

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ПОГОДЖЕНО

Декан факультету

_____ **Ігор БОЛБОТ**
(підпис)

« ____ » _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри комп'ютерних наук

_____ **Белла ГОЛУБ**
(підпис)

« ____ » _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему: «_____ Програмне забезпечення обліку процесів для малого бізнесу
_____»**

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми

к.т.н., доцент

_____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської

кваліфікаційної роботи

Старший викладач

_____ **Юрій МІЛОВІДОВ**
(підпис)

Виконав

_____ **Владислава КОСІНОВА**
(підпис)

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

« ____ » _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення обліку процесів для малого бізнесу»

затверджена наказом від «19» грудня 2025 р. № 3204 «С»

Термін подання завершеної роботи на кафедру « ____ » _____ 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): облік та опис роботи малого бізнесу.

Перелік питань, що підлягають дослідженню:

1. Малий бізнес к об'єкт дослідження
2. Проєктування інформаційного забезпечення
3. Розробка програмного забезпечення
4. Рекомендації щодо впровадження так експлуатації системи

Дата видачі завдання « ____ » _____ 2026 р.

Керівник _____
бакалаврської
кваліфікаційної роботи

(підпис)

Міловідов Ю.О.

Завдання взяв до виконання

(підпис)

Косінова В.О.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	5
ВСТУП	6
1.МАЛИЙ БІЗНЕС ЯК ОБ’ЄКТ ДОСЛІДЖЕННЯ	8
1.1 Опис обліку процесів як предметна область.	8
1.2. Моделювання предметної області.	10
1.2.1 Діаграма активності.	10
1.2.2 Діаграма послідовності.	10
1.3. Огляд існуючих аналогів систем.	11
1.4.Постановка завдання.	14
2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ	17
2.1 Логічна модель даних у вигляді ER— діаграми.	17
2.1.2 Опис зв’язків між сутностями.	19
2.1.3 Перевірка відповідності нормальним формам.	20
2.2 Вибір системи управління базою даних.	21
2.3 Розробка структури інформаційної системи бази даних.	21
2.4. Схема та фізична структура бази даних.	23
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	26
3.1 Абстракції предметної області.	26
3.2 Моделювання структури та взаємодії об’єктів.	27
3.2.1 Діаграма класів.	27
3.2.2 Діаграма кооперацій.	29
3.3. Організаційна структура програмного забезпечення.	33
3.4. Компонентна структура системи.	34
3.5.Вибір інструментарію для створення прикладного програмного забезпечення.	37
3.6.Алгоритмізація та програмування програмних модулів.	39
4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ	51

4.1 Тестування системи.	51
4.1.1.Приклад тестування.	51
4.2. Вимоги до апаратного та програмного забезпечення.	56
4.3. Склад інсталяційного пакету.	58
ВИСНОВКИ	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	62
ДОДАТКИ	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

- **ПЗ** — програмне забезпечення.
- **ТТН** — товарно-транспортна накладна.
- **CRM** — система управління взаємовідносинами з клієнтами.
- **UML** — Unified Modeling Language.
- **ER** — Entity-Relationship.
- **ORM** — Object-Relational Mapping.
- **TLS** — Transport Layer Security.
- **SSL** — Secure Sockets Layer.
- **HTTP** — HyperText Transfer Protocol.
- **HTML** — HyperText Markup Language.
- **CSS** — Cascading Style Sheets.
- **SQL** — Structured Query Language.
- **МСБ** — малий та середній бізнес.
- **СУБД** — система управління базами даних.
- **НФ** — нормальна форма.

ВСТУП

Актуальність. У сучасних умовах розвиток економіки малого бізнесу є важливим аспектом у формуванні конкурентного середовища для роботи. У зв'язку з цим ефективні функції підприємств потребують чіткої організації роботи та системного підходу до управління процесами. Також необхідне оперативне прийняття рішень, а саме здобуття загальної інформації, необхідність та швидкість обробки замовлень, що підвищують потребу в автоматизації керуючих та промислових процесів. Неефективні кола клієнтів, невеликі замовлення та внутрішні операції є однією з ключових проблем малого бізнесу. Їх часто проводять вручну або за допомогою різного ПЗ. Це призводить до зниження продуктивності, помилок у замовленнях, несвоєчасної звітності, крім того збільшення часу та зменшення якості обслуговування клієнтів.

Використання представленого середовища дає змогу розв'язувати питання через збирання даних в одному місці, автоматичний режим звичайних дій і зменшення часу для обслуговування клієнтів.

Особливі актуальні питання щодо створення спеціалізованих інформаційних систем, які становлять облік процесів малих підприємств та не вимагають значних фінансових витрат для впровадження та обслуговування. Такої системи може бути достатньо простими у рамках бізнесу, ручними та адаптованими для більш тісного знайомства із невеликими підприємствами. Розроблення програмного забезпечення для обліку процесів малого бізнесу, дає змогу забезпечити ефективне управління клієнтською базою, звітністю та внутрішніми операціями підприємства. Щоб виконати поставлені завдання, необхідно: провести аналіз предметної галузі, також слід визначити функціональні вимоги, деталізувати структуру баз даних, реалізувати основні модулі продукту, а також оцінити ефективність використання. Це доведено у процесі створення ПЗ управління бізнесом та її автоматизації.

Мета розробки. Метою розробки є створення програмного забезпечення для обліку процесів малого бізнесу, яке забезпечить автоматизацію основних

операцій підприємства, підвищить ефективність управління даними та сприятиме покращенню якості обслуговування клієнтів.

У проєкті визначені наступні завдання:

1. Аналіз наявних програмних забезпечень обліку процесів для малого бізнесу.
2. Розробити авторизацію для користувачів.
3. Розробити генерацію звітів продажу та обліку товару.
4. Розробити графічний інтерфейс для користувача.
5. Розробити програмні компоненти для створення замовлень та додавання нового товару на склад.

Методи та технології. Для реалізації програмного продукту було використано сучасні інструменти розробки, зокрема мову програмування Python, мову структурованих запитів SQL, систему керування базами даних PostgreSQL та веб-фреймворк Django. Застосування цих технологій забезпечило створення зручної, надійної та масштабованої інформаційної системи для обліку процесів малого бізнесу.

1.МАЛИЙ БІЗНЕС ЯК ОБ'ЄКТ ДОСЛІДЖЕННЯ

1.1 Опис обліку процесів як предметна область.

Створення та використання різних систем ведення малого бізнесу витрачають багато часу та ресурсів. На прикладі сфери малого бізнесу — Складу квітів та рослин можна визначити, що для повноцінної роботи потрібно:

- Середовище обліку складу;
- Клієнтська база;
- ПЗ Управління замовленнями;
- Системна програма введення звітністю.

Недоліками описаних ПЗ є те, що це 4 окремі середовища серед яких потрібно постійно перемикатися для ведення такого бізнесу. Через вищевказані мінуси було розроблено систему яка вирішує питання швидкості, легкості та зручності керування бізнесом. Даний софт дозволяє прибрати огріхи людського фактору — помилки при внесенні даних, втрату інформації та сортування замовлень, відсутність аналітики, адже неможливо без допомоги обчислювальної техніки швидко порахувати суму продажів та списати товар зі складу.

Також у створенні будь-якого програмного забезпечення є невід'ємною частиною роботи розробка аналізу вимог до програмної системи даний аналіз зазвичай поділяють на 2 типа вимог: функціональні та нефункціональні.

Першими розглянемо функціональні вимоги до розробленої системи:

- додавання нового товару на склад.,
- Оновлення залишків на складу,
- формування замовлень,
- додавання нових клієнтів,
- формування звітів за місяць.

Тепер розглянемо нефункціональні вимоги:

- цілісність даних,
- продуктивність,

- зручність та інтуїтивність.

Детальніше можна розглянути представлену інформацію на діаграмі прецедентів рис 1.

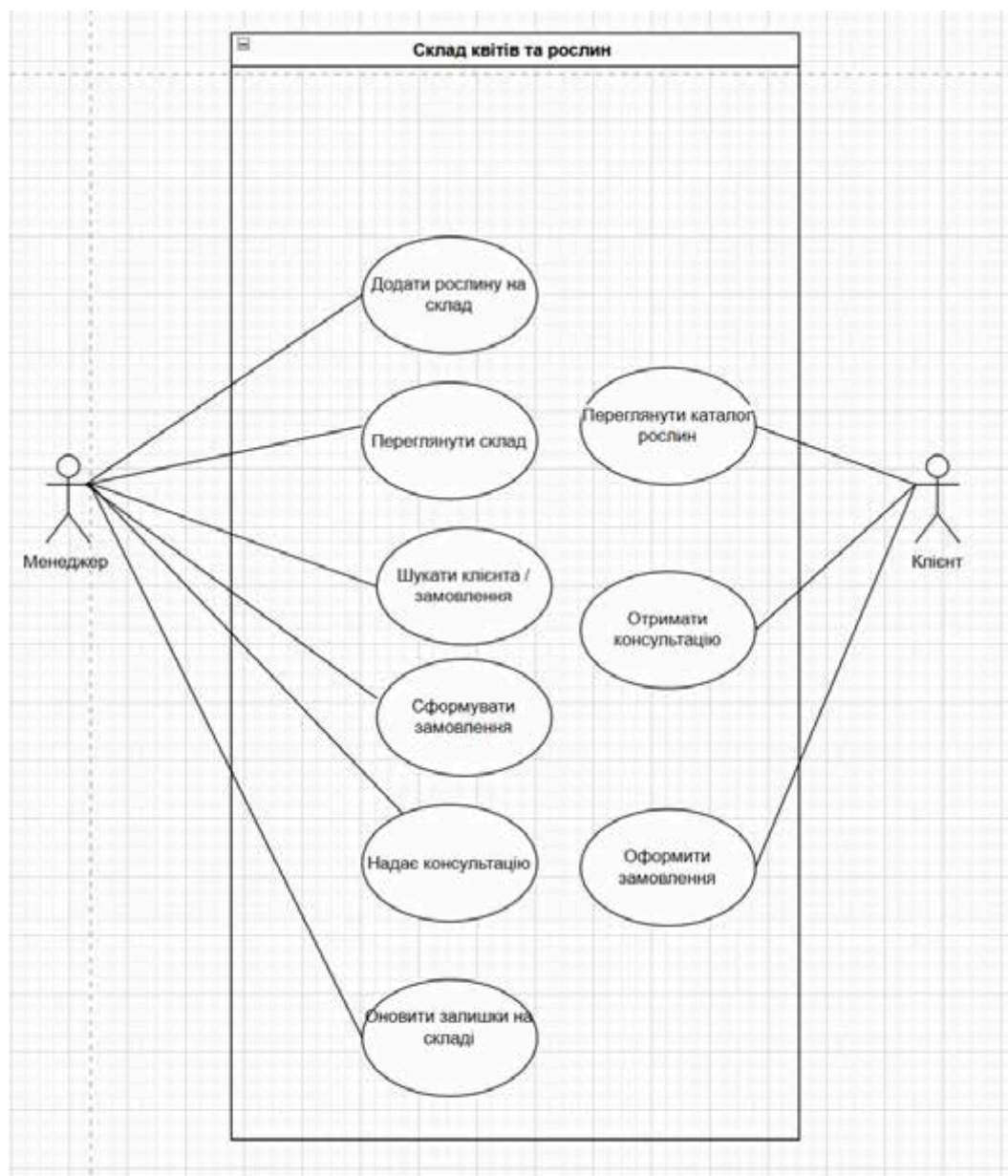


Рис.1.Діаграма прецедентів.

На даній діаграмі чітко зображений процес складського обліку: Менеджер «Переглядає склад», бачить дефіцит і використовує «Оновити залишки» або «Додати рослину» .

1.2. Моделювання предметної області.

Моделювання предметної області – це створення візуальної структури розроблюваної системи. До моделювання входять дані діаграми:

- Діаграма активності;
- Діаграма послідовності;

1.2.1 Діаграма активності.

Наступною розглядається діаграма активності. Даний вид діаграм зображено в Додатку А, який відображає алгоритм роботи системи для досягнення результату. В діаграмі активності представлено алгоритм обробки замовлення менеджером. Процес починається з формування замовлення, далі перевірка залишків на складі, якщо кількості товару вистачає для замовлення, замовлення успішно виконується, якщо товару не вистачає, виконується процес дозамовлення товару, згодом замовлення також успішно виконується.

1.2.2 Діаграма послідовності.

В додатку Б представлена діаграма послідовності. Основний фокус припадає на часовій послідовності обміну повідомленнями між різними об'єктами для виконання різних сценаріїв. На діаграмі змодельовано повний цикл продажу та обліку рослин, що складається з:

- Адміністрування: Менеджер додає нову рослину та отримує підтвердження від системи.
- Передпродажний етап: Менеджер здійснює пошук товару на Складі та надає консультацію Клієнту.
- Етап замовлення: Клієнт приймає рішення та ініціює оформлення замовлення. Менеджер, своєю чергою, створює замовлення в системі.
- Завершення операції: Менеджер підтверджує продаж Клієнту, після чого система автоматично виконує «Облік залишків на складі» (синхронізація даних).

1.3. Огляд існуючих аналогів систем.

Для ведення бізнесу є велика кількість різних допоміжних систем. Основні типи таких систем є customer relationship management, enterprise resource planning, project management. В Україні найпоширенішими системами такого типу є RO App, Sales Drive та if.team, які автоматизують замовлення, склад, продажі та CRM.

RO App використовують здебільшого для сервісного бізнесу а саме автосервіси, ремонти, салони. Система керує замовленням від прийому до видачі, веде склад, фінанси, аналітику; автоматично надсилає смс клієнтам та друкує звіти.

Ціна від 500 грн/міс, мобільний доступ, інтеграції з телефонією та платежами; ідеально для малого бізнесу з ремонтом/обслуговуванням.

На рис 2. Представлено інтерфейс системи RO App.



Рис.2. Інтерфейс системи RO App.

Sales Drive - для інтернет-магазинів та торгівлі з автоматизацією замовлень і складу. Обробляє замовлення з маркетплейсів (Prom.ua, Rozetka),

створює ТТН, веде облік складу, статусів і продажів; велика кількість інтеграцій.

Від 615 грн/міс за 1-2 користувачів (+219 грн за кожного); безкоштовне налаштування, помісячна оплата. Підходить для e-commerce, де потрібно швидко обробляти замовлення без помилок.

На рис 3. Показано інтерфейс Sales Drive

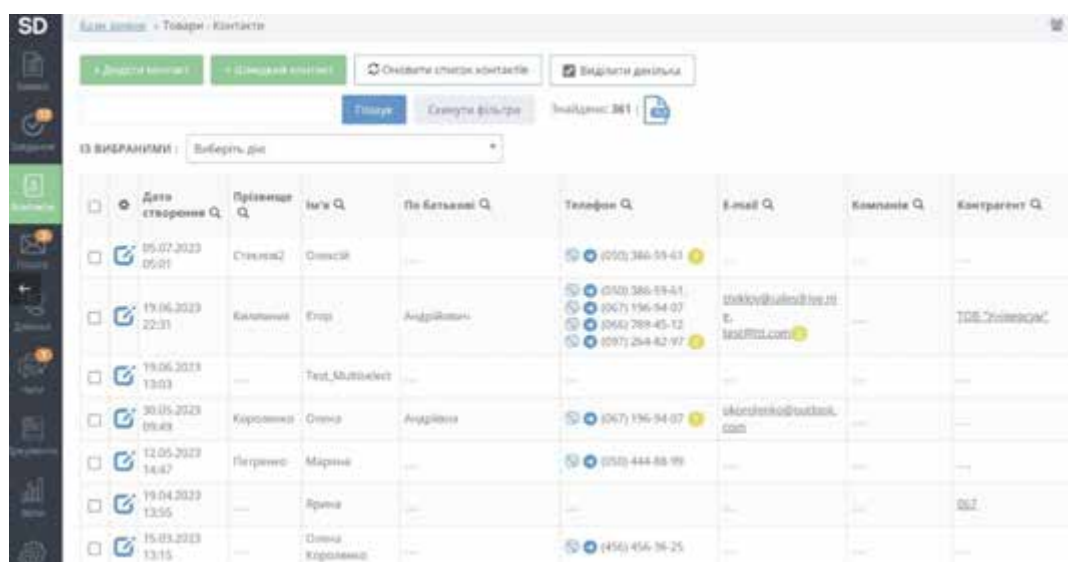


Рис.3.Інтерфейс системи Sales Drive.

if.team - спеціалізуються на проєктних компаніях (агентства, ІТ, послуги), ERP, CRM та контролем витрат. Керує проєктами, CRM (воронка продажів), HR (навантаження, відпустки), фінансами; контролює прогрес і витрати. Хмарна українська ERP для малого/середнього бізнесу; фокус на задачах, рейтах і команді.

Ціна за запитом, але доступна для МСБ; переходять з ClickUp/Trello для повного обліку.

На рис.4 Зображено інтерфейс if.team

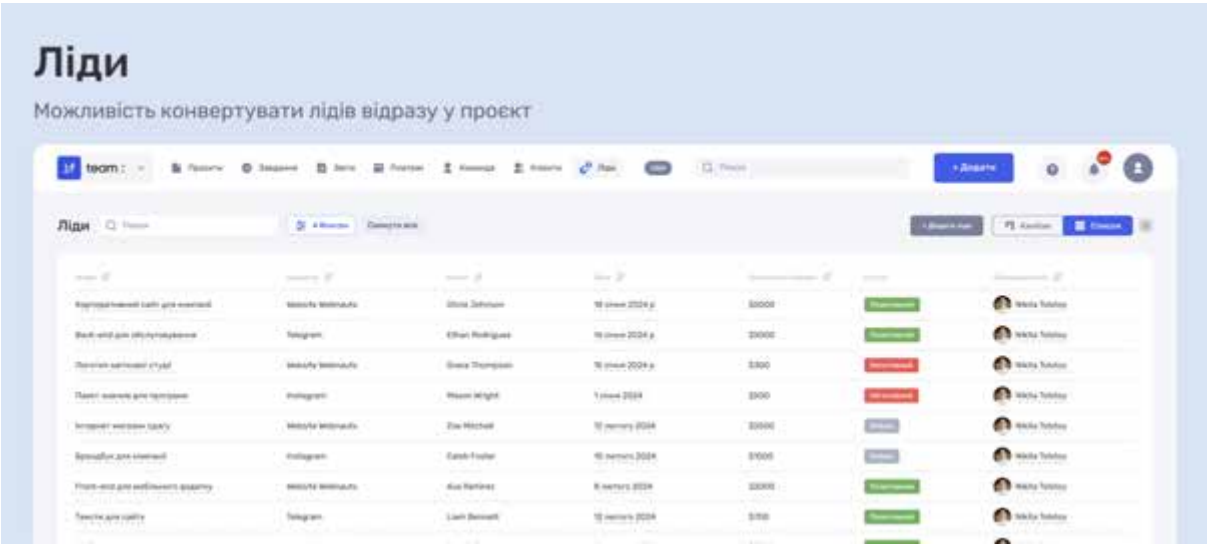


Рис.4.Інтерфейс системи if.team.

Нижче наведено порівняльну таблицю (Табл.1.) з плюсами та мінусами використання даних програм.

Таблиця 1. Порівняння плюсів та мінусів програм.

	Простота інтерфейсу	Ціна	Супровід	Повнота інструментарію	Швидкість навчання з ПЗ
RO App	+	+	-	+	+
Sales Drive	-	+	+	+	-
if.team	-	-	+	+	+

З даного порівняння можна визначити, що всі системи знаходяться на однаковому рівні але система RO App є найбільш зручною, доступною за ціною, повнотою інструментарію та швидким навчання співробітників працювати з даним додатком.

1.4.Постановка завдання.

Розробити веб систему комплексного обліку бізнес процесів для малого бізнесу в Україні яка автоматизує ключові операції, підвищить ефективність персоналу та забезпечить швидкий доступ до аналітичних даних.

Вхідні дані

До вхідних даних належать відомості, які користувач вводить у систему для подальшої обробки:

- дані для авторизації користувача: логін, пароль;
- дані про клієнта: ПІБ, телефон, електронна пошта, адреса, тип клієнта;
- дані про товар: назва, категорія, ціна, кількість, опис;
- дані про замовлення: дата замовлення, клієнт, перелік товарів, кількість, статус замовлення;
- дані для формування звітів: період звіту, тип звіту;
- дані про надходження товару на склад;
- дані про зміну статусу замовлення.

Вихідні дані

Вихідними даними є результати обробки інформації, які система формує для користувача:

- список зареєстрованих клієнтів;
- перелік товарів на складі;
- сформовані замовлення;
- звіти з продажів;?
- звіти з обліку товарів;
- інформація про залишки товарів;
- повідомлення про успішну авторизацію або помилки входу;
- повідомлення про успішне створення або редагування записів;
- аналітичні дані щодо замовлень та продажів.

Умовно-постійні значення

До умовно-постійних значень належать дані, які змінюються рідко і використовуються системою багаторазово:

- перелік ролей користувачів: адміністратор, менеджер, працівник;
- статуси замовлень: нове, в обробці, виконано, скасовано;
- категорії товарів;
- типи клієнтів;
- межі мінімальних залишків товарів;
- шаблони звітів;
- довідники одиниць вимірювання;
- фіксовані параметри доступу до системи;
- налаштування інтерфейсу та форматів відображення даних.

Функціональні вимоги

Функціональні вимоги описують, що саме повинна робити система:

- забезпечувати реєстрацію та авторизацію користувачів;
- надавати доступ до системи відповідно до ролі користувача;
- забезпечувати введення, редагування та видалення даних про клієнтів;
- забезпечувати введення, редагування та видалення даних про товари;
- створювати та редагувати замовлення;
- автоматично оновлювати залишки товарів на складі;
- формувати звіти з продажів за вибраний період;
- формувати звіти з обліку товарів;
- відображати інформацію про поточні залишки;
- надавати пошук і фільтрацію даних;
- забезпечувати збереження інформації в базі даних;
- виводити повідомлення про помилки або успішне виконання дій;

- підтримувати зручний графічний інтерфейс користувача.

Нефункціональні вимоги

Нефункціональні вимоги описують, якою має бути система за якістю роботи:

- зручність використання — інтерфейс має бути інтуїтивним;
- швидкодія — система повинна швидко обробляти запити користувача;
- надійність — дані не повинні втрачатися під час роботи системи;
- безпека — доступ до системи має бути захищений паролем;
- масштабованість — система повинна підтримувати розширення функціоналу;
- супроводжуваність — програму має бути легко оновлювати та змінювати;
- цілісність даних — інформація в базі даних повинна залишатися узгодженою;
- доступність — система має бути доступною для роботи у робочий час без збоїв;
- адаптивність — можливість використання на різних пристроях або з різними роздільними здатностями екрана.

Отже, було визначено основні проблеми, що пов'язані з використанням окремих програмних продуктів та друкованої документації, зокрема, складність ведення обліку, ризик втрати даних та виникнення помилок через людський фактор. Було визначено функціональні та нефункціональні вимоги до системи та проведено предметне моделювання за допомогою UML-діаграм, що дозволило описати структуру системи та взаємодію її компонентів.

Крім того, було проведено аналіз існуючих аналогів, зокрема RO App, SalesDrive та if.team. На основі порівняння було визначено переваги та недоліки сучасних облікових систем. У результаті було розроблено веб-систему, яка автоматизувала облік матеріалів, клієнтів та замовлень, забезпечила

централізоване зберігання даних та підвищила ефективність роботи малого бізнесу.

2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER— діаграми.

Розроблення логічної моделі для обліку процесів є одною з основних етапів створення системи. За допомогою ER— діаграми можна відобразити логіку «сутність — зв'язок». Дана діаграма дозволяє побачити структуру зберігання, обробку та взаємозв'язок між ними, створюючи основу для фізичної структури бази даних. Також за підтримки ER— діаграми є можливість візуалізувати вищевказану логіку, що робить розуміння структури системи набагато легше на етапі проєктування.

На додатку В зображено ER— діаграму.

У розробленій системі обліку процесів для малого бізнесу визначена головні ролі: клієнт, менеджер, рослини, склад, замовлення, позиція замовлення. Кожна з перерахованих сутностей має власний набір атрибутів, що чітко охарактеризую їх або ідентифікує. Для зв'язку сутностей використовуються за зовнішні ключі для гарантування цілісності та зв'язку у базі даних.

2.1.1. Опис сутностей та атрибутів.

Client

- **client_id (PK)** — унікальний ідентифікатор клієнта
- **name** — ім'я або назва клієнта
- **phone** — номер телефону клієнта
- **email** — електронна пошта клієнта
- **address** — адреса клієнта
- **client_type** — тип клієнта (постійний, новий, оптовий тощо)

Order

- **order_id (PK)** — унікальний ідентифікатор замовлення
- **client_id (FK)** — посилання на клієнта, який оформив замовлення
- **manager_id (FK)** — посилання на менеджера, який обробляє

замовлення

- **order_date** — дата створення замовлення
- **status** — поточний статус замовлення
- **total_amount** — загальна сума замовлення

Manager

- **manager_id (PK)** — унікальний ідентифікатор менеджера
- **name** — ім'я менеджера
- **phone** — номер телефону менеджера
- **email** — електронна пошта менеджера

Order_Item

- **order_item_id (PK)** — унікальний ідентифікатор позиції замовлення
- **quantity** — кількість товару в замовленні
- **price** — ціна товару
- **product_id (FK)** — посилання на товар
- **order_id (FK)** — посилання на замовлення

Product

- **product_id (PK)** — унікальний ідентифікатор товару
- **name** — назва товару або рослини
- **category** — категорія товару
- **price** — ціна товару
- **supplier** — постачальник товару

Inventory

- **inventory_id (PK)** — унікальний ідентифікатор запису складу
- **product_id (FK)** — посилання на товар на складі
- **quantity** — кількість товару на складі

- `min_quantity` — мінімальна допустима кількість товару
- `location` — місце зберігання товару
- `arrival_date` — дата надходження товару
- `order_item_id (FK)` — посилання на позицію замовлення

2.1.2 Опис зв'язків між сутностями.

У переважній більшості використовується зв'язок один – до – багатьох (1:M). Лише один зв'язок один – до – одного використовується у `Product` — `Inventory`, оскільки кожен товар має запис про залишок на складі. У системі склад містить інформацію про кількість, місце зберігання та дату надходження товару. Нижче наведено детальний опис зв'язків:

- Таблиця `Client` пов'язана з таблицею `Order` зв'язком один-до-багатьох, оскільки один клієнт може мати багато замовлень, але кожне замовлення належить тільки одному клієнту. У цьому випадку зовнішнім ключем є `client_id` у таблиці `Order`.

- Таблиця `Manager` пов'язана з таблицею `Order` зв'язком один-до-багатьох, тому що один менеджер може обслуговувати багато замовлень, але кожне замовлення закріплене лише за одним менеджером. Зовнішнім ключем є `manager_id` у таблиці `Order`.

- Таблиця `Order` пов'язана з таблицею `Order_Item` зв'язком один-до-багатьох, оскільки одне замовлення може містити кілька позицій, а кожна позиція належить лише одному замовленню. Тут зовнішнім ключем є `order_id` у таблиці `Order_Item`.

- Таблиця `Product` пов'язана з таблицею `Order_Item` зв'язком один-до-багатьох, бо один товар може входити до багатьох позицій замовлень, але кожна позиція містить лише один товар. Зовнішнім ключем є `product_id` у таблиці `Order_Item`.

- Таблиця `Inventory` пов'язана з таблицею `Product` зв'язком один-до-одного або один-до-багатьох, залежно від логіки обліку складу. Якщо для кожного товару ведеться один запис залишку, то це зв'язок один-до-одного.

Якщо ж товар може зберігатися в кількох локаціях, то зв'язок буде один-до-багатьох. Зовнішній ключ — `product_id` у таблиці `Inventory`.

2.1.3 Перевірка відповідності нормальним формам.

Для забезпечення ефективності зберігання обробки та надлишковості даних у базі даних потрібно провести модель до форми що відповідає принципу нормалізації. Нормалізація — це процес організації та відношень у базі даних таким чином щоб зменшити дублювання даних і забезпечити цілісність. Головними нормальними формами є перша (1НФ), друга (2НФ) і третя (3НФ). У процесі створення бази даних було було використано принципи реляційної моделі та дотримано 1НФ, 2НФ нормалізації та дотримано основні аспекти третьої нормальної форми, нижче наведено обґрунтування.

Перша нормальна форма згідно діаграми відповідає тому що:

- Усі атрибути мають атомарні значення;
- Дані клієнтів, замовлень, товарів, менеджерів і складу зберігається в окремих таблицях;
- Кожен запис має унікальний ідентифікатор;

Друга нормальна форма система відповідає даній формі через те що:

- У таблицях відсутні часткові залежності дані про клієнта, менеджера і товару не дублюються;
- Усі ключові атрибути залежать від повного первинного ключа;

Третя нормальна форма система наближене до третьої нормальної форми, оскільки:

- Зменшено дублювання інформації що спрощує оновлення даних і забезпечує цілісність структури;
- Дані розподілені за змістом предметної області тобто кожна таблиця відповідає окремі сутності;

З вище перерахованого можна зробити висновок, що під час проєктування бази даних використано принципи реляційної моделі та нормалізації. Структура таблиць спроектована таким чином, щоб мінімізувати надлишковість даних і

забезпечити високу узгодженість інформації. Однак для окремих сутностей можливе додаткове доопрацювання з метою повного приведення схеми до третьої нормальної форми.

2.2 Вибір системи управління базою даних.

Розроблена модель бази даних перебуває у наближенні до третьої нормальної форми, що надає структурованість, мінімальну надлишковість та логічну цілісність даних для зберігання і обробки даних — найкращим рішенням є реляційна система управління бази даних. Це зумовлено тим що СУБД мають ефективну підтримку складних зв'язків між таблицями, коректність операцій над даними, гарантуючи логіку.

У процесі вибору відповідної СУБД було обрано клієнт-серверну СУБД PostgreSQL. Даний вибір був зроблений через те що: підходить для складних запитів, добре працює з фільтрами, агрегація та аналогічними запитами — зручно для систем, надійність і цілісність даних підтримує обмеження, зовнішні ключі та контроль узгодженості, що важливо для облікових програм, база працює добре і з невеликими проектами, і більшими системами коли даних стає багато. Широкі можливості підтримують додаткові типи даних функції та розширення, тому його можна адаптувати під різні задачі.

Тому, PostgreSQL є зручним інструментом для роботи з Python, оскільки забезпечує надійне зберігання даних, підтримку складних запитів і цілісності інформації. Завдяки сумісності з бібліотеками Python та можливості використання ORM, розробка прикладних систем стає простішою та ефективнішою.

2.3 Розробка структури інформаційної системи бази даних.

Розробка структури інформаційної бази передбачає визначення логічної організації даних, що використовуються в системі. На цьому етапі встановлюються основні таблиці бази даних, їх атрибути, ключі та взаємозв'язки між сутностями.

Правильно спроектована інформаційна база дає змогу ефективно

зберігати, оновлювати та обробляти дані, а також забезпечує можливість швидкого отримання необхідної інформації для формування звітів, аналізу діяльності підприємства та підтримки управлінських рішень. Саме тому етап розробки структури інформаційної бази є одним із ключових у створенні програмного забезпечення для обліку процесів малого бізнесу.

Створення бази даних є найголовнішим етапом у процесі розробки інформаційної системи. Це важлива дія, що забезпечує основу для подальшого зберігання та організації даних. Цей код. На рис.5 показує процес створення нової бази після чого можна приступати до створення таблиць, запитів та інших компонентів.

```
CREATE DATABASE storedipl;
```

Рис.5. Процес створення нової бази.

Створення таблиць важливим етапом у розробці структури бази даних, адже кожна таблиця показує окремий об'єкт або сутність системи що містить відповідні атрибути. Для забезпечення ефективного зберігання даних потрібно визначити правильну структуру таблиць що відповідає вимогам бізнес логіки.

Створення таблиць для зберігання інформації про клієнтів та замовлення було реалізовано за допомогою-запитів, через команду. На прикладі нижче наведених таблиць буде продемонстровано процес створення: фрагмент запиту створення таблиці клієнт що зображено на рис.6, таблиця замовлення на рис.7 .

```
CREATE TABLE clients (
  client_id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  phone VARCHAR(20),
  email VARCHAR(100),
  address TEXT,
  client_type VARCHAR(20) CHECK (client_type IN ('retail', 'wholesale'))
);
```

Рис.6. Фрагмент запиту створення таблиці клієнт.

```
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    client_id INT,
    manager_id INT,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) CHECK (status IN ('new', 'processing', 'completed', 'cancelled')),
    total_amount NUMERIC(10,2),
```

Рис.7. Фрагмент запиту створення таблиці замовлення.

Перед створенням даних таблиць була здійснена перевірка на наявність аналогічних таблиць у базі даних, що зображена на рис.8. Це дозволяє уникнути помилок при спробі створення дубліката таблиці.

```
SELECT EXISTS (
    SELECT FROM information_schema.tables
    WHERE table_name = 'clients'
);
```

Рис.8. Перевірка на наявність аналогічних таблиць у базі даних.

Повний код всіх таблиць надано в додатку Г

Механізм автоматизації контролю за даними в базі є тригери, які виконуються при початку певної дії. Вони потрібні для реалізації логіки перевірки та підтримки цілісності даних.

На рис.9 наведено приклад тригера.

```
CREATE TRIGGER trg_update_inventory
AFTER INSERT ON order_items
FOR EACH ROW
EXECUTE FUNCTION update_inventory_after_sale();
```

Рис.9. Тригер.

У створенні доступу до бази даних було реалізовано механізм створення користувачів за поділом на ролі відповідно до їх функціональних обов'язків. Такий підхід дозволяє ефективно керувати правами доступу аби забезпечити належний рівень безпеки даних. На прикладі створення користувача можна дослідити основні етапи нового користувача до системи. На показано створення облікового запису, на-безпосереднє створення користувача в базі даних.

2.4. Схема та фізична структура бази даних.

Схема бази даних відображає логічну організацію даних у системі та

визначає склад основних таблиць, їхні атрибути, ключі й зв'язки між ними. У розроблюваній інформаційній системі схема бази даних побудована відповідно до предметної області малого бізнесу та включає таблиці для зберігання інформації про клієнтів, замовлення, менеджерів, товари, складські залишки, категорії, статуси та постачальників. Така структура забезпечує впорядковане зберігання даних і створює основу для подальшої обробки інформації.

Фізична структура бази даних описує спосіб реалізації логічної моделі у конкретній системі керування базами даних. У межах даного проєкту фізична реалізація бази даних була виконана за допомогою PostgreSQL, що дало змогу забезпечити надійне зберігання інформації, підтримку зв'язків між таблицями, а також контроль цілісності даних за допомогою первинних і зовнішніх ключів. Для реалізації структури бази даних було створено відповідні таблиці, визначено типи даних для кожного атрибута та налаштовано обмеження, які запобігають виникненню помилок при роботі системи.

Окрему увагу під час фізичного проєктування було приділено оптимізації доступу до даних і зменшенню надлишковості інформації. Для цього основні сутності предметної області було розділено на окремі таблиці, а довідникові значення винесено в окремі структури. Такий підхід дає змогу ефективно виконувати операції додавання, оновлення, пошуку та видалення даних, а також формувати звіти та аналітичну інформацію без втрати цілісності бази даних.

Загалом у базі даних реалізовано 6 основних таблиць, які описують ключові сутності предметної області малого бізнесу:

- Client — зберігає персональні дані клієнтів;
- Manager — містить дані про менеджерів, які працюють із замовленнями;
- Order — зберігає інформацію про замовлення;
- Product — містить дані про товари;
- Order_Item — містить позиції замовлення;
- Inventory — використовується для обліку складських залишків.

Кожна таблиця має первинний ключ, який забезпечує унікальність записів. Для встановлення взаємозв'язків між таблицями використовуються зовнішні ключі. Наприклад, поле `client_id` у таблиці `Order` пов'язане з таблицею `Client`, а `manager_id` — з таблицею `Manager`, що забезпечує зв'язок між замовленнями, клієнтами та відповідальними працівниками. Таблиця `Order_Item` містить зовнішні ключі `order_id` і `product_id`, які пов'язують позиції замовлення з конкретним замовленням і товаром. Таблиця `Inventory` пов'язана з таблицею `Product` через поле `product_id`, що забезпечує облік залишків за кожним товаром.

Для коректного представлення даних використовувалися такі типи:

- `VARCHAR(n)` — для зберігання текстових даних, таких як ПІБ, назви товарів, адреси, телефони та електронні адреси;
 - `INTEGER` — для числових значень, наприклад кількості товарів або ідентифікаторів зовнішніх ключів;
 - `NUMERIC(10,2)` — для зберігання грошових значень, зокрема цін та сум замовлень;
 - `DATE` або `TIMESTAMP` — для зберігання дат створення замовлення або надходження товару.
- На схемі бази даних наочно відображено всі таблиці, поля, типи даних та взаємозв'язки між ними. Така схема дає змогу краще зрозуміти логіку побудови інформаційної бази, зручно аналізувати зв'язки між сутностями та визначати залежності, необхідні для ефективного функціонування системи.

Отже, фізична структура бази даних є логічно впорядкованою, гнучкою та масштабованою. Вона враховує специфіку обліку процесів малого бізнесу, забезпечує надійне зберігання інформації та підтримує механізми цілісності даних. Такий підхід дає змогу легко розширювати систему в майбутньому, додаючи нові сутності або функціональні можливості без шкоди для стабільності роботи та якості даних.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Абстракції предметної області.

На стадії проєктування програмного забезпечення важливою складовою є опис ключових абстракції предметної області, які зображають реальні об'єкти, процеси і учасників, характерні для малого бізнесу. Абстракції-це узагальнені моделі що містять у собі важливі властивості та обов'язки відповідних сутностей, які необхідні для реалізації логіки системи. На рис.10 показано загальну структуру абстракції предметної області, яка є основою для проєктування системи.

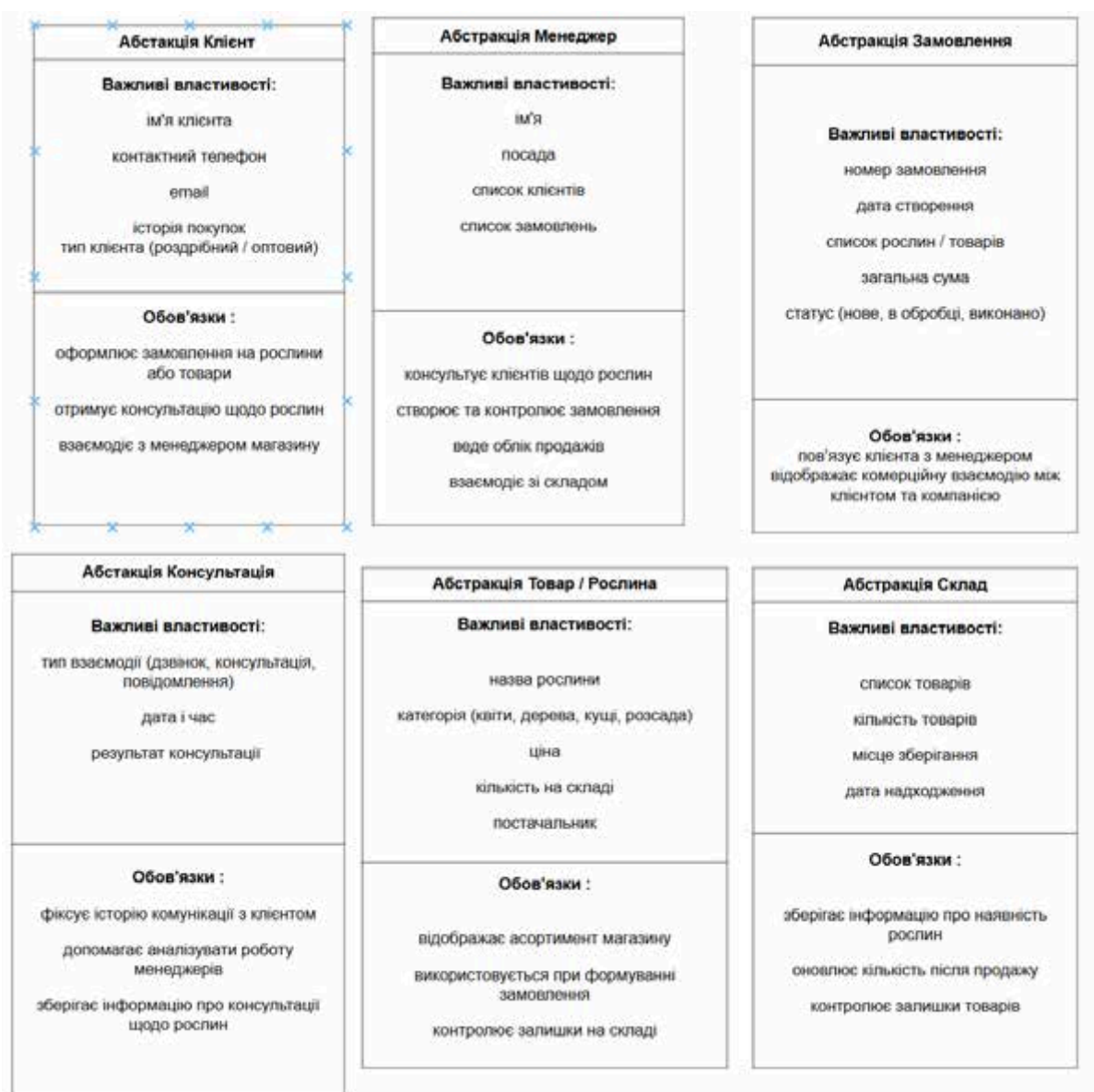


Рис.10. Абстракція предметної області.

Кожна з абстракцій показує типові ролі, елементи та процеси, що мають бути в роботі малого бізнесу. Зокрема, в даних абстракція показані суб'єкти як: клієнт, менеджер, замовлення, товар, склад та консультація.

Усі абстракції визначають характерні властивості та обов'язки які потрібні для функціональної структури системи. Це дозволяє описати логіку взаємодії між користувачами, встановити зв'язки між об'єктами системи та забезпечити належну обробку даних без втрати логіки або даних.

Отже, абстракції предметної області дозволяють створити прозору і логічну модель взаємодії між усіма ключовими учасниками процесів у малому бізнесі. Також це забезпечує структурування даних та функціоналу майбутнього інформаційного середовища, що даватиме автоматизацію основних бізнес-процесів підприємства.

3.2 Моделювання структури та взаємодії об'єктів.

Подальшим етапом є моделювання структури та взаємодії об'єктів, що дає змогу детально визначити складові майбутньої системи та встановити зв'язки між ними. На цьому етапі важливо не лише відобразити загальну логіку роботи програмного продукту, а й чітко описати всі його основні елементи, їхні ролі, властивості та механізми взаємодії в межах виконання окремих сценаріїв. Таке моделювання дозволяє сформувати цілісне уявлення про архітектуру системи, визначити її внутрішню структуру та створити основу для подальшої реалізації програмного забезпечення. Дана модель представлена у додатку І.

Для моделювання використовується стандартна мова UML, яка є загальноприйнятим інструментом для побудови об'єктно-орієнтованих моделей. Особливу роль на цьому етапі відіграють діаграми, що відображають як статичні, так і динамічні аспекти системи.

3.2.1 Діаграма класів.

Діаграма класів є одним із основних типів UML-діаграм, призначеним для зображення об'єктної структури системи. Вона демонструє основні класи, які було ідентифіковано на етапі аналізу, їхні атрибути, методи та зв'язки між ними.

Особливу увагу діаграма приділяє асоціаціям між класами, що визначають логіку взаємодії об'єктів у реальній системі.

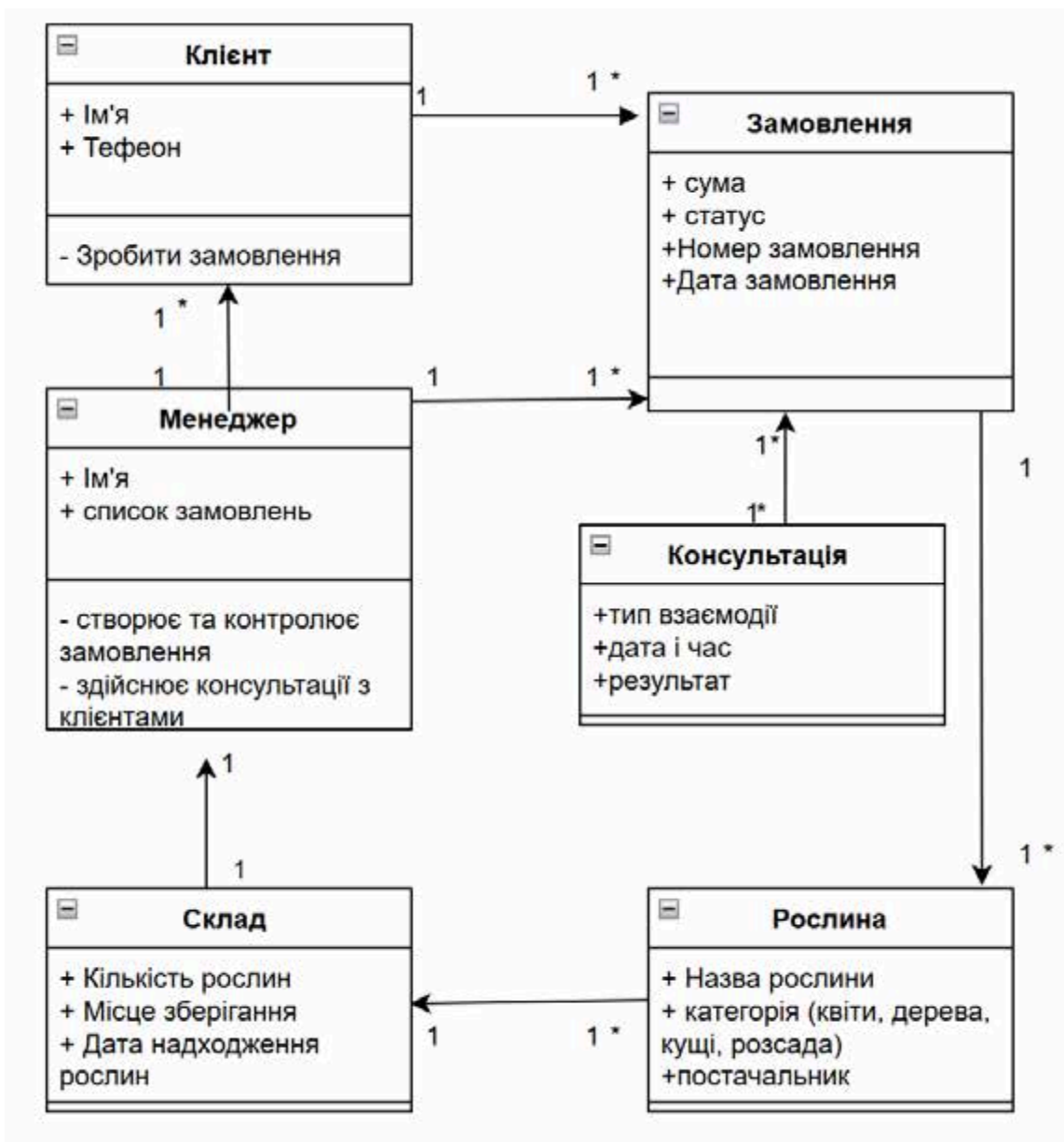


Рис.11. Діаграма класів.

Показана діаграма є основою для подальшого проектування реалізації функціональної системи. Вона демонструє, які саме об'єкти існують у системі, які дані вони зберігають, які дії можуть виконувати, а також як саме дані об'єкти пов'язані між собою.

Діаграма класів включає такі сутності: Клієнт, Замовлення, Менеджер, Консультація, Рослина, Склад. Кожен клас має власний набір атрибутів, які

зберігає ключову інформацію, а також методи, що відповідають за функціонально поведінку системи.

Основний фокус приділено асоціація між класами, які чітко показують ролі та взаємозв'язки. Наприклад, Менеджер зв'язався з Клієнтом та вніс замовлення. Це реалізовано у вигляді асоціації типу один до багатьох.

Клас «Менеджер» пов'язаний з «Замовлення» у відношенні один до багатьох, що показує, що менеджер може бути один, а замовлень може бути багато. Клас «Клієнт» асоційований з класом «Замовлення», адже клієнту під час консультації з менеджером присвоюється номер замовлення. Це дозволяє системі фіксувати замовлення та списувати товар з складу.

Тому, діаграма класів з асоціаціями формує базу для подальшої реалізації системи, оскільки вона узагальнює ключові об'єкти та зв'язки між ними, і на цьому етапі вже продемонстровано структуру майбутнього функціоналу. Вона є інструментом, який поєднує одразу декілька підходів, а саме аналітичний і проєктний, що забезпечує перехід від логічної моделі до фізичної реалізації.

3.2.2 Діаграма кооперацій.

Дана діаграма показує саме вона показує з яких об'єктів складається система та як вони між собою взаємодіють. Вище вказана діаграма представлена у рис.12,13,14 та розділена на 3 окремі блоки, задля кращого розуміння працездатності системи.

Перший блок присвячено взаємодії менеджера та клієнта.

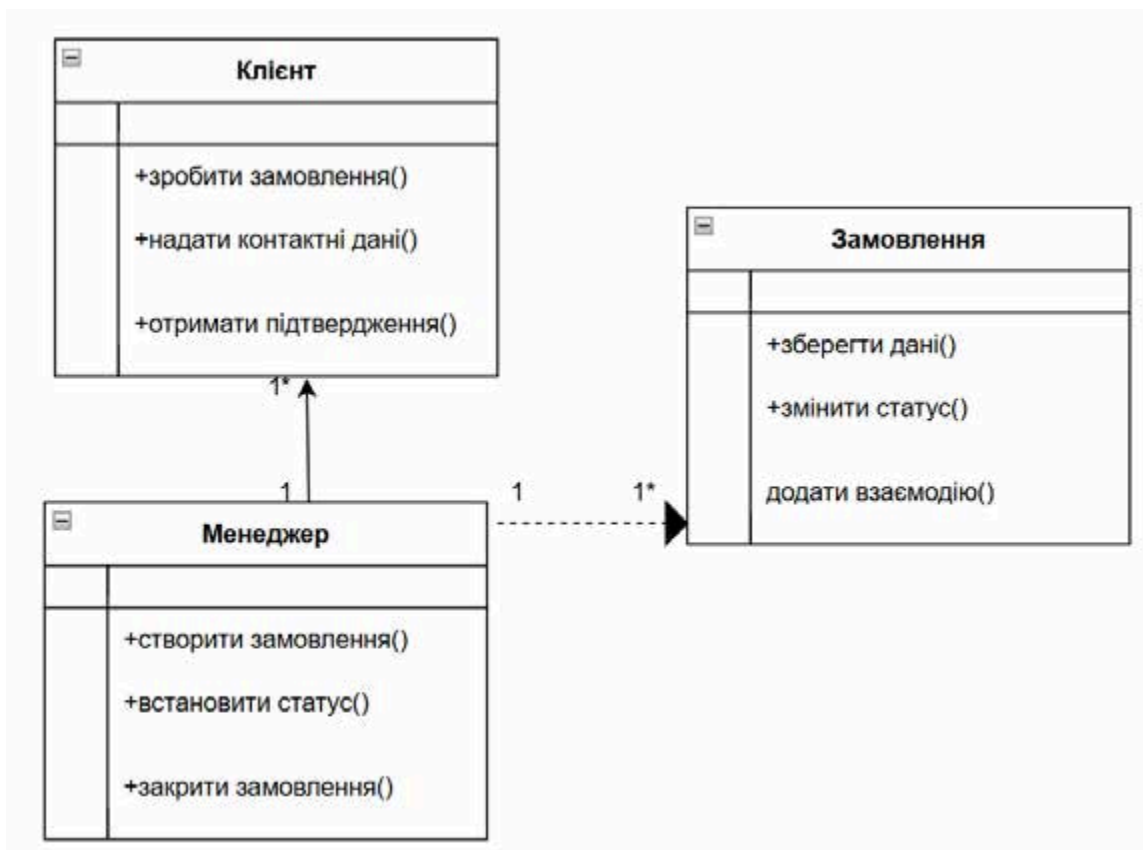


Рис.12. Діаграма кооперації(взаємодія менеджера та клієнта.)

Діаграма складається з 3 сутностей, а саме Клієнт, Менеджер та Замовлення. В побудові діаграм використано зв'язок асоціація з кратністю.

До кожної сутності входять атрибути, що формують інформаційний портрет об'єкта (наприклад, ціна для рослини або статус для замовлення) та дозволяють системі коректно відображати дані користувачеві.

Менеджер – Клієнт:

Використаний зв'язок направлена асоціація. Стрілка від Менеджера до Клієнта вказує на те, що Менеджер володіє інформацією про клієнта та може надати консультацію або підтвердження замовлення.

Менеджер – Замовлення:

Використаний зв'язок асоціація з кратністю. Стрілка від Менеджера до Замовлення вказує, що один менеджер може відпрацьовувати багато замовлень. В даному випадку Менеджер виступає контролем, саме він встановлює статус

та закрити замовлення. Кратність 1* показує, що Менеджер працює в потоковому стані оброблення замовлень.

Другий блок присвячено здійсненню замовлення клієнтом.

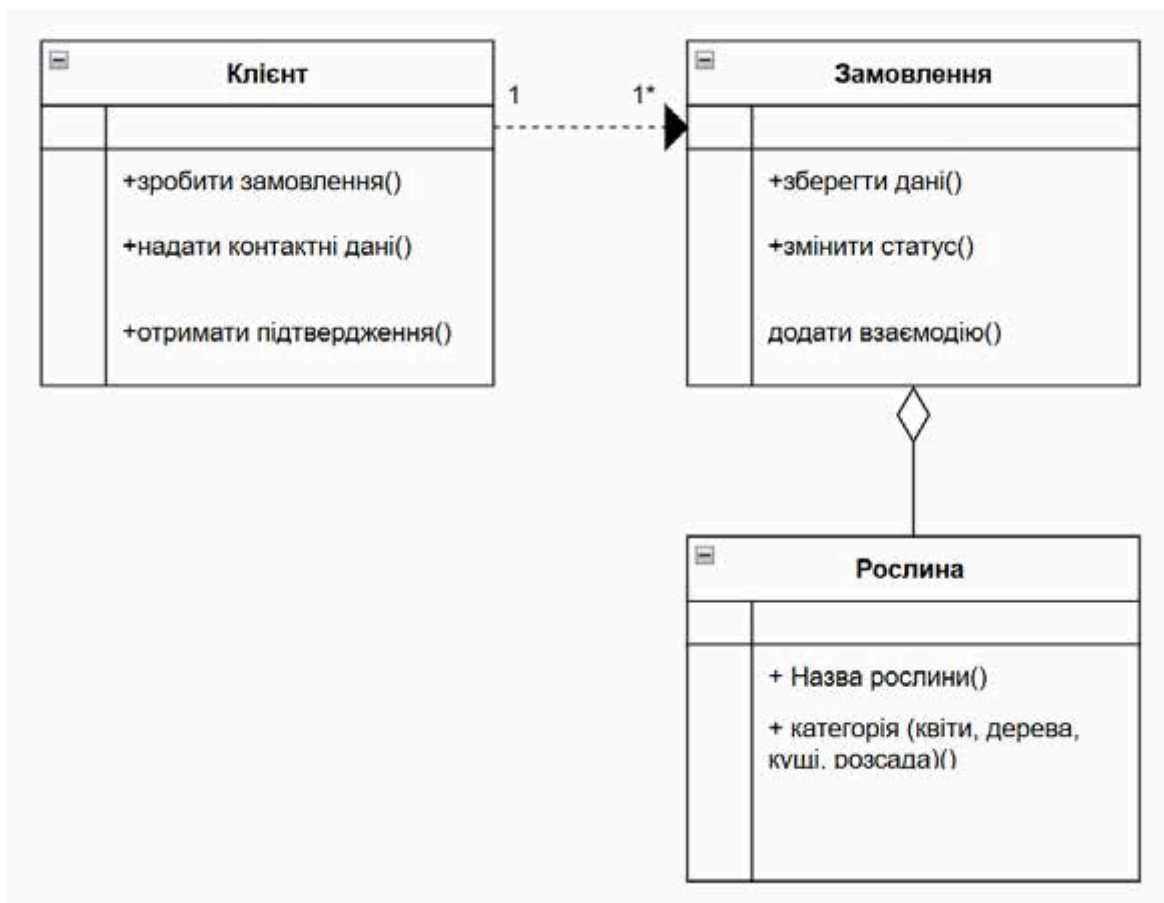


Рис.13. Діаграма кооперації(здійснення замовлення клієнтом).

До даної діаграми входять 3 сутності: Клієнт, Замовлення та Рослина. Використано зв'язки між Клієнт та Замовлення – асоціація з кратністю, між Рослина та Замовлення агрегація. Наведені сутності включають функціональні атрибути, які визначають стан об'єкта (як-от назва рослини чи статус замовлення),а також програмні методи, що дозволяють клієнту ініціювати замовлення, а системі — забезпечувати збереження даних та зворотний зв'язок.

Клієнт – Замовлення:

Зв'язок: асоціація з кратністю. Саме цей зв'язок показує, що один клієнт може робити багато замовлень. Пряма лінія вказує на логічний зв'язок, адже замовлення не може бути неприсвоєним і залишатися самотійним.

Рослина – Замовлення:

Зв'язок: агрегація. Даний зв'язок демонструє, що Рослина є невід'ємною частиною замовлення. Такий вид зв'язку можна інтерпретувати, як «частина – ціле», оскільки вони об'єднанні в одну цілу складову.

Третій блок закріплено за взаємодією менеджером та складом.

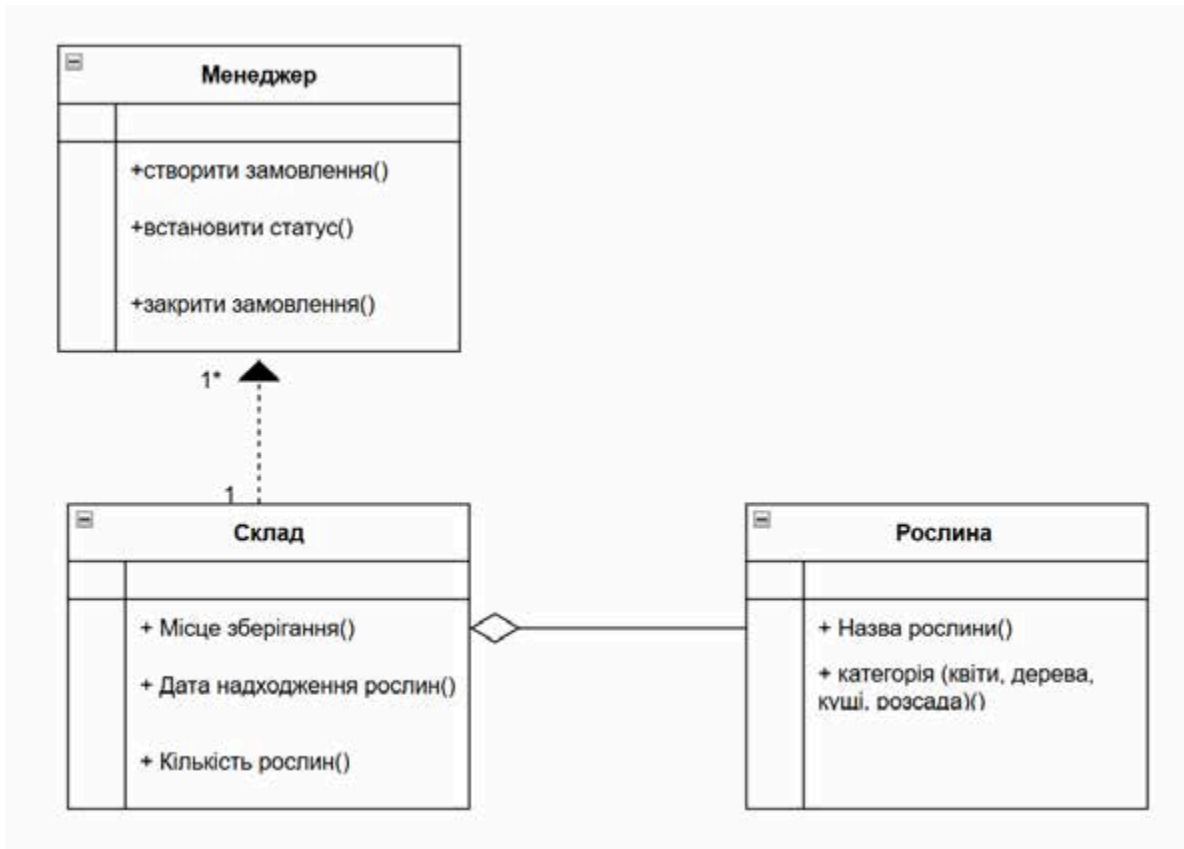


Рис.14. Діаграма кооперації(взаємодією менеджером та складом).

До даної діаграми входять 3 сутності: Менеджер, Склад та Рослина. Наведені сутності включають атрибути, що визначають стан об'єктів (місце зберігання, кількість, категорія), а також програмні методи для управління бізнес-процесами складу та життєвим циклом замовлень.

Менеджер — Склад:

Зв'язок: асоціація з кратністю (1 до 1*).

Даний зв'язок відображає ієрархію управління: один або кілька менеджерів мають доступ до ресурсів складу. Стрілка вказує на те, що Менеджер є активним суб'єктом, який ініціює дії в системі (створення, зміна статусу та закриття замовлень), базуючись на даних, що зберігаються в межах сутності Склад.

Склад — Рослина:

Зв'язок: агрегація (білий ромб).

Цей зв'язок демонструє фізичну та логічну залежність: Рослина розглядається як одиниця зберігання, що входить до складу ресурсів об'єкта Склад. Використання агрегації підкреслює, що хоча Рослина є складовою частиною поточних запасів («частина — ціле»), вона залишається самостійною інформаційною одиницею в базі даних, яка може бути переміщена, списана або реалізована через замовлення.

3.3. Організаційна структура програмного забезпечення.

На діаграмі пакетів відображено структуру програмного забезпечення системи обліку процесів малого бізнесу, а також взаємодію між її основними функціональними модулями. Така діаграма демонструє логічний поділ системи на окремі пакети, кожен із яких відповідає за виконання певного бізнес-процесу. Діаграма зображена в додатку Д.

Основним пакетом системи є «ПЗ обліку процесів малого бізнесу», який об'єднує всі підсистеми та забезпечує їх узгоджену взаємодію.

Пакет «Менеджер взаємодіє зі складом» реалізує функції, пов'язані з управлінням складськими процесами. До його складу входять класи «Менеджер», «Склад» та «Рослина». Менеджер може створювати замовлення, змінювати їхні статуси та завершувати обробку. Клас «Склад» містить відомості про місце зберігання, дату надходження товару та кількість рослин. Клас «Рослина» описує характеристики товару, зокрема його назву та категорію, наприклад квіти, дерева або кущі. Між класами «Склад» і «Рослина» встановлено зв'язок агрегації, що вказує на розміщення рослин на складі.

Пакет «Клієнт здійснює замовлення» відповідає за процес оформлення замовлення клієнтом. До його складу входять класи «Клієнт», «Замовлення» та «Рослина». Клієнт може оформлювати замовлення, надавати свої контактні дані та отримувати підтвердження виконання. Клас «Замовлення» забезпечує збереження інформації, зміну статусу замовлення та додавання пов'язаних

даних. Між класами «Клієнт» і «Замовлення» встановлено зв'язок один до багатьох, оскільки один клієнт може мати декілька замовлень. Крім того, клас «Замовлення» пов'язаний із класом «Рослина», адже в замовленні міститься інформація про вибрані товари.

Пакет «Менеджер обслуговує клієнта» відображає взаємодію менеджера з клієнтом у процесі оформлення та супроводу замовлення. До його складу входять класи «Менеджер», «Клієнт» та «Замовлення». Менеджер створює замовлення, змінює його статус і завершує обробку. Один менеджер може обслуговувати кілька замовлень, що відображається зв'язком один до багатьох. Також менеджер взаємодіє з клієнтом для отримання та підтвердження необхідних даних. Між пакетами існують залежності, які відображають обмін інформацією між окремими підсистемами. Зокрема, пакет оформлення замовлення пов'язаний із модулем складу, оскільки під час створення замовлення система перевіряє наявність рослин на складі. Пакет обслуговування клієнтів, своєю чергою, взаємодіє з пакетом замовлень для зміни їхніх статусів та супроводу процесу продажу.

Таким чином, діаграма пакетів відображає модульну побудову системи, розподіл відповідальності між її компонентами та взаємозв'язки між основними бізнес-процесами, пов'язаними з обліком товарів, клієнтів і замовлень у системі малого бізнесу.

3.4. Компонентна структура системи.

Компонентна структура ПЗ системи обліку процесів малого бізнесу представлена у вигляді діаграми компонентів. Вона моделює логічну архітектуру застосунку, розподілену на чотири основні рівні: інтерфейс користувача, серверну логіку (бізнес-логіка), репозиторії даних та базу даних. Такий багатoshаровий підхід забезпечує високий рівень модульності, розподілу відповідальностей та зручності супроводу системи на базі Django та PostgreSQL. Діаграма компонентів представлена на рис.15.

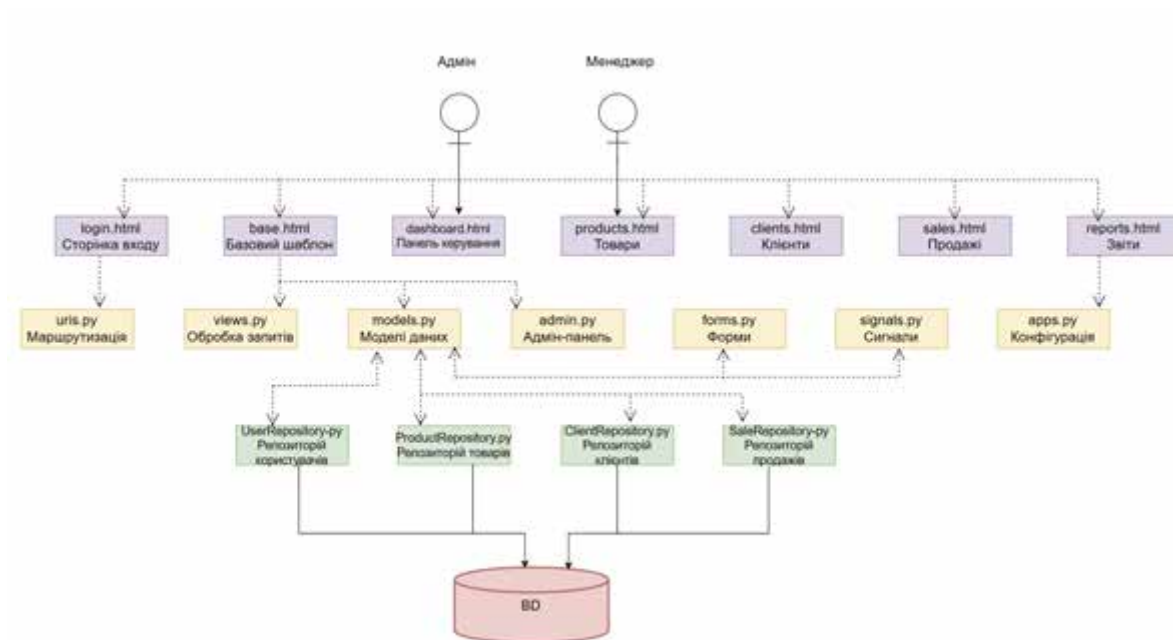


Рис.15. Діаграма компонентів.

Далі розглянемо основні складові діаграми.

Користувачі системи

У системі передбачено два основні типи користувачів, кожен з яких має доступ до певних функціональних модулів:

- Адмін – доступ до: dashboard.html (панель керування), admin.py (адміністрування даних).
- Менеджер – доступ до: products.html (управління товарами), clients.html (управління клієнтами), sales.html (оформлення та перегляд продажів), reports.html (формування звітності).

Всі інші компоненти інтерфейсу доступні залежно від ролі користувача.

Компоненти користувацького інтерфейсу (UI)

Інтерфейс реалізований HTML-шаблонами, які забезпечують взаємодію користувачів із системою та передачу запитів до серверної логіки:

- login.html – сторінка входу;
- base.html – базовий шаблон із спільною структурою;
- dashboard.html – панель керування;

- products.html, clients.html, sales.html, reports.html – спеціалізовані сторінки для ключових операцій.

Ці компоненти викликають відповідну бізнес-логіку через HTTP-запити.

Сервісні компоненти (бізнес-логіка)

Реалізують серверну логіку на Python у Django, обробляючи запити, дані та системні події:

- urls.py – маршрутизація запитів;
- views.py – обробка HTTP-запитів та передача даних до шаблонів;
- models.py – моделі даних та структура таблиць БД;
- admin.py – адміністративна панель;
- forms.py – створення та валідація форм;
- signals.py – обробка подій (наприклад, оновлення залишків після продажу);
- apps.py – конфігурація застосунку.

Репозиторії (доступ до бази даних)

Компоненти, що забезпечують взаємодію бізнес-логіки з БД через ORM Django, реалізуючи CRUD-операції:

- UserRepository.py – робота з користувачами;
- ProductRepository.py – управління товарами;
- ClientRepository.py – клієнтська база;
- SaleRepository.py – облік продажів та замовлень.

База даних

Нижній рівень – реляційна БД PostgreSQL («BD»), де зберігаються дані про користувачів, товари, клієнтів, продажі та звіти. Взаємодія відбувається через моделі та репозиторії.

Зв'язки між компонентами поділяються на два типи: пунктирні стрілки позначають залежності (наприклад, шаблони залежать від views.py, які взаємодіють із репозиторіями), а суцільні – прямі обміни даними (репозиторії з

БД).

Загалом, компонентна структура середовища для малого бізнесу реалізує клієнт-серверну архітектуру з чітким розділенням обов'язків, що спрощує розробку, тестування та еволюцію системи протягом усього життєвого циклу.

3.5.Вибір інструментарію для створення прикладного програмного забезпечення.

Для розробки програмного забезпечення системи обліку процесів малого бізнесу було обрано сучасний набір програмних засобів та технологій, які забезпечують надійність, масштабованість, зручність розробки та подальшого супроводу системи. Основними критеріями вибору інструментарію стали простота інтеграції компонентів, підтримка роботи з базами даних, наявність готових бібліотек, висока продуктивність і популярність серед розробників.

Для реалізації серверної частини системи було обрано мову програмування Python. Вона є однією з найпоширеніших у сфері веброзробки та автоматизації бізнес-процесів. Python відзначається простим і зрозумілим синтаксисом, що значно спрощує процес розробки, тестування та супроводу програмного забезпечення. Основними перевагами Python є висока швидкість розробки, велика кількість бібліотек і фреймворків, підтримка об'єктно-орієнтованого програмування, зручна робота з базами даних, кросплатформеність та активна спільнота розробників. Використання Python дає змогу ефективно реалізувати серверну логіку системи, механізми обробки даних і автоматизацію бізнес-процесів.

Для створення вебзастосунку було обрано Django — високорівневий вебфреймворк для Python, який забезпечує швидку та безпечну розробку вебсистем. Основними причинами вибору Django є наявність вбудованої ORM-системи для роботи з базами даних, готова система автентифікації та авторизації користувачів, вбудована адміністративна панель, підтримка HTML-шаблонів, високий рівень безпеки, підтримка архітектури MVT (Model-View-Template) та швидке створення CRUD-операцій. Фреймворк Django дає змогу реалізувати модулі управління товарами, клієнтами,

продажами та звітністю з мінімальними витратами часу на розробку.

Для зберігання інформації було обрано PostgreSQL — сучасну реляційну систему керування базами даних. PostgreSQL забезпечує високу надійність зберігання даних, підтримку SQL-стандартів, роботу з великими обсягами інформації, підтримку зовнішніх ключів і зв'язків між таблицями, механізми резервного копіювання та високу продуктивність. У базі даних зберігається інформація про користувачів системи, товари, клієнтів, продажі та звіти.

Як середовище розробки було використано Visual Studio Code. Цей редактор коду є одним із найпопулярніших інструментів для розробки програмного забезпечення. Перевагами Visual Studio Code є підтримка Python та Django, інтегрований термінал, підсвічування синтаксису, підтримка розширень, зручна навігація по проєкту та інтеграція із системами контролю версій. Використання VS Code значно спрощує процес написання, налагодження та тестування програмного коду.

Для створення інтерфейсу користувача було використано Bootstrap — CSS-фреймворк для розробки адаптивних вебінтерфейсів. Bootstrap забезпечує швидке створення сучасного дизайну, адаптивність інтерфейсу, готові UI-компоненти, підтримку мобільних пристроїв та просте стилізування HTML-сторінок. Використання Bootstrap дозволило створити зрозумілий і зручний інтерфейс для роботи користувачів із системою.

Для створення клієнтської частини системи використовувалися стандартні вебтехнології: HTML — для побудови структури вебсторінок, CSS — для стилізації інтерфейсу, JavaScript — для реалізації інтерактивних елементів. Застосування цих технологій забезпечує коректне відображення інтерфейсу та зручну взаємодію користувача із системою.

Для адміністрування бази даних використовувалися DBeaver та pgAdmin. Ці інструменти дають змогу створювати таблиці, виконувати SQL-запити, переглядати структуру бази даних, адмініструвати PostgreSQL та здійснювати резервне копіювання бази.

У сукупності обраний інструментарій забезпечує ефективну розробку, супровід і розвиток системи обліку процесів малого бізнесу.

3.6.Алгоритмізація та програмування програмних модулів.

Алгоритмізація функціональних процесів системи є важливою складовою програмної реалізації, що забезпечує послідовність, точність та ефективність виконання основних операцій. На цьому етапі визначаються ключові дії, які виконуються під час взаємодії користувача з інтерфейсом системи, обробки введених даних, їх перевірки та збереження у базу даних. Для ілюстрації рис.16 та рис.22 буде наведено приклади двох алгоритмів: алгоритму оформлення продажу товару менеджером та алгоритму формування звіту про продажі за певний період.

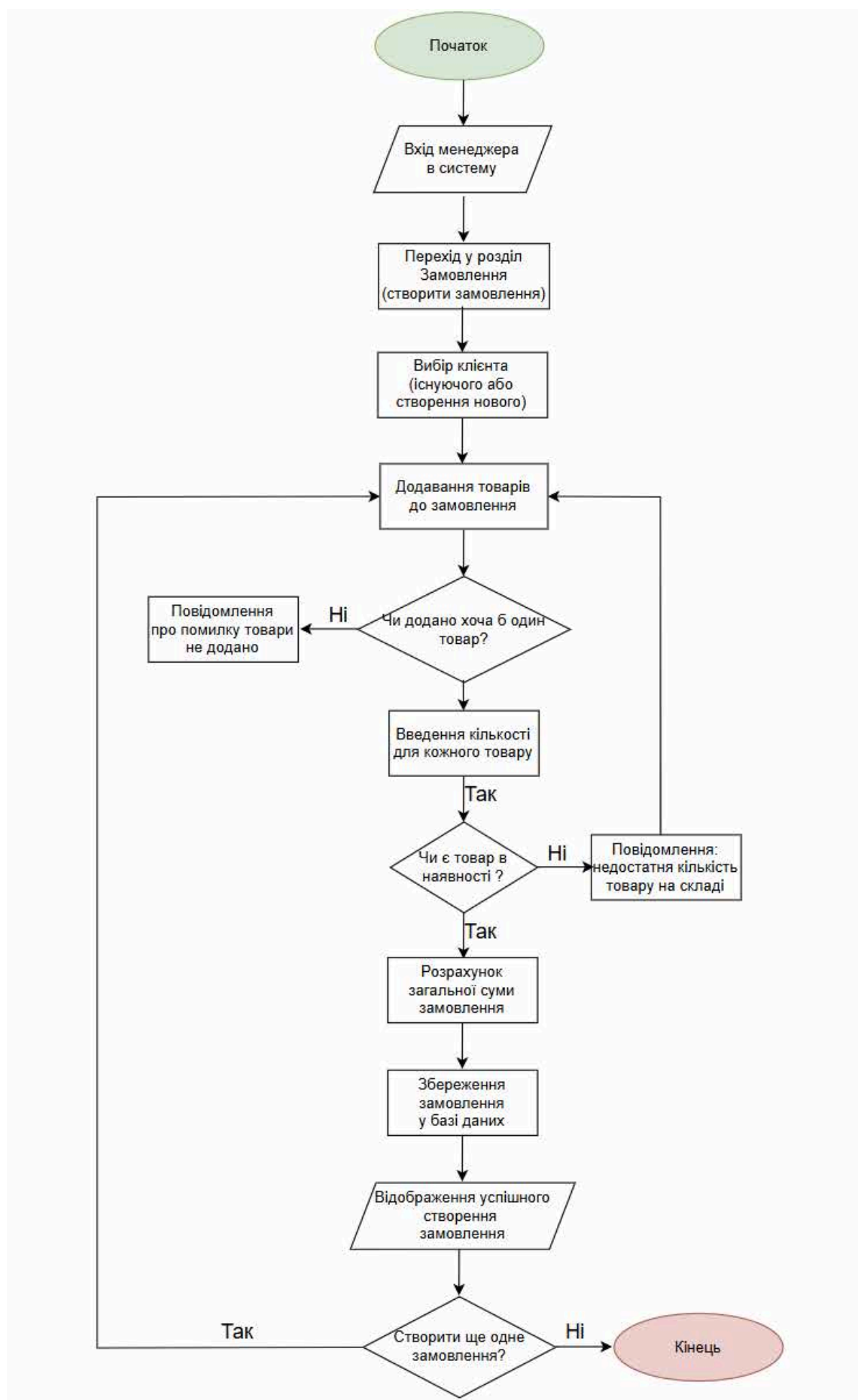


Рис.16. Блок – схема алгоритму оформлення продажу товару менеджером.

Дана блок-схема представляє алгоритм процесу створення замовлення менеджером у програмному забезпеченні системи обліку процесів малого бізнесу. Він демонструє послідовність дій користувача та логіку роботи системи під час оформлення замовлення.

Алгоритм починається з блоку «Початок», після чого менеджер виконує вхід до системи. Після успішної авторизації користувач переходить до розділу «Замовлення», де ініціюється створення нового замовлення. Наступним кроком є вибір клієнта, при цьому система дозволяє використовувати вже існуючого клієнта з бази даних або створити нового у разі його відсутності. Після вибору клієнта виконується додавання товарів до замовлення.

Далі система перевіряє, чи було до замовлення додано хоча б один товар. У разі відсутності товарів систему відображає повідомлення про помилку та повертає менеджера до етапу додавання товарів. Якщо товари додано, менеджер вказує необхідну кількість для кожного позиційного найменування, після чого система перевіряє наявність товару на складі. При недостатній кількості формується повідомлення про нестачу, і процес повертається до етапу редагування замовлення; за умови достатньої кількості система переходить до розрахунку загальної суми замовлення.

Після розрахунку замовлення зберігається в базі даних PostgreSQL, після чого система відображає повідомлення про успішне створення документу. Завершальним етапом є перевірка необхідності створення наступного замовлення. У разі позитивної відповіді процес повторюється з етапу додавання товарів, при від'ємній — алгоритм завершується блоком «Кінець».

Таким чином, блок-схема демонструє повний алгоритм створення замовлення менеджером, включаючи вибір клієнта, формування складового переліку товарів, перевірку даних, контроль наявності товару на складі, обробку помилок та збереження запису у базі даних. Реалізація цього алгоритму забезпечує автоматизацію процесу оформлення замовлень, сприяє зменшенню кількості помилок та підвищує ефективність функціонування системи обліку процесів малого бізнесу.

На рис.17 показано фрагмент коду який здійснює перевірку запиту та авторизацію менеджера. Система в кінцевому вигляді обробляє вхідний HTTP запит у відповідному HttpStatusCode (наприклад, add_order). Спочатку виконується перевірка, чи відправлений POST запит на створення замовлення, і чи є користувач аутентифікованим у системі. Далі здійснюється контроле рівня доступу менеджера, використовуючи атрибути вхідної ролі (is_manager) або вбудовану перевірку прав у Django в'юх (наприклад, @login_required, user_passes_test або UserPassesTestMixin). Якщо обидві умови виконані, менеджер отримує доступ до форми створення нового замовлення; у разі порушення прав доступу запит відхиляється з відповідним HTTP кодом.

```
if is_manager:
    client_id = request.POST.get('client_id')
    product_ids = request.POST.getlist('product_id[]')
    quantities = request.POST.getlist('quantity[]')
    total_amount = request.POST.get('total_amount')
```

Рис.17. Перевірка запиту та авторизації менеджера.

- На рис.18 зображено отримання та перевірку даних замовлення. Після підтвердження авторизації менеджера система отримує дані з веб форми, що містять такі поля:

- клієнт (номер або ідентифікатор),
- список товарів (ID товарів, вибраних на формі),
- кількість кожного товару (кількість для кожної позиції),
- загальна сума замовлення (рассчитана на сервері або у формі)

Далі виконується валідація даних на стороні сервера: перевіряється наявність клієнта у базі, а також той факт, що до замовлення добавлено хоча б один товар. Якщо клієнт відсутній або відсутні позиції товарів, система відображає відповідне помилкове повідомлення і повертає менеджера назад до форми для виправлення даних. У разі коректності вхідних даних система переходить до створення об'єкта замовлення та додавання товарів у замовлення.

```
if request.method == 'POST' and 'add_order' in request.POST:
    if is_manager:
```

Рис.18. Отримання та перевірка даних замовлення.

На рис.19 показано створення замовлення та клієнта. На цьому етапі система виконує створення нового об'єкта замовлення за допомогою виклику `Order.objects.create (order_data)` у Django, де `order_data` містить інформацію про клієнта, дату, статус (наприклад, «Нове») та інші атрибути замовлення. Після успішного збереження замовлення до бази даних система проходить по списку товарів у циклі (наприклад, за допомогою `for product, quantity in items_list`) і для кожної позиції створюється окремий запис у моделі `OrderItem` (order інлайн таблиця). У записі `OrderItem` зберігаються:

- товар (посилання на `Product`),
- кількість (запитана менеджером),
- ціна на момент продажу (фіксована для захисту від можливих змін цін у майбутньому),
- загальна сума по позиції ($\text{quantity} \times \text{price}$).

Додавання товарів реалізується як пакетна операція з збереженням у кожній ітерації чи за допомогою `bulk_create` у разі більш високої продуктивності.

```

if client_id and product_ids:
    new_order = Order.objects.create(
        client_id=client_id,
        manager=request.user,
        status=request.POST.get('status'),
        total_amount=total_amount
    )

    for p_id, qty in zip(product_ids, quantities):
        if not p_id: continue
        try:
            qty = int(qty)
            product_obj = Product.objects.get(id=p_id)
            inv = Inventory.objects.get(product=product_obj)
            OrderItem.objects.create(
                order=new_order,
                product=product_obj,
                quantity=qty,
                price=product_obj.price
            )

```

Рис.19. Створення замовлення та додавання товарів.

На рис.20 зображено оновлення складських залишків та обробка помилок. Після додавання товарів до замовлення система автоматично вносить зміни в складські залишки, викликаючи Inventory модель (наприклад, Product) і зменшуючи поле quantity на відповідну кількість по кожному товару. Ця операція виконується у тому ж транзакційному блоці, що забезпечує або успішне виконання всіх операцій, або відкат у разі помилки. Якщо виникає виняток (наприклад, товар не знайдено, від'ємна кількість або недостатній запас), система перехоплює винятки через try...except, формує відповідне повідомлення про помилку (наприклад, «недостатньо товару на складі») і пропускає проблемну позицію, не завершуючи весь процес створення замовлення. Таким чином, решта коректних товарів може бути додана до замовлення, а недостатні зберігаються у окремому списку для подальшого

розгляду.

```

        inv.quantity -= qty
        inv.save()
    except (Product.DoesNotExist, Inventory.DoesNotExist, ValueError):
        continue
    return redirect('dashboard')

```

Рис.20. Оновлення складських залишків та обробка помилок.

На рис.21 показано зміну статусу замовлення та повернення товару на склад. Під час зміни статусу замовлення система порівнює старий (`previous_status`) і новий (`new_status`) статуси. Якщо замовлення змінюється на статус «Скасовано» (або аналогічний), система виконує зворотну операцію: вона проходить по всіх `OrderItem` пов'язаних з цим замовленням і збільшує кількість товару на складі (`Product.inventory.quantity += order_item.quantity`), потім зберігає кожен змінений товарний об'єкт. Якщо ж статус змінюється з «Скасовано» назад на активний (наприклад, «Прийняте»), система повторно виконує списання товарів зі складу (аналогічно до процесу створення замовлення). Після оновлення складських залишків або відсутності змін у статусі система зберігає новий статус замовлення у базі даних, викликає `order.save()` і, при необхідності, генерує лог або відправлення сповіщення менеджеру або клієнту. Цей підхід забезпечує консистентність складських даних та підтримує прозорість змін у стані замовлення.

```
order = Order.objects.get(id=request.POST.get('order_id'))
old_status = order.status
new_status = request.POST.get('new_status')
items = OrderItem.objects.filter(order=order)

if new_status == "Скасовано" and old_status != "Скасовано":
    for item in items:
        inv = Inventory.objects.get(product=item.product)
        inv.quantity += item.quantity
        inv.save()
elif old_status == "Скасовано" and new_status != "Скасовано":
    for item in items:
        inv = Inventory.objects.get(product=item.product)
        inv.quantity -= item.quantity
        inv.save()

order.status = new_status
order.save()
return redirect('dashboard')
```

Рис.21. Оновлення складських залишків та обробка помилок.

Як зазначалось раніше на рис.22 продемонстровано алгоритм формування звіту про продажі за певний період.

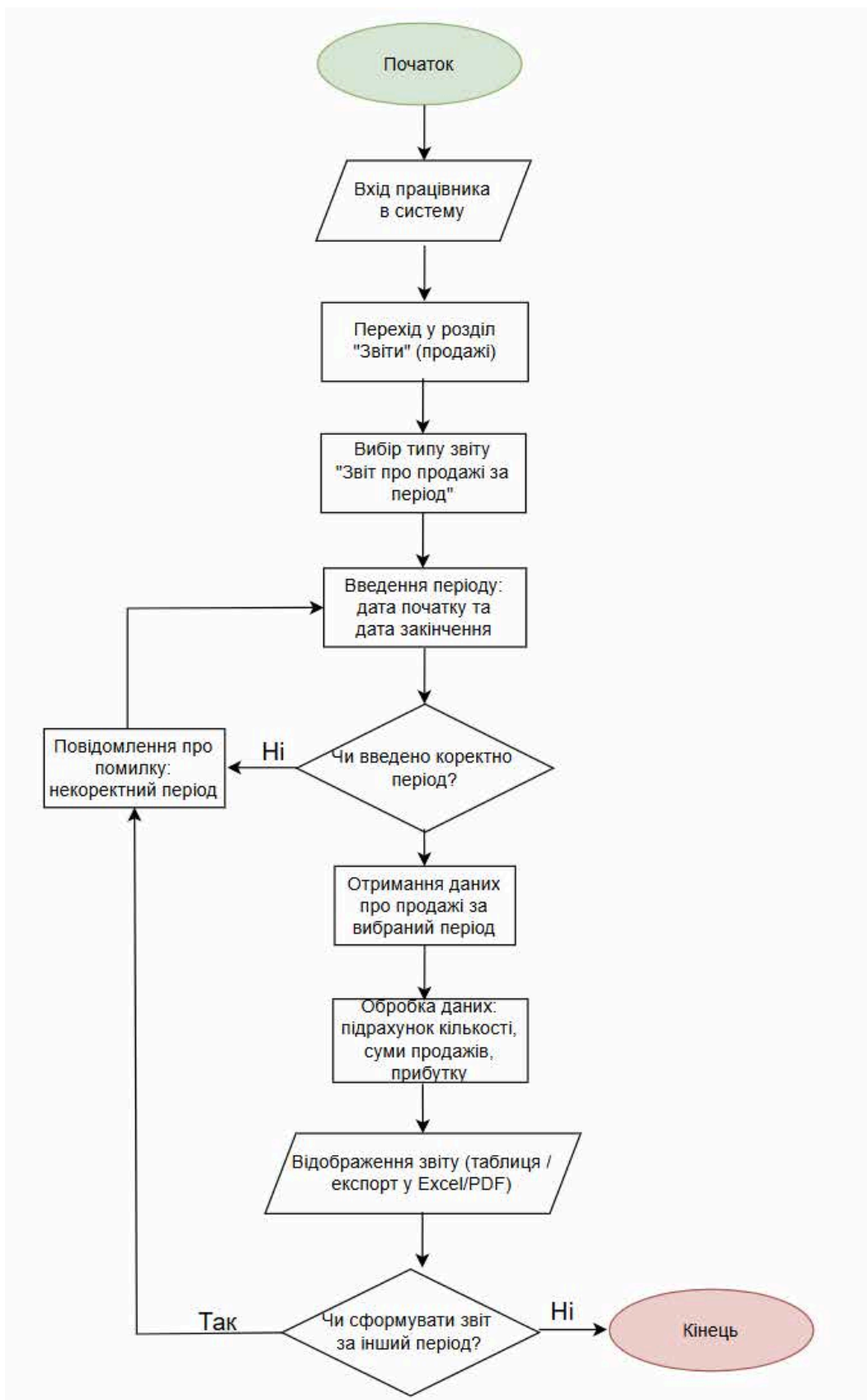


Рис.22. Блок-схема алгоритм формування звіту про продажі за певний період.

Діаграма відображає процес формування звіту про продажі за певний період в інформаційній системі. Процес починається з входу працівника до системи та переходу в розділ «Звіти», де обирається тип звіту «Звіт про продажі за період».

Після цього користувач вводить часовий інтервал, зазначаючи дату початку та дату завершення періоду. Система виконує перевірку правильності введених даних. Якщо період задано некоректно, відображається повідомлення про помилку, після чого користувач повертається до повторного введення параметрів.

У разі коректного введення періоду система здійснює отримання даних про продажі за вказаний проміжок часу. Далі виконується обробка інформації, під час якої обчислюються кількість продажів, сума продажів і прибуток.

Після завершення обробки звіт відображається у табличному вигляді або може бути експортований у форматах Excel чи PDF. На завершальному етапі система пропонує сформувати звіт за інший період. Якщо користувач обирає повторне формування, процес починається знову; якщо ні — робота завершується.

Таким чином, діаграма демонструє послідовність дій від входу в систему до отримання готового звіту про продажі та передбачає перевірку введених даних, обробку статистики й можливість повторного формування звіту.

На рис.23 показано авторизацію та перевірку прав доступу. Процедура формування звіту починається з входу користувача в систему. Для забезпечення контролю доступу застосовується декоратор `@login_required`, який гарантує, що виконання функції можливе лише для автентифікованого користувача. Після успішної автентифікації система додатково перевіряє наявність прав на експорт звітів шляхом аналізу належності користувача до певних груп доступу — Admin або Stafflog — а також перевіряє, чи має він статус супер користувача.

```
@login_required
def export_inventory_report(request):
    # Перевірка прав
    user_groups = request.user.groups.values_list('name', flat=True)
    if not (request.user.is_superuser or "Admin" in user_groups or "Stafflog" in user_groups):
        return HttpResponse("У вас немає прав на вигражу звітів", status=403)
```

Рис.23. Авторизація та перевірка прав доступу.

На рис 24. продемонстровано визначення періоду формування звіту. Після проходження етапу авторизації користувач задає часовий інтервал, за який необхідно сформувати звіт. Для цього з параметрів HTTP-запиту зчитуються значення `start_date` і `end_date`, які відповідають даті початку та даті завершення періоду.

Вказані значення використовуються як критерій відбору даних зі складської бази. Таким чином, звіт формується не для всіх записів, а лише для тих, що належать до заданого діапазону дат. Це дає змогу виконувати аналіз даних у межах конкретного інтервалу часу та підвищує точність результатів.

```
start_date = request.GET.get('start_date')
end_date = request.GET.get('end_date')
```

Рис.24. Визначення періоду формування звіту.

На рис 25. зображено формування структури Excel-звіту. Після визначення параметрів відбору система створює нову книгу Excel за допомогою бібліотеки `openpyxl`. Активному аркушу присвоюється назва "ЗвітСклад", що відображає зміст сформованого документа.

Далі в таблицю додається рядок заголовків, який містить основні показники звіту: товар, залишок, мінімальний запас, локація, всього продано та статус. Наявність структурованих заголовків забезпечує зручність подальшого аналізу, читання та використання звіту в табличному форматі.

```
wb = openpyxl.Workbook()
ws = wb.active
ws.title = "Звіт_Склад"
headers = ['Товар', 'Залишок', 'Мін. запас', 'Локація', 'Всього продано', 'Статус']
ws.append(headers)
```

Рис.25. Формування структури Excel-звіту.

На рис.26 показано отримання та обробку складських даних. На наступному етапі система отримує дані про складські залишки з таблиці Inventory із використанням зв'язку з моделлю товару через `select_related('product')`. Якщо користувач задав діапазон дат, виконується додаткова фільтрація записів за полем `arrival_date` у межах вказаного періоду.

Після цього для кожної позиції складу система виконує окрему обробку. Спочатку обчислюється кількість реалізованого товару шляхом аналізу записів у таблиці замовлень `OrderItem`, при цьому виключаються замовлення зі статусом "Скасовано". Для кожного товару система встановлює його поточний статус залежно від кількості залишків на складі. Якщо кількість товару перевищує мінімальний допустимий рівень, статус залишається "Норма". Якщо залишок дорівнює або менший за мінімально допустиме значення, товар отримує статус "МАЛО". У випадку, коли кількість товару менша або дорівнює нулю, встановлюється статус "НЕМАЄ".

Такий підхід дозволяє не лише відобразити фактичний стан запасів, а й оперативно ідентифікувати проблемні позиції, що потребують поповнення. Отже, звіт виконує контрольну та аналітичну функцію одночасно.

```
ws.append([item.product.name, item.quantity, item.min_quantity, item.location, sold_qty, status])

response = HttpResponse(content_type='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')
response['Content-Disposition'] = f'attachment; filename="inventory_{datetime.now().strftime("%d_%m")}.xlsx"'
wb.save(response)
return response
```

Рис.26. Отримання та обробка складських даних.

На рис.27 зображено формування експорт готового звіту.

Після завершення обробки кожен запис додається до Excel-таблиці у вигляді окремого рядка, що містить назву товару, кількість залишку, мінімальний запас, локацію, сумарний обсяг продажів та визначений статус. Таким чином, формується повноцінна таблична структура звіту.

Після заповнення книги система створює HTTP-відповідь із типом вмісту `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`, що відповідає

формату .xlsx. У заголовку відповіді задається ім'я файлу з поточною датою, після чого книга зберігається безпосередньо у відповідь і передається користувачу для завантаження. Це забезпечує автоматизований експорт підготовленого звіту та його подальше використання в офісних або аналітичних програмах.

```
queryset = Inventory.objects.select_related('product').all()
if start_date and end_date:
    queryset = queryset.filter(arrival_date__range=[start_date, end_date])

for item in queryset:
    sold_qty = OrderItem.objects.filter(
        product=item.product
    ).exclude(order__status="Скасовано").aggregate(total=Sum('quantity'))['total'] or 0

    status = "Норма"
    if item.quantity <= item.min_quantity: status = "МАЛО"
    if item.quantity <= 0: status = "НЕМАЄ"
```

Рис.27. Формування експорт готового звіту.

Таким чином, алгоритм формування звіту охоплює кілька послідовних етапів: перевірку прав доступу, задання періоду аналізу, відбір і обробку складських даних, визначення статусу товарів та експорт результатів у форматі Excel.

4. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи.

Модульне тестування (Unit Testing) виконувалося для перевірки окремих компонентів системи, зокрема механізму авторизації менеджера, обробки форми створення замовлення, валідації вхідних даних, розрахунку загальної суми та змін складських залишків. Метою цього етапу було виявлення логічних помилок у роботі окремих функцій і класів без урахування всієї системи в цілому.

Інтеграційне тестування (Integration Testing) було спрямоване на перевірку взаємодії між компонентами системи, зокрема між формою введення даних, моделями замовлення, таблицею товарів, складським обліком та механізмом зміни статусів. Під час цього тестування перевірялося, чи коректно передаються дані між модулями, чи правильно створюються об'єкти Order і OrderItem, а також чи відбувається синхронне оновлення інформації в базі даних.

Функціональне тестування (Functional Testing) здійснювалося для перевірки відповідності системи її функціональним вимогам. У межах цього тестування аналізувалося, чи може менеджер створити замовлення лише після авторизації, чи правильно обробляються випадки з відсутнім клієнтом або порожнім списком товарів, а також чи коректно виконується зміна статусу замовлення з відповідним оновленням залишків на складі.

Тестування зручності використання (Usability Testing) проводилося для оцінки зручності роботи користувача з інтерфейсом системи. Особлива увага приділялася логічності розташування елементів форми, зрозумілості повідомлень про помилки, а також зручності виконання дій менеджером під час оформлення замовлення та перегляду його статусу. Це дозволило визначити, наскільки інтерфейс є інтуїтивно зрозумілим та придатним для щоденного використання.

Системне тестування (System Testing) охоплювало перевірку всієї системи як єдиного цілого, включаючи серверну логіку, базу даних, механізми авторизації та інтерфейс користувача. На цьому етапі перевірялася стабільність роботи системи в типових сценаріях: вхід менеджера, створення замовлення, додавання товарів, оновлення залишків, скасування замовлення та повторне відновлення товарів на складі.

Тестування “чорної скриньки” (Black Box Testing) використовувалося для перевірки поведінки системи без аналізу її внутрішньої реалізації. Під час такого тестування оцінювалися лише вхідні дані, дії користувача та фактичні результати роботи системи. Це дало змогу перевірити, чи відповідає зовнішня поведінка програми очікуваним результатам з погляду кінцевого користувача.

4.1.1.Приклад тестування.

Нижче наведено приклад тестування за методом “чорної скриньки”, який демонструє перевірку сценарію авторизації менеджера та створення замовлення.

На рис.28 показано відповідне вікно авторизації. Після успішного входу система автоматично перенаправляє користувача до головного меню.



The image shows a login window with a white background and a light gray border. At the top center, the title "Вхід" is displayed in a large, bold, black font. Below the title, there are two input fields. The first field is labeled "Логін" (Login) in a gray font, and it contains the text "Manager1". The second field is labeled "Пароль" (Password) in a gray font, and it contains masked characters represented by seven dots. Below these fields is a prominent blue button with the white text "Увійти" (Login).

Рис.28.Вікно авторизації.

На головному екрані відображаються вкладки: клієнти, замовлення та склад. Це зображено нижче на рис.29.

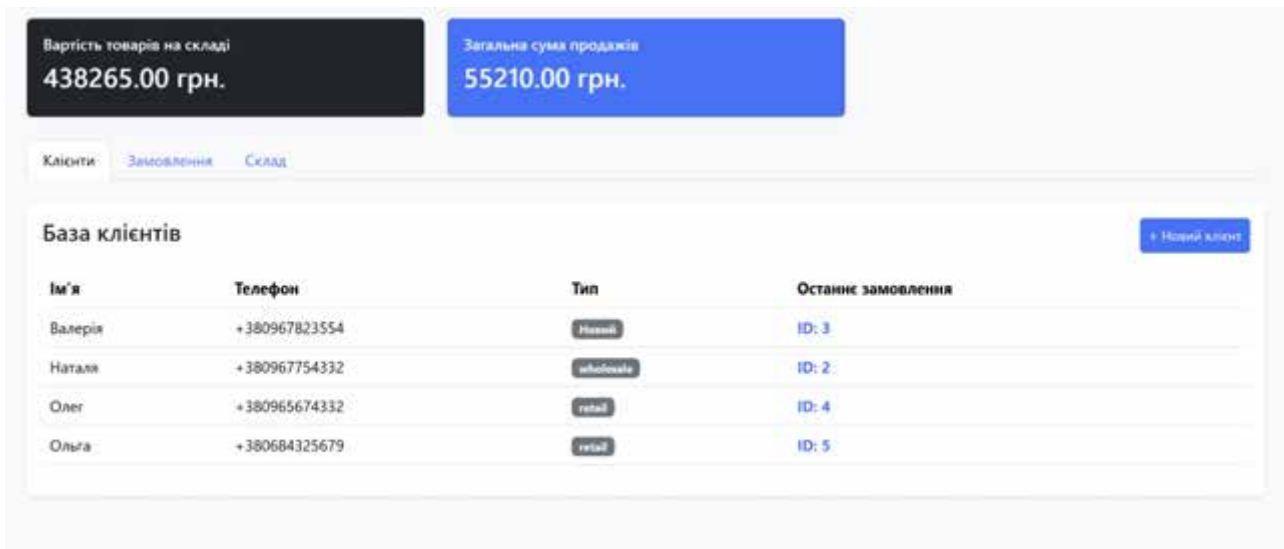


Рис.29.Вікно головне меню.

Користувач натискає на кнопку + Новий клієнт, як показано на рис. 30. Після цього, відповідно до рис. 31, користувач у вікні заповнює усі необхідні поля. Після підтвердження перевірки клієнт з'являється у списку, як показано на рис. 32.



Рис.30. Додавання нового клієнта.

Рис.31. Заповнення даних.

Рис.32. Додано нового клієнта(без замовлення).

Менеджер переходить у вкладку замовлення рис.33 та через кнопку + Створити замовлення обирає необхідного клієнта та додає необхідні товари, кількість(рис.34 також одразу відображається залишок на складі товару, задля зручності менеджера), після чого розрахована сума замовлення буде відображена у формі на рис.35. Після того як замовлення з'явиться у списку рис.36, потрібно обрати статус рис.37.

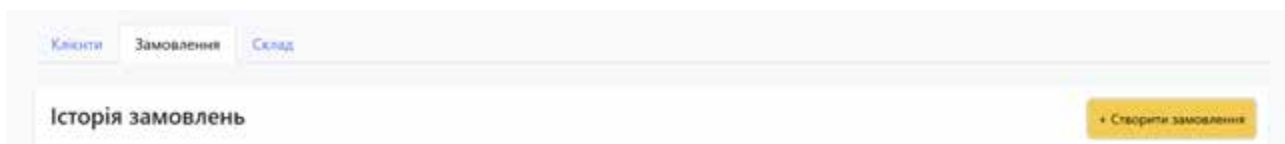


Рис.33. Створення замовлення.

Створити замовлення

Клієнт:
Марчук Олена Вікторівна

Товари:

--- Оберіть товар ---

Троянада Чайно-гібридні (Доступно: 500)

Троянада Плетисті (Доступно: 245)

Жоржини Кактусові (Доступно: 110)

Жоржини Анемоноподібні (Доступно: 360)

Лотос (Доступно: 280)

Троянада Флорибунда (Доступно: 440)

Гортензія (Доступно: 88)

Жоржини Помпонні (Доступно: 200)

Бузок (Доступно: 220)

Лаванда (Доступно: 250)

Монстера (Доступно: 45)

Орхідея Фаленопсис (Доступно: 85)

Самшит (Доступно: 0)

Самшит (Доступно: 100)

1

Підтвердити

17844.00 грн

95620.00 грн

2460.00 грн

Скасовано

Скасовано

Завершено

Рис.34. Відображення залишку товару під час заповнення замовлення.

Створити замовлення

Клієнт:
Марчук Олена Вікторівна

Статус:
Нове

Товари:

Троянада Чайно-гібридні (Доступно: 500) — 130.00 грн	44	
Жоржини Помпонні (Доступно: 200) — 118.00 грн.	12	×
Лотос (Доступно: 280) — 123.00 грн.	33	×
Монстера (Доступно: 45) — 360.00 грн.	19	×
Лаванда (Доступно: 250) — 70.00 грн.	100	×

+ Додати рядок

Загальна сума (грн):
25035,00

Розраховується автоматично на основі доданих товарів.

Закрити Підтвердити

Рис.35. Заповнена форма замовлення з розрахованою сумою.

ID / Дата	Клієнт	Товари	Сума	Статус
#6 15.05.2026 21:13	Марчук Олена Вікторівна	• Троянада Чайно-гібридні (44 шт.) • Жоржини Помпонні (12 шт.) • Лотос (33 шт.) • Монстера (19 шт.) • Лаванда (100 шт.)	25035.00 грн	Нове

Рис.36. Відображення нового замовлення.

#6 15.05.2026 21:13	Марчук Олена Вікторівна	• Троянада Чайно-гібридні (44 шт.) • Жоржини Помпонні (12 шт.) • Лотос (33 шт.) • Монстера (19 шт.) • Лаванда (100 шт.)	25035.00 грн	Нове Нове В роботі Оплачено Завершено Скасовано ОК
------------------------	-------------------------	---	--------------	---

Рис.37. Обрання статусу зі списку.

Результат тестування:

Система успішно пройшла всі кроки зазначеного сценарію без збоїв.

Функції працювали відповідно до заданої логіки, введені дані коректно оброблялись, а результати формувались згідно з очікуванням на всіх етапах взаємодії.

4.2. Вимоги до апаратного та програмного забезпечення.

Для забезпечення стабільної та коректної роботи програмної системи необхідно визначити вимоги до її апаратного та програмного забезпечення. Такі вимоги встановлюють мінімальні умови для функціонування клієнтської частини, веб сервера та сервера бази даних, а також визначають набір програмних засобів, необхідних для розгортання, експлуатації та супроводу системи. Відповідно до діаграми розгортання системи, яка представлена у Додатку Е , програмні компоненти на окремих апаратних вузлах, що відображає фізичну структуру реалізації

У межах даної роботи було сформовано вимоги, що враховують особливості взаємодії користувача з веб застосунком, обробку запитів на сервері, збереження даних у базі та формування звітів у табличному форматі. Це забезпечує надійну роботу системи в умовах реального використання та створює основу для її подальшого масштабування й модернізації.

Клієнтська частина

Для роботи користувача із системою необхідно:

- персональний комп'ютер, ноутбук або смартфон;
- процесор не менше ніж із 2 ядрами;
- оперативна пам'ять обсягом від 4 ГБ;
- стабільне підключення до мережі Інтернет;
- сучасний веббраузер із підтримкою протоколів HTTP та HTTPS.

Вебсервер

Для розміщення вебзастосунку необхідно:

- процесор рівня Intel Core i3 або AMD Ryzen 3 і вище;
- оперативна пам'ять обсягом від 8 ГБ;
- SSD-накопичувач обсягом від 50 ГБ;

- мережеве підключення за протоколом TCP/IP;
- підтримка протоколів HTTP та HTTPS.

Сервер бази даних

Для функціонування системи баз даних необхідно:

- процесор не менше ніж із 2 ядрами;
- оперативна пам'ять обсягом від 8 ГБ;
- SSD-накопичувач для забезпечення швидкої роботи бази даних;
- підтримка SQL-запитів;
- підтримка TCP/IP-з'єднання.

Вимоги до програмного забезпечення

Клієнтська частина

Для доступу до системи з боку користувача необхідно:

- операційна система Windows, Linux, Android або iOS;
- веббраузер Google Chrome;
- веббраузер Mozilla Firefox;
- веббраузер Microsoft Edge.

Серверна частина

Для роботи серверної частини необхідно:

- операційна система Ubuntu Server або Linux;
- операційна система Windows Server;
- вебсервер Nginx або Apache;
- мова програмування Python 3.x;
- backend-фреймворк Django;
- frontend-технології HTML5, CSS3 та JavaScript.

Сервер бази даних

Для зберігання та обробки даних необхідно:

- система керування базами даних PostgreSQL;
- підключення до бази даних через SQL та TCP/IP.

Додаткові програмні компоненти

Для розширення функціональності системи використовуються:

- бібліотека `openpyxl` для експорту Excel-звітів;
- система контролю версій `Git`;
- HTTPS-сертифікат для захищеного з'єднання.

Розроблена система потребує сучасного клієнтського пристрою, вебсервера та сервера бази даних, а також програмного середовища, що включає `Python`, `Django`, `PostgreSQL`, веббраузери та додаткові інструменти для експорту звітів і забезпечення безпеки з'єднання.

4.3. Склад інсталяційного пакету.

Інсталяційний пакет системи обліку бізнес процесів призначено для автоматизації обліку товарів, клієнтів, замовлень та складських операцій у малому бізнесі. Пакет містить серверну частину на базі `Django`, клієнтський веб інтерфейс, `SQL` скрипти для ініціалізації бази даних `PostgreSQL`, конфігураційні файли, додаткові бібліотеки та документацію. Система може працювати в локальній мережі або в хмарному середовищі; доступ здійснюється через браузер по протоколах `HTTP/HTTPS`.

Склад інсталяційного пакета

- Веб застосунок (серверна частина): проєкт `Django`, що містить модулі бізнес логіки, `REST/API` контролери (за потреби), моделі даних, інтерфейс адміністратора та механізми автентифікації й авторизації.
- Клієнтський веб інтерфейс: `HTML`, `CSS`, `JavaScript` файли, шаблони `Django` (або `SPA` компоненти), статичні ресурси (зображення, шрифти).
- Конфігураційний файл: файл із параметрами підключення до БД, налаштуваннями безпеки (секретні ключі, `TLS/SSL`), параметрами середовища (`DEBUG`, `ALLOWED_HOSTS`, логування) та конфігурацією зовнішніх сервісів.
- `SQL` скрипти для ініціалізації БД: набір `SQL` запитів для створення схем, таблиць, обмежень, індексів, початкових довідників (`seed`), ролей і прав доступу у `PostgreSQL`.

- Додаткові бібліотеки та залежності: файл requirements.txt (або Pipfile), що перелічує Python пакети; серед них openruhl (генерація Excel звітів), psycopg2/psycopg binary (драйвер PostgreSQL) та інші бібліотеки.
- Документація: інструкція користувача, керівництво адміністратора, довідка з встановлення і налаштування, рекомендації з резервного копіювання та відновлення даних.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи на тему «Програмне забезпечення системи обліку процесів для малого бізнесу» було здійснено повний цикл розробки інформаційної системи, що охоплює етапи аналізу предметної області, проєктування бази даних, розробки прикладного програмного забезпечення та підготовки рекомендацій щодо впровадження і експлуатації.

Проведений аналіз діяльності малого підприємства дозволив визначити ключові напрями автоматизації, які включають облік товарів, клієнтів, замовлень, складських операцій, а також інші супутні процеси комерційної діяльності. На основі виявлених потреб було сформульовано технічне завдання та розроблено концепцію програмного забезпечення, що забезпечує інтегрований підхід до управління бізнес процесами.

Було побудовано логічну модель даних у вигляді ER діаграми, спроектовано структуру бази даних у середовищі PostgreSQL та реалізовано SQL скрипти для автоматичної ініціалізації всіх необхідних об'єктів бази даних перед початком роботи системи. Це дозволило створити стійку та масштабовану основу для зберігання інформації про товари, клієнтів, постачальників, складські запаси, документи та фінансові операції.

Клієнтська частина програмного забезпечення реалізована як веб застосунок на базі мови Python з використанням фреймворку Django. Веб інтерфейс побудований з використанням HTML, CSS та JavaScript, що дозволило створити сучасний, зручний та зрозумілий інтерфейс для користувачів. Завдяки модульній структурі системи забезпечено просту навігацію, інтуїтивне введення даних та гнучке налаштування режимів роботи.

У склад веб застосунку інтегровано функції генерації Excel звітів за допомогою Python бібліотеки openpyxl, що полегшує підготовку аналітичних звітів, номенклатурних відомостей та звітів з обліку товарів та замовлень. Такий підхід дозволяє власнику та менеджеру малого бізнесу швидко отримувати

актуальну інформацію для прийняття управлінських рішень.

Окремою інноваційною складовою розробленої системи є можливість інтеграції з інтелектуальними інструментами аналітики. Використання моделей штучного інтелекту, розгорнутих локально або в хмарному середовищі, дозволяє формувати рекомендації щодо оптимізації запасів, аналізу попиту та клієнтської поведінки. Такий підхід відкриває перспективи для подальшого розвитку системи у напрямку інтелектуальної підтримки бізнес процесів.

Розроблене програмне забезпечення супроводжується інсталяційним пакетом, що забезпечує його швидке та зручне впровадження в умовах малого підприємства. Інсталятор автоматично встановлює необхідні Python залежності та конфігурує базове середовище, а підготовлені SQL скрипти дозволяють ініціалізувати базу даних PostgreSQL без значних технічних зусиль. Завдяки чіткій структурі та зрозумілим інструкціям запуск системи не потребує висококваліфікованого системного адміністратора, що дозволяє оперативно розпочати її експлуатацію.

У результаті виконаної роботи всі поставлені цілі та завдання були повністю досягнуті. Розроблена система успішно автоматизує ключові процеси обліку товарів, клієнтів, замовлень та складських операцій для малого бізнесу, поєднуючи зручність використання, стабільність роботи, ефективність обробки даних та можливості інтелектуальної підтримки прийняття управлінських рішень. Реалізоване рішення є практичним, функціональним та готовим до застосування в реальних умовах діяльності малого підприємства.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Державний стандарт України ДСТУ 8302:2015 Інформаційно-комунікаційні технології. Бібліографічне посилання. Загальні положення. – К.: ДП «УкрНДНЦ», 2015. – 80 с.
2. Громів І. С. Проектування баз даних інформаційних систем: підручник. – К.: Вид-во КНУ, 2020. – 280 с.
3. Кучерук В. М. Інформаційні системи і технології в бізнесі: навч. посібник. – Львів: Вид-во «Ліга-Прес», 2019. – 324 с.
4. Методика управління малого бізнесу: навч. посібник / О. М. Петренко, І. В. Бондаренко. – Х.: Вид-во «Харківський університет», 2021. – 308 с.
5. Бухгалтерський і управлінський облік малого підприємства: навч. посібник / Н. В. Шевчук. – К.: Вид-во «КНЕУ», 2018. – 264 с.
6. Salesdrive CRM. – URL: <https://salesdrive.com.ua/> (дата звернення: 03.04.2026).
7. if.team – CRM для малого бізнесу. – URL: <https://if.team/> (дата звернення: 03.04.2026).
8. RO-APP (CRM/облік малого бізнесу) – матеріали з офіційного сайту та відгуки. – URL: <https://roapp.com.ua/> (дата звернення: 03.04.2026).
9. UML 2.5 Specification. – URL: <https://www.omg.org/spec/UML/> (дата звернення: 04.04.2026).
10. Студайл файл «Концептуальна, логічна, фізична моделі даних; ER модель». – URL: <https://studfile.net/preview/21632686> (дата звернення: 10.04.2026).
11. Студайл файл «3.1. Ключі та нормалізація даних: основні нормальні форми (1NF, 2NF, 3NF, BCNF)». – URL: <https://studfile.net/preview/21632686/page:2/> (дата звернення: 10.04.2026)
12. PostgreSQL 15 Documentation // The PostgreSQL Global Development Group. – URL: <https://www.postgresql.org/docs/15/> (дата звернення: 12.04.2026).
13. Django 4.2 Documentation. – URL:

<https://docs.djangoproject.com/en/4.2/> (дата звернення: 15.04.2026).

14. Python 3.x Documentation. – URL: <https://docs.python.org/3/> (дата звернення: 15.04.2026).

15. HTML Living Standard. – URL: <https://html.spec.whatwg.org/> (дата звернення: 15.04.2026).

16. CSS-Specifications (W3C). – URL: <https://www.w3.org/Style/CSS/> (дата звернення: 16.04.2026).

17. JavaScript (ECMAScript) documentation (MDN Web Docs). – URL: <https://developer.mozilla.org/uk/docs/Web/JavaScript> (дата звернення: 16.04.2026).

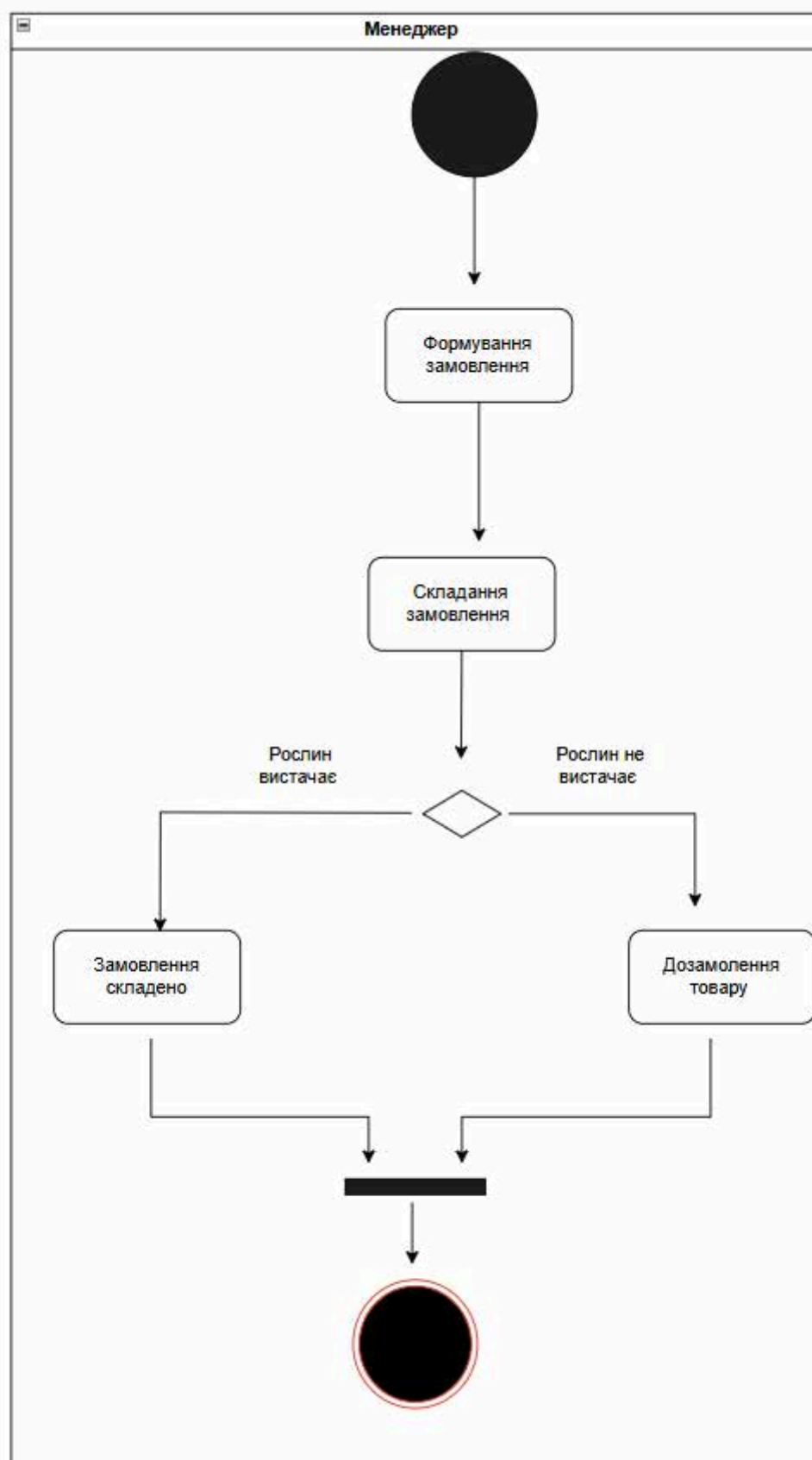
18. HTTP/1.1 and HTTP/2 RFCs (IETF). – URL: <https://datatracker.ietf.org/wg/http/> (дата звернення: 16.04.2026).

19. Матеріал «Рекомендації з побудови діаграми розгортання» (Studfile). – URL: <https://studfile.net/preview/9445425/page:128/> (дата звернення: 27.04.2026).

20. EPAM Campus – «Типи тестування: пам'ятка початківцю. Частина 1». – URL: <https://campus.epam.ua/ua/blog/469> (дата звернення: 01.05.2026).

ДОДАТКИ

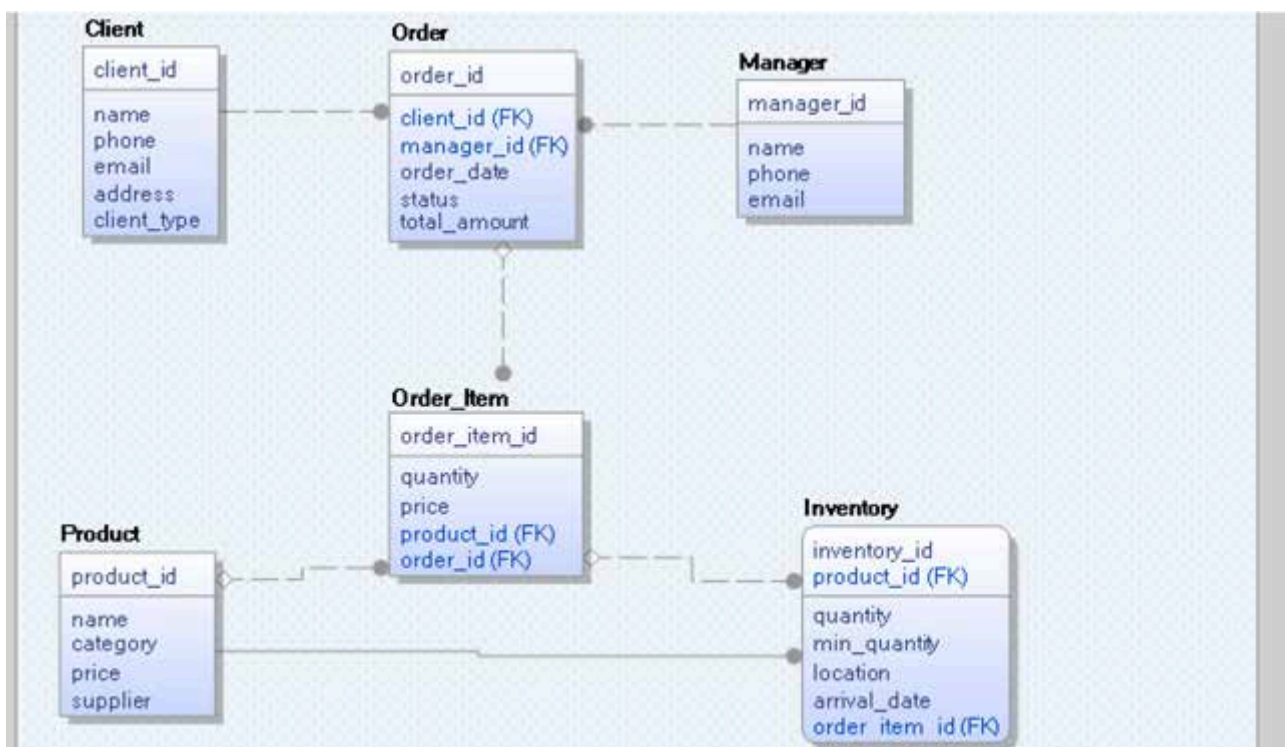
ДОДАТОК А



ДОДАТОК Б



ДОДАТОК В



ДОДАТОК Г

```

CREATE TABLE clients (
    client_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    email VARCHAR(100),
    address TEXT,
    client_type VARCHAR(20) CHECK (client_type IN ('retail', 'wholesale'))
);

CREATE TABLE managers (
    manager_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    email VARCHAR(100),
    user_id INT,
    CONSTRAINT fk_manager_user
        FOREIGN KEY (user_id)
        REFERENCES users(user_id)
        ON DELETE SET NULL
);

CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    login VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(20) CHECK (role IN ('admin', 'manager', 'warehouse')) NOT NULL
);

CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    price NUMERIC(10,2) NOT NULL,
    supplier VARCHAR(100)
);

CREATE TABLE inventory (
    inventory_id SERIAL PRIMARY KEY,
    product_id INT UNIQUE,
    quantity INT DEFAULT 0,
    min_quantity INT DEFAULT 0,
    location VARCHAR(100),
    arrival_date DATE,
    last_update TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT fk_inventory_product
        FOREIGN KEY (product_id)
        REFERENCES products(product_id)
        ON DELETE CASCADE
);

CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    client_id INT,
    manager_id INT,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) CHECK (status IN ('new', 'processing', 'completed',
'cancelled')),

```

```

total_amount NUMERIC(10,2),

CONSTRAINT fk_order_client
    FOREIGN KEY (client_id)
    REFERENCES clients(client_id)
    ON DELETE CASCADE,

CONSTRAINT fk_order_manager
    FOREIGN KEY (manager_id)
    REFERENCES managers(manager_id)
    ON DELETE SET NULL
);

CREATE TABLE order_items (
    order_item_id SERIAL PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT NOT NULL,
    price NUMERIC(10,2) NOT NULL,

    CONSTRAINT fk_item_order
        FOREIGN KEY (order_id)
        REFERENCES orders(order_id)
        ON DELETE CASCADE,

    CONSTRAINT fk_item_product
        FOREIGN KEY (product_id)
        REFERENCES products(product_id)
        ON DELETE CASCADE
);

CREATE TABLE interactions (
    interaction_id SERIAL PRIMARY KEY,
    client_id INT,
    manager_id INT,
    type VARCHAR(50),
    date_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    result TEXT,

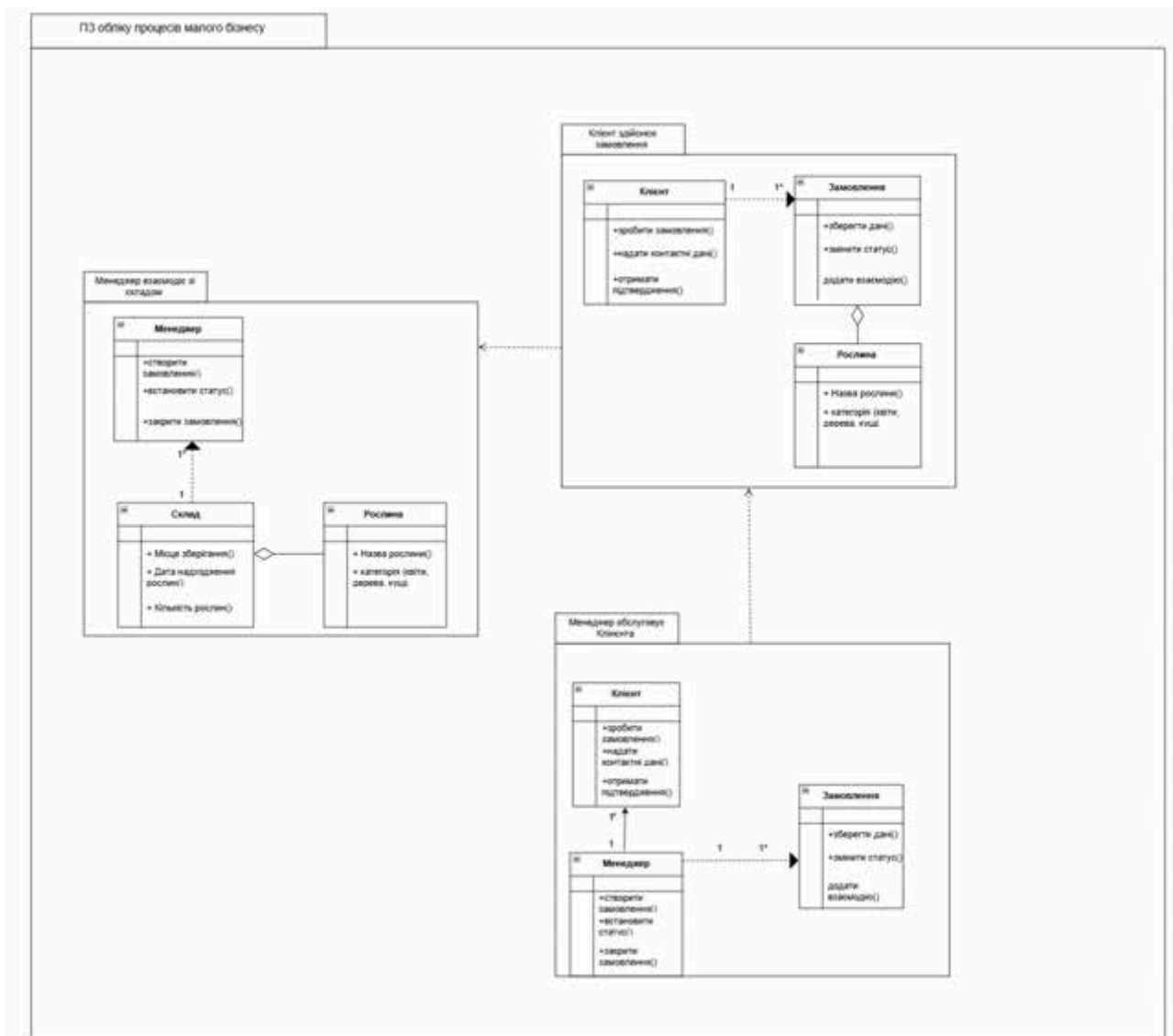
    CONSTRAINT fk_interaction_client
        FOREIGN KEY (client_id)
        REFERENCES clients(client_id)
        ON DELETE CASCADE,

    CONSTRAINT fk_interaction_manager
        FOREIGN KEY (manager_id)
        REFERENCES managers(manager_id)
        ON DELETE SET NULL
);

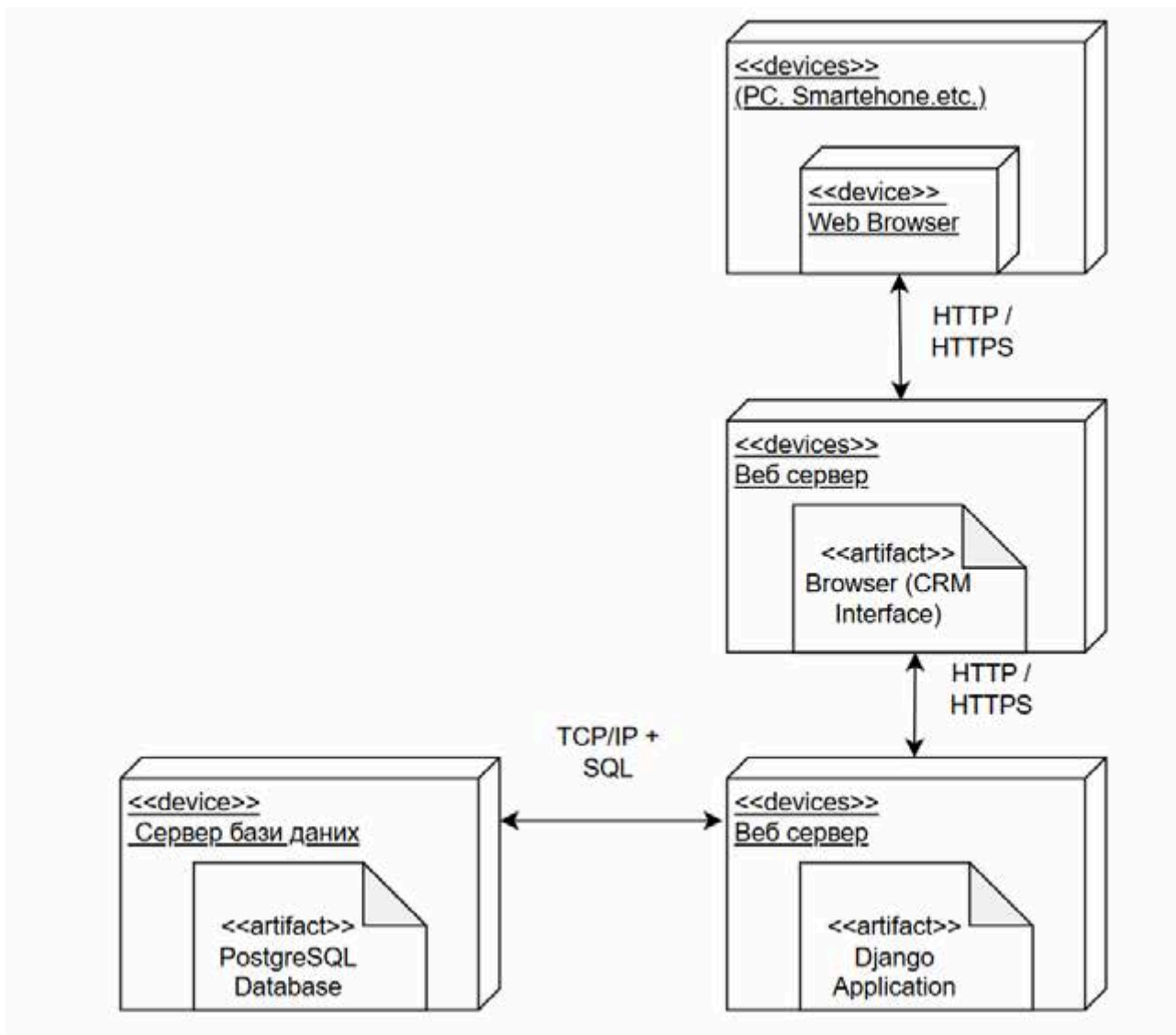
```

ДОДАТОК Г

ДОДАТОК Д



ДОДАТОК Е



ДОДАТОК Ж**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ****Факультет інформаційних технологій****Декларація про академічну доброчесність та використання ШІ****ДЕКЛАРАЦІЯ ЗДОБУВАЧА**

Я, Косінова Владислава Олександрівна, здобувачка НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проект) на тему «Програмне забезпечення обліку процесів для малого бізнесу» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
 - інструмент ШІ (Gemini) були використані для:
 - [] перекладу іншомовних джерел;
 - [✓] стилістичного редагування та перевірки граматики;
 - [] допомоги у написанні/відлагодженні програмного коду;
 - [] інше: _____.

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

«___» _____ 2026 р. _____ **Владислава Косінова**
(підпис)