

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ
УКРАЇНИ
Факультет інформаційних технологій**

ПОГОДЖЕНО
Декан факультету

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних
наук

Ігор БОЛБОТ
(підпис)

_____ **Белла ГОЛУБ**
(підпис)

«___» _____ 2026 р.

«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему: «Інформаційна управляюча система обліку роботи мережі
кав'ярень»**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
д.е.н., професор

(підпис)

Роман РУДЕНСЬКИЙ

**Керівник бакалаврської
кваліфікаційної роботи**
ст. викладач

(підпис)

Віктор ПАНКРАТЬЄВ

Виконала

(підпис)

Маргарита БОРИС

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій**

**ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук**

(назва кафедри)

к.т.н., доцент . Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)

“06” січня 2026 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студентці
Борис Маргариті Юрїївні

Спеціальність 122 – «Комп'ютерні науки»
ОП «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна управляюча система обліку роботи мережі кав'ярень» затверджена наказом ректора НУБіП України від 06.01.2026 №46 «С»
2. Термін подання завершеної роботи на кафедру 2026.05.22
(рік, місяць, число)
3. Вихідні дані до роботи: механізм функціонування інформаційної системи обліку та управління роботою мережі кав'ярень, управління мережею закладів, обліку товарів та інгредієнтів, контролю складських залишків, формування та обробки замовлень клієнтів, управління персоналом, ведення фінансового обліку, зберігання даних про користувачів, продукцію, постачальників і продажі, а також формування статистичної та аналітичної звітності у веб-застосунку.
4. Перелік питань, що розглядаються:
 - Аналіз проблемної області
 - Моделювання предметної області
 - Проектування програмної системи
 - Впровадження та експлуатація системи

Дата видачі завдання “06”січня 2026 р. (дата наказу)

Керівник бакалаврської кваліфікаційної роботи _____ / Панкратьєв В.О. /
(підпис) (прізвище та ініціали)

Завдання прийняла до виконання: _____ / Борис М.Ю. /

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
ВСТУП.....	5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	8
1.1 Опис предметної області.....	8
1.2 Огляд існуючих рішень.....	13
1.3 Постановка завдання.....	17
1.4 Функціональні та нефункціональні вимоги.....	19
1.5 Вимоги до інтерфейсу користувача.....	21
2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ.....	24
2.1 Загальні відомості.....	24
2.2 Об'єктне та функціональне моделювання.....	27
2.3 Абстракції предметної області.....	37
2.4 Діаграма класів.....	39
3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ.....	42
3.1 Логічна модель даних.....	42
3.2 Вибір системи управління базою даних та її реалізація.....	46
3.3 Архітектура програмного забезпечення.....	48
3.4 Організаційна структура програмного забезпечення.....	51
3.5 Вибір інструментарію для створення програмного забезпечення.....	54
4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ.....	57
4.1 Вимоги до апаратного та програмного забезпечення.....	57
4.2 Тестування системи.....	59
4.3 Склад інсталяційного пакету.....	64
ВИСНОВКИ.....	66
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	68
ДОДАТОК А.....	71
ДОДАТОК Б.....	73
ДОДАТОК В.....	75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Скорочення	Розшифрування
API	Application Programming Interface – програмний інтерфейс застосунку
APK	Android Package Kit – інсталяційний пакет Android-застосунку
DBMS	Database Management System – система управління базою даних
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
KYC	Know Your Customer – ідентифікація та верифікація користувача
P2P	Peer-to-Peer – прямий формат взаємодії між користувачами
SDK	Software Development Kit – набір інструментів розробника
SQLite	Вбудована реляційна система управління базою даних
UI	User Interface – інтерфейс користувача
URL	Uniform Resource Locator – адреса ресурсу

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується активним впровадженням цифрових рішень у сферу ресторанного бізнесу та громадського харчування, що зумовлено необхідністю підвищення ефективності управління закладами, автоматизації бізнес-процесів та покращення якості обслуговування клієнтів. Особливої актуальності набувають інформаційні системи обліку та управління мережею кав'ярень, які дозволяють централізовано контролювати діяльність закладів, автоматизувати процеси продажу продукції, вести облік товарів та інгредієнтів, контролювати фінансові операції, а також забезпечувати оперативне отримання статистичної та аналітичної інформації. Використання сучасних веборієнтованих інформаційних систем стало невід'ємною складовою ефективного функціонування мережі кав'ярень, оскільки такі системи забезпечують швидкий доступ до інформації, спрощують процес управління персоналом та дозволяють оптимізувати роботу підприємства загалом.

Однією з головних проблем сучасних закладів громадського харчування є недостатній рівень автоматизації процесів обліку та управління діяльністю мережі. У багатьох випадках облік товарів, контроль складських залишків, ведення фінансової звітності та управління замовленнями виконуються вручну або за допомогою декількох окремих програмних продуктів, що ускладнює централізоване управління даними та підвищує ризик виникнення помилок. Відсутність єдиної інформаційної системи негативно впливає на ефективність роботи закладів, ускладнює процес аналізу продажів та не дозволяє оперативно контролювати діяльність мережі кав'ярень у режимі реального часу.

Актуальність теми бакалаврської роботи полягає у необхідності створення сучасного веб-орієнтованого програмного забезпечення для автоматизації процесів обліку та управління роботою мережі кав'ярень. Реалізація інформаційної системи дозволить забезпечити централізоване

управління даними про товари, замовлення, персонал та фінансові операції, а також підвищити ефективність роботи закладів за рахунок автоматизації основних бізнес-процесів. Крім того, використання сучасних інформаційних технологій у сфері ресторанного бізнесу є перспективним напрямом розвитку цифрових систем управління підприємствами, що активно застосовується у діяльності сучасних мереж громадського харчування.

Метою бакалаврської роботи є покращення процесів обліку та управління діяльністю мережі кав'ярень за допомогою розробки веб-орієнтованого програмного забезпечення, яке забезпечуватиме автоматизацію процесів управління закладами, ведення обліку товарів та інгредієнтів, контролю замовлень клієнтів, управління персоналом, формування фінансової та статистичної звітності, а також централізоване зберігання та обробку даних.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

1. здійснити аналіз предметної області управління мережею кав'ярень та сучасних інформаційних систем автоматизації ресторанного бізнесу;
2. дослідити особливості функціонування систем обліку товарів, замовлень та управління персоналом;
3. визначити функціональні та нефункціональні вимоги до програмного забезпечення;
4. спроектувати архітектуру веб-застосунку та структуру бази даних;
5. реалізувати серверну та клієнтську частини програмного забезпечення;
6. реалізувати механізми обліку товарів, управління замовленнями та формування аналітичної звітності;
7. провести тестування програмного забезпечення та оцінити ефективність його роботи.

Об'єктом дослідження є процес організації обліку та управління діяльністю мережі кав'ярень із використанням інформаційних технологій.

Предметом дослідження є методи та програмні засоби автоматизації процесів обліку, управління замовленнями, контролю товарів та аналізу діяльності мережі кав'ярень.

Під час розробки програмного забезпечення використовуються сучасні методи та технології вебпрограмування, проєктування інформаційних систем та управління базами даних. Серверна частина застосунку реалізується за допомогою мови програмування Python та фреймворку Flask, що дозволяє створити структуровану, масштабовану та безпечну архітектуру програмного забезпечення. Для зберігання та обробки даних використовується реляційна система керування базами даних PostgreSQL. Клієнтська частина програмного забезпечення створюється із застосуванням HTML, CSS та JavaScript для забезпечення зручного та адаптивного користувацького інтерфейсу.

Результати бакалаврської роботи апробовано у тезах доповіді на V Всеукраїнській науково-практичній конференції «Сучасні інформаційні технології та програмне забезпечення», що відбулася у 2026 році.

Структура записки: бакалаврська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 74 сторінок. Список використаних джерел містить 35 посилань. Робота також містить 2 додатки.

У першому розділі наведено аналіз предметної області управління мережею кав'ярень, дослідження сучасних інформаційних систем автоматизації ресторанного бізнесу та особливостей їх функціонування. Другий розділ присвячений проєктуванню програмного забезпечення, моделюванню бази даних та розробці UML-діаграм. У третьому розділі описано процес програмної реалізації вебзастосунку, створення серверної та клієнтської частин системи, а також реалізацію механізмів обліку товарів, управління замовленнями та формування аналітичної звітності. У четвертому розділі наведено результати тестування програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сфера ресторанного бізнесу та громадського харчування є однією з найбільш динамічних галузей сучасної економіки, що активно розвивається під впливом цифрових технологій та автоматизації бізнес-процесів. Особливе місце серед закладів громадського харчування займають мережі кав'ярень, діяльність яких пов'язана з необхідністю постійного контролю товарообігу, обліку продукції, управління персоналом, аналізу фінансових показників та забезпечення високого рівня обслуговування клієнтів. Зростання кількості закладів у межах однієї мережі ускладнює процес управління та потребує використання сучасних інформаційних систем для централізованого контролю діяльності підприємства.

У більшості випадків діяльність мережі кав'ярень охоплює значну кількість взаємопов'язаних процесів, серед яких особливе значення мають управління асортиментом продукції, контроль складських залишків, ведення фінансового обліку, формування замовлень, облік продажів та організація роботи персоналу. Виконання зазначених процесів без використання спеціалізованого програмного забезпечення призводить до збільшення навантаження на працівників, підвищення ризику виникнення помилок та зниження ефективності роботи підприємства [1].

Однією з основних проблем сучасних мереж кав'ярень є недостатній рівень автоматизації управлінських та облікових процесів. Часто різні аспекти діяльності підприємства обробляються окремими програмними засобами або виконуються вручну, що ускладнює централізоване зберігання інформації та оперативний доступ до даних. Відсутність єдиної інформаційної системи може призводити до втрати даних, дублювання інформації, складності контролю фінансових операцій та неефективного управління запасами продукції.

Сучасні інформаційні системи для управління мережею кав'ярень дозволяють автоматизувати більшість бізнес-процесів підприємства та забезпечити централізоване управління всіма закладами мережі. Використання веборієнтованих технологій забезпечує доступ до системи з будь-якого пристрою, підключеного до мережі Інтернет, що значно спрощує процес управління та моніторингу діяльності кав'ярень у режимі реального часу.

Предметна область інформаційної системи обліку та управління роботою мережі кав'ярень включає процеси взаємодії між адміністраторами системи, менеджерами, працівниками закладів та клієнтами. Адміністратор системи здійснює управління користувачами, налаштування прав доступу та контроль функціонування системи. Менеджери мережі відповідають за контроль діяльності закладів, аналіз продажів, управління товарами та формування звітності. Працівники кав'ярень здійснюють оформлення замовлень, ведення обліку продукції та роботу з клієнтами. Клієнти, у свою чергу, формують замовлення та отримують послуги закладу.

Основними функціями інформаційної системи є:

- реєстрація та авторизація користувачів системи;
- управління мережею кав'ярень та окремими закладами;
- ведення обліку товарів, інгредієнтів та складських залишків;
- формування та обробка замовлень клієнтів;
- управління меню та категоріями продукції;
- контроль фінансових операцій та витрат;
- управління персоналом та ролями користувачів;
- формування статистичної та аналітичної звітності;
- централізоване зберігання та обробка даних.

Для реалізації інформаційної системи доцільним є використання сучасних вебтехнологій, що забезпечують масштабованість, безпечність та зручність використання програмного забезпечення. Серверна частина системи реалізується за допомогою Python та фреймворку Flask, які дозволяють створити

гнучку та структуровану архітектуру вебзастосунку. Для зберігання інформації використовується система керування базами даних PostgreSQL, що забезпечує надійне зберігання та ефективну обробку великих обсягів даних.

Таким чином, аналіз предметної області показує, що створення інформаційної системи обліку та управління роботою мережі кав'ярень є актуальним та перспективним завданням. Реалізація такого програмного забезпечення дозволить автоматизувати основні бізнес-процеси підприємства, підвищити ефективність управління мережею закладів, покращити контроль фінансових та товарних операцій, а також забезпечити зручний доступ до аналітичної інформації для прийняття управлінських рішень.

Предметна область даної бакалаврської роботи пов'язана з автоматизацією процесів обліку та управління діяльністю мережі кав'ярень за допомогою сучасних інформаційних технологій. Сфера громадського харчування потребує швидкого обміну інформацією, ефективного контролю ресурсів та оперативного управління бізнес-процесами. У зв'язку з цим використання інформаційних систем є необхідною умовою для забезпечення стабільної та ефективної роботи сучасних закладів [2].

Мережа кав'ярень включає декілька взаємопов'язаних закладів, діяльність яких потребує централізованого контролю та управління. Кожен заклад виконує значну кількість операцій, пов'язаних із продажем продукції, обліком товарів, веденням фінансових операцій, контролем складських залишків та організацією роботи персоналу. При відсутності автоматизованої системи управління більшість цих процесів виконується вручну, що збільшує ризик виникнення помилок та ускладнює контроль діяльності підприємства [2].

Основною метою функціонування інформаційної системи є забезпечення централізованого управління всією мережею кав'ярень та автоматизація основних бізнес-процесів. Система повинна забезпечувати швидкий доступ до інформації про товари, продажі, замовлення, працівників та фінансові показники. Крім того, важливою функцією системи є можливість формування статистичних та аналітичних звітів для оцінки ефективності роботи закладів.

У межах предметної області можна виділити декілька основних категорій користувачів системи. Адміністратор відповідає за налаштування програмного забезпечення, управління користувачами та контроль прав доступу. Менеджери здійснюють управління мережею закладів, аналізують фінансові показники, контролюють товарообіг та формують звітність. Працівники кав'ярень займаються оформленням замовлень, веденням обліку продукції та обслуговуванням клієнтів. Клієнти є безпосередніми споживачами послуг та продукції закладу.

Важливою особливістю предметної області є необхідність постійного оновлення інформації у режимі реального часу. Після оформлення замовлення система повинна автоматично оновлювати дані про продажі та залишки продукції на складі. Це дозволяє уникнути нестачі товарів, забезпечити ефективний контроль ресурсів та спростити процес управління мережею закладів.

Окрему увагу необхідно приділити процесу управління товарами та інгредієнтами. Інформаційна система повинна забезпечувати можливість додавання нової продукції, редагування інформації про товари, ведення категорій продукції та контроль залишків на складах. Автоматизація цих процесів дозволяє значно зменшити навантаження на працівників та підвищити точність обліку.

Не менш важливим елементом предметної області є фінансовий облік та аналіз діяльності мережі кав'ярень. Система повинна забезпечувати можливість ведення історії продажів, контролю прибутків та витрат, а також формування аналітичної звітності для оцінки ефективності діяльності підприємства. Наявність аналітичних інструментів дозволяє керівництву оперативно приймати управлінські рішення та оптимізувати роботу закладів [3].

Для реалізації програмного забезпечення використовуються сучасні вебтехнології та засоби управління базами даних. Серверна частина системи розробляється за допомогою Python та фреймворку Flask, що дозволяє створити гнучку та масштабовану архітектуру вебзастосунку. Для зберігання інформації

використовується PostgreSQL, яка забезпечує надійне зберігання даних та підтримку складних операцій обробки інформації.

Детальний опис класів та атрибутів предметної області наведено у таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	Атрибут	Опис
Кав'ярня	id	унікальний ідентифікатор кав'ярні
	name	назва закладу
	address	адреса розташування кав'ярні
Працівник	id	унікальний ідентифікатор працівника
	fullName	повне ім'я працівника
	position	посада працівника
Товар	id	унікальний ідентифікатор товару
	name	назва товару
	price	вартість товару
Замовлення	id	унікальний ідентифікатор замовлення
	orderDate	дата оформлення замовлення
	totalPrice	загальна вартість замовлення
Позиція замовлення	id	унікальний ідентифікатор позиції замовлення
	quantity	кількість одиниць товару
	price	вартість товару у замовленні

1.2 Огляд існуючих рішень

У сучасних умовах розвитку ресторанного бізнесу автоматизація процесів обліку та управління є важливою складовою ефективного функціонування мережі закладів громадського харчування. Для забезпечення контролю

продажів, обліку товарів, управління персоналом та формування фінансової звітності активно використовуються спеціалізовані інформаційні системи. Існуючі програмні рішення дозволяють автоматизувати основні бізнес-процеси підприємства та забезпечують централізоване управління діяльністю мережі кав'ярень.

Poster POS є сучасною хмарною системою автоматизації закладів громадського харчування, яка використовується у кафе, ресторанах, кав'ярнях та інших підприємствах ресторанного бізнесу. Система призначена для автоматизації процесів продажу, управління товарами, контролю складських залишків, ведення фінансового обліку та формування аналітичної звітності.

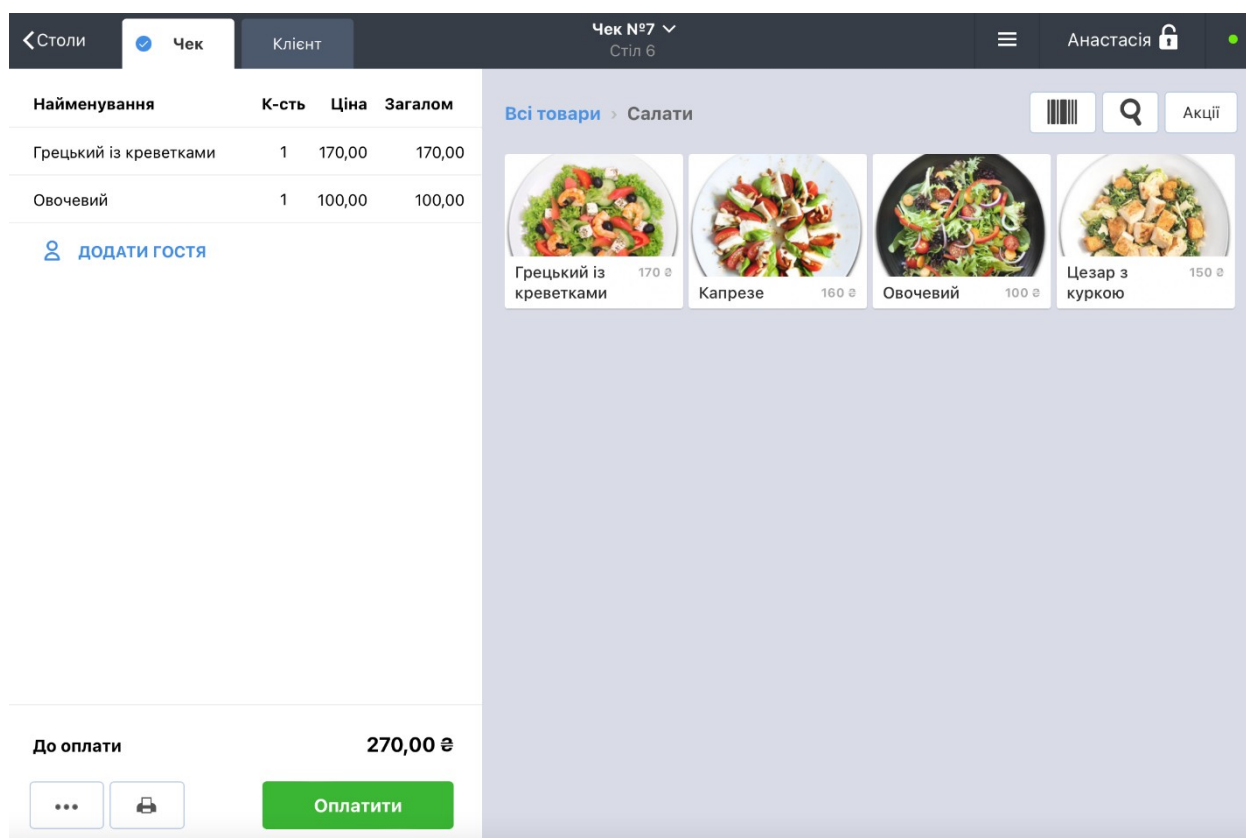


Рисунок 1.1 - Інформаційна система «Poster POS»

Основною метою використання Poster POS є спрощення управління діяльністю закладу та підвищення ефективності виконання щоденних бізнес-процесів. Система дозволяє централізовано контролювати роботу мережі закладів, здійснювати облік продажів, аналізувати фінансові показники та керувати персоналом через єдиний вебінтерфейс [4].

Програмне забезпечення побудоване на основі хмарних технологій, що забезпечує доступ до системи з будь-якого пристрою, підключеного до мережі Інтернет. Це дозволяє власникам та менеджерам закладів контролювати діяльність підприємства у режимі реального часу незалежно від місця перебування. Система підтримує роботу через персональні комп'ютери, планшети, смартфони та POS-термінали.

Poster POS забезпечує функціональні можливості для ведення касових операцій, управління меню, контролю товарів та інгредієнтів, обліку складських залишків і формування фінансових звітів. Важливою особливістю системи є автоматичне оновлення інформації про залишки продукції після здійснення продажів, що дозволяє ефективно контролювати використання товарів та уникати нестачі інгредієнтів.

Однією з переваг системи є простий та зрозумілий користувацький інтерфейс, який спрощує процес навчання персоналу та забезпечує зручність роботи із програмним забезпеченням. Крім того, система підтримує централізоване управління декількома закладами, що є важливим для мережі кав'ярень.

Серед недоліків Poster POS можна виділити залежність від стабільного інтернет-з'єднання, оскільки більшість функцій системи реалізовані через хмарну інфраструктуру. Також окремі функціональні можливості можуть бути доступними лише у платних тарифних планах, що збільшує витрати на використання системи. Додатковим недоліком є обмежена гнучкість налаштування у порівнянні з індивідуально розробленими інформаційними системами.

Таким чином, Poster POS є ефективним програмним рішенням для автоматизації діяльності закладів громадського харчування. Система забезпечує централізоване управління продажами, товарами, персоналом та фінансовими операціями, що дозволяє підвищити ефективність роботи мережі кав'ярень та оптимізувати основні бізнес-процеси підприємства.

Розглянемо ще одне програмне рішення, представлене на ринку електронного навчання.

iiko є комплексною інформаційною системою для автоматизації діяльності закладів громадського харчування, яка широко використовується у ресторанах, кафе, барах та мережах кав'ярень. Система призначена для централізованого управління бізнес-процесами підприємства, автоматизації обліку товарів, контролю фінансових операцій та підвищення ефективності роботи персоналу.

Основною особливістю iiko є комплексний підхід до управління діяльністю закладу. Програмне забезпечення об'єднує функції касового обслуговування, управління складом, контролю продажів, ведення фінансового обліку та аналітики в єдиній інформаційній системі. Це дозволяє забезпечити централізоване управління всіма процесами роботи підприємства та спростити взаємодію між різними підрозділами мережі кав'ярень.

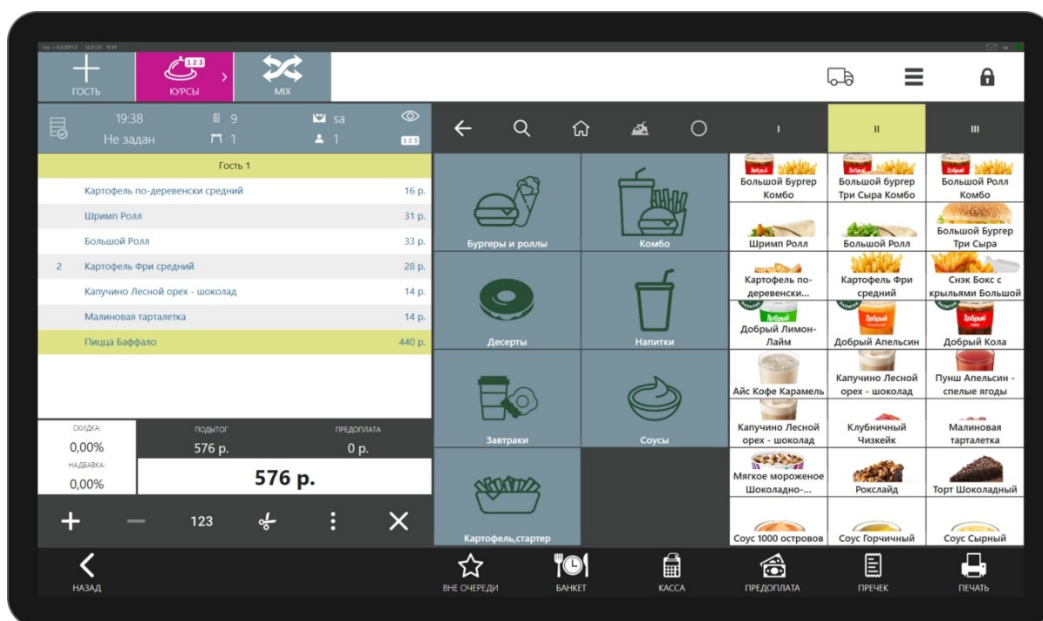


Рисунок 1.2 - Інформаційна система «iiko»

Система підтримує роботу з декількома закладами одночасно, що є важливою перевагою для великих мереж громадського харчування. Менеджери та власники бізнесу мають можливість контролювати продажі, аналізувати фінансові показники та отримувати статистичну інформацію про діяльність закладів у режимі реального часу. Крім того, iiko забезпечує автоматизацію

процесів управління персоналом, включаючи контроль робочого часу працівників та аналіз ефективності їх роботи [5].

Важливою складовою системи є модуль складського обліку, який дозволяє контролювати залишки товарів та інгредієнтів, вести облік поставок та автоматично оновлювати інформацію після виконання продажів. Це забезпечує більш точний контроль використання продукції та дозволяє зменшити фінансові втрати, пов'язані з неправильним обліком товарів.

Програмне забезпечення також підтримує інструменти аналітики та формування звітності. Користувачі можуть отримувати інформацію про обсяг продажів, прибутковість закладів, популярність продукції та інші показники діяльності підприємства. Використання аналітичних інструментів дозволяє оперативно приймати управлінські рішення та підвищувати ефективність роботи мережі кав'ярень.

Однією з переваг ііко є широкий набір функціональних можливостей та підтримка інтеграції з додатковими сервісами, зокрема платіжними системами, програмами лояльності та службами доставки. Це дозволяє адаптувати систему до потреб конкретного підприємства та розширювати функціональність програмного забезпечення.

Разом із перевагами система має певні недоліки. Налаштування та адміністрування ііко потребує додаткових технічних знань, що може ускладнювати процес впровадження програмного забезпечення. Крім того, використання системи може супроводжуватися значними фінансовими витратами, особливо для невеликих кав'ярень та малих підприємств. Деякі функціональні модулі доступні лише у розширених тарифних планах, що також впливає на загальну вартість використання системи.

Таким чином, ііко є потужним програмним рішенням для автоматизації діяльності закладів громадського харчування та мереж кав'ярень. Система забезпечує комплексне управління бізнес-процесами підприємства, автоматизацію обліку та фінансового контролю, а також підтримує аналітичні інструменти для оцінки ефективності діяльності закладів.

Детальний опис порівняння існуючих продуктів наведено у таблиці 1.2.

Таблиця 1.2

Опис порівняння існуючих продуктів

Критерій	Poster POS	iiko	Google Sheets	Розроблюван а ІС
Вартість ліцензії	699+ грн/міс	Недоступно	Безкоштовно	Безкоштовно
Централізація мережі	Так	Так	Ні	Так
Розмежування ролей	Так	Так	Ні	Так (2 ролі)
Автоматична аналітика	Розширена	Розширена	Ручна	Базова
Доступно в Україні	Так	Ні (з 2022)	Так	Так
Без спецобладнання	Ні	Ні	Так	Так
Хмарне розгортання	Так	Ні	Так	Так
Підходить малому бізнесу	Частково	Ні	Частково	Так

Висновок за результатами аналізу: розроблювана система є єдиним варіантом, що одночасно є безкоштовним, доступним в Україні, не потребує спеціального обладнання та забезпечує повноцінне централізоване управління мережею з розмежуванням ролей.

1.3 Постановка завдання

Організація інформаційної системи обліку та управління роботою мережі кав'ярень базується на необхідності забезпечення ефективного управління бізнес-процесами закладів громадського харчування, автоматизації процесів обліку товарів, контролю продажів та координації діяльності персоналу. Система функціонує як централізоване веборієнтоване середовище, у межах якого реалізуються процеси управління мережею закладів, ведення складського та фінансового обліку, оформлення замовлень клієнтів, контролю товарообігу та формування аналітичної звітності.

Основним завданням адміністратора та менеджера у системі є організація та контроль діяльності мережі кав'ярень. Користувачі повинні мати можливість створювати та редагувати інформацію про заклади, управляти меню та товарами, контролювати складські залишки, аналізувати фінансові показники та здійснювати моніторинг продажів. Крім того, система повинна забезпечувати доступ до аналітичної інформації, що характеризує ефективність діяльності підприємства. До таких показників належать обсяг продажів, популярність продукції, рівень витрат, прибутковість закладів та динаміка фінансових показників.

Працівники кав'ярень у межах системи отримують доступ до функцій оформлення замовлень, роботи з касовими операціями, перегляду меню та контролю товарів. Користувачі повинні мати можливість швидко формувати замовлення клієнтів, здійснювати розрахунок вартості продукції та контролювати наявність товарів на складі. Важливою особливістю системи є автоматичне оновлення інформації про залишки продукції після здійснення продажів, що дозволяє забезпечити точний облік товарів та уникнути нестачі інгредієнтів.

Програмний додаток повинен підтримувати централізовану базу даних, у якій зберігається інформація про користувачів, мережу кав'ярень, товари, категорії продукції, замовлення, фінансові операції та результати аналітики. Цілісність та узгодженість даних є важливими вимогами до системи, оскільки програмне забезпечення повинно забезпечувати коректне збереження інформації, уникати втрати даних та запобігати несанкціонованому доступу до конфіденційної інформації.

Основні функціональні можливості системи повинні забезпечувати:

1. створення та адміністрування мережі кав'ярень;
2. управління товарами та категоріями продукції;
3. ведення обліку складських залишків;
4. оформлення та обробку замовлень клієнтів;
5. контроль фінансових операцій та продажів;

6. управління персоналом та ролями користувачів;
7. формування статистичної та аналітичної звітності;
8. моніторинг діяльності закладів у режимі реального часу.

Процес роботи із замовленнями організовується за визначеним життєвим циклом. Після створення замовлення працівником система зберігає інформацію про продукцію, кількість товарів, дату оформлення та загальну вартість замовлення. Після підтвердження замовлення система автоматично оновлює інформацію про складські залишки та фіксує фінансову операцію у базі даних. Усі зміни статусів замовлення та результати фінансових операцій зберігаються у системі та використовуються для подальшого аналізу діяльності мережі кав'ярень.

1.4 Функціональні та нефункціональні вимоги

Функціональні та нефункціональні вимоги є важливою складовою процесу проєктування інформаційної системи обліку та управління роботою мережі кав'ярень. Визначення вимог дозволяє сформулювати перелік функцій програмного забезпечення, встановити характеристики системи та забезпечити відповідність програмного продукту потребам користувачів і особливостям предметної області [7].

Функціональні вимоги визначають основні можливості інформаційної системи та описують процеси, які повинні підтримуватись програмним забезпеченням. Система повинна забезпечувати автоматизацію основних бізнес-процесів мережі кав'ярень, включаючи облік товарів, оформлення замовлень, управління користувачами та формування аналітичної звітності.

До основних **функціональних вимог** системи належать:

1. реєстрація та авторизація користувачів у системі;
2. розмежування прав доступу відповідно до ролей користувачів;
3. створення та редагування інформації про кав'ярні;
4. управління товарами та категоріями продукції;

5. ведення обліку складських залишків;
6. оформлення та обробка замовлень клієнтів;
7. автоматичне оновлення інформації про залишки товарів після продажу;
8. управління персоналом та інформацією про працівників;
9. ведення фінансового обліку та контроль продажів;
10. формування статистичних та аналітичних звітів;
11. пошук та фільтрація інформації у системі;
12. централізоване зберігання даних у базі даних;
13. журналювання основних дій користувачів;
14. адміністрування та налаштування системи.

Окрему увагу під час розробки програмного забезпечення необхідно приділити нефункціональним вимогам, які визначають характеристики якості, надійності, продуктивності та безпечності системи. Нефункціональні вимоги впливають на ефективність використання програмного забезпечення та забезпечують стабільність його роботи.

До основних **нефункціональних вимог** системи належать:

1. забезпечення стабільної роботи вебзастосунку при одночасній роботі декількох користувачів;
2. підтримка швидкого доступу до даних та обробки інформації;
3. забезпечення цілісності та узгодженості даних у базі даних;
4. захист персональних та службових даних користувачів;
5. реалізація механізмів автентифікації та авторизації;
6. забезпечення масштабованості програмного забезпечення;
7. підтримка адаптивного користувацького інтерфейсу;
8. сумісність із сучасними веббраузерами;
9. забезпечення резервного копіювання даних;
10. зручність використання та простота інтерфейсу;
11. можливість подальшого розширення функціональних можливостей системи;

12. підтримка безпечного з'єднання під час передачі даних.

Інформаційна система повинна забезпечувати коректну роботу всіх функціональних модулів та підтримувати централізовану взаємодію між користувачами, товарами, замовленнями та фінансовими операціями. Крім того, програмне забезпечення повинно гарантувати збереження інформації та захист від несанкціонованого доступу до даних.

Для реалізації функціональних та нефункціональних вимог доцільним є використання сучасних технологій веброзробки. Серверна частина системи реалізовується за допомогою Python та фреймворку Flask, що забезпечують створення масштабованого та структурованого вебзастосунку. Для зберігання інформації використовується PostgreSQL, яка забезпечує надійне управління даними та підтримку складних операцій обробки інформації.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача є важливою складовою інформаційної системи обліку та управління роботою мережі кав'ярень, оскільки саме через нього здійснюється взаємодія користувачів із програмним забезпеченням. Від якості реалізації інтерфейсу залежить зручність використання системи, швидкість виконання операцій, ефективність роботи персоналу та загальний рівень продуктивності підприємства. У зв'язку з цим під час проєктування користувацького інтерфейсу необхідно враховувати особливості діяльності мережі кав'ярень, категорії користувачів та специфіку виконання основних бізнес-процесів.

Інтерфейс системи повинен бути зрозумілим, логічно структурованим та простим у використанні для всіх категорій користувачів, включаючи адміністраторів, менеджерів та працівників кав'ярень. Розташування елементів управління повинно забезпечувати швидкий доступ до основних функцій системи та мінімізувати кількість дій, необхідних для виконання операцій. Особливу увагу необхідно приділити зручності оформлення замовлень,

управлінню товарами та роботі з фінансовою інформацією, оскільки саме ці процеси виконуються найбільш часто [8].

Користувацький інтерфейс повинен підтримувати адаптивний дизайн, що забезпечує коректне відображення сторінок на різних типах пристроїв, включаючи персональні комп'ютери, планшети та смартфони. Це дозволить користувачам працювати із системою незалежно від типу пристрою та забезпечить зручний доступ до функціональних можливостей програмного забезпечення у будь-який час.

Однією з важливих вимог до інтерфейсу є забезпечення швидкої навігації між функціональними модулями системи. Для цього необхідно реалізувати зрозуміле меню навігації, систему пошуку та механізми фільтрації інформації. Використання таблиць, форм та інформаційних панелей повинно забезпечувати зручне відображення даних про товари, замовлення, працівників та фінансові операції.

Інтерфейс повинен забезпечувати візуальне розмежування функціональних елементів та підтримувати єдиний стиль оформлення всіх сторінок системи. Використання однакових кольорових схем, шрифтів та елементів управління дозволяє покращити сприйняття інформації та спростити процес роботи користувачів із програмним забезпеченням. Крім того, важливо забезпечити зрозуміле відображення повідомлень про помилки, результатів виконання операцій та системних сповіщень.

Важливою вимогою до інтерфейсу є забезпечення безпечної роботи користувачів із системою. Для цього необхідно реалізувати механізми авторизації, захисту персональних даних та обмеження доступу до окремих функцій відповідно до ролі користувача. Інтерфейс повинен забезпечувати коректне відображення доступних можливостей системи залежно від рівня доступу користувача.

Під час розробки інтерфейсу доцільним є використання сучасних вебтехнологій, що забезпечують високу швидкість роботи, адаптивність та інтерактивність користувацького середовища. Клієнтська частина системи

реалізовується із застосуванням HTML, CSS та JavaScript, що дозволяє створити зручний та сучасний вебінтерфейс для управління діяльністю мережі кав'ярень.

Інтерфейс користувача інформаційної системи повинен забезпечувати зручність використання, швидкий доступ до функціональних можливостей, адаптивність, безпечність та ефективну взаємодію користувачів із програмним забезпеченням. Реалізація якісного користувацького інтерфейсу дозволить підвищити ефективність роботи персоналу та оптимізувати процес управління мережею кав'ярень.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Моделювання предметної області є важливим етапом процесу розробки інформаційної системи обліку та управління роботою мережі кав'ярень. Основною метою моделювання є формування структурованого опису об'єктів системи, визначення взаємозв'язків між ними, а також опис основних бізнес-процесів, які реалізуються у програмному забезпеченні. Використання моделей дозволяє більш детально дослідити особливості функціонування системи, спростити процес проєктування програмного забезпечення та забезпечити коректну реалізацію функціональних можливостей.

Предметна область інформаційної системи охоплює процеси управління мережею кав'ярень, ведення обліку товарів, контролю складських залишків, оформлення замовлень, управління персоналом та формування фінансової й аналітичної звітності. Для ефективної реалізації програмного забезпечення необхідно визначити основні сутності системи, їх атрибути та механізми взаємодії між окремими компонентами [10].

Під час моделювання предметної області особлива увага приділяється опису структури даних та логіки функціонування системи. Це дозволяє визначити склад основних функціональних модулів, встановити взаємозв'язки між користувачами, товарами, замовленнями та фінансовими операціями, а також забезпечити цілісність і узгодженість даних у базі даних.

У межах інформаційної системи можна виділити декілька основних категорій об'єктів предметної області. До них належать користувачі системи, мережа кав'ярень, товари та категорії продукції, замовлення клієнтів, фінансові операції, персонал та складські залишки. Кожен із зазначених об'єктів має власний набір характеристик та функціональне призначення у системі.

Моделювання предметної області виконується із використанням UML-діаграм, які дозволяють графічно представити структуру системи та взаємодію її

компонентів. Використання UML забезпечує формалізований опис програмного забезпечення та спрощує процес розробки інформаційної системи. Зокрема, для моделювання можуть використовуватись діаграми варіантів використання, діаграми класів, діаграми діяльності та інші типи UML-моделей.

Важливим етапом моделювання є визначення ролей користувачів системи та їх функціональних можливостей. Адміністратор системи відповідає за налаштування програмного забезпечення та управління користувачами. Менеджери здійснюють контроль діяльності закладів, аналіз продажів та формування звітності. Працівники кав'ярень виконують оформлення замовлень, ведення касових операцій та роботу з товарами. Розмежування ролей дозволяє забезпечити безпечну та ефективну роботу інформаційної системи.

Під час моделювання також враховуються вимоги до масштабованості, продуктивності та безпечності програмного забезпечення. Інформаційна система повинна забезпечувати стабільну роботу при одночасному використанні декількома користувачами, підтримувати швидкий доступ до даних та гарантувати захист інформації від несанкціонованого доступу.

Для реалізації інформаційної системи використовуються сучасні технології веброзробки. Серверна частина програмного забезпечення створюється за допомогою Python та фреймворку Flask. Для зберігання та обробки інформації використовується PostgreSQL, що забезпечує надійне управління даними та підтримку складних операцій обробки інформації.

UML (Unified Modeling Language) є універсальною мовою моделювання, яка використовується для проєктування, візуалізації та документування програмних систем. UML-діаграми дозволяють графічно представити структуру інформаційної системи, описати взаємодію між її компонентами та відобразити логіку функціонування програмного забезпечення. Використання UML є важливим етапом розробки складних інформаційних систем, оскільки моделювання дозволяє спростити процес проєктування та забезпечити більш зрозумілий опис архітектури програмного забезпечення.

Основною перевагою UML є можливість формалізованого опису різних аспектів системи. За допомогою UML-діаграм можна відобразити структуру класів, взаємодію користувачів із системою, послідовність виконання операцій, логіку бізнес-процесів та взаємозв'язки між окремими компонентами програмного забезпечення. Це дозволяє більш детально дослідити функціонування системи ще на етапі проектування та виявити можливі недоліки архітектури до початку програмної реалізації [11].

У процесі розробки інформаційної системи обліку та управління роботою мережі кав'ярень UML-діаграми використовуються для моделювання структури бази даних, опису ролей користувачів, визначення функціональних можливостей системи та відображення взаємодії між окремими модулями програмного забезпечення. Це дозволяє створити цілісну модель системи та забезпечити узгодженість між функціональними вимогами та програмною реалізацією.

Одним із найбільш поширених типів UML-моделей є діаграма варіантів використання. Вона використовується для опису взаємодії користувачів із системою та визначення основних функцій програмного забезпечення. За допомогою такої діаграми можна відобразити ролі адміністратора, менеджера та працівника кав'ярні, а також функції, доступні кожному користувачу.

Для опису структури даних та взаємозв'язків між сутностями використовується діаграма класів. Вона дозволяє відобразити основні класи предметної області, їх атрибути, методи та зв'язки між об'єктами системи. Діаграма класів є основою для подальшого проектування бази даних та реалізації серверної логіки програмного забезпечення.

Діаграми діяльності застосовуються для моделювання бізнес-процесів та опису послідовності виконання операцій у системі. Вони дозволяють відобразити процес оформлення замовлення, управління товарами, авторизації користувачів або формування звітності. Використання таких діаграм спрощує аналіз логіки функціонування системи та дозволяє більш детально описати алгоритми роботи програмного забезпечення.

Для моделювання взаємодії між компонентами системи також можуть використовуватись діаграми послідовності, які відображають порядок обміну повідомленнями між об'єктами системи у процесі виконання певної операції. Це дозволяє деталізувати процес взаємодії між серверною частиною, базою даних та користувацьким інтерфейсом.

Використання UML-діаграм під час розробки інформаційної системи дозволяє підвищити якість проєктування програмного забезпечення, спростити процес документування системи та забезпечити зрозуміле представлення архітектури програмного продукту. Крім того, UML-моделі спрощують процес командної розробки та подальшого супроводу програмного забезпечення.

2.2 Об'єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма прецедентів є одним із основних типів UML-діаграм, який використовується для опису функціональних можливостей інформаційної системи та взаємодії користувачів із програмним забезпеченням. Основним призначенням діаграми прецедентів є відображення ролей користувачів системи та функцій, які вони можуть виконувати у межах програмного продукту [12].

На діаграмі прецедентів користувачі системи відображаються у вигляді акторів, а функціональні можливості системи — у вигляді прецедентів. Актор являє собою роль, яку виконує користувач або зовнішня система під час взаємодії із програмним забезпеченням. Прецедент описує окрему функцію або сценарій використання системи, наприклад авторизацію користувача, оформлення замовлення чи формування звітності.

Основною метою використання діаграми прецедентів є визначення функціональних вимог до програмного забезпечення та формування загального уявлення про роботу системи. Така діаграма дозволяє візуально представити, які дії можуть виконувати користувачі та яким чином вони взаємодіють із системою.

У межах інформаційної системи обліку та управління роботою мережі кав'ярень діаграма прецедентів використовується для опису взаємодії адміністратора, менеджера та працівника кав'ярні із програмним забезпеченням. Наприклад, адміністратор може здійснювати управління користувачами та налаштування системи, менеджер — контролювати продажі, формувати звітність та управляти товарами, а працівник кав'ярні — оформлювати замовлення та працювати з касовими операціями.

Діаграма прецедентів дозволяє визначити межі системи та встановити перелік основних функцій програмного забезпечення. Крім того, вона спрощує процес аналізу вимог до системи та забезпечує більш зрозуміле представлення функціональних можливостей програмного продукту для розробників і користувачів.

До основних елементів діаграми прецедентів належать актори, прецеденти, межа системи та зв'язки між елементами. Зв'язки використовуються для відображення взаємодії між користувачами та функціями системи, а також для опису залежностей між окремими прецедентами.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

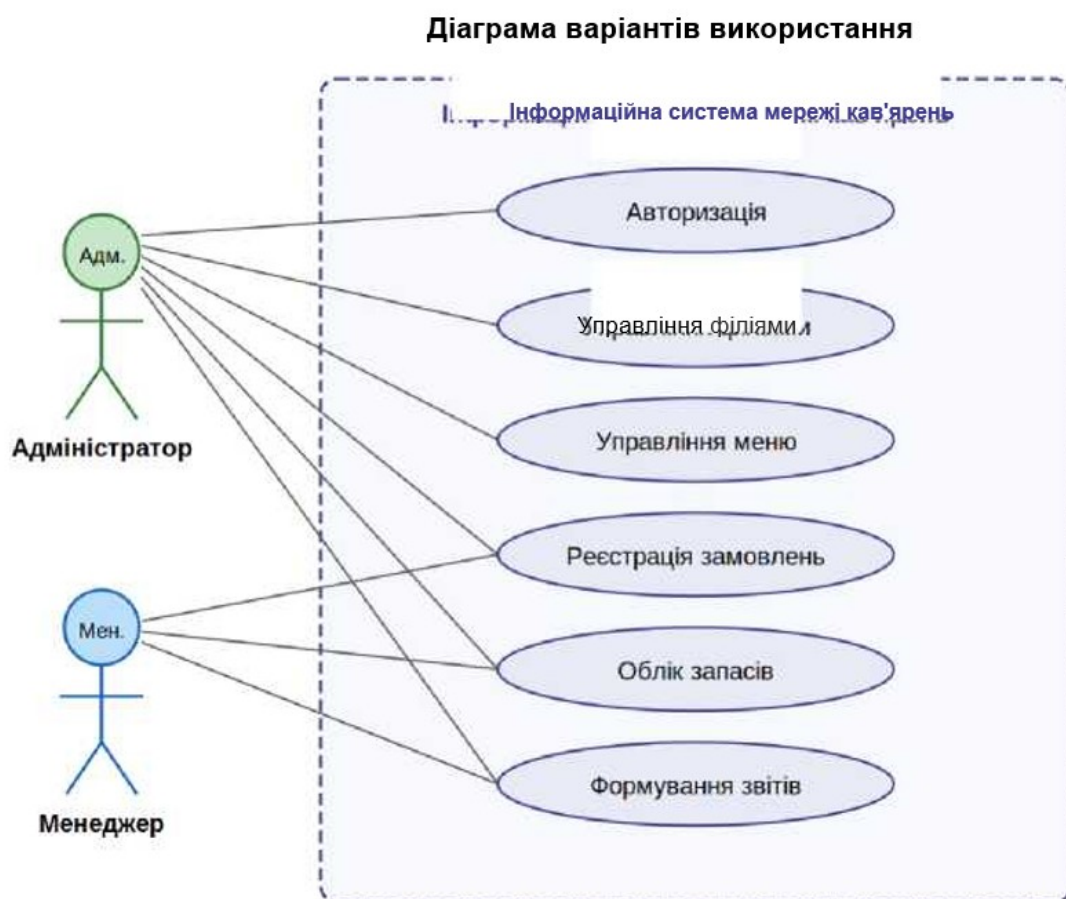


Рис. 2.1 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Студент”;

- “Викладач”.

Прецеденти певним чином залежать одне від одного.

Варіанти використання системи деталізовано нижче. Варіант «Авторизація» є базовим і передує всім іншим — незареєстрований користувач автоматично перенаправляється на сторінку входу. Варіант «Управління філіями» доступний лише Адміністратору і включає CRUD-операції: перегляд списку, додавання нової філії, редагування та видалення. Варіант «Управління меню» також доступний лише Адміністратору: редагування складу меню, управління категоріями, встановлення цін та собівартості. Варіант «Реєстрація замовлень» доступний обом ролям, але Менеджер може реєструвати замовлення лише для своєї філії. Варіант «Облік запасів» аналогічно доступний обом ролям з обмеженням по філії для Менеджера. Варіант «Формування звітів» генерує аналітику за вибраний часовий діапазон та філію.

Таким чином, система реалізує принцип рольового розмежування доступу (RBAC — Role-Based Access Control): кожна роль має чітко визначений набір дозволів. Адміністратор бачить дані всіх філій, Менеджер — лише своєї. Це забезпечує конфіденційність та безпеку операційних даних кожної точки мережі.

Розглянемо детальніше вищеописані прецеденти.

Назва випадку використання: зробити замовлення

Актор: працівник кав’ярні (касир)

Передумови: працівник повинен бути авторизований у системі. У системі повинні бути доступні товари (позиції меню), які можна додати до замовлення. Кав’ярня повинна бути активною та доступною для оформлення продажів.

Основний потік:

1. працівник переходить до розділу оформлення замовлень у системі;
2. система відображає перелік доступних товарів (меню) із цінами та категоріями;
3. працівник обирає необхідні товари для замовлення;

4. працівник вказує кількість кожного обраного товару;
5. система автоматично розраховує проміжну та загальну вартість замовлення;
6. працівник за потреби редагує склад замовлення (додає або видаляє позиції);
7. працівник підтверджує створення замовлення натисканням кнопки «Оформити замовлення»;
8. система перевіряє наявність товарів на складі та коректність введених даних;
9. система зберігає інформацію про замовлення у базі даних;
10. система автоматично оновлює залишки товарів на складі;
11. система фіксує фінансову операцію, пов'язану із замовленням;
12. система відображає повідомлення про успішне створення замовлення.

Альтернативні потоки:

1. якщо працівник не обрав жодного товару, система відображає повідомлення про помилку та не дозволяє створити замовлення;
2. якщо кількість товару перевищує доступні залишки на складі, система повідомляє про недостатню кількість продукції та пропонує змінити замовлення;
3. якщо під час оформлення замовлення працівник скасовує дію, система очищає поточне замовлення та повертає користувача до меню без збереження даних;
4. якщо під час збереження замовлення виникає помилка з'єднання з базою даних або сервером, система відображає повідомлення про технічну помилку та пропонує повторити спробу пізніше.

У даному сценарії описано процес оформлення замовлення в інформаційній системі обліку та управління роботою мережі кав'ярень. Сценарій відображає взаємодію працівника із системою під час формування замовлення,

розрахунку вартості, перевірки наявності товарів, збереження даних та оновлення складських залишків.

Розглянемо інший сценарій.

Назва випадку використання: створити звіт про продажі

Актор: менеджер мережі кав'ярень

Передумови: користувач повинен бути авторизований у системі та мати права доступу до модуля аналітики та звітності. У системі повинні бути наявні дані про продажі за відповідний період, сформовані на основі оформлених замовлень.

Основний потік:

1. менеджер переходить до розділу аналітики та звітності у системі;
2. система відображає доступні типи звітів;
3. менеджер обирає звіт про продажі;
4. система відкриває форму налаштування параметрів звіту;
5. менеджер задає період формування звіту (дата початку та дата завершення);
6. менеджер за потреби обирає додаткові параметри, такі як кав'ярня або група товарів;
7. система перевіряє коректність введених параметрів;
8. менеджер підтверджує формування звіту;
9. система здійснює вибірку даних про продажі з бази даних;
10. система обробляє отримані дані та формує підсумкову інформацію;
11. система відображає сформований звіт про продажі у вигляді таблиці та графічної аналітики;
12. менеджер має можливість переглянути, завантажити або експортувати звіт.

Альтернативні потоки:

1. якщо менеджер не вказав обов'язкові параметри звіту, система відображає повідомлення про помилку та запитує їх введення;

2. якщо вказаний період є некоректним (дата початку пізніше дати завершення), система повідомляє про помилку та не дозволяє сформувати звіт;
3. якщо за вказаний період відсутні дані про продажі, система відображає повідомлення про відсутність інформації та пропонує змінити параметри;
4. якщо під час формування звіту виникає помилка з'єднання з базою даних або сервером, система повідомляє про технічну помилку та пропонує повторити спробу пізніше.

У даному сценарії описано процес формування звіту про продажі в інформаційній системі обліку та управління роботою мережі кав'ярень. Сценарій демонструє взаємодію менеджера із системою під час вибору параметрів звіту, обробки даних про продажі та формування аналітичної інформації для подальшого прийняття управлінських рішень.

2.2.2 Діаграма станів. Діаграма станів (state diagram) — це один із типів UML-діаграм, який використовується для моделювання поведінки об'єкта в системі залежно від його поточного стану та подій, що відбуваються. Вона описує, як об'єкт переходить з одного стану в інший у відповідь на певні події, умови або дії користувача.

Основною метою діаграми станів є відображення життєвого циклу об'єкта предметної області. На відміну від діаграм, що описують структуру системи, діаграма станів зосереджується на динамічній поведінці та зміні станів об'єкта протягом його існування. Це дозволяє більш детально зрозуміти логіку роботи системи та правила переходів між різними етапами процесу [13].

У діаграмі станів основними елементами є стани, переходи, початковий та кінцевий стан. Стан характеризує певну ситуацію або умову, в якій перебуває об'єкт у конкретний момент часу. Перехід відображає зміну стану, яка відбувається під впливом події, наприклад дії користувача або системного сигналу. Початковий стан визначає стартову точку життєвого циклу об'єкта, а кінцевий стан означає завершення його існування або процесу.

У межах інформаційної системи обліку та управління роботою мережі кав'ярень діаграми станів можуть використовуватись для моделювання життєвого циклу замовлення, товару, користувача або фінансової операції. Наприклад, замовлення може перебувати у станах «створено», «в обробці», «оплачено», «виконано» або «скасовано». Перехід між цими станами залежить від дій працівників та автоматичних процесів системи.

Використання діаграми станів дозволяє чітко описати правила зміни станів об'єктів, виявити можливі помилки в логіці бізнес-процесів та забезпечити коректну реалізацію алгоритмів у програмному забезпеченні. Крім того, така діаграма допомагає розробникам краще зрозуміти поведінку системи та забезпечити узгодженість між вимогами та реалізацією.

Таким чином, діаграма станів є важливим інструментом моделювання динамічної поведінки інформаційної системи, який дозволяє описати життєвий цикл об'єктів та процеси переходу між різними станами у системі управління мережею кав'ярень.

Діаграма станів, представлена на Рис. 2.2, містить наступні об'єкти:

- “new”;
- “in_progress”;
- “done”;
- “cancelled”.

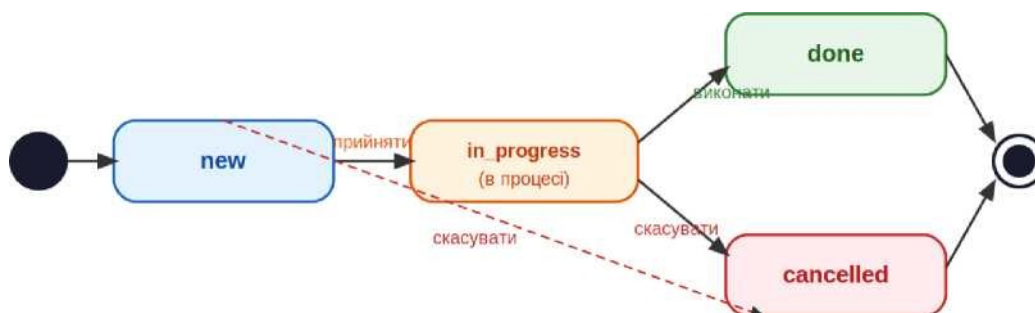


Рисунок 2.2 Діаграма послідовності

Машина станів замовлення реалізована у класі Order через константу `ALLOWED_TRANSITIONS` — словник Python, де ключем є поточний стан, а значенням — список допустимих переходів. Метод `can_transition_to(new_status)` повертає `True` лише якщо перехід є допустимим. Ця перевірка виконується у

контролері перед зміною статусу, що унеможливило некоректні переходи (наприклад, done — new). Тип ENUM PostgreSQL забезпечує ще один рівень захисту на рівні СУБД.

2.2.3 Діаграма активності. Діаграма активності є одним із типів UML-діаграм, яка використовується для моделювання бізнес-процесів та алгоритмів виконання дій у межах інформаційної системи. Вона відображає послідовність виконання операцій, логіку переходів між діями, а також умови, за яких ці переходи відбуваються. Основною метою діаграми активності є опис потоків робіт (workflow) у системі та деталізація процесів, що виконуються користувачами або автоматичними компонентами програмного забезпечення [13].

На відміну від діаграми станів, яка описує життєвий цикл одного об'єкта, діаграма активності фокусується на послідовності дій у межах певного процесу. Вона дозволяє візуалізувати алгоритм виконання задачі від початку до завершення, включаючи розгалуження, цикли та паралельні процеси. Це робить її особливо корисною для опису складних бізнес-процедур у інформаційних системах.

Основними елементами діаграми активності є дії, переходи, початковий та кінцевий вузли, а також умовні розгалуження та паралельні потоки. Дія представляє собою окремий крок процесу, який виконується системою або користувачем. Перехід визначає напрямок руху процесу між діями. Розгалуження використовується для опису умовних сценаріїв виконання, а паралельні потоки дозволяють моделювати одночасне виконання декількох процесів.

У межах інформаційної системи обліку та управління роботою мережі кав'ярень діаграми активності можуть використовуватись для опису таких процесів, як оформлення замовлення, обробка продажів, управління складськими залишками, формування звітності та авторизація користувачів. Наприклад, процес оформлення замовлення може включати вибір товарів,

розрахунок вартості, перевірку наявності на складі, збереження замовлення та оновлення даних у базі.

Використання діаграми активності дозволяє більш наочно представити логіку роботи системи, спростити аналіз бізнес-процесів та виявити можливі недоліки у алгоритмах ще на етапі проєктування. Крім того, вона сприяє кращому розумінню взаємодії між користувачами та системою, а також допомагає у подальшій реалізації програмного забезпечення.

Таким чином, діаграма активності є важливим інструментом моделювання інформаційних систем, який дозволяє описати послідовність виконання процесів та забезпечити структуроване представлення бізнес-логіки системи обліку та управління роботою мережі кав'ярень.

Розроблена діаграма активності представлена на рис. 2.3



Рис. 2.3 Діаграма активності

Діаграма діяльності (Activity Diagram) моделює поведінку системи у вигляді потоку керування між діями. На рис. 2.3 зображено процес реєстрації

нового замовлення — один із найчастіших та найважливіших процесів системи. Початковий вузол (чорна точка) позначає старт процесу. Дія «Вхід до системи» передує всім операціям завдяки декоратору `@login_required`. Ромб «`qty > 0?`» є вузлом рішення: якщо жодної позиції не обрано — потік повертається назад для повторного заповнення; якщо хоча б одну позицію обрано — система автоматично розраховує підсумки та зберігає замовлення. Кінцевий вузол позначає завершення процесу.

2.3 Абстракції предметної області

Абстракції предметної області інформаційної системи обліку та управління роботою мережі кав'ярень являють собою формалізований опис основних сутностей, процесів та взаємозв'язків, які існують у межах досліджуваної системи. Використання абстракцій дозволяє спростити складну предметну область, виділивши її ключові компоненти та відокремивши суттєві характеристики від другорядних деталей реалізації.

У межах розроблюваної інформаційної системи основними абстракціями є користувачі системи, мережа кав'ярень, товари, категорії продукції, замовлення, позиції замовлень, складські залишки та фінансові операції. Кожна з цих абстракцій відображає окремий аспект діяльності підприємства та виконує визначену роль у загальній структурі системи.

Абстракція «користувач» використовується для опису всіх осіб, які взаємодіють із системою, включаючи адміністраторів, менеджерів та працівників кав'ярень. Вона дозволяє узагальнити спільні характеристики користувачів, такі як особисті дані, облікові дані для входу та роль у системі, що визначає рівень доступу до функціональних можливостей.

Абстракція «кав'ярня» описує окремий заклад у межах мережі та містить інформацію, необхідну для його ідентифікації та управління. Вона використовується для організації структури мережі закладів і дозволяє

здійснювати централізоване управління діяльністю кожної кав'ярні окремо та всієї мережі в цілому.

Абстракція «товар» відображає продукцію, яка реалізується у кав'ярні. Вона включає основні характеристики товару, такі як назва, ціна та кількість на складі. Ця абстракція є базовою для реалізації процесів продажу та управління складськими залишками.

Абстракція «замовлення» є однією з ключових у системі та використовується для опису процесу покупки товарів клієнтами. Вона об'єднує інформацію про дату створення замовлення, його загальну вартість та статус виконання, а також пов'язує замовлення з конкретною кав'ярнею та користувачем.

Абстракція «позиція замовлення» деталізує структуру замовлення та визначає конкретні товари, які входять до його складу, включаючи їх кількість та вартість. Вона забезпечує можливість гнучкого формування замовлень із різних товарів та точного обліку продажів.

Абстракція «складські залишки» використовується для відображення кількості товарів, доступних у конкретній кав'ярні. Вона є важливим елементом системи управління ресурсами та дозволяє контролювати наявність продукції, своєчасно оновлювати запаси та уникати дефіциту товарів.

Абстракція «фінансова операція» описує всі грошові транзакції, що відбуваються в системі, включаючи продажі та інші фінансові дії. Вона дозволяє здійснювати облік доходів та витрат, а також формувати аналітичну інформацію для оцінки ефективності діяльності мережі кав'ярень.

Абстракція «замовлення»

Абстракція «замовлення» є ключовою сутністю інформаційної системи обліку та управління роботою мережі кав'ярень, оскільки вона відображає процес оформлення продажу товарів клієнтам. Вона об'єднує інформацію про склад покупки, її вартість, дату створення та поточний статус виконання, а також забезпечує зв'язок між користувачем, кав'ярнею та товарами.

Основні характеристики абстракції «замовлення»:

- ідентифікація кожного замовлення в системі за унікальним ідентифікатором;
- фіксація дати та часу створення замовлення;
- зберігання загальної вартості замовлення;
- визначення статусу виконання замовлення (створено, оплачено, виконано, скасовано);
- зв'язок із конкретною кав'ярнею та працівником, який його оформив.

Абстракція «товар»

Абстракція «товар» описує продукцію, яка реалізується в межах мережі кав'ярень, та є основою для формування меню і обліку продажів. Вона використовується для зберігання інформації про доступні позиції, їх характеристики та вартість, а також для контролю складських залишків.

Основні характеристики абстракції «товар»:

- унікальна ідентифікація товару в системі;
- зберігання назви та опису товару;
- визначення ціни одиниці продукції;
- прив'язка товару до певної категорії;
- контроль кількості доступного товару на складі.

Використання абстракцій предметної області дозволяє формалізувати структуру інформаційної системи, виділити ключові сутності та забезпечити основу для подальшого проєктування бази даних і програмної реалізації системи обліку та управління роботою мережі кав'ярень.

2.4 Діаграма класів

Діаграма класів є однією з основних UML-діаграм, яка використовується для відображення структури інформаційної системи та опису взаємозв'язків між основними сутностями предметної області. Вона дозволяє формалізувати модель

даних системи, визначити класи, їх атрибути, а також зв'язки між об'єктами, що забезпечує цілісне уявлення про архітектуру програмного забезпечення [14].

У межах інформаційної системи обліку та управління роботою мережі кав'ярень діаграма класів відображає основні сутності, які беруть участь у процесах продажу, управління товарами, обліку замовлень та формування фінансової звітності. До таких сутностей належать користувачі системи, мережа кав'ярень, товари, категорії продукції, замовлення, позиції замовлень, складські залишки та фінансові операції.

Клас «User» використовується для зберігання інформації про користувачів системи, включаючи їх особисті дані та роль у системі, яка визначає рівень доступу до функціональних можливостей. Клас «CoffeeShop» описує окремі заклади мережі та містить інформацію про їх назву та адресу.

Клас «Product» відображає товари, які реалізуються у кав'ярнях, та включає основні характеристики продукції, такі як назва та ціна. Клас «Category» використовується для групування товарів за певними ознаками та забезпечує структуроване представлення меню.

Клас «Order» є центральним елементом системи та описує процес оформлення замовлення. Він містить інформацію про дату створення замовлення, його загальну вартість та статус виконання. Для деталізації складу замовлення використовується клас «OrderItem», який містить інформацію про конкретні товари, їх кількість та ціну у межах замовлення.

Клас «Inventory» відповідає за облік складських залишків і дозволяє контролювати кількість товарів у кожній кав'ярні. Клас «FinancialOperation» використовується для фіксації фінансових транзакцій, пов'язаних із продажами та іншими операціями, що забезпечує можливість ведення фінансового обліку та аналітики.

Між класами встановлено взаємозв'язки, які відображають логіку функціонування системи. Користувач може створювати замовлення, кожне замовлення містить одну або декілька позицій, а кожна позиція пов'язана з конкретним товаром. Товари належать до певних категорій, а також пов'язані зі

складськими залишками. Кожне замовлення може генерувати фінансову операцію, що дозволяє відслідковувати доходи системи.

Для цього проєкту було створено спеціальну діаграму класів з використанням об'єктно-орієнтованого підходу (рис. 2.4).

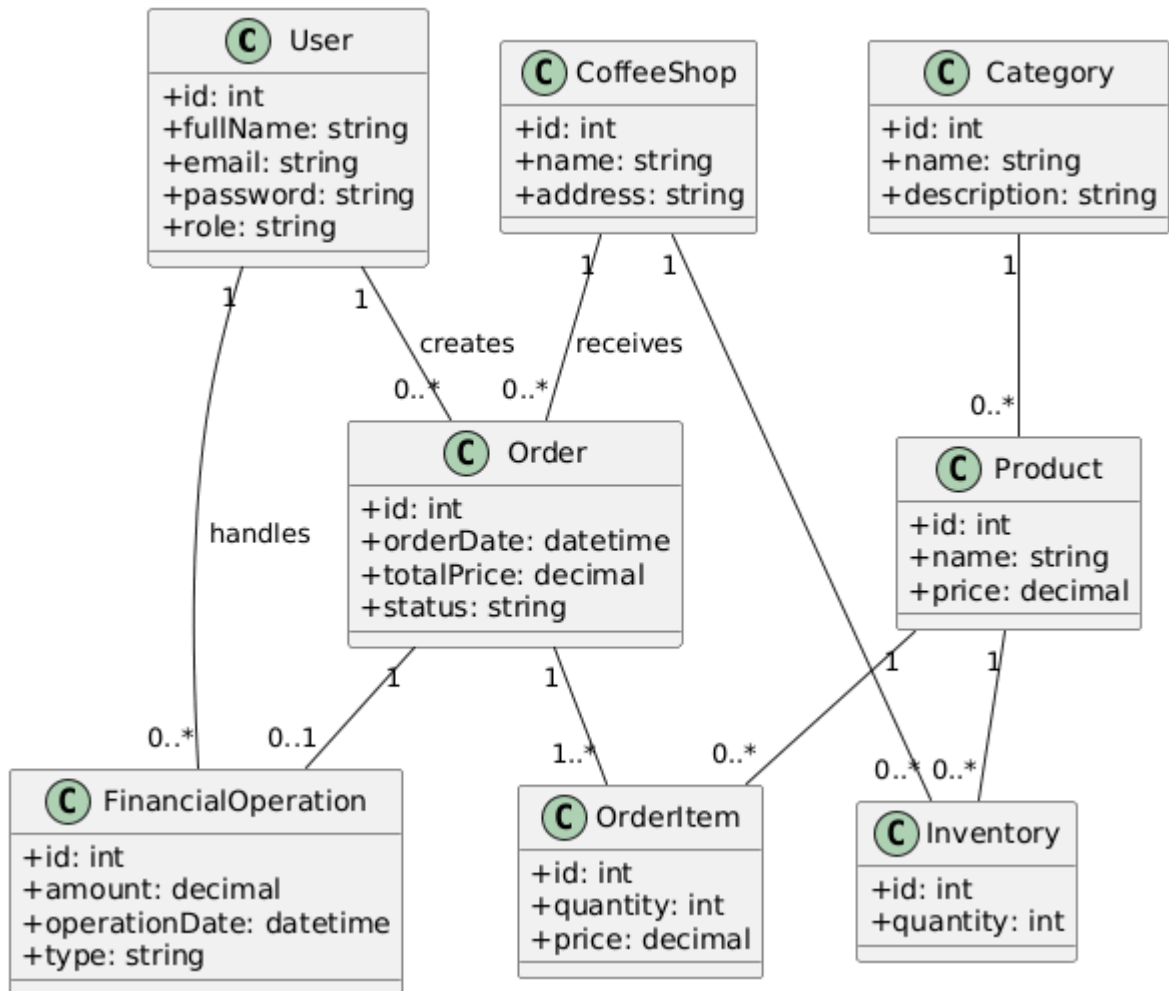


Рис. 2.4 Діаграма класів

Таким чином, діаграма класів забезпечує формалізований опис структури інформаційної системи, дозволяє визначити основні сутності та їх взаємозв'язки, а також є основою для подальшого проєктування бази даних і реалізації програмного забезпечення.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних інформаційної системи обліку та управління роботою мережі кав'ярень відображає структуру даних без прив'язки до конкретної фізичної реалізації бази даних. Вона визначає основні сутності предметної області, їх атрибути та взаємозв'язки, що дозволяє сформулювати цілісне уявлення про організацію зберігання інформації у системі [16].

Логічна модель даних розробляється з метою забезпечення узгодженості інформаційних потоків, мінімізації надлишковості даних та підвищення ефективності їх обробки. Вона є основою для подальшого проєктування фізичної структури бази даних у середовищі PostgreSQL, що використовується для реалізації системи.

У межах логічної моделі даних виділяються основні сутності, які відображають ключові об'єкти предметної області. До них належать користувачі системи, мережа кав'ярень, товари, категорії продукції, замовлення, позиції замовлень, складські залишки та фінансові операції. Кожна сутність має набір атрибутів, що описують її характеристики, а також зв'язки з іншими сутностями системи.

ER-діаграма (Entity-Relationship Diagram) — це інструмент моделювання бази даних, який використовується для графічного відображення структури даних предметної області та зв'язків між її основними сутностями. Вона

дозволяє формалізувати логічну модель даних і є одним із ключових етапів проєктування реляційних баз даних.

Основною метою ER-діаграми є опис того, які сутності існують у системі, які атрибути вони мають, а також як ці сутності пов'язані між собою. Завдяки цьому можна чітко визначити структуру майбутньої бази даних та уникнути дублювання інформації або логічних помилок на етапі реалізації.

У ER-діаграмі використовуються три основні типи елементів: сутності, атрибути та зв'язки. Сутність представляє реальний або абстрактний об'єкт предметної області, наприклад користувача, товар або замовлення. Атрибути описують властивості сутності, такі як назва, ціна або дата створення. Зв'язки визначають взаємодію між сутностями та показують, як об'єкти пов'язані між собою у межах системи [17].

У межах інформаційної системи обліку та управління роботою мережі кав'ярень ER-діаграма використовується для моделювання взаємозв'язків між користувачами, кав'ярнями, товарами, замовленнями, категоріями продукції, складськими залишками та фінансовими операціями. Наприклад, один користувач може створювати багато замовлень, кожне замовлення містить декілька позицій, а кожна позиція пов'язана з конкретним товаром.

Використання ER-діаграми дозволяє забезпечити коректне проєктування бази даних, визначити структуру таблиць та їх зв'язки, а також підвищити ефективність зберігання та обробки даних у системі. Крім того, вона є важливим інструментом комунікації між розробниками та допомагає узгодити логіку роботи системи на етапі проєктування.

Сутність «User» використовується для зберігання даних про користувачів системи, включаючи їх особисту інформацію та роль. Сутність «CoffeeShop» описує окремі заклади мережі та дозволяє організувати їх централізоване управління. Сутність «Product» відповідає за зберігання інформації про товари, які реалізуються у кав'ярнях, тоді як «Category» забезпечує їх логічну класифікацію.

Центральною сутністю системи є «Order», яка відображає процес оформлення покупки. Вона пов'язана з сутністю «OrderItem», що деталізує склад замовлення та містить інформацію про кількість і вартість товарів. Така структура дозволяє гнучко формувати замовлення та забезпечує точний облік продажів.

Сутність «Inventory» використовується для контролю складських залишків і відображає кількість товарів у конкретній кав'ярні. Сутність «FinancialOperation» забезпечує фіксацію всіх фінансових транзакцій, пов'язаних із продажами, що дозволяє здійснювати подальший аналіз прибутковості та ефективності діяльності закладів.

Взаємозв'язки між сутностями логічної моделі даних визначають правила взаємодії між об'єктами системи. Один користувач може створювати декілька замовлень, кожне замовлення містить одну або декілька позицій, а кожна позиція пов'язана з конкретним товаром. Товари належать до певної категорії та пов'язані зі складськими залишками, що забезпечує контроль їх кількості. Кожне замовлення може породжувати фінансову операцію, яка фіксує рух коштів у системі.

Логічна модель системи представлена на рисунку 3.1

ER-діаграма бази даних (логічна модель, ЗНФ)

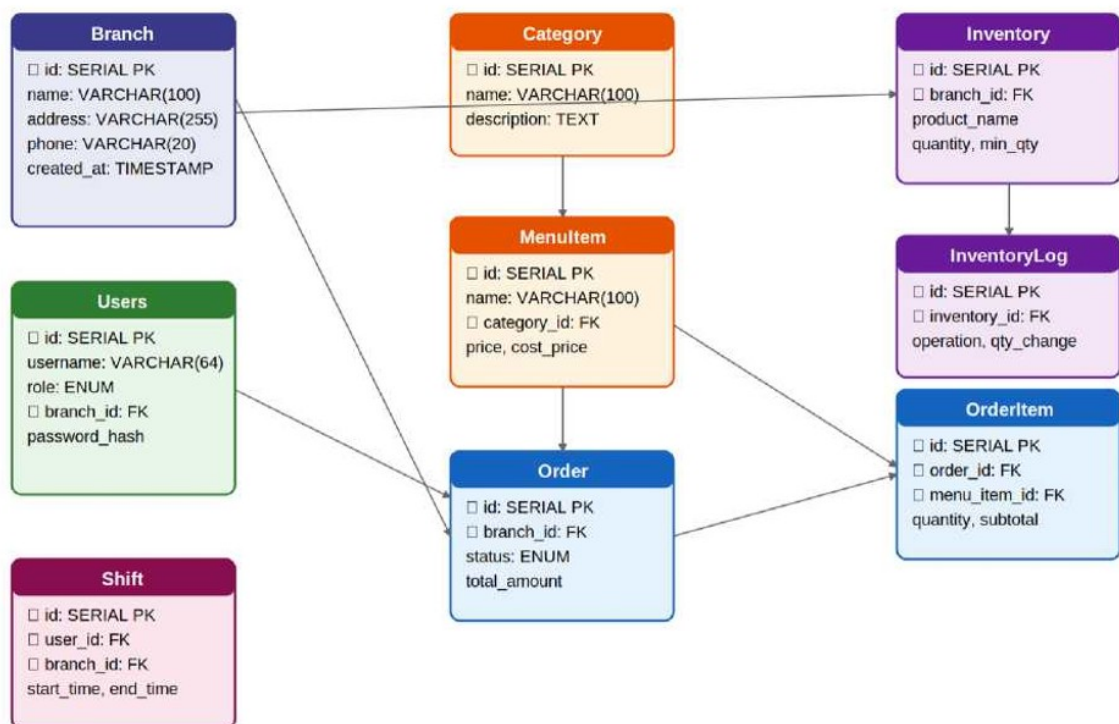


Рис. 3.1 ER-діаграма “estudy_db”

ER-діаграма (рис. 3.1) відображає всі сутності та зв'язки між ними у нотації «пташина лапка». Центральними сутностями є Branch та Order, навколо яких будується вся структура. Домінуючий тип зв'язку — «один до багатьох» (1:N): одна філія має багато замовлень; одне замовлення складається з багатьох позицій; одна категорія включає багато позицій меню.

Центральними сутностями схеми є Branch (філія) та Order (замовлення). Сутність Branch пов'язана з Order, Inventory та Users через зв'язок «один до багатьох» (1:N). Атрибут unit_price у OrderItem зберігає ціну на момент замовлення незалежно від поточної ціни позиції меню — це забезпечує незмінність архівних даних.

Сутність Inventory зберігає поточні залишки складу для кожної філії. Поле min_quantity є мінімальним порогом — якщо залишок нижче мінімуму, система сигналізує про необхідність поповнення. Кожна зміна залишку фіксується у InventoryLog з вказанням типу операції та кількості зміни — це формує повний аудиторський слід.

Детальний опис сутностей логічної моделі даних наведено у таблиці 3.1.

Таблиця 3.1

Опис сутностей логічної моделі даних

Сутність	Ключові атрибути	Призначення
Category	id PK, name UNIQUE, description	Категорія позицій меню: Кава, Чай, Холодні напої, Десерти
MenuItem	id PK, name, category_id FK, price, cost_price, is_available	Позиція меню з ціною та собівартістю
Order	id PK, branch_id FK, status ENUM, total_amount, created_at	Замовлення клієнта. Статус: new, in_progress, done, cancelled
OrderItem	id PK, order_id FK, menu_item_id FK, quantity, unit_price, subtotal	Позиція замовлення. Ціна зберігається на момент замовлення
Inventory	id PK, branch_id FK, product_name, quantity, unit, min_quantity	Складський залишок. min_quantity — поріг сигналізації
InventoryLog	id PK, inventory_id FK, operation, quantity_change, created_at	Журнал операцій руху запасів для аудиту
Users	id PK, username UNIQUE, role ENUM, branch_id FK, password_hash, is_active	Користувач системи з хешованим паролем bcrypt
Shift	id PK, user_id FK, branch_id FK, start_time, end_time	Робоча зміна персоналу

Логічна модель даних є ключовим етапом проектування інформаційної системи, оскільки забезпечує формалізований опис структури даних, визначає основні сутності та їх взаємозв'язки, а також створює основу для подальшої реалізації фізичної бази даних та програмної частини системи обліку та управління роботою мережі кав'ярень.

3.2 Вибір системи управління базою даних та її реалізація

Система управління базами даних (СУБД) є програмним забезпеченням, яке забезпечує створення, зберігання, модифікацію та керування даними в інформаційних системах. Вибір СУБД є одним із ключових етапів проектування програмної системи, оскільки від нього залежить продуктивність, масштабованість, надійність та безпека зберігання інформації [19].

Існує кілька основних типів систем управління базами даних, які відрізняються принципами організації даних та способом їх обробки. Найбільш поширеними є реляційні, нереляційні, об'єктно-орієнтовані та гібридні СУБД.

Реляційні СУБД є найпоширенішим типом систем управління базами даних і базуються на табличній моделі представлення даних. Дані зберігаються у вигляді таблиць, які складаються з рядків і стовпців, а зв'язки між таблицями реалізуються за допомогою ключів. До реляційних СУБД належать такі системи, як PostgreSQL, MySQL, MariaDB, SQLite та Microsoft SQL Server. Вони широко використовуються в корпоративних та вебсистемах завдяки високому рівню надійності та підтримці складних запитів мовою SQL.

Нереляційні СУБД (NoSQL) призначені для роботи з великими обсягами неструктурованих або напівструктурованих даних. Вони забезпечують високу масштабованість та гнучкість структури даних. До таких систем належать MongoDB, Redis та Cassandra. Вони часто використовуються у високонавантажених системах, аналітичних платформах та сервісах реального часу.

Об'єктно-орієнтовані СУБД базуються на концепції об'єктно-орієнтованого програмування та дозволяють зберігати дані у вигляді об'єктів. Вони менш поширені, однак застосовуються у спеціалізованих системах, де важлива тісна інтеграція з об'єктною моделлю програмного забезпечення.

Гібридні або мультимодельні СУБД поєднують у собі можливості різних підходів до зберігання даних, що дозволяє використовувати як реляційну, так і документну або графову модель в межах однієї системи.

Вибір системи управління базами даних є одним із ключових рішень під час проєктування інформаційної системи обліку та управління роботою мережі кав'ярень, оскільки саме СУБД забезпечує зберігання, обробку та цілісність усіх даних системи. Для реалізації даного програмного забезпечення обрано PostgreSQL, яка належить до класу реляційних систем управління базами даних з відкритим вихідним кодом.

Однією з основних причин вибору PostgreSQL є її висока надійність та стабільність роботи. Дана СУБД використовується у багатьох промислових інформаційних системах і довела свою ефективність у задачах, що вимагають обробки транзакційних даних. Для системи обліку кав'ярні це є критично важливим, оскільки будь-які операції з замовленнями, платежами та складськими залишками повинні виконуватись без втрати або спотворення даних.

Ще однією важливою перевагою PostgreSQL є підтримка повноцінних транзакцій із дотриманням принципів ACID (атомарність, узгодженість, ізоляція та довговічність). Це забезпечує коректну обробку операцій, таких як створення замовлення, списання товарів зі складу та формування фінансових записів, навіть у випадку збою системи або одночасної роботи великої кількості користувачів.

PostgreSQL також вирізняється високим рівнем продуктивності та можливістю ефективної роботи з великими обсягами даних. У межах мережі кав'ярень це дозволяє обробляти значну кількість замовлень, аналітичних запитів та операцій у реальному часі без суттєвого зниження швидкодії системи.

Важливою перевагою є гнучкість та розширюваність PostgreSQL. СУБД підтримує створення власних типів даних, функцій та розширень, що дозволяє адаптувати її під специфічні потреби інформаційної системи. Наприклад, це може бути корисно для реалізації аналітичних запитів або додаткових механізмів обробки фінансових даних.

Окремо слід відзначити високий рівень безпеки PostgreSQL. Система підтримує складну модель управління доступом, шифрування даних, а також розмежування прав користувачів. Це є важливим аспектом для системи, яка містить фінансову інформацію, дані про користувачів та операції продажу.

Крім того, PostgreSQL є безкоштовною та відкритою системою, що значно знижує вартість розробки та впровадження програмного забезпечення. Це робить її особливо привабливою для навчальних та комерційних проєктів малого та середнього масштабу.

Вибір PostgreSQL для реалізації інформаційної системи обліку та управління роботою мережі кав'ярень є обґрунтованим завдяки її надійності, продуктивності, підтримці транзакцій, високому рівню безпеки та гнучкості, що повністю відповідає вимогам даного програмного проєкту.

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення визначає загальну структуру інформаційної системи, спосіб організації її компонентів, а також принципи взаємодії між ними. Від правильно обраної архітектури залежить масштабованість, продуктивність, підтримуваність та безпека програмного продукту. У сучасній розробці інформаційних систем використовується кілька основних архітектурних підходів, кожен з яких має свої особливості, переваги та сфери застосування.

Однією з найпростіших є монолітна архітектура, у якій вся система реалізується як єдине ціле. У такому підході всі модулі — інтерфейс користувача, бізнес-логіка та доступ до бази даних — об'єднані в одному застосунку. Монолітна архітектура є простою у розробці та розгортанні, однак при збільшенні масштабу системи може ускладнювати її підтримку та розвиток [20].

Більш сучасним підходом є клієнт-серверна архітектура, яка передбачає розділення системи на дві основні частини: клієнтську та серверну. Клієнт відповідає за взаємодію з користувачем, тоді як сервер забезпечує обробку запитів, бізнес-логіку та роботу з базою даних. Такий підхід широко використовується у вебзастосунках, оскільки дозволяє централізовано керувати даними та забезпечує зручну масштабованість.

Розвитком клієнт-серверної моделі є багаторівнева (n-tier) архітектура, у якій система поділяється на декілька рівнів, зазвичай на рівень представлення (UI), рівень бізнес-логіки та рівень доступу до даних. Такий поділ дозволяє чітко

розмежувати відповідальність компонентів системи, спрощує тестування та подальшу модернізацію програмного забезпечення.

Окремо варто виділити мікросервісну архітектуру, яка передбачає розбиття системи на набір незалежних сервісів, кожен з яких виконує окрему бізнес-функцію. Мікросервіси взаємодіють між собою через API та можуть розгортатися незалежно один від одного. Такий підхід забезпечує високу масштабованість і гнучкість, однак ускладнює розробку, моніторинг і підтримку системи.

Також існує подієво-орієнтована архітектура, у якій взаємодія між компонентами відбувається через події. Кожен компонент реагує на певні події та може генерувати нові, що дозволяє будувати слабкозв'язані та високонавантажені системи.

Для реалізації інформаційної системи обліку та управління роботою мережі кав'ярень найбільш доцільним є використання клієнт-серверної або багаторівневої архітектури. У даному випадку серверна частина реалізується із застосуванням Flask, яка відповідає за обробку бізнес-логіки та взаємодію з базою даних PostgreSQL, тоді як клієнтська частина забезпечує інтерфейс взаємодії користувача з системою через веббраузер.

Вибір архітектури програмного забезпечення є одним із ключових етапів проектування інформаційної системи обліку та управління роботою мережі кав'ярень, оскільки саме архітектурний підхід визначає спосіб організації компонентів системи та характер їх взаємодії. Для реалізації даного проєкту обрано клієнт-серверну архітектуру, яка є найбільш доцільною для веборієнтованих інформаційних систем.

Однією з основних причин вибору клієнт-серверної архітектури є централізоване зберігання та обробка даних. У межах мережі кав'ярень критично важливо забезпечити єдине джерело даних щодо замовлень, товарів, складських залишків та фінансових операцій. Серверна частина системи виконує роль центрального вузла, який забезпечує узгодженість інформації та виключає можливість її дублювання або розсинхронізації між різними користувачами.

Ще однією важливою перевагою є спрощене управління безпекою. Оскільки вся бізнес-логіка та доступ до бази даних реалізуються на сервері, це дозволяє ефективно контролювати права доступу користувачів, реалізовувати авторизацію та захист конфіденційної інформації. Такий підхід є особливо важливим для систем, що містять фінансові дані та інформацію про продажі.

Клієнт-серверна архітектура також забезпечує простоту оновлення та супроводу системи. Усі зміни бізнес-логіки реалізуються на серверній частині, що дозволяє уникнути необхідності оновлення кожного клієнтського пристрою окремо. Це значно спрощує підтримку системи та зменшує витрати на її адміністрування.

Важливою перевагою є масштабованість системи. Серверна архітектура дозволяє обробляти велику кількість одночасних запитів від користувачів, що є необхідним для мережі кав'ярень із великою кількістю працівників та точок продажу. У разі необхідності систему можна масштабувати шляхом підвищення продуктивності сервера або розподілення навантаження між кількома серверами.

Також слід відзначити зручність доступу до системи. Клієнтська частина реалізується у вигляді вебзастосунку, що дозволяє користувачам працювати з системою через браузер без необхідності встановлення додаткового програмного забезпечення. Це особливо зручно для розподілених мереж закладів.

У межах реалізації проєкту серверна частина побудована з використанням Flask, яка забезпечує обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних PostgreSQL. Клієнтська частина відповідає за відображення інтерфейсу та взаємодію користувача із системою.

Вибір клієнт-серверної архітектури є обґрунтованим завдяки централізованому управлінню даними, високому рівню безпеки, простоті супроводу та можливості масштабування, що повністю відповідає вимогам інформаційної системи обліку та управління роботою мережі кав'ярень.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Діаграма пакетів (package diagram) є одним із типів UML-діаграм, яка використовується для логічної декомпозиції програмної системи на окремі групи елементів, що називаються пакетами. Вона дозволяє відобразити структуру програмного забезпечення на рівні модулів та показати залежності між ними, що є особливо корисним при проектуванні великих і середніх інформаційних систем.

Основною метою діаграми пакетів є організація системи у вигляді логічно пов'язаних частин, кожна з яких виконує певну функцію. Пакет може містити класи, інші пакети або компоненти, які об'єднані спільною відповідальністю. Такий підхід дозволяє підвищити структурованість системи, спростити її супровід та забезпечити кращу масштабованість програмного забезпечення.

У діаграмі пакетів основними елементами є самі пакети та залежності між ними. Залежності показують, як один пакет використовує функціональність іншого. Це дозволяє визначити ступінь зв'язності між різними частинами системи та оцінити рівень її модульності.

У межах інформаційної системи обліку та управління роботою мережі кав'ярень діаграма пакетів може використовуватись для розподілу системи на такі логічні модулі, як модуль управління користувачами, модуль роботи з товарами, модуль обробки замовлень, модуль складського обліку та модуль фінансової звітності. Кожен із цих пакетів виконує окрему функціональну роль у системі та взаємодіє з іншими модулями через визначені інтерфейси.

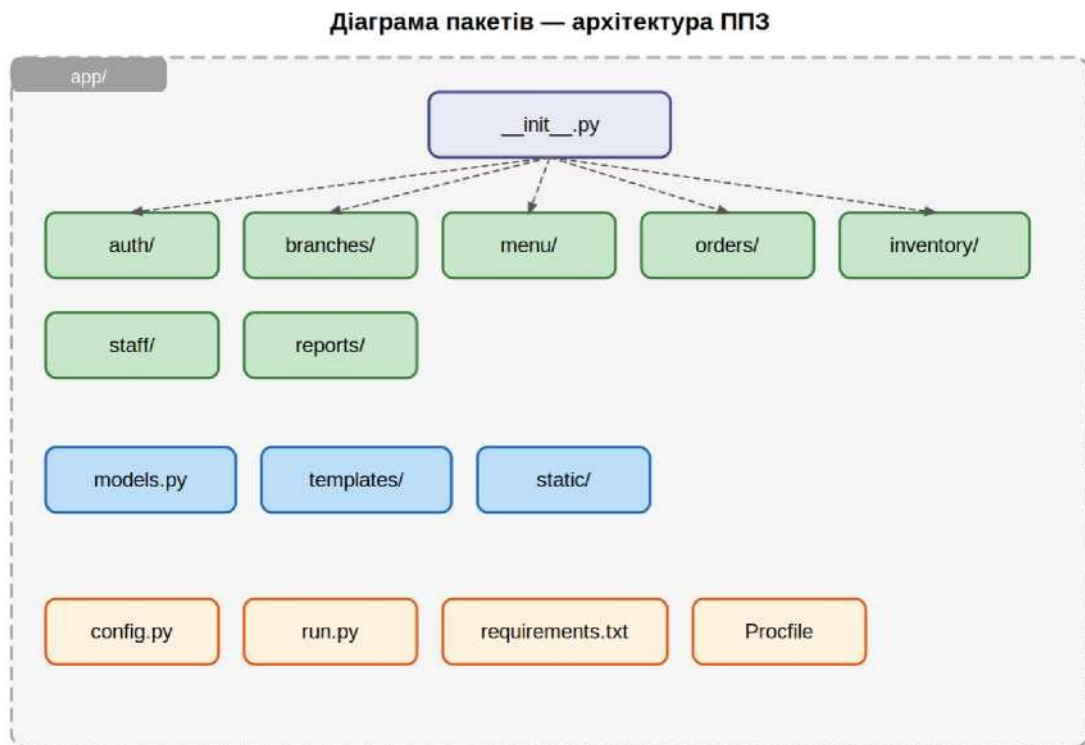


Рис. 3.2 Діаграма пакетів

ППЗ побудоване за архітектурним патерном MVC (Model-View-Controller), реалізованим засобами фреймворку Flask. Патерн MVC забезпечує чітке розмежування відповідальності: модель (SQLAlchemy-класи) відповідає за дані та бізнес-логіку; представлення (Ліца2-шаблони) формує HTML; контролер (Blueprint-маршрути) обробляє HTTP-запити.

Застосунок розбито на 7 Blueprint-модулів за принципом єдиної відповідальності. Кожен Blueprint незалежно розробляється та тестується, зміни в одному модулі не впливають на інші. Діаграма пакетів (рис. 3.2) відображає повну ієрархію каталогів та взаємозв'язки між модулями.

Структура кожного Blueprint однакова: `routes.py` — маршрути та бізнес-логіка; `forms.py` — WTForms-класи для валідації та CSRF-захисту; `templates/` — Ліца2-шаблони HTML. Такий поділ відповідає принципу єдиної відповідальності та спрощує тестування.

Базовий шаблон `base.html` реалізує успадкування Jinja2: всі сторінки мають спільну навігаційну панель, блок flash-повідомлень та Bootstrap 5.3. Статичні файли у `app/static/` обслуговуються `gunicorn` у production. Корпоративна

кольорова схема (темно-коричнева #2C1A0E, бежевий #C8956C) застосована через CSS-змінні.

Детальний опис Blueprint-модулів наведено у таблиці 3.2.

Таблиця 3.2

Опис Blueprint-модулів

Blueprint	Маршрути	Призначення
auth	/auth/login, /auth/logout	Авторизація, керування cookie-сесіями
branches	/branches/, /branches/new, //edit	СНБВ-операції з філіями
menu	/menu/, /menu/new, //edit	Управління позиціями меню та категоріями
orders	/orders/, /orders/new, /, //status	Реєстрація та зміна статусу замовлень
inventory	/inventory/, /inventory/new, //update	Облік складських залишків
staff	/staff/, /staff/new, //edit	Управління обліковими записами персоналу
reports	/reports/	Формування аналітичних звітів

Використання діаграми пакетів дозволяє чітко структурувати архітектуру програмного забезпечення, зменшити складність системи шляхом її поділу на логічні частини та забезпечити легше масштабування і подальший розвиток проєкту. Крім того, вона полегшує розуміння структури системи для розробників та сприяє ефективнішій організації командної роботи.

3.5 Вибір інструментарію для створення програмного забезпечення

Вибір інструментарію для розробки інформаційної системи обліку та управління роботою мережі кав'ярень визначається вимогами до функціональності, продуктивності, безпеки та масштабованості програмного забезпечення. Використання сучасних технологій дозволяє забезпечити

стабільну роботу вебзастосунку, зручність розробки та подальшого супроводу системи.

Основною мовою програмування для реалізації серверної частини обрано Python. Дана мова відзначається простотою синтаксису, високою читабельністю коду та широкими можливостями для розробки вебзастосунків, що робить її доцільною для реалізації інформаційних систем різного рівня складності.

Як основний вебфреймворк використовується Flask, який забезпечує гнучку архітектуру застосунку, підтримку маршрутизації запитів та зручну інтеграцію з іншими бібліотеками Python. Flask дозволяє створювати легкі та масштабовані вебзастосунки з можливістю поступового розширення функціональності [21].

Для взаємодії з базою даних використовується бібліотека Flask-SQLAlchemy, яка є ORM-рішенням і забезпечує об'єктно-реляційне відображення даних. Вона спрощує роботу з базою даних та дозволяє виконувати операції без необхідності написання складних SQL-запитів вручну.

Flask-Login застосовується для реалізації механізму автентифікації та управління сесіями користувачів. Ця бібліотека дозволяє забезпечити захищений доступ до системи та розмежування прав користувачів залежно від їх ролей.

Flask-Migrate використовується для керування міграціями бази даних, що дозволяє відстежувати зміни структури таблиць і забезпечує узгодженість між різними версіями програмного забезпечення.

Flask-Bcrypt відповідає за хешування паролів користувачів, що підвищує рівень безпеки системи та захищає конфіденційні дані від несанкціонованого доступу.

Flask-WTF використовується для створення та валідації вебформ, що дозволяє забезпечити коректність введених користувачем даних та підвищити рівень захисту від типових атак, таких як CSRF.

Як система управління базами даних використовується PostgreSQL, яка забезпечує надійне зберігання структурованих даних, підтримку транзакцій та високу продуктивність при обробці великих обсягів інформації.

Для створення користувацького інтерфейсу застосовується Bootstrap, що дозволяє швидко розробляти адаптивний та сучасний дизайн вебсторінок, який коректно відображається на різних пристроях.

Для візуалізації аналітичних даних використовується Chart.js, яка дозволяє будувати інтерактивні графіки та діаграми на основі даних про продажі та діяльність кав'ярень.

Шаблонізатор Jinja2 використовується для формування динамічних вебсторінок, що дозволяє відокремити бізнес-логіку від представлення даних та підвищує зручність розробки інтерфейсу.

Для розгортання вебзастосунку у виробничому середовищі використовується Gunicorn, який забезпечує стабільну та продуктивну роботу серверної частини системи.

Для тестування програмного забезпечення застосовуються pytest та pytest-cov, які дозволяють виконувати модульне тестування та оцінювати покриття коду тестами, що підвищує якість і надійність програмного продукту.

Детальний опис інструментарію наведено у таблиці 3.3.

Таблиця 3.3

Опис інструментарію

Компонент	Версія	Призначення та обґрунтування
Python	3.11	Основна мова. Читабельний синтаксис, потужна екосистема
Flask	3.0	Мікро-фреймворк. Не нав'язує структуру, гнучкий
Flask-SQLAlchemy	3.1	ORM. Прозора робота з БД, підтримка lazy loading
Flask-Login	0.6	Сесії: @login_required, current_user, remember_me
Flask-Migrate	4.0	Міграції через Alembic: flask db migrate/upgrade
Flask-Bcrypt	1.0	Хешування паролів bcrypt (cost factor 12)
Flask-WTF	1.2	Форми та автоматичний CSRF-захист
PostgreSQL	15	СУБД. Нативна підтримка Render.com, ENUM, BSD
Bootstrap	5.3	CSS. Адаптивна сітка, готові компоненти
Chart.js	4.4	Інтерактивні Canvas-графіки для модуля звітів

Jinja2	3.1	Шаблонізатор HTML (вбудований у Flask)
gunicorn	21.2	WSGI-сервер production: gunicorn run:app
pytest + pytest-cov	7.4 + 4.1	Тестування, вимірювання покриття коду

Таким чином, обраний інструментарій забезпечує повний цикл розробки інформаційної системи обліку та управління роботою мережі кав'ярень, включаючи реалізацію серверної частини, роботи з базою даних, створення інтерфейсу користувача, візуалізацію даних, а також тестування та розгортання системи.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Вимоги до апаратного та програмного забезпечення інформаційної системи обліку та управління роботою мережі кав'ярень визначають мінімальні та рекомендовані характеристики середовища, необхідного для коректного функціонування програмного продукту. Дані вимоги формуються з урахуванням обраної архітектури системи, використаних технологій та передбачуваного навантаження.

З точки зору апаратного забезпечення, для серверної частини системи визначаються наступні мінімальні вимоги:

- процесор із тактовою частотою не нижче 2.0 GHz;
- оперативна пам'ять обсягом від 4 GB;
- дисковий простір не менше 20 GB для зберігання програмного забезпечення, бази даних та користувацьких файлів;
- стабільне підключення до мережі Інтернет для забезпечення доступу користувачів до системи.

Для забезпечення більш стабільної роботи системи в умовах високого навантаження рекомендуються такі параметри серверного обладнання:

- багатоядерний процесор середнього або високого класу;
- оперативна пам'ять обсягом 8 GB і більше;
- використання SSD-накопичувача для підвищення швидкості обробки даних;
- резервне копіювання даних на окремий носій або хмарне сховище.

Апаратна частина системи повинна забезпечувати стабільну роботу серверної складової, бази даних та клієнтських запитів. Для серверного компонента рекомендується використання багатоядерного процесора з

достатньою обчислювальною потужністю, що дозволяє обробляти одночасні запити користувачів без значних затримок. Мінімальний обсяг оперативної пам'яті для серверної частини повинен становити не менше 4 ГБ, однак для стабільної роботи в умовах підвищеного навантаження рекомендовано 8 ГБ і більше. Додатково необхідне достатнє дискове сховище для зберігання даних бази даних, резервних копій та логів системи.

Для клієнтської частини системи вимоги до апаратного забезпечення є менш жорсткими, оскільки основне навантаження припадає на сервер. Користувачі можуть працювати з системою за допомогою персональних комп'ютерів, ноутбуків, планшетів або мобільних пристроїв, які підтримують сучасні веббраузери. Достатньо наявності стабільного інтернет-з'єднання для взаємодії із сервером.

Щодо програмного забезпечення, серверна частина системи реалізується із використанням мови програмування Python та вебфреймворку Flask. Для забезпечення зберігання даних використовується PostgreSQL, яка повинна бути встановлена на сервері або доступна через віддалене підключення.

Операційна система для серверної частини може бути представлена дистрибутивами Linux (наприклад, Ubuntu Server), Windows Server або іншими сумісними середовищами, що підтримують виконання Python-застосунків та роботу PostgreSQL. Вибір операційної системи залежить від вимог розгортання та адміністративних уподобань.

Клієнтська частина системи повинна працювати у сучасних веббраузерах, які підтримують HTML5, CSS3 та JavaScript. До таких браузерів належать Google Chrome, Mozilla Firefox, Microsoft Edge та інші сучасні браузери з актуальними версіями рушіїв.

Для коректної роботи системи також необхідна підтримка бібліотек та фреймворків, які використовуються у проєкті, зокрема Flask-SQLAlchemy, Flask-Login, Flask-WTF, Bootstrap та інших залежностей, що забезпечують функціональність вебзастосунку.

Дотримання визначених вимог до апаратного та програмного забезпечення забезпечує стабільну, безпечну та ефективну роботу інформаційної системи обліку та управління роботою мережі кав'ярень як на серверному, так і на клієнтському рівнях.

Також було зроблено діаграму розгортання для візуального огляду деплою системи (Рис. 4.1).

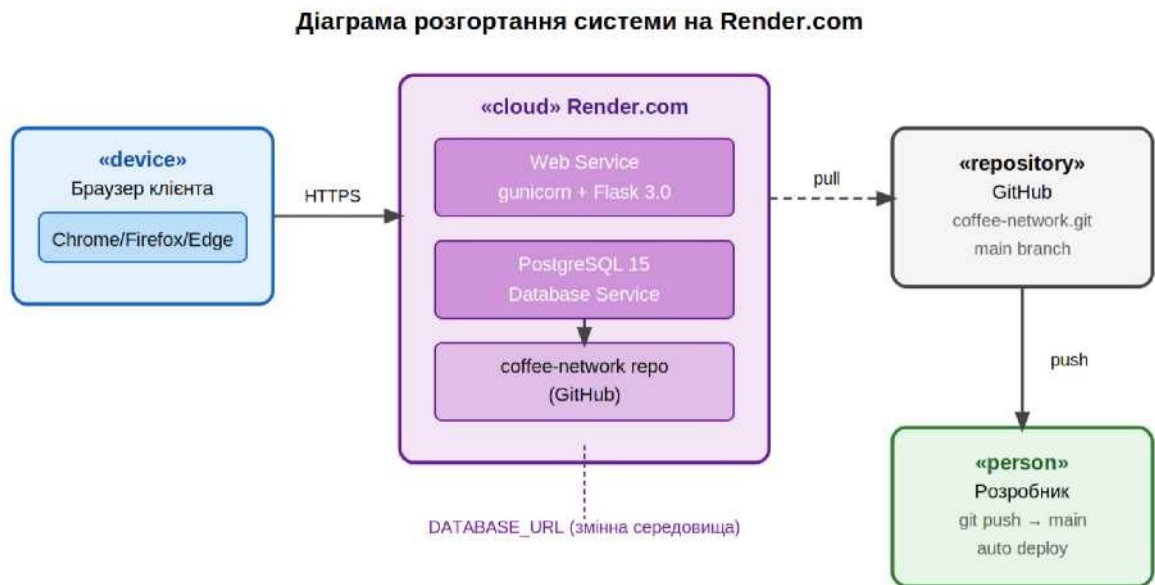


Рис. 4.1 Діаграма розгортання

4.2 Тестування системи

На рис. 4.2-4.7 наведено скріншоти всіх функціональних модулів інтерфейсу системи, розгорнутої на <https://coffee-network.onrender.com>. Дизайн виконано у корпоративних кольорах (темно-коричнева навігаційна панель, бежевий акцент) засобами Bootstrap 5.3.

Після запуску системи відкривається головна сторінка програмного забезпечення (Рисунок 4.2). На головній сторінці відображається список замовлень та головне меню за допомогою якого ми можемо керувати системою.

Модуль замовлень (рис. 4.2) є центральним робочим простором системи. Таблиця відображає всі замовлення з колонками: номер (#), філія, дата та час, загальна сума та поточний статус. Кнопка «+ Нове замовлення» у правому верхньому куті переходить до форми реєстрації. Адміністратор бачить замовлення всіх трьох філій мережі; менеджер бачить виключно замовлення своєї філії. При відсутності замовлень система відображає текст «Замовлень ще немає» замість порожньої таблиці.

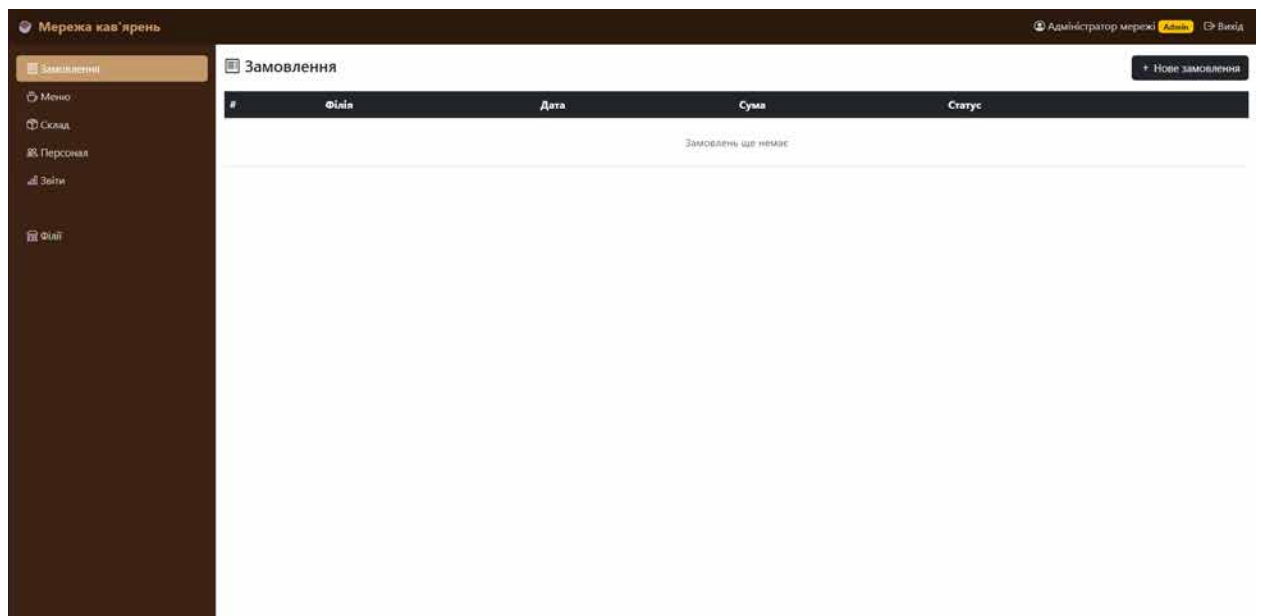


Рисунок 4.2 Головна сторінка

Модуль філій (рис. 4.3) відображає картки трьох діючих точок мережі у Києві: Кав'ярня на Хрещатику (вул. Хрещатик, 22, тел. +380441234567), Кав'ярня на Позняках (пр. Бажана, 14, тел. +380442345678) та Кав'ярня на Оболоні (пр. Героїв Сталінграда, 8, тел. +380443456789). Для кожної картки відображається назва, адреса, телефон та дата відкриття. Доступно лише Адміністратору.

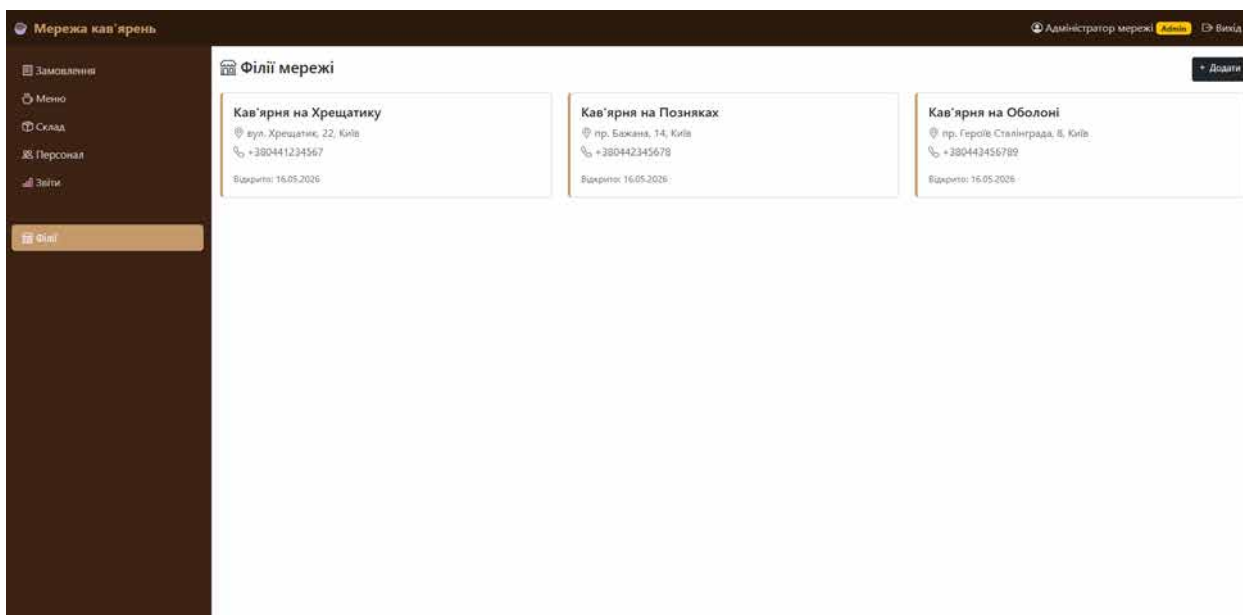


Рисунок 4.3 Філії

Модуль меню (рис. 4.4) групує позиції по чотирьох категоріях. Категорія «Кава»: Еспресо (ціна 55 грн, собівартість 12 грн), Американо (65/14 грн), Капучино (85/22 грн), Латте (90/25 грн). Категорія «Чай»: Чай чорний (55/8 грн). Категорія «Холодні напої»: Лимонад (75/18 грн). Категорія «Десерти»: Круасан (55/20 грн). Відображення собівартості поруч із ціною дозволяє адміністратору миттєво оцінити маржинальність кожної позиції без додаткових розрахунків.

Категорія	Назва	Ціна	Собівартість	Статус	Дія
Кава	Еспресо	55.00 грн	12.00 грн	Доступно	Зняти
	Капучино	85.00 грн	22.00 грн	Доступно	Зняти
	Латте	90.00 грн	25.00 грн	Доступно	Зняти
	Американо	65.00 грн	14.00 грн	Доступно	Зняти
Чай	Чай чорний	55.00 грн	8.00 грн	Доступно	Зняти
Холодні напої	Лимонад	75.00 грн	18.00 грн	Доступно	Зняти
Десерти	Круасан	55.00 грн	20.00 грн	Доступно	Зняти

Рисунок 4.4 Меню

Модуль складу (рис. 4.5) дозволяє переглядати залишки по конкретній філії або по всіх одночасно через фільтр-дропдаун «Всі філії». Таблиця

відображає: назву товару, поточний залишок, одиницю виміру (кг, л, шт), мінімальний поріг та статус. Рядки, де залишок нижче мінімального порогу, виділяються кольором попередження. Кнопки «Додати» та «Списати» відкривають модальне вікно для внесення операції, яка автоматично записується у InventoryLog.

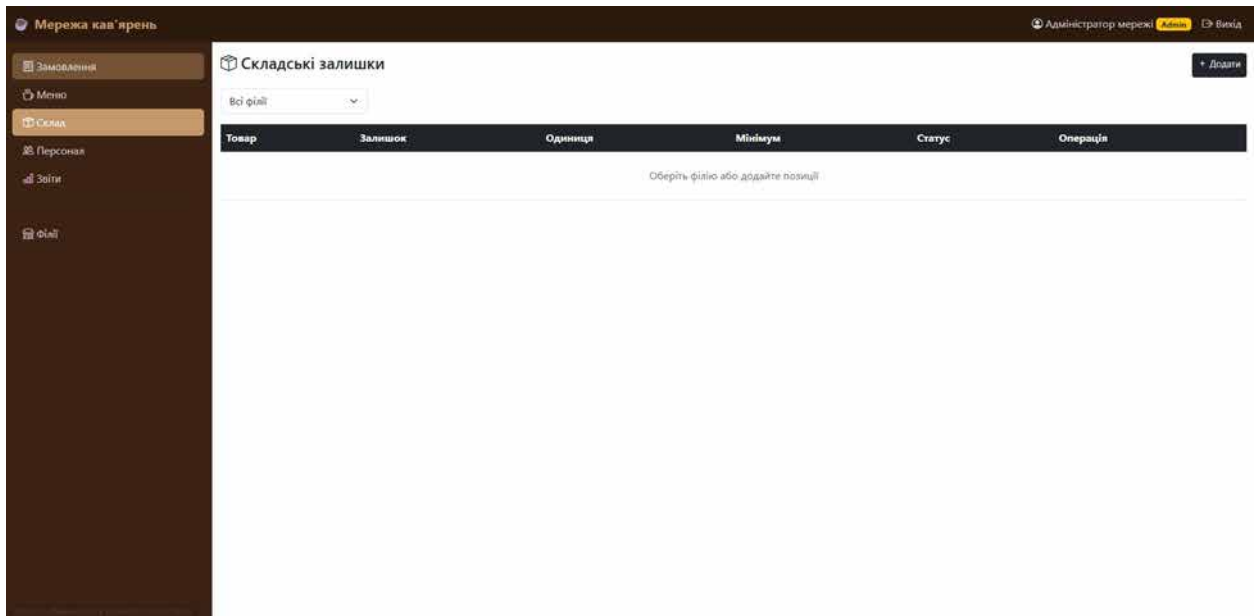


Рисунок 4.5 Склад

Модуль персоналу (рис. 6.6) відображає список облікових записів з колонками: ім'я, логін, роль, прив'язана філія та статус (Активний/Неактивний). Адміністратор може додавати нових користувачів, редагувати їх дані та деактивувати облікові записи без видалення (збереження аудиторської бази). На скріншоті видно менеджера Іваненко Оксану (логін manager1) з прив'язкою до «Кав'ярні на Хрещатику» та статусом «Активний».

Ім'я	Логін	Роль	Філія	Статус
Іваненко Оксана	manager1	manager	Кав'ярня на Хрещатику	Активний

Рисунок 4.6 Персонал

Модуль звітів (рис. 4.7) формує аналітику за вибраний часовий діапазон та конкретну філію або по всіх. Фільтри: поля «Від» та «До» з вбудованим DatePicker та дропдаун «Філія». Три KPI-картки відображають: загальну виручку (грн), кількість виконаних замовлень та середній чек (грн). Нижче розміщено лінійний графік Chart.js з динамікою показників по датах. При відсутності даних за вибраний період система відображає повідомлення «Немає даних за вибраний період».

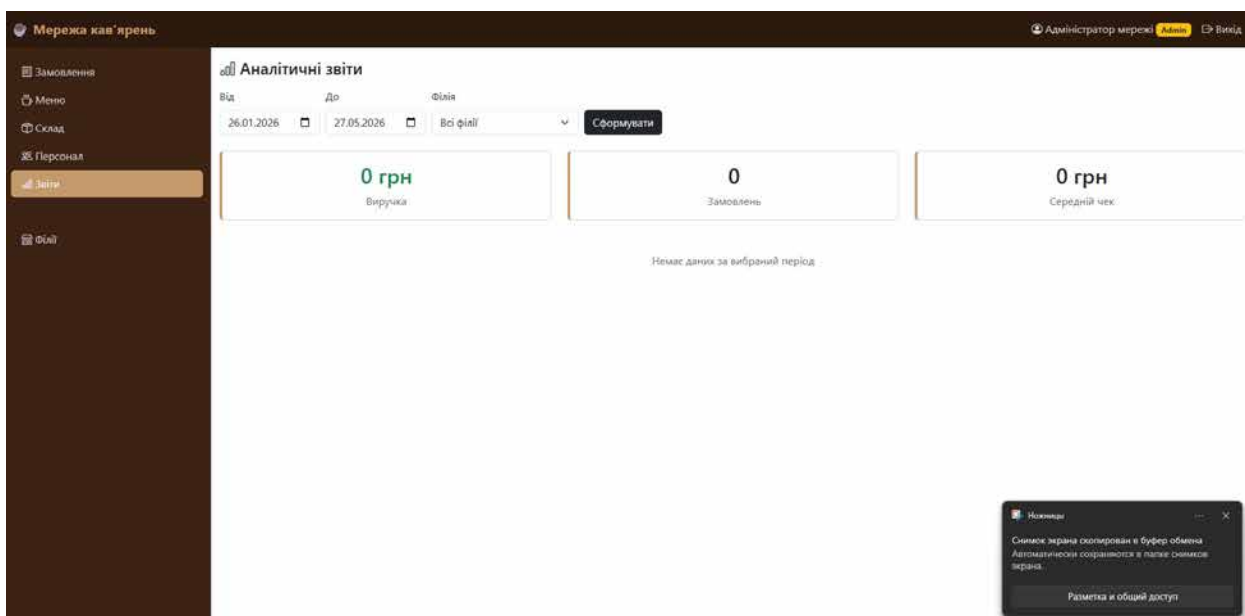


Рисунок 4.7 Звіти

4.3 Склад інсталяційного пакету

Інсталяційний пакет інформаційної системи обліку та управління роботою мережі кав'ярень представляє собою набір файлів і компонентів, необхідних для встановлення, налаштування та запуску програмного забезпечення в робочому середовищі. Його формування забезпечує можливість швидкого розгортання системи на сервері без необхідності додаткового налаштування базових компонентів вручну.

До складу інсталяційного пакету входить серверна частина програмного забезпечення, реалізована за допомогою мови програмування Python та вебфреймворку Flask. Вона включає основну бізнес-логіку системи, модулі обробки запитів, маршрутизацію, а також компоненти для взаємодії з базою даних.

Обов'язковим елементом інсталяційного пакету є конфігураційні файли, які містять параметри підключення до бази даних, налаштування середовища виконання, параметри безпеки та інші системні налаштування. Ці файли дозволяють адаптувати систему до конкретного середовища розгортання без зміни основного програмного коду.

Також до складу пакету входять файли залежностей, які визначають перелік необхідних бібліотек і розширень для коректної роботи системи. Зокрема, це бібліотеки для роботи з базою даних, автентифікації користувачів, обробки форм, візуалізації даних та тестування. Встановлення залежностей здійснюється автоматично за допомогою менеджера пакетів Python.

Окрему частину інсталяційного пакету становлять міграції бази даних, які забезпечують створення та оновлення структури таблиць у системі PostgreSQL. Міграції дозволяють автоматизувати процес ініціалізації бази даних та забезпечують узгодженість її структури з програмним кодом.

До складу також входять статичні ресурси системи, які забезпечують відображення користувацького інтерфейсу. Це файли стилів, скрипти JavaScript, а також шаблони інтерфейсу, реалізовані з використанням Bootstrap та

шаблонізатора Jinja2. Вони відповідають за зовнішній вигляд та інтерактивність вебзастосунку.

До складу інсталяційного пакету входять такі основні компоненти:

1. серверна частина застосунку, реалізована за допомогою Python та Flask, яка містить бізнес-логіку системи, маршрутизацію запитів та модулі взаємодії з базою даних;
2. конфігураційні файли, що визначають параметри роботи системи, включаючи налаштування підключення до бази даних, змінні середовища, параметри безпеки та режим роботи застосунку;
3. залежності проєкту, які містять перелік бібліотек і модулів, необхідних для коректного функціонування системи, включаючи роботу з формами, авторизацією, ORM та іншими компонентами;
4. міграції бази даних для системи PostgreSQL, які забезпечують автоматичне створення та оновлення структури таблиць відповідно до поточної версії програмного забезпечення;
5. статичні ресурси, які включають стилі, скрипти та графічні елементи інтерфейсу, реалізовані з використанням Bootstrap та JavaScript, а також шаблони інтерфейсу на основі Jinja2;
6. скрипти запуску та розгортання системи, зокрема конфігурація для вебсерверу Gunicorn, який використовується для стабільної роботи застосунку у виробничому середовищі.

Інсталяційний пакет розміщено у публічному репозиторії GitHub. Команди розгортання локально: `git clone -+ pip install -r requirements.txt -+ flask db upgrade -+ python seed.py -+ python run.py`.

Для хмарного розгортання на Render.com: підключити CiiНПБ-репозиторій, вказати команду `дппісогп гпп:app`, створити PostgreSQL-сервіс, скопіювати

`DATABASE_URL` до змінних середовища — деплой запускається автоматично.

ВИСНОВКИ

У межах виконання бакалаврської кваліфікаційної роботи було розроблено інформаційну систему обліку та управління роботою мережі кав'ярень, яка забезпечує автоматизацію основних бізнес-процесів закладів громадського харчування, включаючи облік замовлень, управління товарами, контроль складських залишків та формування фінансової інформації. Результатом роботи стало проєктування та часткова реалізація веборієнтованої системи, яка дозволяє ефективно підтримувати діяльність мережі кав'ярень у єдиному інформаційному середовищі.

У процесі виконання роботи було проведено аналіз предметної області, досліджено особливості функціонування систем обліку та управління в закладах громадського харчування, а також визначено основні процеси, що підлягають автоматизації. На основі цього було сформовано вимоги до програмної системи та обґрунтовано необхідність створення централізованого рішення для управління діяльністю мережі кав'ярень.

Також було здійснено проєктування програмної системи, включаючи розробку логічної моделі даних, визначення основних сутностей предметної області та їх взаємозв'язків, а також побудову архітектури програмного забезпечення. Окрему увагу приділено вибору системи управління базою даних PostgreSQL, яка забезпечує надійне зберігання та обробку інформації.

Під час реалізації програмного забезпечення використано сучасні технології веброзробки, зокрема Python, Flask, HTML, CSS та JavaScript, що дозволило створити функціональну серверну частину та зручний користувацький інтерфейс. Це забезпечило гнучкість системи, її масштабованість та зручність у використанні.

Окремим важливим результатом роботи є реалізація механізмів обліку замовлень та фінансових операцій, які дозволяють контролювати процес продажів, аналізувати діяльність закладів та підвищувати ефективність

управління мережею кав'ярень. Такий підхід сприяє підвищенню прозорості бізнес-процесів та оптимізації роботи персоналу.

Таким чином, у результаті виконання бакалаврської роботи було досягнуто поставленої мети та виконано всі визначені завдання, а саме:

- проаналізовано предметну область та існуючі системи обліку і управління закладами громадського харчування;
- визначено функціональні та нефункціональні вимоги до програмної системи;
- розроблено модель предметної області та логічну структуру даних;
- спроектовано архітектуру програмного забезпечення та обрано технологічний стек;
- реалізовано основні модулі інформаційної системи обліку та управління мережею кав'ярень.

Отримані результати підтверджують можливість ефективного використання розробленої інформаційної системи для автоматизації діяльності мережі кав'ярень, підвищення якості управління та оптимізації бізнес-процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flask Documentation. URL: <https://flask.palletsprojects.com> (дата звернення: 27.05.2026).
2. PostgreSQL Official Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 27.05.2026).
3. Python Documentation. URL: <https://docs.python.org/3/> (дата звернення: 27.05.2026).
4. SQLAlchemy Documentation. URL: <https://docs.sqlalchemy.org/> (дата звернення: 27.05.2026).
5. Bootstrap Documentation. URL: <https://getbootstrap.com/docs/> (дата звернення: 27.05.2026).
6. Chart.js Documentation. URL: <https://www.chartjs.org/docs/latest/> (дата звернення: 27.05.2026).
7. Jinja2 Documentation. URL: <https://jinja.palletsprojects.com/> (дата звернення: 27.05.2026).
8. What is Flask? URL: <https://www.geeksforgeeks.org/python-introduction-to-flask/> (дата звернення: 27.05.2026).
9. REST API Architecture. URL: <https://restfulapi.net/> (дата звернення: 27.05.2026).
10. UML Tutorial. URL: <https://www.uml-diagrams.org/> (дата звернення: 27.05.2026).
11. ER Diagram Basics. URL: <https://www.lucidchart.com/pages/er-diagrams> (дата звернення: 27.05.2026).
12. Software Architecture Patterns. URL: <https://martinfowler.com/architecture/> (дата звернення: 27.05.2026).
13. Client-Server Architecture Explained. URL: <https://www.ibm.com/topics/client-server-model> (дата звернення: 27.05.2026).

- 14.MVC Architecture Pattern. URL: <https://developer.mozilla.org/en-US/docs/Glossary/MVC> (дата звернення: 27.05.2026).
- 15.Web Application Architecture. URL: <https://aws.amazon.com/what-is/web-application/> (дата звернення: 27.05.2026).
- 16.OWASP Web Security Basics. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 27.05.2026).
- 17.Gunicorn Documentation. URL: <https://docs.gunicorn.org/> (дата звернення: 27.05.2026).
- 18.Flask-SQLAlchemy Documentation. URL: <https://flask-sqlalchemy.palletsprojects.com/> (дата звернення: 27.05.2026).
- 19.Flask-Login Documentation. URL: <https://flask-login.readthedocs.io/> (дата звернення: 27.05.2026).
- 20.Flask-WTF Documentation. URL: <https://flask-wtf.readthedocs.io/> (дата звернення: 27.05.2026).
- 21.Flask-Migrate Documentation. URL: <https://flask-migrate.readthedocs.io/> (дата звернення: 27.05.2026).
- 22.Flask-Bcrypt Documentation. URL: <https://flask-bcrypt.readthedocs.io/> (дата звернення: 27.05.2026).
- 23.Clean Architecture Overview. URL: <https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html> (дата звернення: 27.05.2026).
- 24.Database Design Basics. URL: <https://www.tutorialspoint.com/dbms/index.htm> (дата звернення: 27.05.2026).
- 25.Introduction to Relational Databases. URL: <https://www.oracle.com/database/what-is-a-relational-database/> (дата звернення: 27.05.2026).
- 26.Software Testing with Pytest. URL: <https://docs.pytest.org/en/stable/> (дата звернення: 27.05.2026).

27. Pytest Coverage Documentation. URL: <https://pytest-cov.readthedocs.io/> (дата звернення: 27.05.2026).
28. What is PostgreSQL? URL: <https://www.ibm.com/topics/postgresql> (дата звернення: 27.05.2026).
29. Frontend Development Basics. URL: https://developer.mozilla.org/en-US/docs/Learn/Front-end_web_developer (дата звернення: 27.05.2026).
30. JavaScript Basics. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (дата звернення: 27.05.2026).
31. Web Application Security Fundamentals. URL: <https://cheatsheetseries.owasp.org/> (дата звернення: 27.05.2026).
32. Agile Software Development Overview. URL: <https://www.agilealliance.org/agile101/> (дата звернення: 27.05.2026).
33. Introduction to APIs. URL: <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces> (дата звернення: 27.05.2026).
34. Software Development Lifecycle. URL: <https://www.atlassian.com/software-development/software-development-life-cycle> (дата звернення: 27.05.2026).
35. Introduction to Web Servers. URL: <https://httpd.apache.org/docs/> (дата звернення: 27.05.2026).

ДОДАТОК А

Фрагменти програмного коду. Функція створення замовлення

```

from flask import render_template, redirect, url_for, flash, request
from flask_login import login_required, current_user
from app.orders import orders_bp
from app.models import Order, OrderItem, MenuItem, Branch
from app import db
from decimal import Decimal

@orders_bp.route('/')
@login_required
def index():
    if current_user.is_admin():
        orders = Order.query.order_by(Order.created_at.desc()).limit(100).all()
    else:
        orders = Order.query.filter_by(branch_id=current_user.branch_id).order_by(Order.created_at.desc()).limit(100).all()
    return render_template('orders/index.html', orders=orders)

@orders_bp.route('/new', methods=['GET', 'POST'])
@login_required
def new():
    menu_items = MenuItem.query.filter_by(is_available=True).all()
    branches = Branch.query.all() if current_user.is_admin() else []
    if request.method == 'POST':
        branch_id = current_user.branch_id if not current_user.is_admin() else request.form.get('branch_id', type=int)
        if not branch_id:
            flash('Оберіть філію', 'danger')
        return render_template('orders/new.html', menu_items=menu_items, branches=branches)
    order = Order(branch_id=branch_id, created_by=current_user.id)
    db.session.add(order)
    total = Decimal('0')
    has_items = False
    for item in menu_items:
        try:
            qty = int(request.form.get(f'qty_{item.id}', 0))
        except:
            qty = 0
        if qty > 0:
            subtotal = item.price * qty
            db.session.add(OrderItem(order=order, menu_item_id=item.id, quantity=qty, unit_price=item.price, subtotal=subtotal))
            total += subtotal
            has_items = True
    if not has_items:
        db.session.rollback()

```

```

        flash('Додайте хоча б одну позицію', 'warning')
        return render_template('orders/new.html', menu_items=menu_items,
branches=branches)
    order.total_amount = total
    db.session.commit()
    flash(f'Замовлення #{order.id} на суму {total} грн створено', 'success')
    return redirect(url_for('orders.detail', id=order.id))
    return render_template('orders/new.html', menu_items=menu_items, branches=branches)

@orders_bp.route('/<int:id>')
@login_required
def detail(id):
    order = Order.query.get_or_404(id)
    if not current_user.is_admin() and order.branch_id != current_user.branch_id:
        flash('Доступ заборонено', 'danger')
        return redirect(url_for('orders.index'))
    return render_template('orders/detail.html', order=order)

@orders_bp.route('/<int:id>/status', methods=['POST'])
@login_required
def change_status(id):
    order = Order.query.get_or_404(id)
    if not current_user.is_admin() and order.branch_id != current_user.branch_id:
        flash('Доступ заборонено', 'danger')
        return redirect(url_for('orders.index'))
    new_status = request.form.get('status')
    if order.can_transition_to(new_status):
        order.status = new_status
        db.session.commit()
        flash(f'Статус: {order.status_label()}', 'success')
    else:
        flash('Недозволений перехід', 'danger')
    return redirect(url_for('orders.detail', id=id))

```

ДОДАТОК Б

Фрагменти програмного коду. Функція створення звіту

```

from flask import render_template, request, jsonify
from flask_login import login_required, current_user
from app.reports import reports_bp
from app.models import Order, OrderItem, MenuItem, Branch
from app import db
from sqlalchemy import func
from datetime import datetime, date, timedelta

@reports_bp.route('/')
@login_required
def index():
    branches = Branch.query.all() if current_user.is_admin() else []
    date_from = request.args.get('date_from', (date.today() - timedelta(days=30)).isoformat())
    date_to = request.args.get('date_to', date.today().isoformat())
    branch_id = request.args.get('branch_id', type=int)

    q = db.session.query(Order).filter(
        Order.status == 'done',
        Order.created_at >= date_from,
        Order.created_at <= date_to + ' 23:59:59'
    )
    if branch_id:
        q = q.filter(Order.branch_id == branch_id)
    elif not current_user.is_admin():
        q = q.filter(Order.branch_id == current_user.branch_id)

    orders = q.all()
    total_revenue = sum(float(o.total_amount) for o in orders)
    order_count = len(orders)
    avg_check = total_revenue / order_count if order_count else 0

    # Top menu items
    top_q = db.session.query(
        MenuItem.name,
        func.sum(OrderItem.quantity).label('total_qty'),
        func.sum(OrderItem.subtotal).label('total_sum')
    ).join(OrderItem).join(Order).filter(
        Order.status == 'done',
        Order.created_at >= date_from,
        Order.created_at <= date_to + ' 23:59:59'
    )
    if branch_id:
        top_q = top_q.filter(Order.branch_id == branch_id)
    elif not current_user.is_admin():
        top_q = top_q.filter(Order.branch_id == current_user.branch_id)

```

```

top_q.group_by(MenuItem.name).order_by(func.sum(OrderItem.quantity).desc()).limit(10).
all()
top_items =

```

```

# Revenue by branch (admin only)
branch_data = []
if current_user.is_admin():
    branch_q = db.session.query(
        Branch.name, func.sum(Order.total_amount).label('revenue')
    ).join(Order).filter(
        Order.status == 'done',
        Order.created_at >= date_from,
        Order.created_at <= date_to + ' 23:59:59'
    ).group_by(Branch.name).all()
    branch_data = branch_q

return render_template('reports/index.html',
    total_revenue=total_revenue, order_count=order_count,
    avg_check=round(avg_check, 2), top_items=top_items,
    branch_data=branch_data, branches=branches,
    date_from=date_from, date_to=date_to, branch_id=branch_

```

Додаток В

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**
**Факультет інформаційних технологій Декларація про академічну
добросесність та використання ШІ**

Я, Борис Маргарита Юріївна, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної добросесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Інформаційна система обліку та управління роботою мережі кав'ярень.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що:
інструменти ШІ Gemini були використані для:
 - + перекладу іншомовних джерел;
 - + стилістичного редагування та перевірки граматики;
 - + допомоги у написанні/відлагодженні програмного коду;
 - + інше: пошуку додаткових інструментів у розробці програмного продукту.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«22» Травня 2026 р. _____

Маргарита БОРИС