Прикарпатський національний університет імені Василя Стефаника

Кафедра інформаційних технологій

УДК 004.05

**КВАЛІФІКАЦІЙНА РОБОТА**

на здобуття першого (бакалаврського) рівня вищої освіти

Тема: Розробка POS системи для закладів громадського харчування з   
інтеграцією штучного інтелекту

Освітня програма Інженерія програмного забезпечення

назва освітньої програми

Спеціальність 121 Інженерія програмного забезпечення

код і назва спеціальності

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

КР.ІПЗ-4225.ПЗ

(позначення)

|  |  |
| --- | --- |
| Рецензент  проф. \_\_\_\_\_\_\_\_*.\_\_.2025* Микола КУЗЬ (посада) (підпис) (дата) (розшифровка підпису)  Нормоконтролер  проф.\_\_\_\_\_\_ \_\_*.\_\_.2025* Микола КУЗЬ  (посада) (підпис) (дата) (розшифровка підпису) | Здобувач вищої освіти  ІПЗ-42 \_\_\_\_\_ \_\_*.\_\_.2025* Вадим ЦВИК  (шифр групи)(підпис) (дата) (розшифровка підпису)  Керівник кваліфікаційної роботи  доц.\_\_\_\_\_ \_\_*.\_\_.2025* Валерій ТКАЧУК (посада) (підпис) (дата) (розшифровка підпису)  Допускається до захисту  Завідувач кафедри  \_\_\_\_\_\_\_ \_\_.*06.2025* Борис НЕЗАМАЙ (підпис) (дата) (розшифровка підпису) |

2025 (рік)

Прикарпатський національний університет імені Василя Стефаника

Факультет математики та інформатики Кафедра інформаційних технологій

Освітня програма Інженерія програмного забезпечення

Спеціальність 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

В.о. завідувача кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ігор ЛАЗАРОВИЧ

„23” жовтня 2024 р.

**ЗАВДАННЯ**

**НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

Цвику Вадиму Степановичу

(прізвище, ім’я, по батькові здобувача освіти)

1. Тема роботи: “ Розробка POS системи для закладів громадського харчування з інтеграцією штучного інтелекту”

затверджена розпорядженням по факультету математики та інформатики від „23 жовтня” 2024 р. №4

2. Термін здачі здобувачем вищої освіти закінченої роботи: 29 травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: Python, Django, PostgreSQL, HTML5, CSS3, Bootstrap 5, PyTest, Llama 3 (Ollama API), SQLite (тестова база), Twilio API (WhatsApp), Django ORM, Django Templates, PostgreSQL ERD (pgAdmin), pytest-django, Pillow, Chart.js.

4. Зміст пояснювальної записки (перелік питань, що їх належить опрацювати):

1. Аналіз предметної області \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Проєктування програми \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Розробка програмного забезпечення системи

4. Бізнес-план\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу

6. Дата видачі завдання: 23.10.2024 р.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_ Валерій ТКАЧУК

(підпис) (розшифровка підпису)

Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_\_\_ Вадим ЦВИК

(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів кваліфікаційної роботи | Терміни | Примітка |
| 1. | Затвердження теми і керівника кваліфікаційної роботи (далі “роботи”) на засіданні випускової кафедри. | 25.10.2024 - 10.11.2024 | Виконано |
| 3. | Обґрунтування актуальності, формулювання мети, завдання, предмету та об’єкту дослідження. Підготовка вступу. | 10.11.2024 - 20.11.2024 | Виконано |
| 4. | Узгодження орієнтовних тем, виду та термінів подачі до друку результатів дослідження. | 20.11.2024 - 15.12.2024 | Виконано |
| 5. | Опрацювання джерел з теми роботи. Підготовка І розділу роботи. | 15.12.2024 - 15.01.2025 | Виконано |
| 6. | Виправлення зауважень. Підготовка ІІ розділу роботи. | 15.01.2025 - 15.02.2025 | Виконано |
| 7. | Виправлення зауважень. Підготовка ІІІ розділу роботи. | 15.02.2025 - 15.03.2025 | Виконано |
| 8. | Виправлення зауважень. Підготовка IV розділу роботи. | 15.03.2025 - 15.04.2025 | Виконано |
| 9. | Виправлення зауважень. Підготовка висновків. | 15.04.2025 - 01.05.2025 | Виконано |
| 10. | Оформлення роботи згідно вимог. | 01.05.2025 - 15.05.2025 |  |
| 11. | Виправлення зауважень. Надсилання науковим керівником роботи нормоконтролеру. | 15.05.2025 - 22.05.2025 |  |
| 12. | Перевірка роботи нормоконтролером. | 22.05.2025 - 29.05.2025 |  |
| 13. | Виправлення зауважень нормоконтролера. Надсилання роботи випусковою кафедрою відповідальній особі від факультету для перевірки на академічний плагіат. | 29.05.2025 |  |
| 14. | Попередній захист роботи на випусковій кафедрі та допуск її до захисту в екзаменаційній комісії. | за 15 днів до захисту |  |
| 15. | Подача роботи із усіма необхідними матеріалами на випускову кафедру | за 10 днів до захисту |  |
| 16. | Рецензування. | за 5 днів до захисту |  |
| 17. | Захист роботи |  |  |

Здобувач вищої освіти \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Вадим ЦВИК

(підпис) (розшифровка підпису)

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Валерій ТКАЧУК

(підпис) (розшифровка підпису)

РЕФЕРАТ

Пояснювальна записка: 79 сторінок (без додатків), 15 рисунків, 1 таблиця, 27 джерел та 1 додаток.

Ключові слова: POS-СИСТЕМА, АВТОМАТИЗАЦІЯ, DJANGO, POSTGRESQL, РЕКОМЕНДАЦІЇ, MISTRAL 7B, OLLAMA API, HORECA.

Об’єкт дослідження: система автоматизації HoReCa.

Мета роботи: аналіз ринку сучасних POS-систем, проєктування та розробка POS системи Bublydr для автоматизації роботи закладів громадського харчування з інтеграцією рекомендаційної підсистеми на базі штучного інтелекту.

Стислий опис тексту пояснювальної записки:

Проведено порівняльний аналіз існуючих POS-систем і виявлено їх обмеження, що обґрунтовує необхідність створення нової системи. Розроблено архітектуру POS системи на основі фреймворку Django та СКБД PostgreSQL. Реалізовано модулі управління користувачами, замовленнями, фінансовими операціями, обліку запасів та інтегровано рекомендаційну підсистему на основі мовної моделі Mistral 7B через Ollama API.   
 Проведено тестування системи та розрахунок економічної ефективності впровадження. Розроблений бізнес-план підтверджує доцільність використання розроблюваної системи у сфері ресторанного бізнесу.

ABSTRACT

Explanatory note: 79 pages (without appendices), 15 figures, 1 table, 27 sources and 1 appendix.

Keywords: POS SYSTEM, AUTOMATION, DJANGO, POSTGRESQL, RECOMMENDATIONS, MISTRAL 7B, OLLAMA API, HORECA.

Object of research: HoReCa automation system.

Purpose: to analyze the market of modern POS systems, design and develop the Bublydr POS system for automating restaurant operations with AI-powered recommendation integration.

Brief description of the explanatory note:

A comparative analysis of existing POS systems was conducted, and their limitations identified, justifying the development of a new system. POS system architecture was designed using the Django framework and PostgreSQL database. Modules for user management, order processing, financial operations, inventory control, and an AI-powered recommendation subsystem based on the Mistral 7B language model via Ollama API were implemented.   
 The system was tested, and its economic efficiency was evaluated. The business plan confirms that the system is suitable for deployment in the restaurant business sector.

**ЗМІСТ**

[ВСТУП 10](#_Toc199288111)

[РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АНАЛОГІВ 11](#_Toc199288112)

[1.1 Аналіз предметної області цифровізації у сфері HoReCa 11](#_Toc199288113)

[1.2 Огляд і порівняльний аналіз існуючих POS-систем 12](#_Toc199288114)

[1.2.1 Основні функції сучасних POS-систем 12](#_Toc199288115)

[1.2.2 Аналіз поширених платформ 14](#_Toc199288116)

[1.2.3 Порівняння з Bublydr: інновації та переваги 17](#_Toc199288117)

[1.2.4 Обмеження існуючих рішень і ніша для Bublydr 19](#_Toc199288118)

[1.3 Висновки до першого розділу 20](#_Toc199288119)

[РОЗДІЛ 2. ПРОЄКТУВАННЯ POS-СИСТЕМИ BUBLYDR 21](#_Toc199288120)

[2.1 Опис методів дослідження 21](#_Toc199288121)

[2.2 Вибір технологій та інструментів 22](#_Toc199288122)

[2.2.1 Обґрунтування вибору фреймворку Django 22](#_Toc199288123)

[2.2.2 Стек технологій для серверної частини 24](#_Toc199288124)

[2.2.3 Стек технологій для клієнтської частини 25](#_Toc199288125)

[2.2.4 Забезпечення безпеки у системі 27](#_Toc199288126)

[2.3 Архітектура системи 29](#_Toc199288127)

[2.3.1 Загальна структура програмного комплексу 29](#_Toc199288128)

[2.3.2 Взаємодія компонентів системи 30](#_Toc199288129)

[2.3.3 Модульна організація системи 32](#_Toc199288130)

[2.3.4 Візуалізація архітектури в контексті C4-моделі 35](#_Toc199288131)

[2.3.4 Побудова ER-діаграми 36](#_Toc199288132)

[2.4 Вимоги до UI/UX дизайну 40](#_Toc199288133)

[2.5 Очікувані результати розробки системи Bublydr 41](#_Toc199288134)

[2.6 Висновки до другого розділу 42](#_Toc199288135)

[РОЗДІЛ 3. РОЗРОБКА POS-СИСТЕМИ BUBLYDR 44](#_Toc199288136)

[3.1 Процес створення таблиць у базі даних 44](#_Toc199288137)

[3.2 Реалізація графічного інтерфейсу системи 46](#_Toc199288138)

[3.2.1 Інтерфейс для офіціантів та адміністраторів 46](#_Toc199288139)

[3.2.2 Інтерфейс відвідувача закладу 48](#_Toc199288140)

[3.2.3 Практична реалізація дизайну інтерфейсу користувача 50](#_Toc199288141)

[3.3 Конфігурація проєкту та система міграцій бази даних 52](#_Toc199288142)

[3.3.1 Організація структури керування та конфігурації проєкту 52](#_Toc199288143)

[3.3.2 Система керування міграціями бази даних 54](#_Toc199288144)

[3.4 Реалізація функціональних модулів. 55](#_Toc199288145)

[3.4.1 Модуль управління меню 55](#_Toc199288146)

[3.4.2 Модуль бронювання столів 56](#_Toc199288147)

[3.4.3 Модуль оформлення замовлень 60](#_Toc199288148)

[3.4.4 Модуль управління запасами 63](#_Toc199288149)

[3.4.5 Модуль управління користувачами та ролями 68](#_Toc199288150)

[3.4.6 Реалізація push-сповіщень 71](#_Toc199288151)

[3.5 Інтеграція модуля штучного інтелекту 72](#_Toc199288152)

[3.5.1 Моделі та алгоритми побудови рекомендацій 72](#_Toc199288153)

[3.5.2 Архітектура та інтеграція рекомендаційої системи 73](#_Toc199288154)

[3.5.3 Реалізація функціоналу розумних рекомендації 75](#_Toc199288155)

[3.6 Забезпечення безпеки та контролю доступу 76](#_Toc199288156)

[3.6.1 Рівні доступу користувачів 76](#_Toc199288157)

[3.6.2 Захист персональних даних 78](#_Toc199288158)

[3.6.3 Безпека транзакцій 80](#_Toc199288159)

[3.7 Тестування системи 81](#_Toc199288160)

[3.8 Висновки до третього розділу 84](#_Toc199288161)

[РОЗДІЛ 4 БІЗНЕС-ПЛАН 85](#_Toc199288162)

[4.1 Планування витрат 85](#_Toc199288163)

[4.2 Аналіз ринку та очікуваний прибуток 86](#_Toc199288164)

[ВИСНОВКИ 88](#_Toc199288165)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 90](#_Toc199288166)

[ДОДАТОК А 95](#_Toc199288168)

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI — Artificial Intelligence

AJAX — Asynchronous JavaScript and XML

API — Application Programming Interface

CSRF — Cross-Site Request Forgery

GPT — Generative Pretrained Transformer

HoReCa — Hotel Restaurant Café/Catering

HTTP — HyperText Transfer Protocol

JSON — JavaScript Object Notation

ORM — Object-Relational Mapping

POS — Point Of Sale

SQL — Structured Query Language

СКБД — система керування базами даних

XSS — Cross-Site Scripting

ВСТУП

Стрімкий розвиток інформаційних технологій зумовлює необхідність автоматизації бізнес-процесів у сфері громадського харчування. Заклади працюють у висококонкурентному середовищі, що вимагає покращення якості обслуговування клієнтів, оптимізації роботи персоналу та скорочення витрат. Вирішити ці завдання допомагає впровадження сучасних технологій.

Цифровізація HoReCa передбачає автоматизацію управління замовленнями, обліку запасів, фінансових розрахунків і аналітики діяльності. Системи автоматизації закладів підвищують прозорість управління, мінімізують людський фактор і забезпечують швидке обслуговування клієнтів. За аналітичними даними, використання сучасних POS-систем дозволяє збільшити обсяги продажів на 10–15% завдяки кращому обліку товарів і оптимізації роботи персоналу [1].

Актуальність теми: Розроблена система спрямована на підвищення ефективності управління закладами громадського харчування шляхом автоматизації ключових бізнес-процесів. Це дозволяє оптимізувати роботу персоналу, покращити якість обслуговування клієнтів і зменшити операційні витрати [18].

Мета роботи: Розробка POS системи Bublydr для автоматизації роботи закладів громадського харчування з інтеграцією системи рекомендацій до замовлення на базі штучного інтелекту.

Об’єкт дослідження: POS системи для автоматизації закладів громадського харчування.

Практична цінність роботи: Розроблена система автоматизує ключові бізнес-процеси, включає підсистему на базі мовної моделі для рекомендацій релевантних позицій до замовлення, що сприяє підвищенню середнього чека і покращення обслуговування. Бізнес-план підтверджує економічну доцільність впровадження Bublydr у закладах громадського харчування.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АНАЛОГІВ

# 1.1 Аналіз предметної області цифровізації у сфері HoReCa

Сфера HoReCa (Hotel, Restaurant, Café/Catering) охоплює підприємства, діяльність яких пов’язана з наданням послуг у галузі гостинності та харчування. В умовах зростання конкуренції та високих очікувань споживачів підприємства цього сектору змушені шукати нові шляхи підвищення ефективності бізнес-процесів, оптимізації витрат і покращення якості обслуговування клієнтів [23].

Одним із ключових напрямів підвищення конкурентоспроможності є впровадження цифрових технологій. Автоматизація бізнес-процесів у сфері HoReCa дозволяє вирішувати низку критичних завдань: скорочення часу обробки замовлень, мінімізацію помилок персоналу, оптимізацію складського обліку, покращення фінансового контролю та підвищення лояльності клієнтів. За даними досліджень, використання сучасних інформаційних систем дозволяє зменшити час обслуговування клієнтів на 15–20%, а також збільшити середній чек на 10–15% завдяки ефективнішому управлінню пропозиціями товарів та послуг [24].

Важливим фактором є і впровадження аналітичних інструментів, що дозволяють отримувати точні дані про товарообіг, популярність окремих позицій меню та поведінку споживачів. Без належної цифровізації заклади громадського харчування стикаються з такими проблемами, як:

* неточний облік товарно-матеріальних запасів, що призводить до надлишкових витрат або дефіциту продукції;
* складність формування актуальних фінансових звітів;
* неможливість швидкого аналізу попиту та прогнозування потреб клієнтів;
* високий вплив людського фактору на якість обслуговування та управлінські рішення .

Значну увагу у сучасних умовах приділяють питанням безпеки даних та конфіденційності інформації про клієнтів. У зв’язку з посиленням законодавчих вимог до захисту персональних даних (зокрема, в рамках GDPR та національного законодавства України), інформаційні системи для сфери HoReCa повинні забезпечувати належний рівень захищеності даних.

Крім того, зростає попит на мобільні та хмарні рішення, які дозволяють керівникам закладів отримувати доступ до аналітики та управління бізнес-процесами в режимі реального часу, незалежно від місцезнаходження. За даними світових досліджень, понад 65% підприємств сфери HoReCa у 2024 році планували або вже впровадили хмарні POS-системи для оптимізації своєї діяльності [17]. З чого випливає, що цифровізація у сфері HoReCa є об’єктивною необхідністю для забезпечення стабільного розвитку підприємств, підвищення їх конкурентоспроможності та задоволення сучасних вимог ринку.

# 1.2 Огляд і порівняльний аналіз існуючих POS-систем

## 1.2.1 Основні функції сучасних POS-систем

З розвитком цифрових технологій автоматизація бізнес-процесів у сфері громадського харчування стала ключовим фактором ефективного управління [25]. Сучасні POS-системи перетворилися з простих інструментів для фіксації продажів на комплексні програмні рішення, які охоплюють критично важливі аспекти щоденної діяльності закладів. Вони забезпечують повноцінну підтримку процесів обробки замовлень і оплат, включаючи інтеграцію з платіжними терміналами, безконтактними способами оплати та мобільними додатками. Крім цього, системи дозволяють вести точний облік товарів, інгредієнтів та витратних матеріалів, що є надзвичайно важливим для забезпечення контролю за складом і попередження нестач.

POS-рішення дедалі частіше включають модулі для управління персоналом — з можливістю відстеження змін, обліку робочого часу, нарахування бонусів і контролю ефективності співробітників. Вбудовані аналітичні інструменти надають керівникам актуальні дані про обсяги продажів, середній чек, найпопулярніші позиції меню та пікові години навантаження. Це створює умови для ухвалення обґрунтованих управлінських рішень на основі даних, а не інтуїтивних припущень.

Суттєво зросло значення інтеграції з CRM-системами, які дозволяють формувати клієнтські бази, аналізувати поведінку відвідувачів, запроваджувати програми лояльності та здійснювати персоналізовані маркетингові кампанії. Такий рівень взаємодії з клієнтом є важливою конкурентною перевагою для сучасного закладу харчування, особливо в умовах високої ринкової конкуренції.

Окрему увагу слід приділити хмарним технологіям, які дедалі частіше стають основою для побудови сучасних POS-систем. Вони забезпечують доступ до управління в режимі реального часу з будь-якого пристрою, дозволяючи власникам і адміністраторам оперативно реагувати на зміни, аналізувати результати роботи чи вносити корективи незалежно від фізичної присутності в закладі. Це особливо важливо для мережевих ресторанів і мобільних точок продажу, які потребують централізованого контролю без втрати гнучкості.

Разом із цим, незважаючи на значне розширення функціоналу, більшість сучасних рішень усе ще залишаються прив’язаними до типової логіки бізнес-процесів. Вони часто не враховують унікальні особливості конкретного закладу, що ускладнює адаптацію під індивідуальні потреби [9]. Це обмеження створює попит на більш гнучкі та налаштовувані системи, які здатні підтримувати нетипові моделі обслуговування, меню чи внутрішні правила організації роботи персоналу.

## 1.2.2 Аналіз поширених платформ

Poster POS — це зручне хмарне рішення, орієнтоване насамперед на невеликі ресторани, кав’ярні та фастфуди. Його головні переваги полягають у простоті налаштування, мінімальних вимогах до апаратного забезпечення та можливості швидко розпочати роботу без залучення технічних спеціалістів. Система пропонує повноцінну роботу на планшетах або бюджетних ноутбуках, що особливо зручно для невеликих закладів.

Разом із тим, гнучкість платформи обмежена. Для складніших бізнес-сценаріїв — таких як масштабовані мережі, детальна аналітика, автоматизація кухні або глибока кастомізація бізнес-процесів — Poster пропонує відповідні можливості лише на дорогих тарифних планах. Розширена інтеграція з бухгалтерією, сторонніми CRM-системами чи системами лояльності потребує окремої оплати або API-доступу з обмеженнями.

Одним із ключових факторів при виборі POS-системи є інтерфейс користувача (UI), і Poster демонструє тут високий рівень продуманості. Інтерфейс вирізняється мінімалізмом та інтуїтивністю: основні дії — додавання замовлення, зміна статусу столика, проведення оплати — доступні буквально у два кліки. Інтерфейс адаптований під сенсорне управління, має контрастне візуальне оформлення та підтримує темну і світлу теми, що зручно при роботі в різних умовах освітлення.

Особливістю є панель швидкого доступу до популярних страв, кастомізовані кнопки на головному екрані POS-терміналу, а також гнучкий редактор меню — все це дозволяє офіціантам мінімізувати час взаємодії з системою.

Адміністративний інтерфейс, у свою чергу, реалізований через окрему веб-панель з адаптивним дизайном, що дає змогу керувати бізнесом дистанційно — з будь-якого пристрою.

На рис. 1.1 можна побачити інтерфейс системи Poster

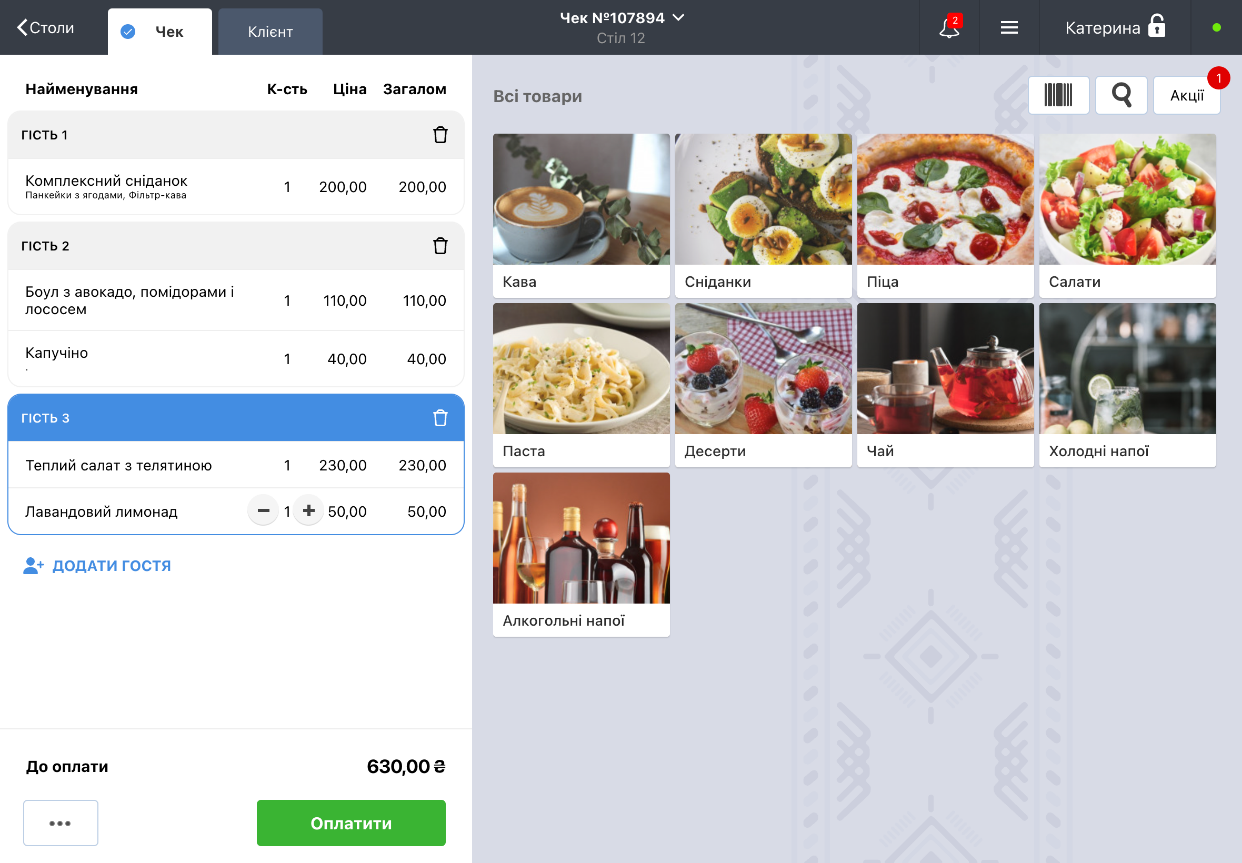


Рисунок 1.1 – Інтерфейс системи Poster

У підсумку, Poster є хорошим вибором для невеликих закладів з простими бізнес-процесами, але при масштабуванні або у складніших структурах управління його можливості можуть виявитися недостатніми без переходу на розширені платні пакети.

Syrve (раніше відома як iiko) позиціонується як повноцінна ERP-система для галузі HoReCa, розрахована на середній та великий бізнес. Платформа охоплює не лише класичні функції POS-системи, а й широкі можливості з управління виробництвом, фінансами, логістикою, персоналом та аналітикою. Це дозволяє підприємствам інтегрувати в єдине середовище процеси постачання, інвентаризації, розрахунку зарплат, контролю ефективності та навіть бухгалтерії.

З технічної точки зору Syrve має такі нелоліки. Система є складною у впровадженні та вимагає залучення сертифікованих інтеграторів, спеціального навчання персоналу та значних витрат часу. Це також підвищує загальну вартість володіння платформою, роблячи її малопридатною для малого чи локального бізнесу.

Інтерфейс користувача в Syrve створений для досвідчених операторів: він технократичний, перевантажений функціональністю та неінтуїтивний для новачків (див. рис. 1.2).

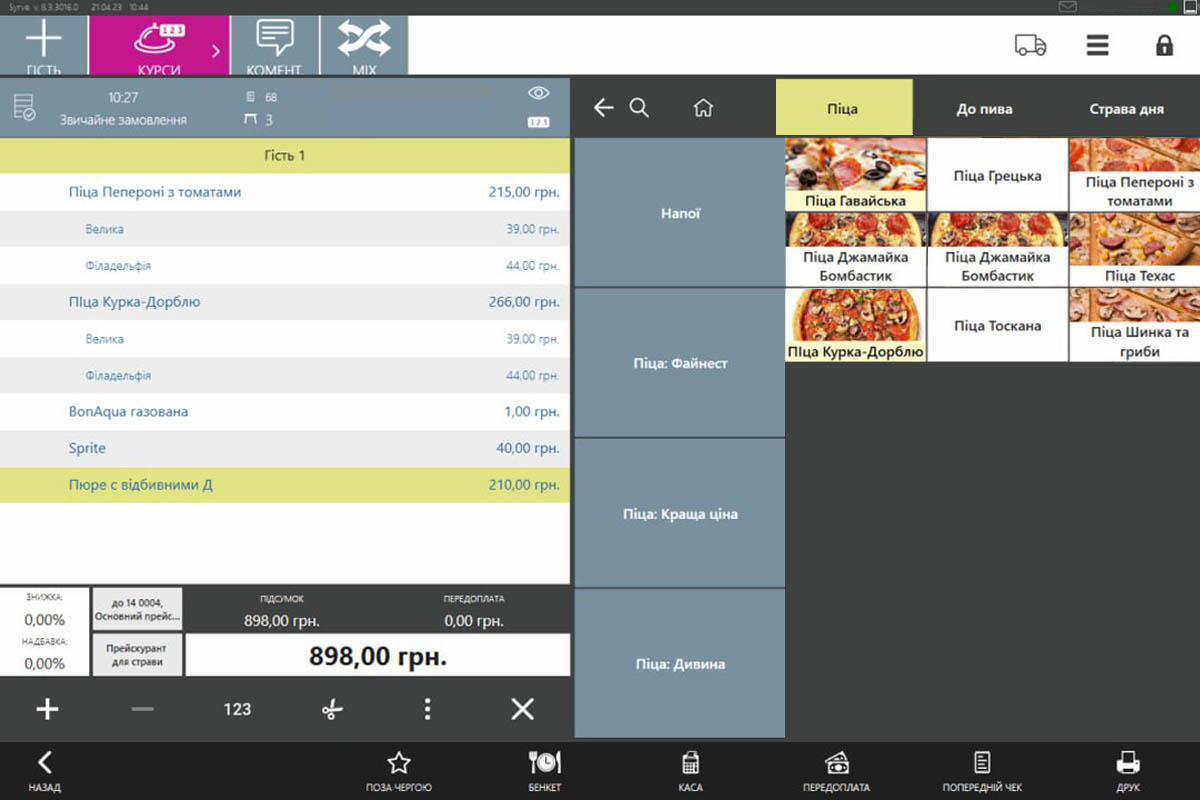


Рисунок 1.2 – Інтерфейс ситсеми Syrve

Для керівного складу передбачені дашборди, графіки та звіти з можливістю налаштування KPI, однак усі ці можливості не виправдовують критичних ризиків, пов’язаних з походженням і потенційною небезпекою використання цього програмного забезпечення в українських реаліях.

R-Keeper відома своєю комплексністю — це одне з найпоширеніших рішень для мережевих ресторанів. Система підтримує облік продажів, складський облік, інструменти для CRM, доставку та фінансовий моніторинг. Її мінусами є складне ліцензування, висока вартість впровадження та складність оновлення програмного забезпечення, що знижує її ефективність у динамічному середовищі малого підприємництва.

Square POS — сучасна хмарна система, особливо популярна серед малого бізнесу завдяки безкоштовній базовій версії та простому інтерфейсу [26]. Серед недоліків — залежність від власної платіжної інфраструктури, що обмежує гнучкість, і потреба у додаткових підписках для доступу до розширеного функціоналу [14].

## 1.2.3 Порівняння з Bublydr: інновації та переваги

З метою ефективної автоматизації бізнес-процесів у сфері громадського харчування була розроблена POS-система Bublydr. Її архітектура забезпечує продуктивність, стабільність, захист даних і масштабованість, що дозволяє використовувати її як у невеликих закладах, так і в мережевих структурах із високим навантаженням.

Система орієнтована на малий і середній бізнес, пропонуючи сучасне рішення з мінімальними витратами на впровадження й обслуговування. До ключових переваг належать простота розгортання, інтуїтивно зрозумілий адаптивний інтерфейс, що не потребує тривалого навчання, та модульна структура, яка дозволяє гнучко розширювати функціонал без значних витрат і перебудови системи.

На відміну від багатьох конкурентів, Bublydr не обмежує жорсткими умовами ліцензування й не прив’язує до закритої інфраструктури, забезпечуючи гнучкість налаштування та адаптацію під специфіку закладу. Система також підтримує інтеграцію з іншими обліковими й управлінськими сервісами, формуючи єдину інформаційну екосистему.

Додатковою перевагою є вбудована рекомендаційна підсистема, що пропонує релевантні позиції до замовлення, підвищує середній чек, оптимізує обслуговування та покращує клієнтський сервіс.

У таблиці нижче (див. табл. 1.1) представлено узагальнене порівняння аналізованих рішень, що дозволяє візуально оцінити їхні ключові характеристики, переваги та обмеження:

Таблиця 1.1 – Порівняння систем Poster, Syrve і Bublydr

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Poster | Syrve | Bublydr |
| Інтерфейс | Інтуїтивно зрозумілий, швидке навчання персоналу. | Професійний, але може вимагати більше часу на навчання персоналу. | Сучасний та адаптивний, розроблений для швидкого доступу до основних функцій. |
| Обладнання | Підтримка роботи на планшетах, що зменшує витрати на обладнання. | Підтримка різних платформ, включаючи стаціонарні термінали та мобільні пристрої. | Кросплатформенність, підтримка роботи на різних пристроях, включаючи пк,планшети та смартфони. |
| Функціонал | Обробка замовлень, складський облік, аналітика продажів, програми лояльності, інтеграція з платіжними терміналами. | Розширене управління замовленнями, складом, фінансами, персоналом; детальна аналітика та звітність. | Стандартні можливості POS-системи з простим та зрозумілим інтерфейсом. Зручним керуванням закладом, та рекомендаційним модулем. |

|  |  |  |  |
| --- | --- | --- | --- |
| Додаткові можливості | QR-меню, мобільний додаток для керування бізнесом, інтеграція з програмним РРО. | Інтеграція з програмами лояльності, онлайн-замовленнями, доставкою. | Інтеграція з CRM-системами, підтримка програм лояльності, модуль прогнозування продажів. |
| Недоліки та обмеження | - Обмежена функціональність для великих закладів. - Можливі труднощі з інтеграцією специфічного обладнання. | - Складність у налаштуванні та інтеграції без відповідної технічної підтримки. | - Новизна продукту може означати обмежену кількість впроваджень та відгуків. - Потреба в додаткових ресурсах для впровадження та навчання персоналу. |

Таким чином, система Bublydr вирізняється серед конкурентів своєю адаптивністю, доступністю для малого й середнього бізнесу, відкритістю до інтеграцій та наявністю інтелектуальних функцій, що робить її перспективним інструментом для автоматизації закладів громадського харчування.

## 1.2.4 Обмеження існуючих рішень і ніша для Bublydr

Хоча ринок POS-систем насичений, більшість рішень або надто прості й обмежені (Poster, Square POS), або складні й дорогі (Syrve, R-Keeper), тоді як Bublydr поєднує доступність, функціональність і можливість поступового розширення, що робить її актуальною альтернативою для українського малого й середнього підприємництва [27].

# 1.3 Проблематика та постановка задачі

З огляду на проведений аналіз предметної області, опис існуючих аналогів і виявлені недоліки сучасних POS-систем, актуальним є розроблення інформаційної системи для автоматизації бізнес-процесів у сфері громадського харчування. Така система має забезпечувати підвищення ефективності операційної діяльності, спрощення управлінських процесів і покращення якості обслуговування клієнтів, з можливістю подальшого розширення функціональності.

Метою проєкту є розробка POS системи Bublydr для автоматизації бізнес-процесів закладів громадського харчування з використанням сучасних інформаційних технологій та впровадженням рекомендаційної підсистеми для підтримки прийняття управлінських рішень.

Для досягнення поставленої мети необхідно:

* проаналізувати існуючі рішення на ринку POS-систем та визначити їх основні недоліки;
* розробити вимоги до системи Bublydr, враховуючи потреби малого та середнього бізнесу;
* спроєктувати модульну архітектуру системи з урахуванням можливостей масштабування і безпеки;
* реалізувати основні функціональні модулі: управління замовленнями, облік товарно-матеріальних запасів, аналітика фінансової діяльності та формування управлінських звітів;
* впровадити рекомендаційну підсистему для покращення взаємодії з клієнтами та підтримки персоналу під час обслуговування замовлень;
* протестувати систему в умовах реальної експлуатації та провести аналіз її ефективності;
* розробити бізнес-план впровадження системи та оцінити економічну доцільність її використання для закладів HoReCa.

# 1.4 Висновки до першого розділу

У першому розділі проаналізовано сучасний стан ринку POS-систем, виявлено основні тенденції розвитку та ключові проблеми впровадження ІТ у сфері громадського харчування. Аналіз рішень Poster, Syrve, R-Keeper і Square POS показав: одні не забезпечують персоналізації та гнучкості, інші вимагають значних ресурсів для впровадження й підтримки.

Poster і Square POS більше підходять малим закладам, але мають обмежений функціонал. Syrve та R-Keeper орієнтовані на великі підприємства, однак є дорогими й складними, що ускладнює їх застосування.

Визначено вимоги до архітектури й функціоналу системи Bublydr, що враховує потреби малого й середнього бізнесу. Обрано сучасний стек технологій: Python, Django, PostgreSQL, інтеграцію моделі Mistral 7B через Ollama API для рекомендацій.

Системний аналіз бізнес-процесів і сучасні методи проєктування дали змогу сформувати концепцію гнучкої, безпечної та масштабованої системи. Bublydr має забезпечити покращення сервісу, автоматизацію, зниження витрат і зростання прибутковості, що робить її актуальним рішенням для автоматизації закладів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ POS-СИСТЕМИ BUBLYDR

# 2.1 Опис методів дослідження

У процесі розробки POS системи Bublydr було застосовано комплекс методів, що охоплюють як аналіз предметної області, так і підходи до проєктування та впровадження сучасних інформаційних систем.

На етапі вивчення потреб закладів громадського харчування використовувався системний аналіз бізнес-процесів у сфері HoReCa. Це дозволило ідентифікувати ключові функціональні зони діяльності закладів, визначити типові сценарії обслуговування клієнтів, особливості управління замовленнями, складськими запасами та фінансовими потоками. Отримані результати лягли в основу формування вимог до архітектури програмного забезпечення та вибору необхідного функціоналу.

При проєктуванні архітектури системи застосовувалися методи побудови та оптимізації систем, що передбачають використання модульного підходу та принципів розділення відповідальностей.

Зокрема, архітектура була побудована на основі фреймворку Django з дотриманням патернів проєктування, що забезпечують простоту масштабування, підтримки та подальшого розширення системи.

Для структуризації даних та візуалізації логіки взаємодії між сутностями інформаційної системи використовувалися інструменти моделювання баз даних, зокрема ER-діаграми (Entity-Relationship) для визначення сутностей та їх зв’язків, а також UML-діаграми, які дозволили формалізувати основні бізнес-процеси та механізми взаємодії між компонентами системи. Це забезпечило чітке розуміння структури даних і логіки роботи системи як на етапі розробки, так і в подальшій експлуатації.

Особливу увагу в межах дослідження було приділено вивченню та впровадженню інструментів для інтеграції елементів інтелектуальної підтримки рішень у веб-застосунки. Зокрема, застосовувалися методи впровадження інтелектуальних систем на базі мовних моделей. Для цього використовувалася модель Mistral 7B, інтегрована через Ollama API, що дозволило створити рекомендаційну підсистему для формування персоналізованих підказок персоналу без необхідності розгортання ресурсомістких систем машинного навчання [13, 14, 35].

Застосування вищезазначених методів дослідження дало змогу побудувати високопродуктивну, гнучку та безпечну інформаційну систему, яка відповідає сучасним вимогам бізнесу у сфері громадського харчування.

# 2.2 Вибір технологій та інструментів

## 2.2.1 Обґрунтування вибору фреймворку Django

У процесі розробки POS-системи Bublydr було обґрунтовано доцільність використання фреймворку Django як основної технологічної платформи для реалізації серверної частини системи. Django є високорівневим веб-фреймворком мови програмування Python, який побудовано на основі архітектурного патерну Model-View-Template (MVT), що забезпечує чітке розділення рівнів логіки даних, обробки запитів та представлення інформації [4].

Вибір Django обумовлений низкою технічних характеристик, які відповідають сучасним вимогам до інформаційних систем у сфері ресторанного бізнесу. Одним із ключових факторів є вбудовані засоби автоматизації розробки, зокрема генерація структурованого каркасу проєкту за допомогою інструменту manage.py, що дозволяє виконувати типові операції, такі як запуск тестового сервера, застосування міграцій бази даних, керування користувачами та обробка статичних файлів. У проєкті Bublydr цей механізм активно використовується для управління понад 30 URL-маршрутами та інтеграції семи основних модулів, що забезпечують повний цикл роботи ресторанної системи [10].

Значною перевагою Django є наявність вбудованої системи автентифікації та авторизації, яка дозволяє ефективно розмежовувати права доступу до функціональних підсистем без необхідності розробки додаткових механізмів. У системі Bublydr реалізовано ієрархію ролей користувачів, що включає адміністратора, менеджера, офіціанта, кухаря, касира та клієнта. Розмежування доступу реалізовано шляхом використання декораторів @login\_required та власного декоратора @role\_required, що забезпечує контроль доступу до ресурсів системи відповідно до призначених ролей.

Завдяки застосуванню об’єктно-реляційного перетворення (ORM) фреймворку Django, у системі Bublydr реалізовано понад 40 моделей, що описують сутності предметної області, включаючи користувачів, замовлення, платежі, страви меню, склади продуктів та фінансові операції. Це дозволяє автоматизувати процес створення та оновлення структури бази даних, спрощує підтримку зв’язків між таблицями та значно зменшує обсяг SQL-запитів, які розробник має писати вручну.

Важливою особливістю є наявність вбудованих засобів захисту від поширених вразливостей, описаних у стандарті OWASP Top 10. Django забезпечує захист від SQL-ін’єкцій за допомогою параметризованих запитів ORM, автоматично інтегрує захист від XSS-атак через екранування вмісту у шаблонах, а також підтримує захист від CSRF-атак завдяки використанню відповідних токенів у формах. Незважаючи на те, що додаткові механізми безпеки, такі як SECURE\_HSTS\_SECONDS, CSRF\_COOKIE\_SECURE та SESSION\_COOKIE\_SECURE, не були активовані на етапі розробки системи, архітектура дозволяє легко інтегрувати ці параметри при розгортанні у промисловому середовищі [16].

Важливим фактором вибору Django стала його повна сумісність із середовищем PostgreSQL, що дозволяє ефективно працювати з великими обсягами даних і забезпечувати підтримку транзакцій, складних зв’язків між таблицями та оптимізацію запитів за допомогою індексів. Для зберігання фінансової інформації та даних про запаси в системі застосовано типи полів DecimalField, що гарантує точність обчислень при роботі з грошовими одиницями.

Окремо слід відзначити можливості Django щодо інтеграції з бібліотеками машинного навчання та інструментами обробки даних, що входять до складу екосистеми Python. У рамках реалізації інтелектуальної підсистеми рекомендацій у Bublydr використано взаємодію з мовною моделлю Mistral 7B, інтегрованою через API платформи Ollama. Завдяки цьому система формує рекомендації для клієнтів на основі історії замовлень, покращуючи якість обслуговування та збільшуючи середній чек.

## 2.2.2 Стек технологій для серверної частини

Архітектура серверної частини системи Bublydr побудована на базі перевірених рішень, які забезпечують надійність, розширюваність і відповідність сучасним вимогам розробки веб-систем. Усі основні компоненти конфігурації чітко задекларовані у файлі settings.py, що демонструє прозору структуру та грамотний підхід до організації програмного середовища.

Для зберігання даних у системі використовується реляційна СКБД PostgreSQL, яка забезпечує збереження структурованої інформації, зокрема даних про замовлення, користувачів, меню, оплати та складські залишки. Вибір PostgreSQL обумовлений її сумісністю з ORM-ядром Django, підтримкою транзакцій і механізмів забезпечення цілісності даних відповідно до принципів ACID. Параметри підключення до бази даних задаються в конфігураційному блоці DATABASES проєкту з використанням відповідних драйверів [5].

Управління конфіденційними параметрами, такими як секретні ключі, облікові дані до бази даних та змінні середовища, реалізовано за допомогою бібліотеки python-dotenv. Зчитування параметрів із файлу .env дозволяє розділити програмний код і налаштування середовища, що підвищує безпеку проєкту та спрощує процес розгортання системи в різних середовищах — розробки, тестування та продуктивного використання.

До складу додаткових бібліотек, інтегрованих у серверний стек, входять django-crispy-forms та crispy-bootstrap5, що відповідають за відображення форм у сучасному вигляді відповідно до стилістики Bootstrap. Це дозволяє швидко формувати елементи інтерфейсу для авторизації, реєстрації, редагування профілів тощо.

Кодова база розділена на низку функціональних додатків, кожен з яких виконує конкретне завдання, така модульність сприяє гнучкому масштабуванню системи без негативного впливу на стабільність загальної архітектури.

Окремо варто зазначити правильну локалізацію програмного середовища. У налаштуваннях системи задано українську мову інтерфейсу (LANGUAGE\_CODE = 'uk-ua') та київський часовий пояс (TIME\_ZONE = 'Europe/Kyiv), що забезпечує коректне відображення інформації у відповідності до національного контексту.

У результаті, обраний стек для серверної частини Bublydr демонструє вдале поєднання ефективності, безпеки, гнучкості та відповідності практикам сучасної веб-розробки. Система готова до масштабування, легко адаптується до змін бізнес-логіки та дозволяє впроваджувати нові функціональні модулі без ризику порушення роботи основного середовища.

## 2.2.3 Стек технологій для клієнтської частини

Клієнтська частина системи Bublydr реалізована з використанням класичних вебтехнологій, які забезпечують кросплатформеність, адаптивність інтерфейсу та зручність користування на всіх сучасних пристроях. Для розробки інтерфейсу було обрано технологічний стек, що включає HTML5, CSS3, JavaScript, а також шаблонізатор Django Templates, доповнений фреймворком стилів Bootstrap 5 [6].

Структура вебсторінок формується за допомогою мови розмітки HTML5, що визначає ієрархію контенту та структуризацію елементів інтерфейсу. Усі сторінки базуються на єдиному базовому шаблоні base.html, який використовується як каркас для наслідування у підшаблонах, що дозволяє уникнути дублювання коду та полегшує подальше масштабування інтерфейсу.

Візуальне оформлення реалізовано за допомогою CSS3 та фреймворку Bootstrap 5, що забезпечує уніфікований дизайн і кросбраузерну сумісність.

Основні компоненти інтерфейсу — кнопки, форми, спливаючі вікна, навігаційні панелі — створені на основі вбудованих класів Bootstrap, що спрощує адаптацію сторінок під різні розміри екранів за допомогою системи Flexbox та Grid. Для окремих сторінок застосовані кастомні стилі, що розміщені у структурі проєкту в каталозі static/css/, що дозволяє реалізувати індивідуальні дизайнерські рішення відповідно до корпоративного стилю ресторану.

Шаблонізатор Django Templates виступає проміжною ланкою між серверною та клієнтською частинами, дозволяючи безпосередньо вбудовувати динамічні дані в HTML-сторінки. Це дає змогу зручно відображати інформацію про користувачів, меню, замовлення та платежі без використання окремого API, що суттєво спрощує архітектуру системи.

Усі статичні ресурси системи організовані в структурі каталогу static/, де зберігаються файли стилів, скриптів, зображень і SVG-іконок. Ця структура дозволяє ефективно організувати кешування ресурсів та оптимізувати швидкість завантаження сторінок. Під час розгортання застосунку використовується механізм збирання статичних файлів за допомогою команди collectstatic, що централізує всі ресурси для зручної доставки через вебсервер. Клієнтський інтерфейс реалізовано з використанням шаблонів Django, HTML5, CSS3, Bootstrap 5. Всі сторінки мають адаптивну верстку та автоматично переходить у мобільний формат з бургер-меню.

Використання стандартних компонентів Bootstrap спрощує навчання персоналу — інтерфейс побудований за принципом «одна сторінка — одна дія», що дозволяє офіціантам виконувати замовлення без тривалого пошуку функцій.

## 2.2.4 Забезпечення безпеки у системі

Інформаційна безпека є важливим аспектом розробки системи Bublydr, оскільки її функціонування пов’язане з обробкою персональних даних клієнтів, фінансової інформації та внутрішніх даних підприємств сфери громадського харчування. Для забезпечення надійного захисту інформації було реалізовано комплекс заходів, спрямованих на мінімізацію ризиків витоку даних, несанкціонованого доступу та втрати інформації.

На рівні серверної частини системи безпека забезпечується шляхом використання параметризованих запитів у механізмі об’єктно-реляційного відображення (ORM) фреймворку Django. Це дозволяє ефективно запобігати SQL-інʼєкціям, унеможливлюючи впровадження шкідливих інструкцій через користувацькі запити. Для захисту від міжсайтових скриптових атак (XSS) застосовується механізм автоматичного екранування вмісту HTML-шаблонів, що унеможливлює виконання небезпечного коду на стороні клієнта [8]. Захист від міжсайтових запитів (CSRF) реалізовано за допомогою вбудованого механізму генерації унікальних токенів, які перевіряються при кожному запиті, що надходить до серверної частини системи. Додатково впроваджено захист від повторного використання токенів, що знижує ризики атак повторного надсилання (replay attacks).

Паролі користувачів зберігаються у захищеному вигляді шляхом застосування криптографічних алгоритмів хешування, таких як PBKDF2 або Argon2, що гарантує високий рівень стійкості до атак методом перебору. Додаткове використання сольових значень значно підвищує безпечність збереження облікових даних та унеможливлює застосування попередньо підготовлених словників для злому. У разі підозрілих дій з боку користувача система може ініціювати багатофакторну автентифікацію для підвищення захисту.

Безпека мережевої взаємодії між клієнтською та серверною частинами забезпечується використанням протоколу HTTPS із застосуванням сучасних стандартів шифрування SSL/TLS. Це гарантує конфіденційність та цілісність даних, що передаються в процесі обміну інформацією між клієнтами та сервером. Архітектура системи також передбачає можливість увімкнення додаткових параметрів безпеки, таких як SECURE\_HSTS\_SECONDS, CSRF\_COOKIE\_SECURE та SESSION\_COOKIE\_SECURE, що дозволяє значно підвищити рівень захисту при розгортанні системи в промисловому середовищі.

Контроль доступу до ресурсів інформаційної системи реалізовано за допомогою вбудованих механізмів автентифікації та авторизації, що дозволяють гнучко розмежовувати права доступу відповідно до призначених ролей користувачів. У системі передбачено ієрархію ролей, яка включає адміністратора, менеджера, офіціанта, кухаря, касира та клієнта. Розмежування доступу до функціональних можливостей забезпечується шляхом використання стандартних засобів фреймворку Django, зокрема декораторів @login\_required та спеціально розробленого декоратора @role\_required, що реалізує контроль доступу відповідно до ролей користувачів. У перспективі передбачено впровадження політик доступу на основі атрибутів (ABAC.

Завдяки впровадженню зазначених заходів, POS-система Bublydr забезпечує належний рівень захисту персональних і фінансових даних, що відповідає сучасним вимогам інформаційної безпеки та створює передумови для безпечної та стабільної роботи підприємств у сфері громадського харчування.

# 2.3 Архітектура системи

## 2.3.1 Загальна структура програмного комплексу

POS система Bublydr спроєктована відповідно до сучасних вимог до автоматизації бізнес-процесів у сфері громадського харчування. Архітектура програмного комплексу базується на фреймворку Django, який забезпечує чітке розділення логіки за принципом MVT (Model–View–Template), вбудовану підтримку безпеки, інтеграцію з ORM, систему керування міграціями та механізми модульного розширення.

Система реалізована як мультимодульний моноліт: функціональність розподілено між окремими Django-додатками (apps), кожен із яких виконує ізольовані задачі в межах загальної бізнес-логіки. До складу програмного комплексу входять такі модулі:

* accounts — забезпечує автентифікацію користувачів, розмежування доступу за ролями (офіціант, адміністратор, кухар, клієнт тощо), зберігання профільної інформації та базові налаштування облікових записів;
* orders — відповідає за повний життєвий цикл замовлення: створення, обробка, фіналізація, зміна статусів та взаємодія з іншими модулями (наприклад, списання запасів);
* menu — реалізує управління асортиментом страв, категоріями, цінами та параметрами доступності;
* inventory — здійснює облік матеріальних ресурсів, контроль мінімальних залишків, формування даних для звітів та автоматичне списання інгредієнтів;
* payments — обробляє фінансові транзакції та підтримує різні методи оплати;
* tables — надає інструменти для адміністрування залу: бронювання, візуалізація статусів столиків, контроль завантаженості;
* dashboard — формує аналітичну інформацію та ключові показники;
* recommendations — інтегрує систему штучного інтелекту на базі моделі Mistral 7B, яка генерує контекстно релевантні поради для клієнтів і персоналу.

Усі модулі працюють у межах одного серверного процесу, спільної бази даних PostgreSQL та єдиного середовища виконання, що підтверджує монолітний характер системи. Водночас логічне розділення за модулями забезпечує гнучкість, спрощує розробку, тестування й подальше масштабування.

Візуальна частина реалізована за допомогою Django Templates у поєднанні з Bootstrap 5. Завдяки адаптивній верстці інтерфейс коректно відображається як на POS-терміналах, так і на мобільних пристроях.

Динамічне формування контенту відбувається через передавання змінних у шаблони, що дозволяє ефективно розділяти бізнес-логіку й подання даних.

Кожен модуль містить власний набір міграцій, що дає змогу незалежно оновлювати структуру бази даних без конфліктів між компонентами. Це дозволяє підтримувати узгодженість даних, зменшує ризики під час розгортання оновлень і забезпечує надійність функціонування системи.

Таким чином, застосунок поєднує переваги централізованого розгортання з модульною архітектурою, що забезпечує високу підтримуваність і гнучкість у розвитку програмного продукту.

## 2.3.2 Взаємодія компонентів системи

Програмний комплекс Bublydr побудований за модульною архітектурою із чітким розмежуванням функціональних блоків, що забезпечує високий рівень масштабованості, гнучкості та надійності системи.

Архітектурною основою комплексу є високорівневий веб-фреймворк Django, який забезпечує централізоване керування запитами, обробку транзакцій та цілісність даних.

Механізм маршрутизації запитів реалізовано через головний файл urls.py, який визначає основні напрямки обробки запитів та делегує їх відповідним підсистемам:

- accounts — управління користувачами;

- menu — управління меню та категоріями страв;

- tables — управління статусами столиків;

- orders — обробка замовлень;

- payments — фінансові операції;

- inventory — облік запасів;

- dashboard — формування аналітичних звітів.

Організацію контролю доступу зебезпечує те, що доступ до системних ресурсів реалізовано через централізований механізм автентифікації та авторизації, побудований на основі вбудованої моделі User, розширеної додатковими атрибутами, такими як тип користувача, телефон, адреса та фото профілю. Ролі користувачів визначаються через поле user\_type і охоплюють наступні категорії: адміністратор, менеджер, офіціант, кухар, касир та клієнт.

Для контролю доступу використовується стандартний декоратор @login\_required, який обмежує доступ до більшості ресурсів системи лише для автентифікованих користувачів. Винятком є перегляд меню та списку вільних столиків, що доступні без входу в систему.

Додатково, застосовується кастомний декоратор @role\_required, який перевіряє належність користувача до певної ролі, необхідної для виконання конкретної дії. До прикладу, доступ до функціоналу управління замовленнями, платежами та користувачами дозволено лише персоналу ресторану, відповідно до призначених ролей.

Взаємозв’язок між модулями системи забезпечується за допомогою моделі запитів та відповідей у форматі MVC (Model-View-Controller), яка є основною архітектурною парадигмою у фреймворку Django.

Модуль обробки замовлень тісно інтегрований із підсистемами управління столиками та обліку запасів. При створенні замовлення відповідний стіл автоматично отримує статус «зайнятий», а після завершення обслуговування — статус повертається на «вільний». Це дозволяє оперативно контролювати завантаженість закладу та оптимізувати розподіл клієнтів.

Модуль обліку запасів взаємодіє з підсистемою замовлень, автоматично списуючи інгредієнти зі складу відповідно до рецептури приготованих страв. Такий підхід забезпечує точний облік залишків у режимі реального часу та мінімізує ризики виникнення дефіциту товарів.

Фінансові операції здійснюються через модуль payments, який підтримує різні способи оплати, включаючи готівку, банківські картки та онлайн-платежі. Після завершення транзакції змінюється статус замовлення та формується відповідний фінансовий документ (чек), який може бути роздрукований або збережений у цифровому форматі.

Для підвищення прозорості операцій та контролю за діяльністю системи використовується модуль dashboard, який формує звіти щодо фінансових операцій, обсягів продажу, статистики завантаженості столиків та стану складських запасів.

## 2.3.3 Модульна організація системи

Архітектура програмного комплексу Bublydr побудована відповідно до принципів модульності, що дозволяє досягти високого рівня структурованості, забезпечити гнучкість у процесі розробки та подальшої підтримки системи. Модульна організація передбачає поділ системи на функціонально незалежні компоненти, кожен з яких виконує визначені завдання та взаємодіє з іншими модулями через стандартизовані інтерфейси. Це забезпечує зменшення зв’язності між модулями та підвищує рівень повторного використання програмних компонентів, що є критично важливим для створення сучасних корпоративних систем.

Центральним елементом модульної архітектури є модуль управління користувачами, який реалізує механізми автентифікації, авторизації та контролю доступу. Цей модуль базується на розширеній моделі користувача, що включає не лише стандартні атрибути, а й додаткові поля, необхідні для класифікації користувачів за ролями та збереження контактної інформації. Завдяки чіткому визначенню типів користувачів забезпечується розмежування доступу до функціональних можливостей системи, що підвищує її безпеку та надійність.

Функціонування системи також передбачає використання модуля управління меню, який відповідає за зберігання та обробку інформації про страви, їх категорії, ціни та склад. Цей модуль тісно інтегрований з підсистемою обліку запасів, що дозволяє автоматично визначати можливість приготування певної страви залежно від наявності інгредієнтів на складі. Такий підхід забезпечує актуальність інформації в меню та дозволяє оперативно реагувати на зміни в асортименті продукції.

Окреме місце в структурі системи займає модуль управління замовленнями, який реалізує повний життєвий цикл замовлення — від моменту створення до завершення та оплати. Цей модуль взаємодіє з модулями управління столиками, обліку запасів і фінансових операцій. Зміна статусу замовлення автоматично супроводжується оновленням інформації про завантаженість залу, актуалізацією запасів на складі та обробкою відповідних платіжних операцій. Це дозволяє значною мірою автоматизувати роботу персоналу та підвищити ефективність обслуговування клієнтів.

Модуль управління фінансовими операціями виконує завдання з обробки платежів та ведення обліку фінансових транзакцій. Завдяки підтримці різноманітних способів оплати, включаючи готівкові розрахунки, оплату банківськими картками та онлайн-платежі, забезпечується висока гнучкість фінансових операцій. Модуль реалізує механізми обліку платежів за замовленнями, формування фіскальних чеків і генерації фінансової звітності для керівництва закладу.

Модуль управління запасами відповідає за облік товарних залишків, контроль надходжень і витрат інгредієнтів, а також зберігання даних про постачальників. Інтеграція з модулем замовлень дозволяє автоматично списувати інгредієнти після оформлення замовлення відповідно до рецептурної карти страв.

Це забезпечує підтримку актуального стан складу та дозволяє оптимізувати процеси закупівель шляхом своєчасного формування замовлень постачальникам.

Модуль аналітики та звітності реалізує формування стандартних звітів, таких як: фінансові підсумки за день і місяць, аналіз продажів за категоріями страв, облік залишків на складі та звіти про продуктивність зміни. Дані виводяться у вигляді графіків і таблиць за допомогою бібліотеки Plotly.js (або інша, вкажіть яка), що спрощує аналіз і прийняття управлінських рішень.

Модуль рекомендаційної системи інтегрує мовну модель Mistral 7B через платформу Ollama для генерації релевантних порад щодо додаткових страв або напоїв. Рекомендації формуються на основі аналізу поточного замовлення та структури меню й доступні як для персоналу (підвищення середнього чека), так і для клієнтів. Взаємодія між модулями здійснюється через REST API та обробляється у вигляді окремих сервісів, що спрощує подальше масштабування системи.

Модульна архітектура дає змогу застосунку розширювати функціональність системи шляхом додавання нових компонентів без зміни існуючих.

## 2.3.4 Візуалізація архітектури в контексті C4-моделі

У межах архітектурного аналізу проєкту Bublydr було розроблено C4-діаграми першого та другого рівня відповідно до методології візуального моделювання систем. C4-модель дозволяє представити архітектуру з різним ступенем деталізації — від загального контексту до внутрішньої організації контейнерів системи, що дає змогу ефективно узгодити бачення архітектури між технічними й нетехнічними учасниками команди. Такий підхід покращує комунікацію, пришвидшує розробку та полегшує подальшу підтримку [22].

На рівні контекстної діаграми (Level 1) система Bublydr розглядається як єдиний програмний комплекс, який взаємодіє з кількома категоріями користувачів: клієнтами, офіціантами, адміністраторами, а також з зовнішніми сервісами — зокрема, базою даних PostgreSQL для зберігання інформації та AI-платформою Ollama, що забезпечує рекомендаційну функціональність на основі моделі Mistral 7B (див. рис. 2.1).

Зображення, що містить текст, схема, ряд, Графік

Вміст, створений ШІ, може бути неправильним.

Рисунок 2.1 – Контекстна діаграма системи

На другому рівні деталізації (Level 2) зображено внутрішню архітектуру системи, реалізовану у вигляді мультимодульного моноліту. Кожен з модулів — accounts, orders, menu, tables, payments, inventory, dashboard, recommendations — виконує окрему функціональну роль. Усі модулі працюють в одному середовищі та мають доступ до спільної бази даних (див. рис. 2.2).

Запити до зовнішнього AI-сервісу здійснюються з модуля recommendations, що інтегрується з платформою Ollama через API. Такий підхід дозволяє досягти високої структурованості системи, централізованого керування та легкості розширення функціоналу.

Зображення, що містить текст, схема, План, Паралель

Вміст, створений ШІ, може бути неправильним.

Рисунок 2.2 – Діаграма контейнерів програмної системи Bublydr

Представлені C4-діаграми дають змогу чітко уявити архітектуру системи Bublydr як на макрорівні, так і на рівні компонентів, що підвищує зрозумілість проєкту для розробників, аналітиків і сторонніх учасників.

## 2.3.5 Побудова ER-діаграми

ER-діаграма (Entity-Relationship Diagram) є візуальним представленням логічної структури бази даних, що дозволяє наочно відобразити основні сутності системи, атрибути та взаємозв’язки між ними. У контексті POS системи Bublydr ця діаграма слугує основою для формування фізичної моделі бази даних і дозволяє побачити взаємозалежність ключових об’єктів ресторанного середовища.

Серед основних сутностей, представлених на діаграмі, можна виділити користувачів, замовлення, страви, категорії меню, столики, інгредієнти, платежі та ролі. Кожна сутність має чітко визначений набір атрибутів, які відповідають характеристикам об’єкта реального світу. Наприклад, сутність User містить поля, пов’язані з особистими даними користувача, такими як ім’я, прізвище, email, тип користувача та шлях до зображення профілю. Також для забезпечення безпеки та контролю доступу реалізовано зв’язок з ролями, що зумовлює можливості кожного користувача в системі.

Сутність Order займає центральне місце в структурі, оскільки безпосередньо пов’язана із більшістю інших об’єктів: користувачем-офіціантом, клієнтом, столиком, списком страв, статусами та оплатами. Зв’язок між замовленням та користувачем реалізований через зовнішній ключ waiter\_id, який допускає NULL у випадку видалення офіціанта, що забезпечує збереження історії замовлень.

Окрему роль відіграє сутність Dish, яка відповідає за опис кожної окремої страви в меню. Кожна страва зв’язана з відповідною категорією (Category) через зовнішній ключ, що дозволяє логічно групувати асортимент. Додатково в структурі передбачено атрибути для ціни, опису, зображення, статусу наявності, часу приготування тощо. Страви включаються до замовлень через проміжну таблицю OrderItem, що реалізує зв’язок типу «багато-до-багатьох» між Order та Dish. У цій таблиці також фіксується кількість одиниць кожної страви в межах конкретного замовлення, що дозволяє точно обраховувати суму та формувати рахунок.

Взаємодія між замовленнями, стравами та користувачами уможливлює відстеження повного життєвого циклу обслуговування клієнта — від моменту створення замовлення до завершення платежу. Завдяки фіксації кожного етапу, система може надавати аналітичну інформацію, наприклад, про середній час обробки замовлень, популярність страв або продуктивність працівників.

Сутність Table відповідає за фізичне розміщення клієнтів у закладі й має зв’язок із замовленням. Через цей зв’язок можна оцінити завантаженість залу, організувати бронювання та візуально відображати статуси столиків у реальному часі. Поля number, capacity, location і status описують конфігурацію посадкових місць і забезпечують гнучкість при масштабуванні інтерфейсу залу.

У структурі також присутній модуль керування запасами, де сутності Product, Recipe, RecipeIngredient, Purchase, PurchaseItem і Consumption реалізують повний цикл обліку інгредієнтів. Страви мають прив’язку до рецептур (Recipe), де через RecipeIngredient задається точний склад продуктів для списання. Кожне списання (Consumption) може бути прив’язане до певного замовлення або операції внутрішнього обліку, що забезпечує точність контролю залишків і аналітичну прозорість складу. Крім того, таблиця Purchase з підлеглими PurchaseItem дозволяє фіксувати всі закупівлі, джерела постачання та планові дати доставки.

У фінансовому модулі реалізовано зв’язок один-до-одного між Order та Payment. Кожен платіж зберігає інформацію про суму, спосіб оплати, статус, дату завершення операції та, за потреби, фіскальний номер. Додатково існує сутність Receipt, яка забезпечує ведення обліку друкованих чеків і може бути інтегрована з фіскальними реєстраторами.

Для системи доступу реалізовано логіку взаємодії користувачів, груп (auth\_group), прав (auth\_permission) та сесій. Через проміжні таблиці accounts\_user\_groups і accounts\_user\_user\_permissions система підтримує розмежування доступу не лише за типом користувача (офіціант, адміністратор, клієнт), а й за індивідуальними повноваженнями. Це дозволяє адаптувати систему до складних внутрішніх політик закладів.

Типи зв’язків у діаграмі охоплюють як класичні один-до-багатьох, так і більш складні сценарії: один-до-одного (Order–Payment), багато-до-багатьох через проміжні таблиці (Order–Dish), каскадні оновлення та обмеження видалення через PROTECT, SET NULL або CASCADE.

Така структура дозволяє реалізувати гнучку й стабільну логіку поведінки даних у реальному бізнес-середовищі (на рисунку 2.3 і 2.4 зображено ERD).

Зображення, що містить текст, знімок екрана, число, Паралель

Вміст, створений ШІ, може бути неправильним.

Рисунок 2.3 – ER-діаграма системи Bublydr

Зображення, що містить текст, знімок екрана, число, Паралель

Вміст, створений ШІ, може бути неправильним.

Рисунок 2.4 – Продовження ER-діаграми

ER-діаграма охоплює ключову логіку ресторанної системи — від обслуговування клієнтів до контролю ресурсів і фінансів. Вона формує основу для фізичної структури бази даних, забезпечує масштабованість і підтримує розвиток системи.

# 2.4 Вимоги до UI/UX дизайну

Інтерфейс користувача системи Bublydr розроблятиметься відповідно до сучасних принципів UI/UX дизайну, з урахуванням потреб закладів громадського харчування та різнорівневої підготовки персоналу. Основною вимогою є створення зрозумілого, адаптивного і візуально цілісного середовища, що забезпечує зручність щоденного використання навіть у стресових умовах високої завантаженості.

Особлива увага приділятиметься мінімізації кількості кліків до ключових функцій. Основні операції — створення замовлення, перегляд меню, зміна статусу столика — повинні бути доступні не більше ніж у два кліки. Це скорочує час обслуговування клієнтів і підвищує загальну ефективність роботи персоналу.

Інтерфейс має бути повністю українськомовним. Всі елементи управління, повідомлення та текстові підказки оформлюються чіткими, однозначними формулюваннями. Важливою вимогою є забезпечення достатнього розміру шрифтів і контрастності тексту, зокрема для використання системи в умовах недостатнього освітлення або на мобільних пристроях із невеликими екранами.

Колірна палітра повинна бути не лише естетичною, але й функціональною. Наприклад, індикатори стану столиків мають бути стандартизовані: зелений означає «вільно», червоний — «зайнято», жовтий — «заброньовано». Це дозволяє швидко оцінювати ситуацію в залі без потреби у додаткових діях.

Дизайн інтерфейсу має бути мінімалістичним. Повинні відображатись лише ті елементи, що необхідні на поточному етапі взаємодії. Технічні параметри й додаткові налаштування приховані від звичайного користувача, щоб уникнути перевантаження інтерфейсу і зменшити ризик помилкових дій.

Підтвердження важливих і незворотних дій, таких як видалення або скасування замовлення, має бути обов’язковим. Усі дії повинні супроводжуватись зворотним зв’язком — повідомленням про успішне виконання чи помилку з відповідним поясненням.

Обов’язковим є впровадження механізмів автентифікації та авторизації. Доступ до адміністративних функцій повинен бути обмежений лише тими користувачами, які мають відповідні ролі. Це забезпечує не лише безпеку, але й розмежування відповідальності в межах системи.

# 2.5 Очікувані результати розробки системи Bublydr

Розробка POS системи Bublydr спрямована на досягнення комплексного покращення показників діяльності закладів громадського харчування за рахунок підвищення ефективності бізнес-процесів, покращення якості обслуговування клієнтів та оптимізації витрат.

Одним із ключових очікуваних результатів є підвищення якості обслуговування клієнтів. Це досягається за рахунок впровадження сучасних інструментів управління замовленнями, автоматизації обробки транзакцій та використання рекомендаційної підсистеми, яка допомагає у формуванні більш релевантні пропозиції на основі замовлених страв та поточної наявності. Такий підхід дозволяє покращити клієнтський досвід і сприяє підвищенню лояльності постійних клієнтів.

Завдяки модульній архітектурі та впровадженню механізмів інтеграції з зовнішніми сервісами очікується підвищення рівня автоматизації операційних процесів. Це стосується не лише процесів обробки замовлень, але й управління товарно-матеріальними запасами, фінансового обліку та формування аналітичних звітів. Автоматизація дозволяє зменшити кількість помилок, пов’язаних із людським фактором, і забезпечити більш прозоре управління ресурсами закладу.

Ще одним важливим результатом є зниження витрат на персонал завдяки інтеграції елементів штучного інтелекту. Рекомендаційна система, що використовує мовну модель Mistral 7B, допомагає персоналу оперативно приймати рішення щодо формування додаткових пропозицій і оптимального обслуговування клієнтів, що зменшує потребу в залученні висококваліфікованого та численного персоналу. Це дає змогу оптимізувати кадрову політику та спрямувати зекономлені ресурси на розвиток основної діяльності.

Завдяки інтеграції рекомендаційної підсистеми очікується також підвищення прибутковості закладів. Формування персоналізованих рекомендацій дозволяє збільшити середній чек замовлень і стимулювати повторні покупки, що позитивно впливає на фінансові результати підприємства. Впровадження таких механізмів створює конкурентні переваги на ринку та сприяє довгостроковому розвитку бізнесу.

# 2.6 Висновки до другого розділу

У другому розділі дипломної роботи обґрунтовано архітектурні, функціональні та алгоритмічні засади розробки POS-системи Bublydr. При проєктуванні системи застосовано принципи модульної побудови, масштабованості та використання сучасних технологій обробки й представлення даних.

Розроблено структуру бази даних із використанням системи керування базами даних PostgreSQL, що забезпечує повну підтримку нормалізованих структур і коректне встановлення зв’язків між основними сутностями системи, зокрема: користувачами, замовленнями, позиціями меню, платежами та іншими об’єктами бізнес-процесів. Визначено логічні й фізичні моделі даних, що забезпечують ефективне зберігання та обробку інформації.

Особливу увагу в проєкті приділено розробці рекомендаційної підсистеми, яка інтегрує сучасні мовні моделі на основі технології Ollama. Для формування персоналізованих рекомендацій використовується модель Mistral 7B, що дозволяє генерувати контекстно релевантні поради на основі поточного складу замовлення. Хоча система не передбачає складної аналітики на основі історичних даних чи навчання моделей на специфічних наборах транзакцій, її архітектура дозволяє в подальшому розширювати функціональні можливості рекомендаційного модуля.

Інтеграція базових елементів штучного інтелекту сприяє підвищенню рівня клієнтського сервісу та створює передумови для подальшого вдосконалення бізнес-процесів у закладах громадського харчування.

РОЗДІЛ 3. РОЗРОБКА POS-СИСТЕМИ BUBLYDR

# 3.1 Процес створення таблиць у базі даних

Розробка таблиць у базі даних системи Bublydr здійснювалася з урахуванням вимог до продуктивності, масштабованості та підтримки цілісності даних у транзакційному середовищі. В якості системи керування базами даних обрано PostgreSQL, яка забезпечує гнучке керування типами даних, зовнішніми ключами, індексацією та транзакційною обробкою відповідно до стандарту ACID.

Створення таблиць бази даних розпочинається з аналізу сутностей предметної області, які відображають основні бізнес-об'єкти системи: користувачі, замовлення, страви, столики, платежі, інгредієнти. На основі цього формується логічна модель, яка потім трансформується у фізичну структуру шляхом генерації SQL-таблиць.

У процесі створення таблиць кожна сутність отримує унікальний первинний ключ (PRIMARY KEY), що забезпечує ідентифікацію записів. Зв’язки між таблицями реалізуються за допомогою зовнішніх ключів (FOREIGN KEY) із вказаними політиками поведінки при видаленні (ON DELETE). Наприклад, при створенні таблиці замовлень зв’язок із таблицею офіціантів (accounts\_user) реалізується з опцією ON DELETE SET NULL, що дозволяє зберегти замовлення навіть у разі видалення працівника:

waiter = models.ForeignKey(

User, on\_delete=models.SET\_NULL, null=True,

related\_name='waiter\_orders', verbose\_name=\_('Офіціант')

)

Натомість у випадках, де критично важливо зберегти цілісність історії (наприклад, щодо страв), використовується обмеження PROTECT, яке забороняє видалення записів, що мають посилання в інших таблицях:

python

Копіювати код

dish = models.ForeignKey(

Dish, on\_delete=models.PROTECT,

related\_name='+', verbose\_name=\_('Страва')

)

Вибір типів полів здійснюється відповідно до їх призначення: INTEGER — для числових ідентифікаторів, VARCHAR і TEXT — для текстової інформації, DATE і TIMESTAMP — для дат і часових міток. Для грошових полів, таких як ціна чи сума, використовується DECIMAL(10, 2), що дозволяє точно оперувати значеннями з фіксованою точністю, без втрати десяткової частини.

Процес створення таблиць також включає визначення значень за замовчуванням (DEFAULT), що дозволяє зменшити кількість помилок під час введення нових записів. Наприклад, поля status та payment\_status у таблиці замовлень отримують початкові значення 'new' та 'pending', відповідно. Окрім того, до структури включено поля created\_at, updated\_at, які дозволяють відслідковувати хронологію змін:

CREATE TABLE orders\_order (

id SERIAL PRIMARY KEY,

table\_id INTEGER REFERENCES tables\_table(id) ON DELETE SET NULL,

waiter\_id INTEGER REFERENCES accounts\_user(id) ON DELETE SET NULL,

client\_id INTEGER REFERENCES accounts\_user(id) ON DELETE CASCADE,

status VARCHAR(20) NOT NULL DEFAULT 'new',

payment\_status VARCHAR(20) NOT NULL DEFAULT 'pending',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

completed\_at TIMESTAMP,

notes TEXT,

customer\_name VARCHAR(100),

discount\_percent DECIMAL(5, 2) DEFAULT 0

);

Для зберігання мультимедійних ресурсів (зображення профілів, фотографії страв) у таблицях зберігаються лише шляхи до відповідних файлів, тоді як самі файли зберігаються в директорії media/ на файловій системі. Такий підхід дає змогу зменшити навантаження на базу даних і оптимізувати резервне копіювання.

Відображення зображень відбувається через URL-шляхи, які автоматично кешуються вебсервером для покращення продуктивності інтерфейсу.

Під час створення таблиць було дотримано принципів нормалізації до третьої нормальної форми (3НФ), що дозволило уникнути дублювання даних та забезпечити логічну узгодженість у структурі. Також для ключових полів, що використовуються у фільтрації та сортуванні (наприклад, status, user\_id, created\_at), додатково створено індекси для пришвидшення виконання запитів.

У підсумку, процес створення таблиць у базі даних Bublydr є ретельно спроєктованим етапом, що охоплює проєктування структури, визначення типів даних, встановлення зв’язків і обмежень, а також враховує специфіку обробки мультимедійної інформації та аудиту змін у системі.

# 3.2 Реалізація графічного інтерфейсу системи

## 3.2.1 Інтерфейс для офіціантів та адміністраторів

Інтерфейс користувача для офіціантів та адміністраторів у системі Bublydr розроблений таким чином, щоб забезпечити максимальну простоту та інтуїтивність роботи навіть для користувачів без спеціальних технічних знань. Основна мета цього інтерфейсу — надати швидкий доступ до ключових функцій ресторану: управління замовленнями, столиками, переглядом і редагуванням меню, а також аналітики діяльності закладу.

Після входу в систему користувач потрапляє на головну сторінку (панель керування), де представлено зведену інформацію: кількість замовлень за обрані періоди, сума доходів, статус столиків, найпопулярніші страви, активні замовлення, найближчі бронювання та стан залишків на складі. Для цього використовується інтерактивна панель із віджетами та аналітичними діаграмами, реалізована у модулі Dashboard (dashboard/views.py, функція dashboard\_home). Інтерфейс адаптується до типу пристрою, забезпечуючи зручність роботи як з десктопів, так і з планшетів або мобільних терміналів.

На рисунку 3.1 зображено панель керування, за допомогою якої працівники закладу можуть відстежувати поточний стан закладу.

Зображення, що містить текст, знімок екрана, програмне забезпечення, Веб-сторінка

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.1 – Сторінка «Панель керування»

Офіціант або адміністратор має змогу переглядати список усіх замовлень, фільтрувати їх за статусом виконання та оплатою, ознайомлюватися з деталями кожного замовлення, а також створювати нові. При створенні замовлення можна одразу обрати столик, додати до замовлення необхідні позиції меню (страви та напої), вказати кількість і зафіксувати особливі побажання клієнта. Якщо обраний стіл був вільний — його статус автоматично змінюється на «зайнятий». Ця функціональність реалізована у модулі обробки замовлень (orders/views.py, функції order\_list, create\_order, add\_order\_item).

Для зручного управління столиками передбачена окрема сторінка, де наочно відображається поточний стан кожного столика (вільний, зайнятий, зарезервований), а також активні замовлення та бронювання, прив’язані до них. Офіціанти та адміністратори можуть безпосередньо з цієї сторінки створювати нові замовлення або отримувати детальну інформацію про поточне обслуговування (tables/views.py, функція table\_list).

Адміністратор має змогу додавати нові позиції в меню, редагувати існуючі страви, змінювати ціни, опис і склад інгредієнтів. Для цього передбачено окремі представлення в модулі меню (menu/views.py, функції menu\_list, category\_detail, dish\_detail).

Для забезпечення високого рівня безпеки доступ до всіх основних функцій можливий лише після авторизації. Контроль прав доступу реалізовано через декоратори @login\_required та @role\_required, які обмежують доступ до окремих функціональних можливостей системи відповідно до ролі користувача.

Загальний вигляд інтерфейсу відзначається зрозумілими підписами на кнопках, простими формами для введення даних, наочними повідомленнями про успішне виконання операцій або виникнення помилок. Усі ці можливості забезпечують ефективну організацію основних бізнес-процесів ресторану, підвищують продуктивність персоналу та якість обслуговування клієнтів.

## 3.2.2 Інтерфейс відвідувача закладу

Інтерфейс відвідувача закладу розроблений з урахуванням принципів максимальної простоти, зручності та інтуїтивної зрозумілості для користувачів будь-якого рівня володіння сучасними технологіями.

Основне призначення цього інтерфейсу — надати гостям закладу можливість самостійно переглядати актуальне меню, обирати страви та формувати замовлення без необхідності додаткового звернення до персоналу.

При відкритті клієнтської сторінки відображається меню, яке структуровано за категоріями страв, зокрема: салати, гарячі страви, напої, десерти тощо. Кожна категорія містить перелік страв з фотографіями, коротким описом, зазначенням ціни та інформацією про наявність позиції в меню. Це дозволяє швидко зорієнтуватися у виборі та оцінити пропозиції закладу.

На рис. 3.2 можна побачити сторінку меню.

Зображення, що містить текст, Закуска, Фаст-фуд, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.2 – Сторінка «Меню»

Для отримання детальнішої інформації про певну страву користувач може натиснути на відповідну позицію, після чого відкривається додаткове вікно з розширеним описом, складом та зазначенням можливих алергенів. Страва може бути додана до кошика одним натисканням кнопки (див. рис. 3.3).

Зображення, що містить Закуска, випічка, їжа, текст

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.3 – Детальна сторінка страви

Після завершення вибору клієнт переходить до кошика, де відображаються всі обрані позиції із зазначенням кількості та загальної суми замовлення. При необхідності користувач може змінити кількість обраних страв або видалити окремі позиції. Підтвердження замовлення здійснюється в один клік, після чого воно автоматично надходить до системи обробки замовлень закладу.

Важливою особливістю клієнтського інтерфейсу є інтеграція модуля рекомендацій на основі штучного інтелекту. Система аналізує поточний вибір клієнта та надає персоналізовані поради щодо додаткових страв і напоїв, які логічно доповнюють замовлення. Рекомендації формуються за допомогою сучасної мовної моделі Llama 3, що дозволяє покращити досвід клієнтів і сприяє підвищенню середнього чеку закладу.

Особлива увага при проєктуванні клієнтського інтерфейсу приділялась ергономіці: всі елементи управління розміщені у доступних зонах екрана, а кнопки мають зрозумілі підписи. Для забезпечення високої читабельності використано контрастну колірну гаму, оптимальні розміри шрифтів та лаконічні графічні елементи. Інтерфейс виключає зайві візуальні ефекти, що можуть ускладнювати взаємодію з системою.

## 3.2.3 Практична реалізація дизайну інтерфейсу користувача

У реалізації інтерфейсу Bublydr використано шаблони Django Templates у поєднанні з фреймворком Bootstrap 5, що забезпечує адаптивність і кросплатформеність рішення. Завдяки використанню системи блоків Django ({% block %}) та розширення шаблонів ({% extends %}), забезпечено централізоване керування структурою сторінок, зокрема розміткою навігаційного меню, футера, заголовків, повідомлень і головного контенту. Такий підхід спрощує підтримку інтерфейсу та гарантує його однорідність у межах усієї системи.

Інтерфейс проєкту побудований на базі сітки Bootstrap 5 із використанням компонентів container, row, col-md-\*, що дозволяє ефективно масштабувати вміст на різні екрани. Структура шаблонів побудована за принципом «адаптивної деградації»: елементи автоматично перебудовуються у вертикальні блоки на мобільних пристроях без втрати функціональності. Компоненти, такі як navbar, dropdown, alert, card, table-responsive, активно використовуються для створення інтуїтивного й адаптивного користувацького інтерфейсу.

Зміст інтерфейсу формується динамічно на основі контексту, переданого з представлень Django, із використанням змінних ({{ ... }}), фільтрів форматування (|date, |default, |floatformat) та умовних конструкцій ({% if %}, {% for %}). Це дозволяє відображати лише актуальні дані (наприклад, статус замовлення або наявність активного облікового запису) і реалізувати логіку персоналізованої взаємодії з користувачем.

Доступ до функціональних елементів інтерфейсу регулюється умовами, які враховують статус автентифікації (user.is\_authenticated) та роль користувача (user.user\_type, user.is\_staff, user.is\_admin). На основі цих умов динамічно відображаються або приховуються окремі кнопки, пункти меню чи блоки сторінки. Наприклад, адміністратор бачить додаткові елементи навігації для керування користувачами, меню та фінансами, а клієнту вони не відображаються взагалі.

Інтерфейс орієнтований на швидкий доступ до найважливіших функцій. Кнопки дій (btn, btn-primary, btn-outline-\*) розташовуються у верхній частині інтерфейсних блоків або в логічних місцях таблиць, карток і панелей. Для забезпечення кращої доступності використовуються великі шрифти, контрастні кольори і піктограми із бібліотеки Bootstrap Icons (<i class="bi bi-...">). Наприклад, статуси об'єктів оформлюються через бейджі badge bg-success, bg-danger, bg-warning, які додають інтерфейсу наочності й дозволяють швидко візуально розрізняти об’єкти за станом.

Інтерфейс реалізовано за принципом контекстної мінімалізації: сторінки містять лише ті елементи, які необхідні на даному етапі роботи, що значно підвищує зручність використання та зменшує навантаження на користувача. Для цього застосовуються умовні блоки, що приховують технічні деталі або функції, доступні лише певним ролям, зокрема адміністраторам або менеджерам. Складні або рідковживані налаштування згруповані в окремі розділи або приховані за допомогою collapsible-елементів Bootstrap (accordion, collapse), що дозволяє зберігати візуальну чистоту та впорядкованість інтерфейсу навіть при значному обсязі функцій. Для зворотного зв’язку реалізовано систему повідомлень із категоріями success, info, warning, danger, яка виводить результат дії користувача одразу після її виконання. Повідомлення додаються на стороні сервера через messages.add\_message() і автоматично відображаються у шаблонах у відповідному стилі Bootstrap Alerts, що дозволяє миттєво інформувати персонал про результат дій без потреби оновлення сторінки.

Профілі користувачів, замовлення, меню, столики та інші об'єкти виводяться у вигляді таблиць (table, table-striped, table-hover) або карток (card, card-body, card-header), забезпечуючи зрозумілу структуру даних та швидкий доступ до потрібної інформації. При цьому всі компоненти підтримують адаптивність завдяки table-responsive і row-cols-\*, що дає змогу відображати великі обсяги інформації навіть на пристроях із невеликим екраном, таких як смартфони або невеликі планшети.

Таким чином, інтерфейс Bublydr не лише відповідає сучасним стандартам дизайну, а й реалізований із технічної точки зору як модульна, адаптивна і логічно організована система шаблонів із гнучкою підтримкою ролей, станів і типів даних.

# 3.3 Конфігурація проєкту та система міграцій бази даних

## 3.3.1 Організація структури керування та конфігурації проєкту

У проектованому застосунку важливу роль у забезпеченні керованості та гнучкості системи відіграє організація структури проєкту та підхід до зберігання конфігураційних параметрів. Центральним елементом є файл [manage.py](vscode-file://vscode-app/c:/Users/vadim/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), який виступає універсальним інтерфейсом для взаємодії розробника чи адміністратора із системою. Саме через цей файл здійснюється запуск локального серверного середовища, ініціалізація та застосування міграцій, створення суперкористувача, виконання тестів, а також інші адміністративні операції, необхідні для підтримки життєвого циклу проєкту.

Файл [manage.py](vscode-file://vscode-app/c:/Users/vadim/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) містить точку входу для виконання команд Django, автоматично підключаючи основні налаштування проєкту через змінну оточення DJANGO\_SETTINGS\_MODULE. Це дозволяє централізовано керувати всіма параметрами системи, не змінюючи структуру коду при переході між різними середовищами. До прикладу, запуск сервера розробки здійснюється командою «python manage.py runserver».

Нижче надано код файлу manage.py:

def main():

"""Run administrative tasks."""

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'bublydr.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

main()

Всі основні конфігураційні файли розміщені у директорії bublydr, яка містить налаштування підключення до бази даних, параметри безпеки, шляхи до статичних і медіафайлів, а також визначає перелік встановлених додатків. Такий підхід забезпечує модульність і масштабованість системи, дозволяючи легко додавати чи змінювати функціональні компоненти без ризику порушення цілісності проєкту.

Для безпеки використовується файл .env, у якому містяться чутливі параметри: секретний ключ, облікові дані для підключення до бази даних, налаштування зовнішніх API тощо. Завдяки цьому критично важлива інформація не потрапляє до загальнодоступного репозиторію, що відповідає сучасним вимогам інформаційної безпеки та дозволяє розмежувати відкритий код і приватні налаштування.

## 3.3.2 Система керування міграціями бази даних

У процесі розробки POS системи Bublydr важливою складовою стало впровадження механізму керування структурними змінами бази даних. Для цього використано вбудовану систему міграцій, яку надає фреймворк Django. Вона забезпечує узгоджене створення, оновлення та збереження історії змін у схемі бази даних без необхідності ручного написання SQL-скриптів.

Кожен модуль системи, наприклад accounts, orders, inventory або menu, містить власний каталог migrations/, у якому зберігаються файли міграцій у хронологічному порядку. Ці файли формуються автоматично за допомогою команди python manage.py makemigrations та відображають усі зміни, внесені до моделей: додавання або видалення полів, створення нових таблиць, зміна типів даних тощо.

Застосування міграцій здійснюється командою python manage.py migrate, яка генерує відповідні SQL-запити й застосовує їх до активної бази даних. Усі виконані міграції фіксуються у службовій таблиці django\_migrations, що дозволяє контролювати стан схеми бази на будь-якому етапі розробки чи розгортання.

Використання цього механізму дозволяє забезпечити безперервність розробки навіть у командному середовищі, коли кілька розробників одночасно працюють із різними частинами системи. Django гарантує, що зміни будуть застосовані у правильній послідовності, зберігаючи цілісність даних.

Також система підтримує створення кастомних або ручних міграцій, що дозволяє гнучко реалізовувати складні сценарії, наприклад, перенесення даних між таблицями або застосування умовних змін.

# 3.4 Реалізація функціональних модулів.

## 3.4.1 Модуль управління меню

Модуль управління меню відіграє ключову роль у функціональній структурі системи Bublydr, оскільки саме через нього здійснюється керування основним ресурсом ресторану — меню. Цей модуль забезпечує повний життєвий цикл управління інформацією про страви, їх категорії, доступність та супровідні характеристики. Зокрема, адміністратори та менеджери мають можливість оперативно оновлювати дані, що є критично важливим для підтримки актуальності інформації про пропозиції закладу.

Архітектурно модуль побудовано відповідно до патерну Model-View-Template (MVT), який є базовим для фреймворку Django. У моделі (Model) визначаються сутності предметної області, такі як Category та Dish. Представлення (View) реалізують бізнес-логіку та обробляють запити користувачів, а шаблони (Template) відповідають за відображення інформації у браузері. Взаємодія між цими компонентами дозволяє підтримувати чистоту коду, розділення відповідальностей і спрощує супровід системи.

Загальна логіка обробки HTTP-запитів зосереджена у файлі views.py модуля menu. Функції цього модуля відповідають за відображення списку категорій, детальну інформацію про конкретну категорію разом зі стравами та відображення окремих позицій меню. Приклади ключових представлень:

def menu\_list(request):

categories = Category.objects.filter(is\_active=True)

return render(request, 'menu/list.html', {'categories': categories})

def category\_detail(request, category\_id):

category = get\_object\_or\_404(Category, id=category\_id, is\_active=True)

dishes = category.dishes.filter(is\_available=True)

return render(request, 'menu/category\_detail.html', {'category': category, 'dishes': dishes})

def dish\_detail(request, dish\_id):

dish = get\_object\_or\_404(Dish, id=dish\_id, is\_available=True)

return render(request, 'menu/dish\_detail.html', {'dish': dish})

Завдяки використанню ORM Django реалізовано безпечну роботу з базою даних, що зменшує ймовірність виникнення помилок, пов’язаних із некоректними SQL-запитами. Важливо, що у всіх запитах враховується доступність категорій і страв за допомогою фільтрації за полями is\_active та is\_available, що забезпечує коректне відображення лише актуальних позицій.

Важливим аспектом є обробка статичних ресурсів, таких як CSS-стилі, JavaScript-скрипти та медіафайли (зображення страв). Вони зберігаються у каталозі static/, а шаблони інтерфейсу — у templates/menu/.

Модуль управління меню тісно інтегровано з іншими компонентами системи, зокрема з модулями замовлень, обліку складу та фінансових розрахунків. Це створює єдину інформаційну екосистему, в якій всі дії користувачів та адміністрації узгоджені й автоматизовані.

## 3.4.2 Модуль бронювання столів

Модуль бронювання столиків є важливою складовою програмного комплексу Bublydr, що відповідає за ефективне управління просторовими ресурсами закладу громадського харчування. Його основним призначенням є організація процесу попереднього бронювання місць, контроль завантаженості залу, забезпечення оптимального розподілу клієнтських потоків і підтримка високого рівня обслуговування (див. рис. 3.4).

Зображення, що містить текст, знімок екрана, програмне забезпечення, Комп’ютерна піктограма

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.4 – Сторінка «Бронювання столиків»

Реалізація модуля сприяє мінімізації конфліктів під час посадки гостей, підвищенню продуктивності персоналу і, як наслідок, покращенню загального клієнтського досвіду. Крім того, завдяки автоматизації цього процесу, значно зменшуються витрати часу на ручне ведення розкладу та усуваються помилки, пов’язані з людським фактором (див. рис. 3.5).

Зображення, що містить текст, знімок екрана, програмне забезпечення, число

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.5 – Сторінка «Створення нового бронювання»

З архітектурної точки зору модуль реалізовано у вигляді окремого додатка tables фреймворку Django. Такий підхід забезпечує логічну ізоляцію бізнес-логіки, спрощує процес масштабування системи та полегшує її підтримку. Функціональні можливості модуля охоплюють створення, редагування, перегляд і скасування бронювань, а також інтеграцію з іншими модулями системи, зокрема з модулем керування замовленнями. Взаємодія між модулями побудована таким чином, щоб усі зміни, що стосуються статусу столиків або їх бронювань, миттєво відображалися у відповідних розділах системи, забезпечуючи актуальність інформації в режимі реального часу.

У моделі даних сутність Table описує характеристики об’єктів бронювання — столиків. Для відображення актуального статусу столика використовується атрибут status, який може набувати значень: вільний (free), зайнятий (occupied), зарезервований (reserved) або недоступний (unavailable). Додаткові атрибути моделі включають кількість місць, розташування в залі, наявність приміток і статус активності.

Така структура дозволяє не лише вести облік доступних ресурсів, а й формувати різноманітні звіти про ефективність використання залу, що є корисним інструментом для менеджменту закладу.

Нижче наведено приклад опису сутності Table у вигляді Django-моделі, що ілюструє структуру зберігання даних про столики в системі Bublydr:

class Table(models.Model):

STATUS\_CHOICES = (

('free', 'Вільний'),

('occupied', 'Зайнятий'),

('reserved', 'Зарезервований'),

('unavailable', 'Недоступний'),

)

number = models.IntegerField(unique=True, verbose\_name=\_('Номер столика'))

capacity = models.IntegerField(verbose\_name=\_('Кількість місць'))

status = models.CharField(max\_length=20, choices=STATUS\_CHOICES, default='free')

location = models.CharField(max\_length=100, verbose\_name=\_('Розташування'))

Модель Reservation реалізує збереження інформації про здійснені бронювання, включаючи дату та час резервування, тривалість перебування, контактні дані клієнтів і статус бронювання. Це дозволяє системі відстежувати як майбутні, так і вже завершені або скасовані бронювання.

Для забезпечення актуальної інформації про статуси столиків і відповідних бронювань реалізовано спеціальні представлення (views). Наприклад, функція table\_list забезпечує отримання переліку всіх доступних столиків разом із прив’язкою активних замовлень, що дозволяє оперативно визначити завантаженість залу.

def table\_list(request):

tables = Table.objects.filter(is\_active=True)

tables\_with\_orders = []

for table in tables:

active\_order = None

try:

from orders.models import Order

active\_order = Order.objects.filter(

table=table,

status\_\_in=['new', 'processing', 'ready', 'delivered']

).latest('created\_at')

except:

pass

tables\_with\_orders.append({'table': table, 'active\_order': active\_order})

return render(request, 'tables/list.html', {'tables\_with\_orders': tables\_with\_orders})

Формування і редагування бронювань здійснюється через форми, розроблені з використанням Django Forms і інтегровані з фреймворком Bootstrap для покращення зовнішнього вигляду та забезпечення зручної валідації даних.

class ReservationForm(forms.ModelForm):

reservation\_date = forms.DateField(widget=forms.DateInput(attrs={'type': 'date'}))

reservation\_time = forms.TimeField(widget=forms.TimeInput(attrs={'type': 'time'}))

class Meta:

model = Reservation

fields = ['table', 'customer\_name', 'customer\_phone', 'customer\_email',

'reservation\_date', 'reservation\_time', 'duration',

'guests\_count', 'status', 'notes']

Усі дії, пов’язані з керуванням столиками та бронюванням, захищені механізмами автентифікації та авторизації. Для обмеження доступу до функціоналу використовуються вбудовані механізми Django через декоратори @login\_required та спеціалізовані перевірки ролей, що запобігають несанкціонованим змінам даних.

Інтеграція з модулем замовлень дозволяє системі автоматично змінювати статус столика при створенні замовлення або бронювання, а також виводити актуальну інформацію про активні замовлення безпосередньо у вікні керування столиками. Це підвищує прозорість бізнес-процесів і дозволяє адміністрації закладу швидко реагувати на зміну ситуації в залі.

Завдяки чітко визначеній структурі модулів, використанню сучасних вебтехнологій та підтримці взаємодії з іншими компонентами системи, модуль бронювання столиків виконує роль центрального елемента управління просторовими ресурсами ресторану.

## 3.4.3 Модуль оформлення замовлень

Модуль оформлення замовлень є функціональною віссю серверної частини POS системи Bublydr, що реалізує повний життєвий цикл взаємодії між клієнтом, офіціантом та кухнею ресторану. Його реалізація базується на фреймворку Django, що дозволяє забезпечити високий рівень масштабованості, підтримку авторизації, гнучку роботу з ORM, а також інтеграцію з іншими підсистемами платформи.

У структурі системи модуль оформлення замовлень представлено окремим Django-додатком orders. Архітектура додатку побудована з урахуванням принципів логічної ізоляції, де моделі, представлення, форми та шаблони організовані в окремих відповідальних компонентах. Основними моделями, які беруть участь у реалізації функціоналу, є Order та OrderItem. Перша описує загальні мета-дані замовлення, зокрема дату та час створення, відповідального офіціанта, прив’язку до столика, статус виконання та оплати, суму, а також коментарі.

Модель OrderItem, у свою чергу, відповідає за деталізацію кожної позиції замовлення, включаючи посилання на конкретну страву, кількість одиниць, ціну та статус приготування.

Формування нових замовлень відбувається за допомогою OrderForm, яка реалізує первинну логіку створення об'єкта Order. Після заповнення форми та її валідації система виконує ряд критичних дій, зокрема автоматичне прив’язування замовлення до авторизованого користувача, який виконує роль офіціанта. У випадку, якщо вказаний стіл є вільним, його статус оновлюється до “зайнятий”. Це забезпечує унеможливлення ситуацій, коли кілька офіціантів одночасно використовують один і той самий ресурс.

Нижче наведено фрагмент відповідного представлення, що відповідає за створення нового замовлення:

@login\_required

def create\_order(request):

if request.method == 'POST':

form = OrderForm(request.POST)

if form.is\_valid():

order = form.save(commit=False)

order.waiter = request.user

order.save()

if order.table and order.table.status == 'free':

order.table.status = 'occupied'

order.table.save()

messages.success(request, \_('Замовлення успішно створено!'))

return redirect('orders:add\_item', order\_id=order.id)

Після створення замовлення система автоматично перенаправляє користувача до маршруту для додавання позицій. Тут використовується форма OrderItemForm, яка дозволяє обрати страву, вказати кількість та додати коментар.

Перегляд замовлень реалізовано через функцію order\_list із фільтрацією за статусом виконання та оплати. GET-параметри status і payment формують ORM-запит, результати якого виводяться у зворотному хронологічному порядку.

Сторінку перегляду активних замовлень зображено на рисунку 3.6.

@login\_required

def order\_list(request):

status\_filter = request.GET.get('status')

payment\_filter = request.GET.get('payment')

orders = Order.objects.all()

if status\_filter and status\_filter != 'all':

orders = orders.filter(status=status\_filter)

if payment\_filter and payment\_filter != 'all':

orders = orders.filter(payment\_status=payment\_filter)

orders = orders.order\_by('-created\_at')

return render(request, 'orders/list.html', {

'orders': orders,

'status\_filter': status\_filter,

'payment\_filter': payment\_filter,

'status\_choices': Order.STATUS\_CHOICES,

'payment\_choices': Order.PAYMENT\_STATUS\_CHOICES,

})

Зображення, що містить текст, знімок екрана, програмне забезпечення, Комп’ютерна піктограма

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.6 – Сторінка «Панель керування»

Важливо зазначити, що всі операції з базою даних здійснюються виключно через ORM Django. Це гарантує захищеність запитів, унеможливлює SQL-ін’єкції та забезпечує зрозумілу абстракцію над структурою таблиць. Міжмодульна взаємодія реалізується через зв’язки типу ForeignKey, які дозволяють встановлювати чіткі логічні залежності між замовленням, позиціями, офіціантом, столиком та іншими бізнес-об’єктами.

Особлива увага приділяється авторизації та валідації. Доступ до всіх маршрутів модуля обмежений декоратором @login\_required, а контроль прав доступу до перегляду та редагування реалізується через рольову систему. Усі форми містять внутрішні механізми перевірки даних, а результати дій відображаються користувачеві через Django-фреймворк повідомлень messages, що істотно підвищує зручність використання системи в умовах ресторанного навантаження.

Модуль також є тісно інтегрованим з іншими підсистемами: зокрема, при створенні або редагуванні замовлень оновлюється статус відповідного столика, у випадку зміни статусу замовлення на “оплачено”.

Після цього викликаються функції із модуля оплат, а дані про замовлення доступні для модулів аналітики в агрегованому вигляді. Це дозволяє розглядати оформлення замовлень не як ізольовану дію, а як частину цілісного бізнес-процесу ресторану.

Отже, модуль оформлення замовлень у системі Bublydr не лише реалізує базову функцію взаємодії між клієнтом і рестораном, а й слугує ключовим координуючим центром для обробки інформації в межах загальної архітектури системи. Його реалізація відповідає вимогам безпеки, підтримує масштабованість, забезпечує логічну зв’язаність з іншими модулями та дозволяє досягти високої продуктивності обслуговування в реальних умовах експлуатації ресторанного бізнесу.

## 3.4.4 Модуль управління запасами

Модуль управління запасами у системі Bublydr виступає важливим елементом, безпосередньо впливаючим на ефективність і стабільність функціонування ресторанного бізнесу. Запаси матеріальних ресурсів, що охоплюють продукти, напівфабрикати, складові для приготування страв та супутні матеріали, повинні бути контрольовані та оптимально збалансовані. Надлишкові запаси призводять до марнотратства, псування продуктів і збільшення операційних витрат, у той час як дефіцит — до зупинки виробництва і незадоволення клієнтів [13].

Модуль спроєктовано так, щоб забезпечити надійний, прозорий і автоматизований облік ресурсів, що є основою для обґрунтованих управлінських рішень (див. рис. 3.7). Він інтегрується з підсистемою замовлень: при оформленні замовлення відбувається автоматичне списання інгредієнтів відповідно до рецептур. Додатково, система дозволяє формувати звіти щодо руху запасів, контролювати мінімальні залишки та своєчасно ініціювати закупівлі.

Зображення, що містить текст, знімок екрана, програмне забезпечення, Веб-сторінка

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.7 – Сторінка «Контроль запасів ресторану»

Архітектурно модуль реалізований як окремий додаток Django із чіткою предметною доменною моделлю, що відображає всі аспекти матеріального забезпечення ресторану. Основні сутності системи включають моделі, які репрезентують товари, закупівлі, списання, постачальників та рецепти.

Зокрема, модель Product є фундаментальною. Вона описує кожен товар або інгредієнт, включно з характеристиками, такими як назва, категорія, одиниця виміру, поточна кількість, мінімальна кількість для сповіщення про дефіцит, ціна за одиницю та статус активності. Наявність поля мінімальної кількості дозволяє автоматизувати контроль за запасами та генерувати своєчасні оповіщення для персоналу, що зменшує ризик зупинки виробництва через нестачу сировини .

Модель Supplier містить інформацію про постачальників, що є важливою для відстеження якості та надійності постачання. Це дозволяє не лише вести облік контактних даних, але й аналізувати ефективність співпраці з різними контрагентами.

Для фіксації надходжень використовується модель Purchase, що зберігає дані про закупівлі: дату, постачальника та перелік товарів із їхніми кількостями та цінами. Деталізація закупівель реалізована через модель PurchaseItem, яка містить конкретні позиції закупівлі.

Важливою є модель Consumption, що відображає факти списання запасів, зокрема використання у виробництві, псування, або інші причини. Ця модель дозволяє вести історію списань і здійснювати контроль за коректністю обліку.

Для автоматизації списання матеріалів під час приготування страв використовується комплекс моделей Recipe та RecipeIngredient, які зберігають інформацію про склад кожної страви та кількість інгредієнтів, необхідних для її приготування. Такий підхід забезпечує високу точність списань і дає змогу уникнути помилок, що часто виникають при ручному обліку.

Ключовим функціональним елементом модуля є панель управління запасами (dashboard), яка надає персоналу зручний огляд поточного стану ресурсів. На панелі відображаються загальна кількість активних продуктів, кількість позицій з низьким залишком, сумарна вартість запасів, а також останні операції закупівель і списань. Для візуалізації даних використовуються складні ORM-запити із застосуванням агрегацій і аннотацій, що значно підвищує продуктивність системи.

Управління товарами реалізовано через стандартні CRUD-операції з розширеними валідаціями, які запобігають внесенню некоректних даних. Особливо важливою є логіка, що забороняє списання запасів у кількості, більшій за поточний залишок, що виключає можливість виникнення "негативних" залишків.

Закупівельна частина модуля дозволяє фіксувати постачання ресурсів, автоматично оновлюючи складські залишки. Цей механізм не лише підтримує актуальність даних, а й дає змогу накопичувати історичні дані для подальшого аналізу та прогнозування потреб.

Механізм списань має гнучку структуру: списання можуть бути як ручними (при псуванні, втраті тощо), так і автоматичними, що відбуваються при оформленні замовлень у системі. Автоматичне списання на основі рецептів гарантує точність і зменшує трудовитрати персоналу.

Модуль базується на повноцінному використанні можливостей Django ORM, що забезпечує безпеку (захист від SQL-ін’єкцій), високу продуктивність і масштабованість. Для обчислення загальної вартості запасів застосовується комбінація ExpressionWrapper, F() та агрегатних функцій (Sum), що дозволяє виконувати складні обчислення безпосередньо на рівні бази даних, зменшуючи навантаження на сервер додатку.

Валідація введених даних реалізована як на рівні форм, так і безпосередньо у моделях, що гарантує цілісність і достовірність даних. Для інформування користувачів використовується фреймворк повідомлень Django messages, що підвищує зручність роботи персоналу.

Важливою є система розмежування доступу, яка дозволяє обмежувати функціонал модуля відповідно до ролей користувачів. Так, керування запасами доступне лише авторизованим співробітникам із відповідними правами, що виключає несанкціоновані дії.

Модуль управління запасами інтегрований із системою замовлень, що дозволяє автоматично списувати необхідні інгредієнти при кожному оформленні замовлення відповідно до заданих рецептур. Така інтеграція забезпечує високу точність обліку, мінімізує людський фактор і запобігає помилкам, характерним для ручного ведення складських записів.

Крім того, модуль тісно взаємодіє з фінансовою підсистемою: усі операції зі списання та закупівель надходять у звітну систему, де використовуються для розрахунку витрат на одиницю продукції, формування щомісячної аналітики й оцінки ефективності витрат. Це створює аналітичне підґрунтя для прогнозування закупівель, бюджетування та стратегічного управління ресурсами закладу.

Аналітичні дані з модуля управління запасами служать джерелом для оцінки ефективності роботи постачальників, популярності окремих продуктів, а також дозволяють приймати оперативні рішення щодо корекції закупівельної політики.

Приклад реалізації ключової функції – панелі управління запасами:

@login\_required

def inventory\_dashboard(request):

total\_products = Product.objects.filter(is\_active=True).count()

low\_stock\_count = Product.objects.filter(

is\_active=True,

current\_quantity\_\_lt=F('minimum\_quantity')

).count()

total\_value\_expr = ExpressionWrapper(

F('current\_quantity') \* F('price\_per\_unit'),

output\_field=DecimalField()

)

total\_value = Product.objects.filter(is\_active=True).annotate(

value=total\_value\_expr

).aggregate(total=Sum('value'))['total'] or 0

recent\_purchases = Purchase.objects.order\_by('-created\_at')[:5]

recent\_consumptions = Consumption.objects.order\_by('-created\_at')[:5]

low\_stock\_products = Product.objects.filter(

is\_active=True,

current\_quantity\_\_lt=F('minimum\_quantity')

).order\_by('category\_\_name', 'name')[:10]

return render(request, 'inventory/dashboard.html', {

'total\_products': total\_products,

'low\_stock\_count': low\_stock\_count,

'total\_value': total\_value,

'recent\_purchases': recent\_purchases,

'recent\_consumptions': recent\_consumptions,

'low\_stock\_products': low\_stock\_products,

})

Даний код демонструє роботу з базою даних за допомогою ORM Django. Застосування виразів F() і ExpressionWrapper дозволяє виконувати розрахунки безпосередньо на стороні бази даних, що суттєво скорочує час обробки запитів і зменшує навантаження на сервер.

Модуль управління запасами є ключовим елементом системи Bublydr, що забезпечує точний облік ресурсів і підтримку актуальних залишків. Його використання дозволяє підвищити ефективність роботи закладу, мінімізувати втрати та приймати обґрунтовані управлінські рішення щодо закупівель та оптимізації витрат.

Отримані аналітичні дані інтегруються з модулем фінансової звітності, що дозволяє здійснювати комплексний аналіз витрат і доходів у розрізі окремих категорій товарів та постачальників.

## 3.4.5 Модуль управління користувачами та ролями

Модуль управління користувачами та ролями є критично важливим компонентом POS-системи Bublydr, оскільки забезпечує захист персональних даних, контроль доступу до функціональних можливостей і організацію внутрішньої ієрархії користувачів. Основна мета модуля — реалізація процесів реєстрації, автентифікації, авторизації та централізованого управління правами доступу через систему ролей і рівнів доступу.

З технічної точки зору, модуль реалізовано у вигляді окремого Django-додатка accounts, який базується на розширеній моделі користувача, успадкованій від AbstractUser. Це дозволяє зберігати повну сумісність із вбудованими механізмами автентифікації та одночасно розширити модель необхідними полями, специфічними для ресторанної діяльності. Зокрема, це поля для типу користувача (user\_type), email, номера телефону, фізичної адреси та аватару.

У межах системи реалізовано ролі: адміністратор, менеджер, офіціант, кухар, касир і клієнт. Ці ролі визначають функціональні можливості кожного користувача, відображаються в інтерфейсі й використовуються у декораторах @login\_required та @role\_required для контролю доступу до окремих маршрутів. Наприклад, лише адміністратори можуть додавати нових користувачів або змінювати налаштування облікових записів, тоді як офіціанти мають доступ лише до функцій, пов’язаних із замовленнями та столиками.

Завдяки модульній реалізації та чіткому поділу відповідальності, система може легко адаптуватися до різних структур управління персоналом у закладах громадського харчування.

Нижче наведено фрагмент коду, який демонструє реалізацію кастомної моделі користувача:

from django.contrib.auth.models import AbstractUser

from django.db import models

class User(AbstractUser):

USER\_TYPE\_CHOICES = (

('admin', 'Адміністратор'),

('manager', 'Менеджер'),

('waiter', 'Офіціант'),

('cook', 'Кухар'),

('cashier', 'Касир'),

('client', 'Клієнт'),

)

user\_type = models.CharField(max\_length=20, choices=USER\_TYPE\_CHOICES, default='waiter')

phone = models.CharField(max\_length=15, unique=True, verbose\_name='Телефон')

address = models.TextField(blank=True, null=True)

profile\_image = models.ImageField(upload\_to='profile\_images/', blank=True, null=True)

Для забезпечення безпеки всі паролі користувачів зберігаються у хешованому вигляді з використанням криптографічно стійких алгоритмів. Завдяки цьому унеможливлюється відновлення початкового пароля навіть у випадку отримання доступу до бази даних сторонніми особами. Механізми реєстрації та входу до системи реалізовані через стандартні класи Django UserCreationForm та AuthenticationForm, які були модифіковані з урахуванням специфічних вимог системи, включаючи додаткові поля профілю та посилену валідацію введених даних.

Візуальна частина форм реалізована з використанням бібліотеки crispy\_forms, що покращує зручність введення інформації, забезпечує адаптивне відображення інтерфейсу на різних пристроях і підвищує якість контролю правильності даних перед їх обробкою.

Контроль доступу до функціональних можливостей реалізовано через систему ролей, що перевіряється як на рівні представлень (views), так і безпосередньо у шаблонах інтерфейсу.

Для цього використовується декоратор @role\_required, що дозволяє обмежити доступ до окремих маршрутів на основі ролі користувача:

from django.http import HttpResponseForbidden

from functools import wraps

def role\_required(allowed\_roles):

def decorator(view\_func):

@wraps(view\_func)

def \_wrapped\_view(request, \*args, \*\*kwargs):

if request.user.user\_type in allowed\_roles:

return view\_func(request, \*args, \*\*kwargs)

return HttpResponseForbidden("Вам заборонено доступ до цієї сторінки.")

return \_wrapped\_view

return decorator

Модуль також реалізує підтримку персональних профілів користувачів, які містять детальну інформацію про співробітника або клієнта. Адміністратори мають розширені повноваження з перегляду та редагування профілів інших користувачів, що спрощує процес керування персоналом і забезпечує актуальність інформації в системі.

Інтеграція модуля з іншими компонентами системи є тісною та повною. Наприклад, офіціанти мають доступ до створення замовлень, кухарі — до перегляду списку страв для приготування, касири — до модулів обробки оплат, а менеджери — до аналітичних звітів і фінансових показників. Адміністратори, у свою чергу, мають повний контроль над системою, включаючи створення нових користувачів, керування їхніми правами та доступ до адміністративної панелі.

Особлива увага приділяється захисту критичних функцій. Використання декораторів @login\_required та вбудованої системи повідомлень messages дозволяє забезпечити як технічну, так і організаційну безпеку, а також підвищує інформативність системи для кінцевого користувача.

## 3.4.6 Реалізація push-сповіщень

Оперативна комунікація з клієнтами є важливою складовою стратегії підвищення лояльності, збільшення повторних продажів і формування емоційного зв’язку із закладом. У системі Bublydr реалізовано повноцінний модуль push-сповіщень, який інтегровано безпосередньо в адміністративний веб-інтерфейс. Завдяки цьому адміністратори мають можливість самостійно створювати, налаштовувати та запускати розсилки без залучення технічних спеціалістів, що значно підвищує гнучкість маркетингових кампаній.

Функціональні можливості модуля дозволяють адміністратору обирати тип розсилки — періодичну або разову — залежно від поставленої мети. Для розсилок доступне гнучке налаштування умов вибірки одержувачів, наприклад, можна сформувати групу клієнтів, які не здійснювали замовлень протягом останніх 30 днів. Крім цього, передбачена можливість ручного введення тексту повідомлення або використання заздалегідь підготовлених шаблонів, у яких підтримується підстановка змінних, таких як ім’я клієнта, індивідуальна знижка або дата останньої покупки. Важливо, що адміністратор може обрати як негайне надсилання повідомлень, так і запланувати їх відправлення на визначену дату та час. Також передбачена можливість попереднього перегляду сформованого повідомлення для різних типів клієнтів, що дозволяє переконатися у коректності персоналізації перед розсилкою.

Контакти клієнтів збираються під час реєстрації у системі. Кожен користувач може надати згоду на отримання сповіщень, що фіксується у полі allow\_notifications моделі User. На основі цього параметра адміністратор може здійснювати сегментацію клієнтів через веб-інтерфейс модуля розсилок. Наприклад, для повернення неактивних користувачів формується вибірка клієнтів, дата останнього замовлення яких є більш ніж на 30 днів давнішою за поточну.

До повідомлення підставляється ім’я клієнта, пропонується індивідуальна знижка та враховуються особливості попередніх замовлень. Надсилання повідомлень виконується через інтеграцію з платформою Twilio WhatsApp API, що дозволяє швидко доставляти інформацію на найбільш популярний серед клієнтів канал комунікації. Усі запуски розсилок та результати доставки фіксуються в системі за допомогою моделі PushLog. Це дозволяє адміністраторам у будь-який момент переглянути історію проведених кампаній, оцінити статус доставки та проаналізувати ефективність кожної розсилки.

З метою забезпечення безпеки персональних даних та відповідності нормативним вимогам, користувачі мають можливість у будь-який момент відмовитись від отримання сповіщень, змінивши відповідне налаштування у своєму профілі. Усі персональні дані обробляються згідно з вимогами регламенту GDPR. Дані зберігаються у зашифрованому вигляді, а доступ до модуля розсилок мають виключно адміністратори з відповідними правами. Крім того, реалізовано аудит усіх вихідних повідомлень, що гарантує прозорість і контроль за маркетинговою активністю системи.

# 3.5 Інтеграція модуля штучного інтелекту

## 3.5.1 Моделі та алгоритми побудови рекомендацій

У підсистемі рекомендацій Bublydr застосовується трансформерна мовна модель Mistral 7B, інтегрована через платформу Ollama. Це забезпечує універсальний інтерфейс для роботи з різними мовними моделями штучного інтелекту та дозволяє оперативно змінювати або оновлювати модель без суттєвих змін у кодовій базі [10].

Генерація рекомендацій відбувається на основі динамічного контексту, сформованого з поточного замовлення клієнта та актуального асортименту закладу. Історичні дані про попередні замовлення не використовуються: запит до моделі побудований у режимі реального часу, що істотно спрощує інфраструктуру та уникає потреби в зберіганні великих обсягів даних.

Алгоритм роботи системи починається зі збору інформації про страви, які вже додані до замовлення, та формування переліку доступних для рекомендації позицій із виключенням тимчасово недоступних. Далі формується текстовий prompt у природній мові з чіткими інструкціями: рекомендувати не більше трьох страв або напоїв, які логічно доповнюють наявний вибір, або у випадку відсутності релевантних варіантів повернути стандартну фразу про неможливість додати щось до замовлення. Після цього система відправляє HTTP-запит до Ollama API з повідомленням користувача й обробляє відповідь моделі: якщо вона повертає фразу "There is nothing to add to the order", інтерфейс повідомляє про відсутність рекомендацій, в інших випадках – виводить користувачеві до трьох запропонованих доповнень.

Такий підхід дозволяє отримувати релевантні й практичні поради без складних ETL-процесів та попередньої підготовки даних. Архітектуру легко масштабувати і в майбутньому розширити за рахунок інтеграції історичних даних, врахування харчових обмежень, алергенів та індивідуальних уподобань клієнтів.

## 3.5.2 Архітектура та інтеграція рекомендаційої системи

Архітектурно рекомендаційний модуль Bublydr складається із трьох основних компонентів: сервісу збору інформації про поточне замовлення, підсистеми генерації запитів до мовної моделі та механізму обробки відповіді й відображення результатів у клієнтському інтерфейсі. Сервіс збору даних збирає перелік вже вибраних страв із замовлення та формує список доступних для рекомендації позицій, виключаючи ті, що тимчасово недоступні.

Підсистема генерації запитів реалізована в окремому модулі gpt\_service.py. Вона формує текстовий prompt на основі списків обраних і доступних страв, додаючи чіткі інструкції для моделі: рекомендувати до трьох доповнень або повернути фіксовану фразу про відсутність релевантних варіантів. Для взаємодії із застосованою локальною або віддаленою мовною моделлю використовуються HTTP-запити до API платформи Ollama, що забезпечує універсальний інтерфейс для різних версій моделей (Mistral 7B, Llama 3 тощо) [11,12].

Отримана від моделі відповідь надходить у вигляді тексту, який підсистема обробляє та передає далі (див. рис. 3.8). Якщо текст містить повідомлення про відсутність рекомендацій, в інтерфейсі відображається відповідне інформаційне повідомлення, інакше – користувачу або офіціанту показуються до трьох запропонованих страв чи напоїв.

Інтеграція в клієнтський інтерфейс реалізована через AJAX-запит: при натисканні кнопки «Рекомендація від GPT» відправляється список уже доданих страв, а результат виводиться у спеціальному блоці «gpt-recommendation-result». Це дозволяє асинхронно отримувати й відображати рекомендації без перезавантаження сторінки.

Зображення, що містить текст, електроніка, знімок екрана, програмне забезпечення

Вміст, створений ШІ, може бути неправильним.

Рисунок 3.8 – Сторінка перегляду замовлення з ШІ рекомендацією

Використання Ollama API та підхід із локальною розгорткою моделей забезпечують високу гнучкість: зміну або модернізацію моделі можна здійснити без зміни основного коду, а всі дані обробляються локально, що відповідає вимогам безпеки та конфіденційності.

## 3.5.3 Реалізація функціоналу розумних рекомендації

Нижче наведено приклад реалізації запиту до рекомендаційного сервісу в модулі gpt\_service.py. Функція get\_recommendation формує prompt на основі списків вибраних та доступних страв, відправляє HTTP-запит до платформи Ollama та повертає текст рекомендацій:

import requests

OLLAMA\_URL = 'http://localhost:11434/api/chat'

def get\_recommendation(dishes, available\_dishes):

if not dishes:

return "Please select at least one dish."

prompt = (

f"Ordered: {', '.join(dishes)}. "

f"Available items: {', '.join(available\_dishes)}. "

"Recommend up to 3 items that best complement the order, "

"or return 'There is nothing to add to the order'."

)

try:

response = requests.post(

OLLAMA\_URL,

json={

"model": "mistral:7b-instruct",

"messages": [{"role": "user", "content": prompt}],

"stream": False

},

timeout=60

)

data = response.json()

text = data['message']['content'].strip()

if "There is nothing to add to the order" in text:

return "There is nothing to add to the order"

return text

except Exception:

return "Failed to get recommendation. Please try again later."

У клієнтському інтерфейсі JavaScript реалізовано AJAX-виклик за допомогою Fetch API. Нижче фрагмент коду, який відправляє список замовлених страв на сервер і відображає отримані рекомендації у блоці з ідентифікатором

gpt-recommendation-result.

document.getElementById('gpt-recommend-btn').addEventListener('click', async () => {

const orderId = event.target.dataset.orderId;

const dishes = getSelectedDishes(orderId); // користувацька функція

const response = await fetch(`/api/orders/${orderId}/recommend`, {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ dishes })

});

const resultBlock = document.getElementById('gpt-recommendation-result');

if (response.ok) {

const { recommendation } = await response.json();

resultBlock.textContent = recommendation;

resultBlock.classList.remove('d-none');

} else {

resultBlock.textContent = 'Error fetching recommendation.';

resultBlock.classList.remove('d-none');

}

});

Приклад HTML-розмітки, який забезпечує інтеграцію кнопки та блоку для відображення рекомендацій:

<button id="gpt-recommend-btn" class="btn btn-outline-info mb-2" type="button" data-order-id="{{ order.id }}">

<i class="bi bi-stars"></i> Рекомендація від GPT

</button>

<div id="gpt-recommendation-result" class="alert alert-info d-none"></div>

# 3.6 Забезпечення безпеки та контролю доступу

## 3.6.1 Рівні доступу користувачів

У системі Bublydr реалізовано багаторівневу модель доступу, яка забезпечує захист даних, розмежування повноважень персоналу та контроль над виконанням бізнес-процесів. Вона побудована на основі розширеної моделі користувача фреймворку Django із застосуванням кастомної моделі User, яка наслідує стандартну AbstractUser.

Ключовим атрибутом моделі є поле user\_type, що визначає роль користувача в системі. Підтримуються такі ролі: адміністратор (admin), менеджер (manager), офіціант (waiter), кухар (cook), касир (cashier) та клієнт (client). Це дозволяє ефективно регламентувати доступ до ресурсів системи залежно від посадових обов’язків персоналу.

Фрагмент визначення моделі користувача наведено нижче:

class User(AbstractUser):

USER\_TYPE\_CHOICES = (

('admin', 'Адміністратор'),

('manager', 'Менеджер'),

('waiter', 'Офіціант'),

('cook', 'Кухар'),

('cashier', 'Касир'),

('client', 'Клієнт'),

)

user\_type = models.CharField(max\_length=20, choices=USER\_TYPE\_CHOICES, default='waiter')

Контроль доступу до представлень реалізовано через комбінацію декораторів @login\_required та власного декоратора @role\_required. Це дозволяє обмежувати доступ до певних функцій залежно від ролі користувача без дублювання перевірок у кожному методі.

Фрагмент декоратора обмеження доступу:

def role\_required(allowed\_roles: list[str]):

def decorator(view\_func):

@wraps(view\_func)

def \_wrapped\_view(request, \*args, \*\*kwargs):

if hasattr(request.user, 'user\_type') and request.user.user\_type in allowed\_roles:

return view\_func(request, \*args, \*\*kwargs)

return HttpResponseForbidden("Вам заборонено доступ до цієї сторінки.")

return \_wrapped\_view

return decorator

На рівні інтерфейсу управління відображенням елементів реалізовано динамічне приховування кнопок та посилань за допомогою шаблонної мови Django. Це дозволяє уникнути спроб доступу до заборонених функцій навіть на рівні інтерфейсу.

Фрагмент шаблону контролю видимості елементів:

{% if user.user\_type == 'manager' or user.user\_type == 'admin' %}

<a href="{% url 'inventory:dashboard' %}">Запаси</a>

{% endif %}

Завдяки такій архітектурі кожен працівник має доступ лише до тих функцій, які необхідні для виконання його службових обов’язків. Це підвищує безпеку системи, мінімізує людські помилки та забезпечує простоту масштабування й адаптації системи під нові бізнес-потреби.

## 3.6.2 Захист персональних даних

У системі реалізовано комплекс заходів щодо захисту персональних даних користувачів, який базується на рекомендаціях фреймворку Django та сучасних принципах інформаційної безпеки. Основним механізмом захисту автентифікаційних даних є криптографічне хешування паролів із використанням сольових значень, що виключає можливість зберігання паролів у відкритому вигляді та унеможливлює компрометацію облікових записів навіть у разі витоку бази даних.

Додатково у системі застосовуються стандартні валідатори надійності паролів, які визначають мінімальні вимоги до складності паролів та запобігають використанню слабких або поширених комбінацій. Конфігурація механізмів валідації паролів представлена наступним чином:

AUTH\_PASSWORD\_VALIDATORS = [

{'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidato},

{'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator'},

{'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator'},

{'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator'},]

Доступ до функціональних модулів системи захищений за допомогою декоратора @login\_required для перевірки автентифікації.

Також було реалізовано власний декоратор @role\_required, який забезпечує контроль доступу на основі ролей користувачів. Наприклад, доступ до модуля керування користувачами обмежено адміністраторами та менеджерами:

@login\_required

@role\_required(['admin', 'manager'])

def users\_list\_view(request):

Особливу увагу приділено обмеженню видимості персональних даних. Користувачі мають можливість переглядати та редагувати лише власний профіль, тоді як розширені повноваження з управління обліковими записами доступні лише адміністраторам через відповідний функціонал панелі керування. Для безпечної роботи з автентифікацією встановлені стандартні налаштування перенаправлення користувачів після входу та виходу із системи:

LOGIN\_REDIRECT\_URL = 'home'

LOGIN\_URL = 'accounts:login'

LOGOUT\_REDIRECT\_URL = 'accounts:login'

Захист поширюється також і на завантажувані користувачами файли, включаючи зображення профілів. Всі медіафайли зберігаються у спеціальній директорії MEDIA\_ROOT, доступ до якої контролюється через вебсервер і обробники файлів у Django. Фізичний доступ до контенту можливий лише для авторизованих користувачів, а шляхи до файлів зберігаються у захищеній базі даних.

Для забезпечення конфіденційності даних реалізовано механізм відмови від розсилки сповіщень, який доступний у профілі користувача. Усі персональні дані обробляються відповідно до вимог регламенту GDPR, зберігаються у зашифрованому вигляді, а доступ до критичних компонентів системи мають лише адміністратори з відповідними повноваженнями. Крім цього, у системі запроваджено аудит усіх дій, пов’язаних із обробкою персональних даних, що дозволяє підтримувати високий рівень прозорості та контролю.

## 3.6.3 Безпека транзакцій

Фінансові операції в системі реалізовані з урахуванням вимог до безпеки транзакцій у середовищі ресторанного бізнесу. Основна увага приділяється контролю доступу, валідації дій, узгодженості статусів і захисту від несанкціонованих змін.

Усі дії з платежами — створення, завершення, повернення чи перегляд — дозволені лише авторизованим користувачам. Для цього використовується декоратор @login\_required, який блокує доступ стороннім особам. Додатково перевіряється роль користувача: завершення або повернення платежу дозволене лише для cashier, admin, waiter або manager:

@login\_required

@role\_required(['waiter', 'admin', 'manager', 'cashier'])

def complete\_payment(request, payment\_id):

    """Завершення платежу"""

    payment = get\_object\_or\_404(Payment, id=payment\_id)

Ключовим елементом захисту є контроль статусів транзакцій. Кожен об’єкт моделі Payment має визначений набір статусів: pending, completed, failed, refunded. Це дозволяє системі фіксувати стан кожної операції на будь-якому етапі її життєвого циклу. Перед завершенням платежу виконується перевірка, що запобігає повторному проведенню тієї ж транзакції:

if payment.status == 'completed':

messages.error(request, \_('Цей платіж вже завершено!'))

return redirect('payments:detail', payment\_id=payment.id)

У разі успішного завершення платежу відбувається автоматичне оновлення пов’язаного об'єкта замовлення, де змінюється поле payment\_status на значення paid. Це гарантує, що подвійна оплата за одне й те саме замовлення неможлива:

def complete(self):

self.status = 'completed'

self.completed\_at = timezone.now()

self.save()

self.order.payment\_status = 'paid'

self.order.save()

Усі суми платежів генеруються автоматично на основі загальної суми замовлення, що мінімізує ризик помилок або маніпуляцій з фінансовими даними. Перед створенням нового платежу система перевіряє, чи не був відповідний рахунок уже оплачений. Логіка повернення також реалізована з перевірками: дозволяється лише для платежів, статус яких completed, а не, наприклад, failed або refunded.

Система зберігає детальну інформацію про кожну транзакцію, включно з унікальним номером, способом оплати, часом завершення та коментарями. Це дозволяє здійснювати повноцінний аудит фінансових подій через адміністративний інтерфейс Django. Усі записи можна фільтрувати за статусом, датою або іншими параметрами. Додатково реалізована можливість генерації фіскального чеку, придатного для друку, що забезпечує відповідність обліковим вимогам закладу.

Для запобігання CSRF-атакам усі форми у шаблонах системи містять CSRF-токени, обробка яких є обов’язковою для POST-запитів:

<form method="post">

{% csrf\_token %}

...

</form>

Безпека транзакцій у Bublydr забезпечується на кількох рівнях: через аутентифікацію користувачів, авторизаційні обмеження, контроль станів об'єктів моделі Payment, валідацію даних при проведенні транзакцій, а також застосування механізмів захисту від зовнішніх атак. У поєднанні ці технічні рішення формують надійний каркас для обробки фінансових операцій у межах ресторанної інформаційної системи.

# 3.7 Тестування системи

У процесі розробки POS-системи Bublydr (системи для обслуговування клієнтів у ресторані) особлива увага була приділена якості виконання програмних компонентів та їх відповідності поставленим вимогам.

Для забезпечення надійності використовувався комплексний підхід до тестування, що включав кілька рівнів перевірки.

Модульне тестування — це перевірка окремих частин програми, наприклад, функцій чи методів, які відповідають за конкретні завдання. Такі тести перевіряють, чи правильно працює кожна маленька частина системи окремо від інших. Для цього використовувався спеціальний інструмент PyTest, який допомагає автоматично перевіряти, чи працює програмний код так, як очікується.

Інтеграційне тестування дозволяє перевірити, як різні частини системи працюють разом. Наприклад, чи правильно взаємодіють між собою модуль замовлень та модуль обліку продуктів. Системне тестування — перевірка роботи всієї системи в цілому, яке виконувалося вручну через інтерфейс користувача. Функціональне тестування визначало, чи відповідає система всім заявленим функціональним вимогам.

Для успішного проходження тестування було визначено такі критерії: правильне виконання бізнес-логіки (наприклад, правильний розрахунок вартості замовлення), стабільна робота системи при різних умовах, швидкий відгук системи та відсутність серйозних помилок.

Нижче наведено приклад тесту, який перевіряє, чи правильно працює функція позначення замовлення як виконаного:

import pytest

from django.utils import timezone

from orders.models import Order, OrderItem

from accounts.models import User

from tables.models import Table

from menu.models import Dish, Category

from inventory.models import Product, Unit

@pytest.mark.django\_db

def test\_mark\_order\_as\_completed\_full():

user = User.objects.create(username='test\_waiter')

table = Table.objects.create(number=5, capacity=4)

unit = Unit.objects.create(name='Кілограм', short\_name='кг')

product = Product.objects.create(

name='Борошно', current\_quantity=100, price\_per\_unit=10, unit=unit

)

category = Category.objects.create(name='Десерти', description='Солодкі страви')

dish = Dish.objects.create(name='Млинець', price=50, category=category)

order = Order.objects.create(waiter=user, table=table)

OrderItem.objects.create(order=order, dish=dish, quantity=2, price=dish.price)

order.mark\_as\_completed()

assert order.status == 'completed'

assert order.completed\_at is not None

assert timezone.now() - order.completed\_at < timezone.timedelta(seconds=5)

for item in order.items.all():

assert item.status == 'completed'

Цей тест створює необхідні об'єкти для імітації реальної ситуації в ресторані: офіціанта, столик, продукти, страви та замовлення. Потім він викликає функцію mark\_as\_completed (позначити як виконане), яка має змінити статус замовлення. Після цього тест перевіряє, чи правильно змінився статус і чи правильно зафіксовано час виконання. Успішне завершення тестування зображено на рисунку 3.9.

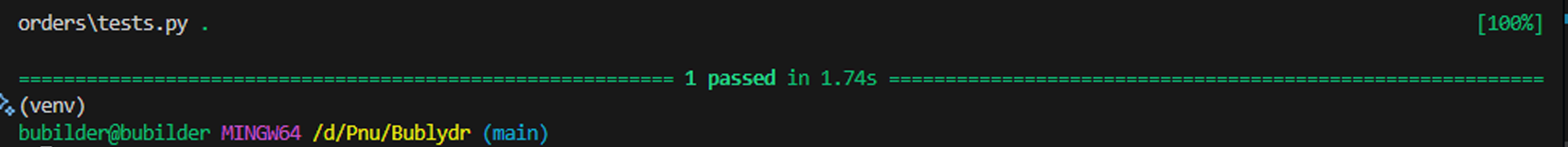


Рисунок 3.9 – Результат проходження тесту в середовищі розробки

Аналіз результатів тестування показав, що перевірені частини системи працюють правильно. Система коректно змінює статуси замовлень, фіксує час їх завершення та працює стабільно як зі звичайними, так і з нестандартними даними.

Окрім автоматичних тестів, було проведено й ручне тестування інтерфейсу, яке підтвердило, що всі процеси (створення замовлень, зміна їх статусів, перегляд інформації про товари) реалізовані правильно. Інтерфейс системи виявився зручним і зрозумілим для користувачів, навіть без спеціальної підготовки.

Загальні результати тестування дозволяють стверджувати, що система працює надійно, відповідає всім поставленим вимогам і готова до використання в реальних ресторанах.

# 3.8 Висновки до третього розділу

У межах третього розділу здійснено детальний аналіз реалізації ключових компонентів системи Bublydr, спрямованих на автоматизацію операцій у ресторанному бізнесі.

Робота охопила функціональні модулі та засоби захисту даних, що забезпечують стійку, масштабовану й безпечну роботу системи під реальним навантаженням.

Архітектура базується на принципах модульності. Бізнес-процеси — керування користувачами, замовленнями, запасами, оплатами — реалізовано як окремі логічні модулі з чіткими межами відповідальності. Це спрощує підтримку, тестування та масштабування.

Важливою особливістю є інтеграція елементів ШІ: реалізовано рекомендаційну систему для персоналізованих пропозицій клієнтам і підказок офіціантам, що підвищує ефективність обслуговування і збільшує середній чек.

Система аутентифікації побудована на розширеній моделі користувача з ролями, хешуванням паролів із криптографічною сіллю та перевірками доступу на рівні представлень і шаблонів.

Для захисту персональних даних реалізовано механізми згідно з вимогами Django: користувач має доступ лише до власного профілю, обробка даних вимагає відповідних дозволів, файли завантажуються через захищене середовище.

Фінансові операції захищено аутентифікацією, перевірками ролей, валідацією, обмеженням змін і аудитом. Критичні дії виконуються через POST-запити з CSRF-захистом.

Загалом система Bublydr демонструє ефективне поєднання модульної архітектури, інтелектуальних алгоритмів і принципів інформаційної безпеки. Це забезпечує не лише автоматизацію, а й потенціал для подальшого розвитку у сфері аналітики та бізнес-інтелекту.

РОЗДІЛ 4 БІЗНЕС-ПЛАН

# 4.1 Планування витрат

У процесі розробки POS системи Bublydr було визначено ключові етапи робіт, кожен із яких потребує відповідного планування ресурсів та фінансових витрат. Першочергово здійснюється аналіз вимог замовника та проєктування архітектури системи, що включає в себе узгодження функціональних модулів, інформаційних потоків і технічних специфікацій. Для виконання цих завдань передбачено близько 40 годин роботи бізнес-аналітика та системного архітектора за середньою ставкою 20 дол. США/год, що забезпечує всебічну підготовку до подальшої розробки та мінімізує ризики масштабних змін на наступних стадіях.

Основна частина витрат припадає на безпосередню розробку програмного коду, створення користувацького інтерфейсу та інтеграцію підсистеми рекомендацій. Загальний обсяг робіт оцінюється в межах 250–300 годин із залученням двох фріланс-розробників із середньою ставкою 18 дол. США/год. У рамках цього етапу реалізуються модулі управління замовленнями, складом, фінансовою звітністю та движок генерації рекомендацій на базі мовної моделі Mistral 7B. Одночасно передбачено 30 годин роботи DevOps-інженера для налагодження безперервної інтеграції й доставки (CI/CD) та автоматизованого тестування, виходячи зі ставки 22 дол. США/год.

Тестування системи складається з поєднання модульних перевірок, інтеграційних сценаріїв та приймального тестування силами як автоматизованих засобів (PyTest), так і ручного тестування інтерфейсів. Орієнтовний обсяг цих робіт становить 100 годин із середньою ставкою фріланс-тестувальника 12 дол. США/год. Такий підхід дозволяє виявити та усунути критичні помилки до стадії вводу в експлуатацію, гарантувати відповідність системи функціональним вимогам та знизити вартість подальшої підтримки.

До витрат впровадження належать налаштування серверного середовища, розгортання бази даних PostgreSQL, конфігурація API-зв’язків із системою рекомендацій через Ollama API та навчання персоналу. На ці задачі передбачено приблизно 50 годин роботи системного адміністратора (15 дол. США/год) і 20 годин корпоративного навчання (витрати на підготовку — умовно 10 дол. США/год за годину тренінгу). Окрім того, до початкових інвестицій належать одноразова покупка доменного імені (15–20 дол. США) та організація базової системи резервного копіювання (200 дол. США). Щомісячні витрати на VPS-сервер середнього класу оцінено в 30 дол. США, що на рік становить близько 360 дол. США.

Таким чином, сумарні одноразові витрати на реалізацію та впровадження системи Bublydr становлять орієнтовно 10 500–12 000 дол. США, а річні операційні витрати — близько 360 дол. США на хостинг та 240 дол. США на оновлення домену й резервні копії. Таке планування забезпечує прозорість бюджету проєкту та дозволяє здійснювати контроль витрат на всіх ключових етапах розробки й експлуатації системи.

# 4.2 Аналіз ринку та очікуваний прибуток

Аналіз ринку впровадження інформаційних систем у сегменті HoReCa підтверджує зростаючий попит на рішення для автоматизації обробки замовлень і управління запасами. Попри поки що невисокий рівень проникнення таких систем в українських закладах громадського харчування, міжнародний досвід демонструє суттєву економію витрат і зростання клієнтської лояльності завдяки використанню інтелектуальних платформ. За даними досліджень, середньорічне зростання цього ринку становить близько 12 % [1], що зумовлено потребою оптимізації операцій на тлі підвищення витрат на персонал та енергоносії.

В Україні та на суміжних ринках Східної Європи основною перешкодою залишається недостатня обізнаність керівників щодо переваг інтегрованих рішень. Водночас заклади, які вже впровадили ПЗ для обліку й управління, демонструють зростання середнього чеку на 8–10 % і скорочення витрат на 15–20 % протягом першого року роботи [2]. Це створює сприятливі умови для позиціонування Bublydr як комплексного рішення з модулем інтелектуальних рекомендацій, що посилює конкурентоспроможність.

Цінова аналітика на міжнародних фріланс-платформах свідчить: базові рішення без рекомендаційного модуля коштують від 1 000 до 3 000 дол. США. Враховуючи локальні особливості, рекомендовано встановити стартову вартість Bublydr на рівні 5 000 грн із підпискою на обслуговування за 1 500 грн/міс. Така модель знижує вхідний бар’єр для малого бізнесу та забезпечує стабільні надходження [14,15].

За консервативними оцінками, прогнозований дохід упродовж перших шести місяців становитиме 300 000–400 000 грн за рахунок укладення 20–30 контрактів із малими та середніми закладами. Щомісячні доходи від супроводу (30 000–45 000 грн) дозволяють очікувати річний прибуток на рівні 600 000–700 000 грн і окупність проєкту вже через 4–5 місяців.

Таким чином, інтеграція інтелектуального модуля в архітектуру системи робить Bublydr привабливим як для закладів, що лише розпочинають цифрову трансформацію, так і для тих, хто вже використовує автоматизовані платформи. Помірна вартість, підтверджені кейси ефективності й потенціал масштабування дозволяють прогнозувати стабільний попит з 2026 року — у тому числі на ринках сусідніх країн.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було спроектовано та реалізовано інформаційну систему Bublydr для автоматизації процесів обробки замовлень і управління запасами в закладах громадського харчування з інтегрованим модулем інтелектуальних рекомендацій на базі мовної моделі Mistral 7B. Система забезпечує повний цикл роботи: від реєстрації замовлення до списання інгредієнтів і формування звітів, при цьому інтерфейс користувача залишає інтуїтивно зрозумілим і доступним як для персоналу фронт-офісу, так і для адміністративного складу.

Порівняно з існуючими аналогами, які зазвичай працюють у хмарних сервісах і потребують постійного підключення до Інтернету, Bublydr функціонує локально або в приватному середовищі замовника. Такий підхід гарантує відсутність ризиків витоку конфіденційних даних, мінімізує простої в разі відсутності з’єднання та знижує експлуатаційні витрати на оренду хмарних ресурсів. Окрім того, модуль рекомендацій працює безпосередньо на стороні замовника через Ollama API, що дозволяє обробляти дані швидше та з меншими затримками, ніж конкурентні рішення з обробкою в публічній хмарі.

Особливою перевагою розробки є інтегрована підсистема формування персоналізованих рекомендацій, що аналізує історію замовлень і пропонує оптимальні «супутні» позиції меню, сприяючи збільшенню середнього чеку. Завдяки модульній архітектурі Bublydr може бути легко адаптовано до різних форматів закладів — від невеликих кав’ярень до мережевих ресторанів — із подальшою опцією масштабування.

Економічна оцінка впровадження системи продемонструвала короткий строк окупності — близько 3–5 місяців за рахунок щомісячної економії на 20–30 % у витратах персоналу, обробці замовлень і оптимізації закупівель. У порівнянні з вітчизняними та міжнародними аналогами, де строк окупності може сягати 9–12 місяців, це рішення є більш вигідним для малих і середніх підприємств. Фактично одноразові витрати на розробку і впровадження компенсуються завдяки підвищенню продуктивності та зниженню операційних витрат.

Отже, розроблена система Bublydr повністю відповідає сучасним вимогам до автоматизації бізнес-процесів у сфері громадського харчування, забезпечує високий рівень надійності та продуктивності, а також створює передумови для стабільного зростання фінансових показників підприємства й подальшого розширення функціоналу за рахунок AI-модулів і нових каналів взаємодії з клієнтами. Систему можна рекомендувати до впровадження як ефективний та економічно вигідний інструмент цифрової трансформації закладів HoReCa.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

REFERENCES

1. Deloitte (2022). The future of restaurants: how technology is transforming the restaurant industry. Deloitte Insights, Jan. 2022. URL: https://www2.deloitte.com/…/future-of-restaurants (дата звернення: 12.05.2025).
2. Toast Inc. (2024). How Restaurants Are Embracing AI to Drive Efficiency in 2025. Toast Inc., 2024. URL: https://pos.toasttab.com/blog/… (дата звернення: 10.05.2025).
3. Global Growth Insights (2023). AI-powered POS analytics can increase sales conversion up to 25%. Global Growth Insights, 2023. URL: https://globalgrowthinsights.com/…/pos (дата звернення: 15.05.2025).
4. Djangostars (2021). Why use Django: top 14 pros of the Django framework. Djangostars Blog, 2021. URL: https://djangostars.com/blog/… (дата звернення: 14.05.2025)
5. Bytebase Blog (2023). MySQL vs PostgreSQL: comparison of architecture and performance. Bytebase, 2023. URL: https://bytebase.com/blog/mysql-vs-postgres (дата звернення: 11.05.2025).
6. Abdulsamad, M. (2021). Bootstrap 5 – The Most Popular CSS Framework for Responsive Design. Medium, 2021. URL: https://medium.com/…/bootstrap-5 (дата звернення: 13.05.2025).
7. OWASP Foundation (2023). Django Security Cheat Sheet. OWASP, 2023. URL: https://cheatsheetseries.owasp.org/…/Django\_Security\_Cheat\_Sheet (дата звернення: 14.05.2025).
8. Django Project (2024). Security in Django. Django Documentation, 2024. URL: https://docs.djangoproject.com/…/security/ (дата звернення: 14.05.2025).
9. GeeksforGeeks (2022). How to design ER diagram for a POS system. GeeksforGeeks, 2022. URL: https://www.geeksforgeeks.org/…/design-er-diagram (дата звернення: 10.05.2025).
10. Zhang Y., Qiu Z., Liao C., Zhou Z., Li T., Wu Z. (2023). Joint AI and Location-Based Restaurant Recommendation with LLMs. Applied Sciences, 13(11), 7095; https://doi.org/10.3390/app13117095.
11. Ollama AI (2024). Mistral – 7B parameters. The new state-of-the-art open model. Ollama AI, 2024. URL: https://ollama.com/mistral (дата звернення: 12.05.2025).
12. Ollama (2024). Ollama – the Lightweight Local LLM Framework. GitHub, 2024. URL: https://github.com/jmorganca/ollama (дата звернення: 13.05.2025).
13. National Restaurant Association (2024). Restaurateurs use tech to manage inventory, save money. National Restaurant Association, 15 Oct 2024. URL: https://restaurant.org/…/restaurateurs-use-tech (дата звернення: 12.05.2025).
14. B2BReviews (2024). Restaurant POS Market Size, Share & Forecast (2023–2031). B2BReviews, 2024. URL: https://www.b2breviews.com/…/restaurant-pos (дата звернення: 15.05.2025).
15. Research and Markets (2024). Foodservice Global Industry Guide 2024: A $4.33 Trillion Market in 2023. Research and Markets, 5 Nov 2024. URL: https://www.globenewswire.com/…/2974684 (дата звернення: 10.05.2025).
16. CareerFoundry (2023). 9 pros and cons of the Django framework. CareerFoundry Blog, Aug. 2023. URL: https://careerfoundry.com/blog/web-development/django-framework-guide (дата звернення: 13.05.2025).
17. Popmenu (2022). 55% of Restaurants Plan to Adopt New Technologies in 2023. RestaurantTechnology News, 8 Dec 2022. URL: https://restauranttechnologynews.com/…/new-technologies-2023 (дата звернення: 12.05.2025).
18. National Restaurant Association (2024). Restaurant Technology Trends Report 2024. National Restaurant Association, 2024. URL: оптимізації витрат і покращення якості обслуговування клієнтів https://restaurant.org/…/technology-trends-2024 (дата звернення: 12.05.2025).
19. Anser, S. (2021). Django ORM — Pros and Cons. Medium, 27 May 2021. URL: https://medium.com/…/django-orm-pros-and-cons (дата звернення: 14.05.2025).
20. Hospitality Technology (2023). 2024 POS Software Trends Study: Next-Level Integration. Hospitality Technology, 2023. URL: https://hospitalitytech.com/2024-pos-study (дата звернення: 12.05.2025).
21. Sabiri, B., et al. (2025). Hybrid Quality-Based Recommender Systems: A Systematic Literature Review. Journal of Imaging, 11(1), 12; https://doi.org/10.3390/jimaging11010012.
22. C4Model (2024). The C4 model for visualising software architecture. C4Model.com, 2024. URL: https://c4model.com/ (дата звернення: 13.05.2025).
23. Capgemini (2023). How Digital is Reinventing the Hospitality Sector. Capgemini Research Institute. URL: https://www.capgemini.com/research/hospitality-digital-transformation (дата звернення: 20.05.2025).
24. Hospitality (2023). The ROI of Restaurant Technology. Oracle White Paper. URL: <https://www.oracle.com/industries/hospitality/roi-restaurant-tech> (дата звернення: 20.05.2025).
25. IBM (2024). How AI and Cloud are Transforming Retail and Hospitality. IBM Insights. URL: https://www.ibm.com/blogs/industries/ai-cloud-hospitality (дата звернення: 20.05.2025).
26. Square, Inc. (2024). Square POS – Features and Benefits for Small Businesses. Square Resources. URL: https://squareup.com/pos/restaurant (дата звернення: 20.05.2025).
27. European Commission (2024). Digital Transformation of SMEs in the Hospitality Sector. EU Digital Economy Reports. URL: <https://digital-strategy.ec.europa.eu/en/library/hospitality-smes> (дата звернення: 20.05.2025).

ДОДАТОК А

Файлова структура проєкту:

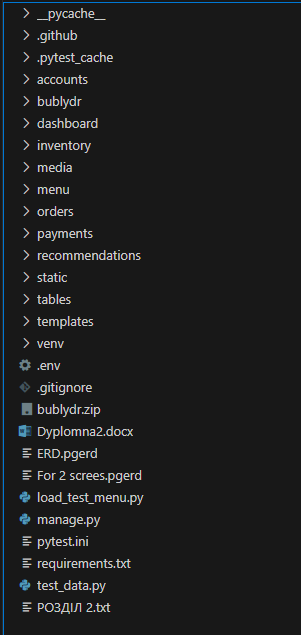


Рисунок А.1 – Файлова структура проєкту

Код файлу manage.py

#!/usr/bin/env python

"""Django's command-line utility for administrative tasks."""

import os

import sys

def main(

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'bublydr.settings')

    try:

        from django.core.management import execute\_from\_command\_line

    except ImportError as exc:

        raise ImportError(

            "Couldn't import Django. Are you sure it's installed and "

            "available on your PYTHONPATH environment variable? Did you "

            "forget to activate a virtual environment?"

        ) from exc

    execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

Код файлу settings.py

import os

from pathlib import Path

from dotenv import load\_dotenv

load\_dotenv()

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

SECRET\_KEY = os.getenv('SECRET\_KEY', 'django-insecure-default-key-change-in-production')

DEBUG = os.getenv('DEBUG', 'False').lower() == 'true'

ALLOWED\_HOSTS = os.getenv('ALLOWED\_HOSTS', 'localhost,127.0.0.1').split(',')

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'crispy\_forms',

    'crispy\_bootstrap5',

    'accounts',

    'menu',

    'tables',

    'orders',

    'payments',

    'inventory',

    'dashboard',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'bublydr.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [BASE\_DIR / 'templates'],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'bublydr.wsgi.application'

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql',

        'NAME': os.getenv('DB\_NAME', 'bublydr'),

        'USER': os.getenv('DB\_USER', 'postgres'),

        'PASSWORD': os.getenv('DB\_PASSWORD', 'postgres'),

        'HOST': os.getenv('DB\_HOST', 'localhost'),

        'PORT': os.getenv('DB\_PORT', '5432'),

    }

}

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',},]

LANGUAGE\_CODE = 'uk-ua'

TIME\_ZONE = 'Europe/Kiev'

USE\_I18N = True

USE\_TZ = True

STATIC\_URL = '/static/'

STATIC\_ROOT = BASE\_DIR / 'staticfiles'

STATICFILES\_DIRS = [BASE\_DIR / 'static']

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / 'media'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

CRISPY\_ALLOWED\_TEMPLATE\_PACKS = "bootstrap5"

CRISPY\_TEMPLATE\_PACK = "bootstrap5"

AUTH\_USER\_MODEL = 'accounts.User'

LOGIN\_REDIRECT\_URL = 'home'

LOGIN\_URL = 'accounts:login'

LOGOUT\_REDIRECT\_URL = 'accounts:login'

Код файлу templates/base.html

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>{% block title %}Bublydr - Ресторанна система{% endblock %}</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css">

</head>

<body>

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

        <div class="container">

            <a class="navbar-brand" href="/">Bublydr</a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav">

                    <li class="nav-item">

                        <a class="nav-link" href="{% url 'home' %}">Головна</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="{% url 'menu:list' %}">Меню</a>

                    </li>

                    {% if user.is\_authenticated %}

                        <li class="nav-item">

                            <a class="nav-link" href="{% url 'orders:list' %}">Замовлення</a>

                        </li>

                        <li class="nav-item">

                            <a class="nav-link" href="{% url 'tables:list' %}">Столики</a>

                        </li>

                        {% if user.is\_staff or user.is\_admin %}

                        <li class="nav-item">

                            <a class="nav-link" href="{% url 'payments:list' %}">Платежі</a>

                        </li>

                        <li class="nav-item">

                            <a class="nav-link" href="{% url 'inventory:dashboard' %}">Запаси</a>

                        </li>

                        <li class="nav-item">

                            <a class="nav-link" href="{% url 'dashboard:home' %}">

                                <i class="bi bi-speedometer2"></i> Панель керування

                            </a>

                        </li>

                        {% endif %}

                    {% endif %}

                </ul>

                <ul class="navbar-nav ms-auto">

                    {% if user.is\_authenticated %}

                    <li class="nav-item dropdown">

                        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown">

                            {% if user.profile\_image %}

                            <img src="{{ user.profile\_image.url }}" class="rounded-circle" width="24" height="24">

                            {% else %}

                            <i class="bi bi-person-circle"></i>

                            {% endif %}

                            {{ user.first\_name|default:user.username }}

                        </a>

                        <ul class="dropdown-menu dropdown-menu-end">

                            <li><a class="dropdown-item" href="{% url 'accounts:profile' %}">Мій профіль</a></li>

                            {% if user.is\_admin %}

                            <li><a class="dropdown-item" href="{% url 'accounts:users\_list' %}">Користувачі</a></li>

                            <li><a class="dropdown-item" href="{% url 'admin:index' %}">Адмін-панель</a></li>

                            <li><hr class="dropdown-divider"></li>

                            {% endif %}

                            <li><a class="dropdown-item" href="{% url 'accounts:logout' %}">Вийти</a></li>

                        </ul>

                    </li>

                    {% else %}

                    <li class="nav-item">

                        <a class="nav-link" href="{% url 'accounts:login' %}">Увійти</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="{% url 'accounts:client\_register' %}">Реєстрація</a>

                    </li>

                    {% endif %}

                </ul>

            </div>

        </div>

    </nav>

    <div class="container mt-4">

        {% block content %}{% endblock %}

    </div>

    <footer class="mt-5 py-3 bg-light">

        <div class="container text-center">

            <p>© 2025 Bublydr - Система управління рестораном</p>

        </div>

    </footer>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>