID)
        if not cart:
            cart = self.session[settings.CART_SESSION_ID] = {}
        self.cart = cart

        self.coupon_id = self.session.get('coupon_id')

    def add(self, food_item, quantity=1, override_quantity=False):
        food_id = str(food_item.id)
        if food_id not in self.cart:
            self.cart[food_id] = {'quantity': 0, 'price': str(food_item.price)}

            if override_quantity:
                self.cart[food_id]['quantity'] = quantity
            else:
                self.cart[food_id]['quantity'] += quantity

        self.save()

    def save(self):
        self.session.modified = True

    def remove(self, food_item):
        food_id = str(food_item.id)
        if food_id in self.cart:
            del self.cart[food_id]
            self.save()

    def __iter__(self):
        food_ids = self.cart.keys()
        food_items = FoodItem.objects.filter(id__in=food_ids)
        cart = copy.deepcopy(self.cart)

        for food_item in food_items:
            cart[str(food_item.id)]['food_item'] = food_item

        for item in cart.values():
            item['price'] = Decimal(item['price'])
            item['total_price'] = item['price'] * item['quantity']

```

```

        yield item

    def __len__(self):
        return sum(item['quantity'] for item in self.cart.values())

    def get_total_price(self):
        return sum(Decimal(item['price']) * item['quantity'] for item in
self.cart.values())

    def clear(self):
        del self.session[settings.CART_SESSION_ID]
        if 'coupon_id' in self.session:
            del self.session['coupon_id']
        self.save()

    @property
    def coupon(self):
        if self.coupon_id:
            try:
                return Coupon.objects.get(id=self.coupon_id)
            except Coupon.DoesNotExist:
                pass
        return None

    def get_discount(self):
        if self.coupon:
            return (self.coupon.discount / Decimal(100)) * self.get_total_price()
        return Decimal(0)

    def get_total_price_after_discount(self):
        return self.get_total_price() - self.get_discount()

```

Б.2 Текст файлу models.py

```

class Order(models.Model):
    class Status(models.TextChoices):
        CREATED = 'CREATED', _('Створено')
        PROCESSING = 'PROCESSING', _('Готується')
        SHIPPING = 'SHIPPING', _('В дорозі')
        DELIVERED = 'DELIVERED', _('Доставлено')
        CANCELLED = 'CANCELLED', _('Скасовано')

    user = models.ForeignKey(

```

```

        settings.AUTH_USER_MODEL,
        related_name='orders',
        on_delete=models.CASCADE,
        verbose_name=_('Користувач')
    )
    first_name = models.CharField(max_length=50, verbose_name=_('Ім\'я'))
    last_name = models.CharField(max_length=50, verbose_name=_('Прізвище'))
    email = models.EmailField(verbose_name=_('Email'))
    phone_number = models.CharField(max_length=20, verbose_name=_('Номер телефону'))
    address = models.CharField(max_length=250, verbose_name=_('Адреса доставки'))

    created_at = models.DateTimeField(auto_now_add=True, verbose_name=_('Час створення'))
    updated_at = models.DateTimeField(auto_now=True, verbose_name=_('Час оновлення'))

    status = models.CharField(max_length=20, choices=Status.choices, default=Status.CREATED, verbose_name=_('Статус'))
    paid = models.BooleanField(default=False, verbose_name=_('Оплачено'))
    stripe_id = models.CharField(max_length=250, blank=True)
    payment_method = models.CharField(max_length=20, choices=[('cash', 'Готівкою при отриманні'), ('card', 'Карткою онлайн')], default='cash', verbose_name=_('Спосіб оплати'))

    coupon = models.ForeignKey(Coupon, related_name='orders', null=True, blank=True, on_delete=models.SET_NULL, verbose_name=_('Купон'))
    discount = models.IntegerField(default=0, validators=[MinValueValidator(0), MaxValueValidator(100)], verbose_name=_('Знижка (%)'))

    class Meta:
        ordering = ['-created_at']
        verbose_name = _('Замовлення')
        verbose_name_plural = _('Замовлення')

    def __str__(self):
        return f"Замовлення #{self.id} від {self.first_name}"

    def get_total_cost(self):
        return sum(item.get_cost() for item in self.items.all())

    def get_total_cost_after_discount(self):
        total = self.get_total_cost()
        if self.discount:

```

```

    return total - (total * self.discount / 100)
return total

```

```

class OrderItem(models.Model):
    order = models.ForeignKey(Order, related_name='items',
on_delete=models.CASCADE, verbose_name=_('Замовлення'))
    food_item = models.ForeignKey(FoodItem, related_name='order_items',
on_delete=models.SET_NULL, null=True, verbose_name=_('Страва'))
    price = models.DecimalField(max_digits=10, decimal_places=2,
verbose_name=_('Ціна'))
    quantity = models.PositiveIntegerField(default=1, verbose_name=_('Кількість'))

    class Meta:
        verbose_name = _('Позиція замовлення')
        verbose_name_plural = _('Позиції замовлення')

    def __str__(self):
        return f'{self.id}'

    def get_cost(self):
        return self.price * self.quantity

```

Б.3 Текст файлу tasks.py

```

@shared_task
def order_created_task(order_id):
    try:
        order = Order.objects.get(id=order_id)

        bot_token = settings.TELEGRAM_BOT_TOKEN
        chat_id = settings.TELEGRAM_CHAT_ID

        if bot_token and chat_id:
            items_text = ""
            for item in order.items.all():
                item_name = item.food_item.name if item.food_item else "Видалена страва"
                items_text += f"• {item_name} (x{item.quantity}) – {item.get_cost():.2f} грн\n"

            total_cost = order.get_total_cost()
            message = (
                f"<b>Нове замовлення #{order.id}</b>\n\n"

```

```

f"Клієнт: {order.first_name} {order.last_name}\n"
f"Телефон: {order.phone_number}\n"
f"Email: {order.email}\n"
f"Адреса: {order.address}\n"
f"Спосіб оплати: {order.get_payment_method_display()}\n\n"
f"<b>Замовлення:</b>\n"
f"{items_text}\n"
)

if order.discount:
    final_cost = order.get_total_cost_after_discount()
    message += (
        f"Проміжна сума: <s>{total_cost:.2f} грн</s>\n"
        f"Знижка: {order.discount}%\n"
        f"<b>До сплати: {final_cost:.2f} грн</b>"
    )
else:
    message += f"<b>До сплати: {total_cost:.2f} грн</b>"

url = f"https://api.telegram.org/bot{bot_token}/sendMessage"
payload = {
    "chat_id": chat_id,
    "text": message,
    "parse_mode": "HTML"
}
response = requests.post(url, json=payload)
if response.status_code != 200:
    print(f"Помилка відправки в Telegram: {response.text}")
else:
    print(f"Замовлення {order_id} успішно відправлено в Telegram.")
else:
    print("Telegram токени не налаштовані.")

if order.email:
    html = render_to_string('orders/order/pdf.html', {'order': order})

    pdf = weasyprint.HTML(string=html).write_pdf()

    subject = f"Чек до замовлення #{order.id} - Food Delivery"
    body = f"Шановний/а {order.first_name},\n\nДякуємо за ваше замовлення.\nУ вкладенні ви знайдете чек.\n\nСмачного!"
    email = EmailMessage(
        subject=subject,
        body=body,
        from_email=settings.EMAIL_HOST_USER,

```

```

        to=[order.email]
    )

    email.attach(f"order_{order.id}.pdf", pdf, "application/pdf")

    email.send()
    print(f"Email чек для замовлення {order.id} успішно відправлено на
{order.email}.")
    else:
        print(f"Замовлення {order_id} не має Email адреси, пропускаємо
відправку листа.")

except Order.DoesNotExist:
    print(f"Помилка: Замовлення з ID {order_id} не знайдено.")
except Exception as e:
    print(f"Сталася непередбачена помилка: {e}")

```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Початковий слайд

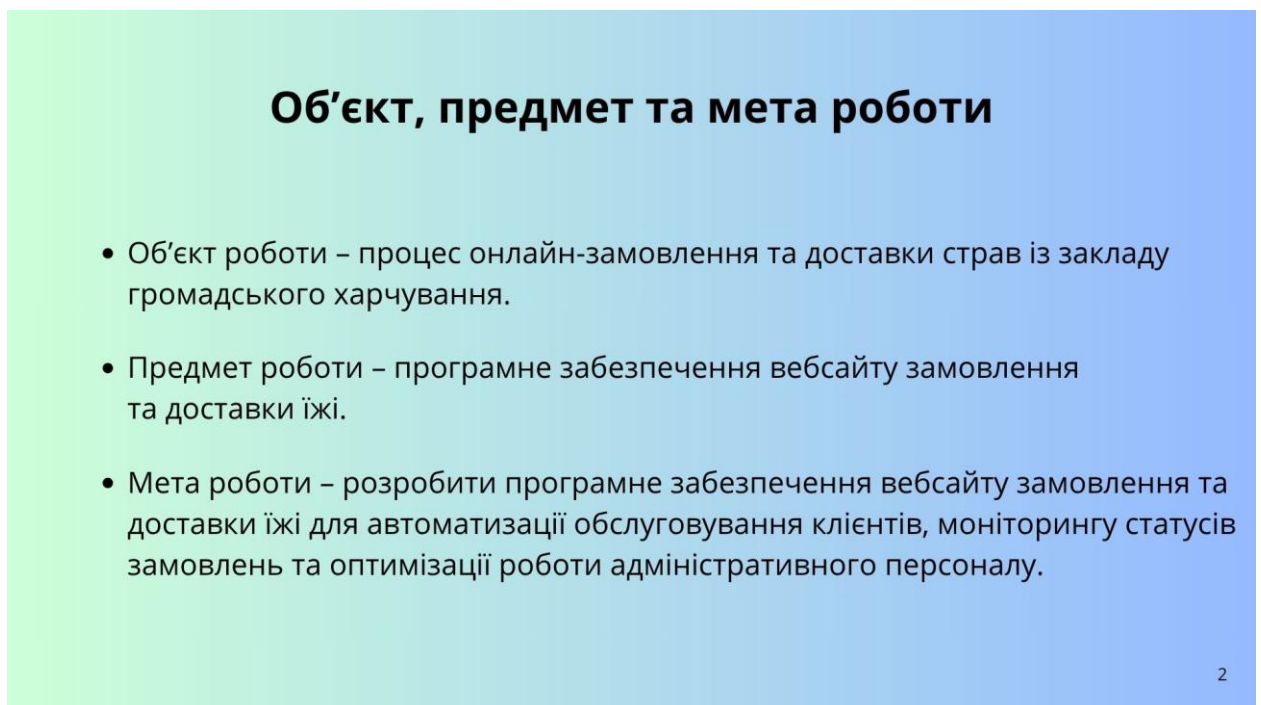


Рисунок В.2 – Об'єкт, предмет та мета роботи

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра треба розв'язати такі задачі:

- проаналізувати предметну область та існуючі процеси онлайн-замовлення їжі;
- спроектувати архітектуру та базу даних програмного забезпечення вебсайту замовлення та доставки їжі;
- розробити програмне забезпечення вебсайту замовлення та доставки їжі;
- дослідити швидкодію розробленої системи та ефективність асинхронної обробки замовлень.

3

Рисунок В.3 – Завдання роботи

Результат порівняння аналогів вебзастосунків для підтримки роботи служби доставки

Критерій	Платформи-агрегатори (на прикладі Glovo)	Локальні сайти ресторанів (на прикладі Sir Lancelot)	Розроблювана система
Брендування та впізнаваність	Просування бренду агрегатором. Заклад губиться серед конкурентів	Власний бренд, проте загальна презентація часто страждає від застарілого дизайну	Повний контроль над брендом, сучасний дизайн на базі Bootstrap 5
Фінансові умови	Висока комісія	Відсутність комісій посередникам	Відсутність комісій. Оплата лише фіксованого відсотка еквайрингу за онлайн-платежі
Оформлення та Оплата	Сучасні онлайн-платежі (вбудовані в застосунок агрегатора)	Часто обмежені виключно готівкою або переказом на картку	Безпечна інтеграція зі Stripe API для миттєвих безготівкових розрахунків
Аналітика	Базові статистичні дані, що надаються з затримкою	Аналітика зазвичай відсутня або потребує додаткового ПЗ	Інтерактивний дашборд (Chart.js)
Гнучкість бізнес-логіки та кастомізація	Низька: заклад обмежений жорсткими шаблонами та правилами агрегатора	Середня: обмежена можливостями базової CMS та доступними плагінами	Максимальна: повний архітектурний контроль над моделями, процесами
Доступ до клієнтських даних (маркетинг)	Відсутній: клієнтська база повністю належить агрегатору	Повний: дані зберігаються у власній базі, але аналітичні інструменти зазвичай базові	Повний: можливість будувати складні вибірки, дашборди та персоналізовані розсилки
Формування електронної звітності	Автоматизоване надання стандартних чеків	Потребує встановлення важких, часто платних сторонніх плагінів	Нативна генерація PDF-документів у пам'яті сервера (бібліотека WeasyPrint)

4

Рисунок В.4 – Результат порівняння аналогів вебзастосунків для підтримки роботи служби доставки

Результат порівняння вебфреймворків		
Показник	Django	Flask
Придатність для реалізації складної бізнес-логіки (e-commerce)	Розробка ведеться мовою Python, доступна широка екосистема для фінансових розрахунків та фонових задач	Розробка ведеться мовою Python, підтримується інтеграція будь-яких пакетів
Наявні готові засоби розробки	Автентифікація, механізм шаблонізації, адміністративна панель, об'єктно-реляційне відображення (ORM)	Потребує самостійного встановлення та налаштування засобів розробки
Рівень вбудованої безпеки	Високий (захист від CSRF, XSS, SQL-ін'єкцій інтегровано в ядро)	Базовий (безпека реалізується розробником вручну)
Час розробки	Швидкий	Порівняно більший

5

Рисунок В.5 – Результат порівняння вебфреймворків



Рисунок В.6 – Діаграма варіантів використання сайту клієнтом

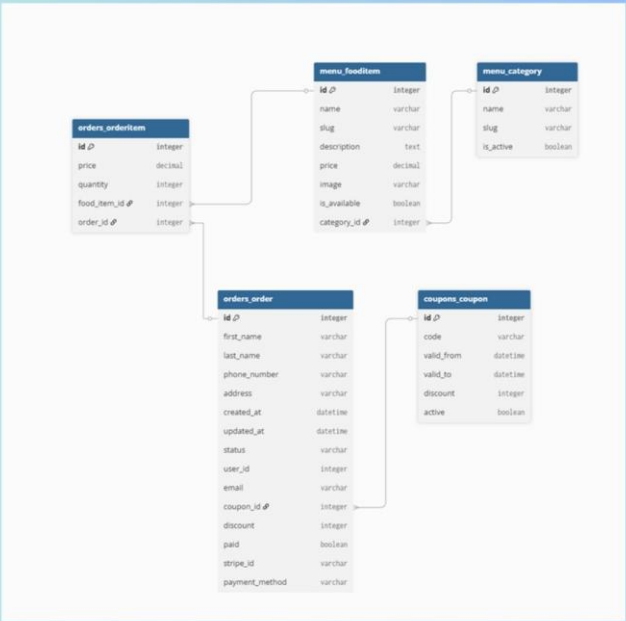
Діаграма варіантів використання сайту менеджером



7

Рисунок В.7 – Діаграма варіантів використання сайту менеджером

Схема бази даних вебзастосунку



8

Рисунок В.8 – Схема бази даних вебзастосунку

Алгоритм обробки замовлення

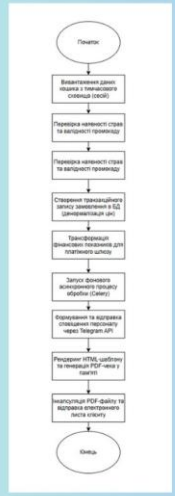


Рисунок В.9 – Алгоритм обробки замовлення

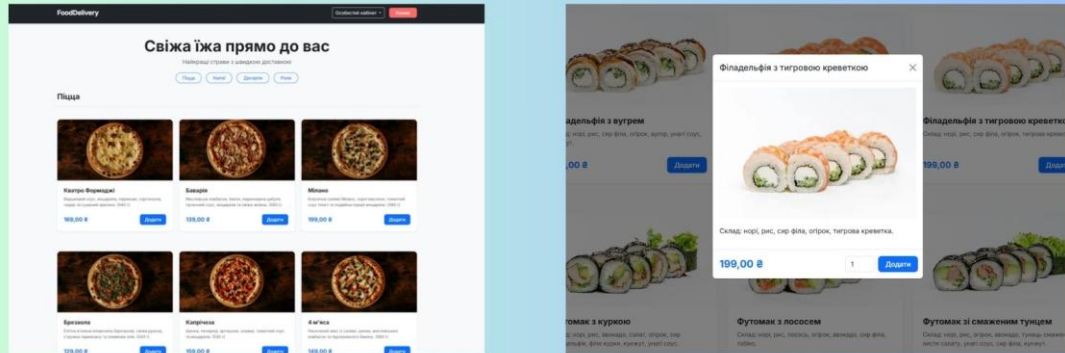
Результати дослідження швидкодії обробки замовлень при використанні асинхронних черг завдань

Етап обробки (підзадача)	Середній час виконання, мс	Відсоток від загального часу,
Валідація даних та запис у	57,2	2,15
Генерація PDF-чека (WeasyPrint)	194,6	7,32
Запит до Telegram API (Bot)	435,6	16,37
Відправка Email з вкладенням	1972,6	74,16
Загальний час блокування	2660	100

Тип архітектурного рішення	Середній час очікування, мс	Пропускна здатність запитів/сек
Синхронна обробка	2660	~0,38
Асинхронна обробка (Celery + Redis)	115	~8,69
Покращення показника	у ~23 рази	

Рисунок В.10 – Результати дослідження швидкодії обробки замовлень при використанні асинхронних черг завдань

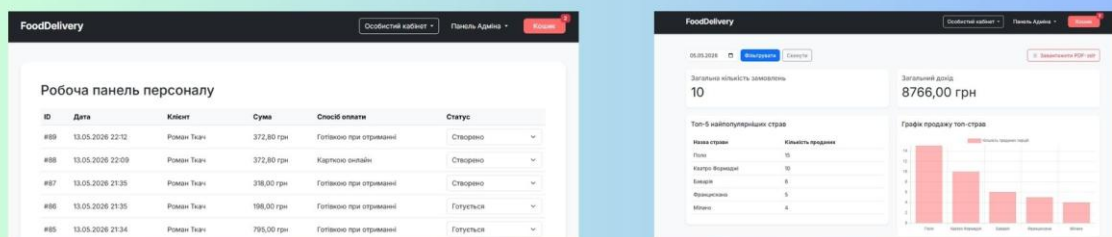
Інтерфейс програми: головна сторінка сайту зі сторони клієнта



11

Рисунок В.11 – Інтерфейс програми: головна сторінка сайту зі сторони клієнта

Інтерфейс програми: панель персоналу та аналітика продажів



12

Рисунок В.12 – Інтерфейс програми: панель персоналу та аналітика продажів

Чек-лист тестування сайту

Функція	Неавторизований користувач	Авторизований користувач (Клієнт)
Перегляд каталогу страв	+	+
Навігація за категоріями	+	+
Перегляд детальної інформації про страву	+	+
Додавання страви до кошика	+	+
Керування вмістом кошика	+	+
Застосування промокоду	+	+
Оформлення замовлення (оплата готівкою при отриманні)	НЗ	+
Оформлення замовлення (онлайн-оплата Stripe)	НЗ	+
Реєстрація нового облікового запису	+	НЗ
Авторизація в системі	+	НЗ
Вихід з облікового запису	НЗ	+
Перехід до особистого кабінету користувача	НЗ	+
Перегляд історії минулих та активних замовлень	НЗ	+
Детальний перегляд збереженого чека/замовлення	НЗ	+

13

Рисунок В.13 – Чек-лист тестування сайту

Висновки

- У роботі шляхом успішного виконання всіх поставлених завдань було досягнуто мету розробити програмне забезпечення вебсайту замовлення та доставки їжі для автоматизації обслуговування клієнтів та оптимізації роботи персоналу.
- Застосування розробленої системи базується не лише на наданні зручного цифрового каталогу страв для покупців, але й на автоматизації внутрішніх бізнес-процесів закладу. Для цього програма надає ряд спеціалізованих інструментів, що полегшують ці процеси: від динамічного керування сповіщеннями до адміністративного дашборда з генерацією PDF-звітності. Усе це, підкріплене асинхронною обробкою, що дозволяє використовувати програму як незалежну, стійку до навантажень платформу електронної комерції, що усуває необхідність залучення високовартісних агрегаторів.

14

Рисунок В.14 – Висновки