**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»**

Інститут комп’ютерних систем

Кафедра інженерії програмного забезпечення

КУРСОВИЙ ПРОЕКТ

з\_\_\_\_\_\_\_\_\_\_\_\_\_\_Розробка інформаційних систем\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва дисципліни)

на тему:\_\_\_\_\_\_\_\_\_»Відділ збуту готової продукції булочна»\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки)\_\_\_\_\_\_\_Барабаш Д.Є.\_\_\_\_\_\_\_\_\_

групи\_\_\_АС-226\_\_

спеціальності \_Інженерія програмного

забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник\_\_\_\_\_\_Зіноватна C.Л.\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Кандидат технічних наук, доцент Доцент\_\_\_

Кафедри інженерії програмного\_\_\_\_\_\_\_\_

забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь)

Зміст

АНОТАЦІЯ 3

ВСТУП 4

1 УТОЧНЕННЯ ВИМОГ5

2 ВИЗНАЧЕННЯ КОНЦЕПТУАЛЬНИХ КЛАСІВ 6

3 ПОБУДОВА ДІАГРАМ ВЗАЄМОДІЇ З ВИКОРИСТАННЯМ ШАБЛОНІВ ПРОЕКТУВАННЯ 8

4 РОЗРОБКА КОНЦЕПТУАЛЬНОЇ ТА РЕЛЯЦІЙНОЇ МОДЕЛІ 9

5. СКЛАДАННЯ СПЕЦИФІКАЦІЙ ПРОГРАМНИХ КЛАСІВ 11

6 СТВОРЕННЯ ЗАПИТІВ НА SQL 13

6.1-- Запит 1: Вибір з декількох таблиць із сортуванням 13

6.2-- Запит 2: Завдання умови відбору з використанням предиката LIKE 13

6.3-- Запит 3: Завдання умови відбору з використанням предиката

BETWEEN 13

6.4-- Запит 4: Агрегатна функція без угруповання 13

6.5-- Запит 5: Агрегатна функція з угрупованням 14

6.6-- Запит 6: Використання предиката ALL або ANY 15

6.7-- Запит 7: Корельований підзапит 15

6.8-- Запит 8.1: Запит на заперечення (LEFT JOIN) 16

6.9-- Запит 8.2: Запит на заперечення (IN) 16

6.11— Запит 9: Операція об'єднання UNION із включенням коментарю в кожен рядок 17

7 НАПИСАННЯ КОДУ 18

8 ТЕСТУВАННЯ 33

8.1Тестування окремих методів класу з застосуванням методів білого ящику 35

8.2 Тестування окремих класів з застосуванням методів сірого ящику 36

8.3 Тестування програми із застосуванням методу чорного ящику 37

9 СКРІНШОТИ ДОДАТКУ 37

**АНОТАЦІЯ**

У даній курсовій роботі розроблено систему управління продажами продукції з використанням мови програмування Python та бази даних PostgreSQL. Система включає функціонал для адміністраторів і клієнтів, що дозволяє здійснювати замовлення продукції, керувати замовленнями та постачанням, а також надавати інформацію про продукцію, замовників і постачальників. У роботі також виконано аналіз вимог, розробку концептуальних і реляційних моделей даних, створено SQL-запити, написано програмний код та проведено тестування.

**ВСТУП**

Актуальність:

У сучасних умовах розвитку бізнесу важливо забезпечити ефективне управління процесами продажів та постачання продукції. Інформаційні системи відіграють ключову роль у цьому процесі, дозволяючи автоматизувати рутинні операції, зменшити людські помилки та покращити обслуговування клієнтів. Розробка системи управління продажами продукції є актуальною задачею, оскільки дозволяє оптимізувати процеси замовлення та постачання, що є критичними для успішного функціонування компанії.

Мета розробки:

Метою даної роботи є розробка системи управління продажами продукції, яка забезпечує:

• можливість клієнтам робити замовлення, переглядати доступні продукти та статус замовлень;

• адміністрування замовлень, продуктів і постачань;

• зберігання та обробку даних у реляційній базі даних;

• надання інтерфейсу для взаємодії з системою як для клієнтів, так і для адміністраторів.

**1** **УТОЧНЕННЯ ВИМОГ**

Виявлення зацікавлених осіб і їхніх потреб:

- Замовники: Потребують зручного способу замовлення продуктів.

- Постачальники: Потребують інформації про замовлення та постачання продуктів.

- Магазини: Потребують обліку замовлень та управління запасами.

Виявлення основних виконавців:

- Користувачі системи: Замовники, адміністратори магазинів, постачальники.

- Системні адміністратори: Підтримка та управління базою даних.

Визначення загальних вимог до системи та створення документа «Бачення":

- Система повинна дозволяти користувачам робити замовлення, переглядати доступні продукти та статус замовлень.

- Постачальники повинні мати доступ до інформації про замовлення для організації постачання.

- Адміністратори магазинів повинні мати змогу керувати запасами та обробляти замовлення.

Створення словника предметної області:

- Замовник (Customer): Особа, яка робить замовлення.

- Продукт (Product): Товар, який можна замовити.

- Замовлення (Order): Документ, що фіксує замовлення продукту.

- Магазин (Store): Місце, де зберігаються продукти і здійснюються продажі.

- Постачальник (Supplier): Організація, що постачає продукти.

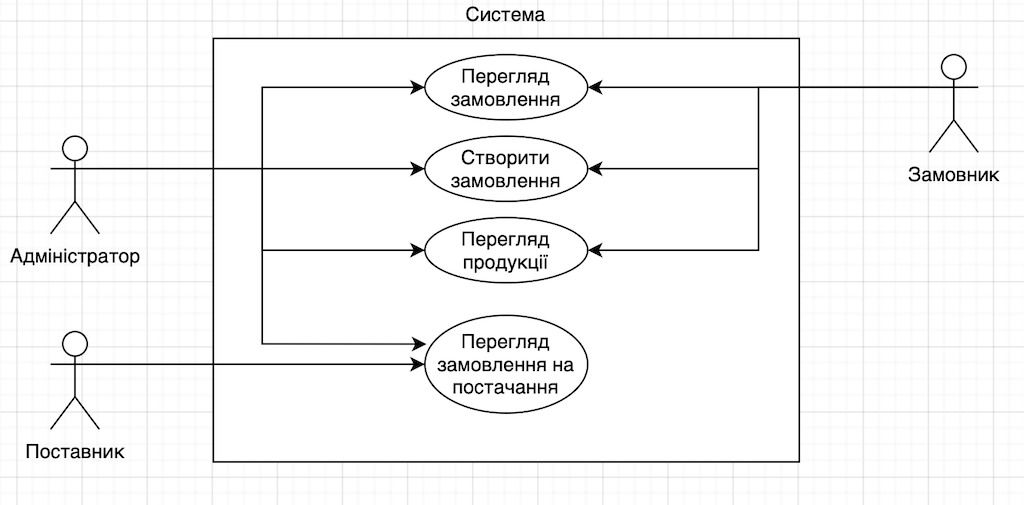
Створення діаграми прецедентів:

Рисунок 1.1— Діаграма прецендентів

Визначення програмного модуля:

- Модуль замовлень: Реалізує функції створення, редагування та перегляду

замовлень.

Опис двох прецедентів:

1. Створення замовлення:

- Користувач обирає продукти зі списку продуктів у вікні створення замовлення.

- Користувач у вікні створення замовлення обирає кількість продукції, дату доставку

- Користувач підтверджує замовлення натискаючи кнопку зберегти у вікні створення замовлення.

- Система зберігає замовлення в базі даних.

2. Перегляд замовлень:

- Користувач в основному вікні дій натискає на кнопку «Переглянути замовлення».

- Система бере інформацію з бази даних і відображає замовлення із зазначенням їхніх характеристик(кількість, дата доставки).

- Клієнт отримує таблицю і має змогу видалити замовлення(вибрати рядок у таблиці і натиснути кнопку «Видалити»)

Сценарії варіантіввикористання

Для користувача:

1. Створення замовлення:

• Користувач обирає продукти зі списку продуктів у вікні створення замовлення.

• Користувач у вікні створення замовлення обирає кількість продукції, дату доставку.

• Користувач підтверджує замовлення натискаючи кнопку зберегти у вікні створення замовлення

Система зберігає замовлення в базі даних

2. Перегляд замовлень:

• Користувач переглядає список своїх замовлень, відображених у системі.

• Система відображає деталі кожного замовлення (продукт, кількість, дата доставки).

3. Видалити замовлення:

• Користувач обирає замовлення, яке він хоче видалити і натискає кнопку «Видалити».

• Система видаляє замовлення з бази даних.

Для адміністратора:

1. Переглянути замовлення:

• Адміністратор переглядає список всіх замовлень у системі.

• Система відображає деталі кожного замовлення (продукт, кількість, замовник, дата доставки).

2. Видалити замовлення:

• Адміністратор обирає замовлення, яке він хоче видалити.

• Система видаляє замовлення з бази даних.

3. Переглянути замовників:

• Адміністратор переглядає список всіх замовників у системі.

• Система відображає деталі кожного замовника (ім’я, адреса, телефон).

4. Видалити замовника:

• Адміністратор обирає замовника, якого він хоче видалити.

• Система видаляє замовника з бази даних.

5. Переглянути продукти:

• Адміністратор переглядає список всіх продуктів у системі.

• Система відображає деталі кожного продукту (назва, ціна, опис).

6. Додати поставку:

• Адміністратор додає нову поставку продуктів до системи.

• Система зберігає деталі поставки у базі даних (продукт, постачальник, дата поставки).

7. Переглянути поставки:

• Адміністратор переглядає список всіх поставок у системі.

• Система відображає деталі кожної поставки (продукт, постачальник, дата поставки).

8. Видалити поставку:

• Адміністратор обирає поставку, яку він хоче видалити.

• Система видаляє поставку з бази даних.

**2 ВИЗНАЧЕННЯ КОНЦЕПТУАЛЬНИХ КЛАСІВ**

Сутності:

- Продукт (Product)

- product\_id (Ідентифікатор продукту)

- назва (Назва продукту)

- ціна (Ціна продукту)

- рік\_випуску (Рік випуску продукту)

- виконавець (Автор або виконавець)

- Замовник (Customer)

- customer\_id (Ідентифікатор замовника)

- ім'я (Ім'я замовника)

- адреса (Адреса замовника)

- телефон (Контактний телефон)

- Замовлення (Order)

- order\_id (Ідентифікатор замовлення)

- кількість (Кількість продукту, що було замовлено)

- дата (Дата замовлення)

- Магазин (Store)

- store\_id (Ідентифікатор магазину)

- назва (Назва магазину)

- адреса (Адреса магазину)

- графік\_роботи (Графік роботи магазину)

- Постачальник (Supplier)

- supplier\_id (Ідентифікатор постачальника)

- назва (Назва постачальника)

- контактна\_особа (Контактна особа)

- адреса (Адреса постачальника)

Зв'язки:

- Замовник -- (має) --> Замовлення

- Замовлення -- (містить) --> Продукт

- Замовлення -- (розміщене\_в) --> Магазин

- Продукт -- (постачається\_від) --> Постачальник

Обмеження:

- Кожен продукт має унікальний ідентифікатор (product\_id).

- Кожен замовник має унікальний ідентифікатор (customer\_id).

- Кожне замовлення має унікальний ідентифікатор (order\_id).

- Кожен магазин має унікальний ідентифікатор (store\_id).

- Кожен постачальник має унікальний ідентифікатор (supplier\_id).

- Кількість продукту на одне замовлення обмежена.

- Кожне замовлення повинно бути пов'язане з існуючим замовником, продуктом і магазином.

- Ідентифікатори замовників, продуктів і магазинів, використані у замовленнях, повинні існувати.

**3 ПОБУДОВА ДІАГРАМ ВЗАЄМОДІЇ З ВИКОРИСТАННЯМ ШАБЛОНІВ ПРОЕКТУВАННЯ**

Опис системних операцій:

1. Система зберігає замовлення в базі даних.

• Після підтвердження замовлення користувачем, система автоматично зберігає дані про замовлення у базі даних.

2. Система видаляє замовлення з бази даних.

• Після вибору користувачем або адміністратором замовлення для видалення, система автоматично видаляє відповідні дані.

3. Система зберігає поставки в базі даних.

• Після додавання нової поставки адміністратором, система зберігає відповідні дані у базі даних.

4. Система видаляє поставку з бази даних.

• Після вибору адміністратором поставки для видалення, система автоматично видаляє відповідні дані.

5. Система надає інформацію про замовлення.

• Система відображає відповідні дані про замовлення за запитом користувача або адміністратора, забезпечуючи актуальну та точну інформацію.

6. Система надає інформацію про поставки.

• Система відображає відповідні дані про поставки за запитом користувача або адміністратора, забезпечуючи актуальну та точну інформацію.

7. Система надає інформацію про замовників.

• Система відображає відповідні дані про замовників за запитом адміністратора, забезпечуючи актуальну та точну інформацію.

8. Система надає інформацію про постачальників.

• Система відображає відповідні дані про постачальників за запитом адміністратора, забезпечуючи актуальну та точну інформацію.

9. Система надає інформацію про продукти.

• Система відображає відповідні дані про продукти за запитом користувача або адміністратора, забезпечуючи актуальну та точну інформацію.

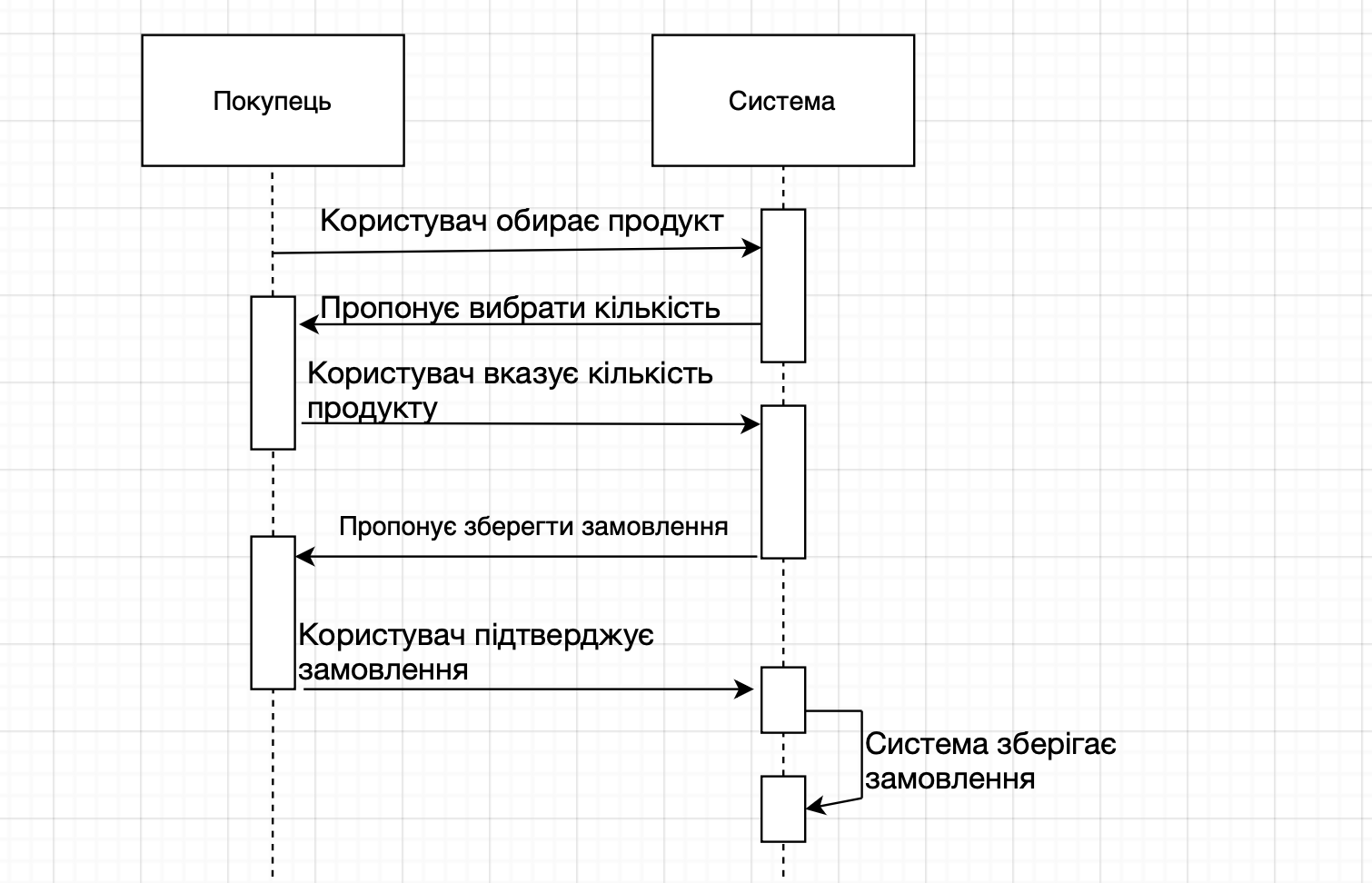
Діаграма послідовностей:

Рисунок 3.1— Діаграма послідовностей

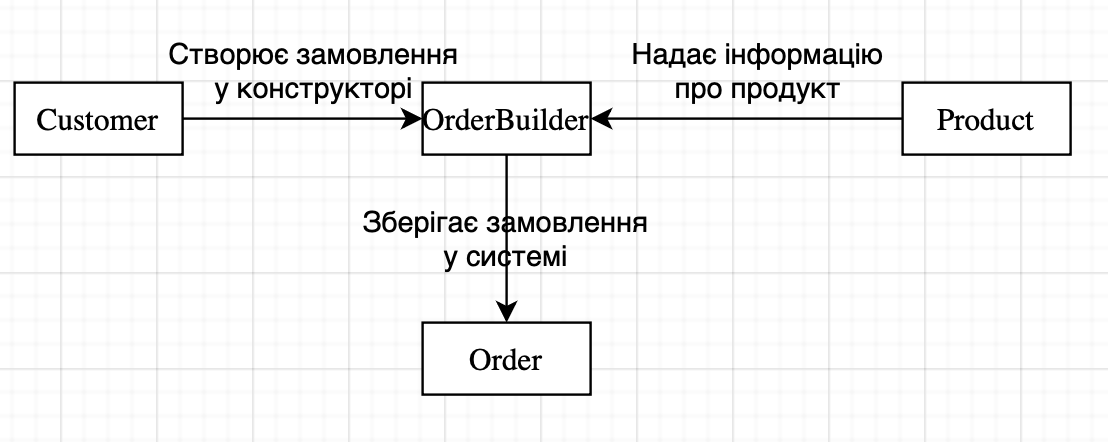
Діаграми кооперації:

Рисунок 3.1— Діаграми кооперації

**4 РОЗРОБКА КОНЦЕПТУАЛЬНОЇ ТА РЕЛЯЦІЙНОЇ МОДЕЛІ**

Типи зв'язків:

- Замовник - Замовлення (багато до багатьох)

- Замовлення - Продукт (один до багатьох)

- Замовлення - Магазин (багато до одного)

- Продукт - Постачальник (багато до багатьох)

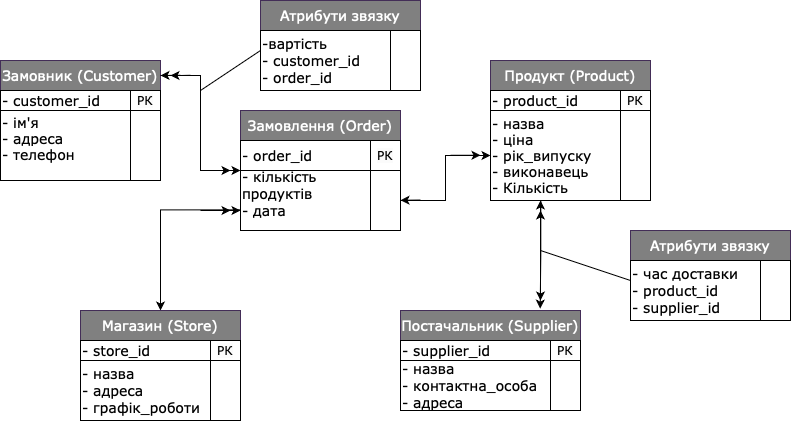
Діаграма "сутність-зв'язок":

Рисунок 3.1— Діаграма "сутність-зв'язок"

Реляційну модель даних:

1) Продукт (**product\_id**, назва, ціна, рік\_випуску, виконавець, order\_id(зовнішній ключ))

2) Замовник (**customer\_id**, ім’я, адреса, телефон)

3) Замовлення (**order\_id**, кількість, дата, customer\_id(Зовнішній ключ))

4) Магазин (**store\_id**, назва, адреса, графік\_роботи)

5) Постачальник (**supplier\_id**, назва, контактна\_особа, адреса)

6) Доставка (час доставки, product\_id, supplier\_id)

7) Кількість грошей (вартість, ﻿customer\_id, ﻿order\_id)

Обмеження:

- Кожен продукт має унікальний ідентифікатор (product\_id).

- Кожен замовник має унікальний ідентифікатор (customer\_id).

- Кожне замовлення має унікальний ідентифікатор (order\_id).

- Кожен магазин має унікальний ідентифікатор (store\_id).

- Кожен постачальник має унікальний ідентифікатор (supplier\_id).

- Кількість продукту на одне замовлення обмежена.

- Кожне замовлення повинно бути пов'язане з існуючим замовником, продуктом і магазином.

- Ідентифікатори замовників, продуктів і магазинів, використані у замовленнях, повинні існувати.

**5 СКЛАДАННЯ СПЕЦИФІКАЦІЙ ПРОГРАМНИХ КЛАСІВ**

—Клас Product

Атрибути:

- product\_id: int

- name: str

- price: float

- release\_year: int

- performer: str

Методи:

- get\_product\_details(): Отримує детальну інформацію про продукт.

- set\_product\_details(name, price, release\_year, performer): Встановлює детальну інформацію про продукт.

—Клас Customer

Атрибути:

- customer\_id: int

- name: str

- address: str

- phone: str

Методи:

- get\_customer\_details(): Отримує детальну інформацію про замовника.

- set\_customer\_details(name, address, phone): Встановлює детальну інформацію про замовника.

—Клас Order

Атрибути:

- order\_id: int

- quantity: int

- order\_date: datetime.date

- customer\_id: int

- product\_id: int

Методи:

- create\_order(customer\_id, product\_id, quantity, order\_date): Створює нове замовлення.

- get\_order\_details(): Отримує детальну інформацію про замовлення.

—Клас Store

Атрибути:

- store\_id: int

- name: str

- address: str

- working\_hours: str

Методи:

- get\_store\_details(): Отримує детальну інформацію про магазин.

- set\_store\_details(name, address, working\_hours): Встановлює детальну інформацію про магазин.

—Клас Supplier

Атрибути:

- supplier\_id: int

- name: str

- contact\_person: str

- address: str

Методи:

- get\_supplier\_details(): Отримує детальну інформацію про постачальника.

- set\_supplier\_details(name, contact\_person, address): Встановлює детальну інформацію про постачальника.

—Клас Delivery

Атрибути:

- delivery\_id: int

- delivery\_time: datetime.datetime

- product\_id: int

- supplier\_id: int

Методи:

- get\_delivery\_details(): Отримує детальну інформацію про поставку.

- set\_delivery\_details(delivery\_time, product\_id, supplier\_id): Встановлює детальну інформацію про поставку.

**6 СТВОРЕННЯ ЗАПИТІВ НА SQL**

6.1-- Запит 1: Вибір з декількох таблиць із сортуванням

CREATE OR REPLACE FUNCTION query1()

RETURNS TABLE (

customer\_id INT,

customer\_name VARCHAR,

product\_id INT,

product\_name VARCHAR,

order\_id INT,

order\_date DATE

) AS $$

BEGIN

RETURN QUERY

SELECT

c.customer\_id,

c.name AS customer\_name,

p.product\_id,

p.name AS product\_name,

o.order\_id,

o.order\_date

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

JOIN

products p ON o.product\_id = p.product\_id

ORDER BY

c.name, o.order\_date;

END;

$$ LANGUAGE plpgsql;

6.2-- Запит 2: Завдання умови відбору з використанням предиката LIKE

CREATE OR REPLACE FUNCTION query2()

RETURNS TABLE (

customer\_id INT,

name VARCHAR,

address VARCHAR

) AS $$

BEGIN

RETURN QUERY

SELECT

c.customer\_id,

c.name,

c.address

FROM

customers c

WHERE

c.name LIKE 'А%';

END;

$$ LANGUAGE plpgsql;

6.3-- Запит 3: Завдання умови відбору з використанням предиката

BETWEEN

CREATE OR REPLACE FUNCTION query3()

RETURNS TABLE (

order\_id INT,

product\_id INT,

quantity INT,

order\_date DATE

) AS $$

BEGIN

RETURN QUERY

SELECT

o.order\_id,

o.product\_id,

o.quantity,

o.order\_date

FROM

orders o

WHERE

o.order\_date BETWEEN '2024-01-01' AND '2024-12-31';

END;

$$ LANGUAGE plpgsql;

6.4-- Запит 4: Агрегатна функція без угруповання

CREATE OR REPLACE FUNCTION query4()

RETURNS TABLE (

total\_orders BIGINT

) AS $$

BEGIN

RETURN QUERY

SELECT

COUNT(\*) AS total\_orders

FROM

orders

WHERE

order\_date >= CURRENT\_DATE - interval '7 days';

END;

$$ LANGUAGE plpgsql;

6.5-- Запит 5: Агрегатна функція з угрупованням

CREATE OR REPLACE FUNCTION query5()

RETURNS TABLE (

customer\_id INT,

total\_orders BIGINT

) AS $$

BEGIN

RETURN QUERY

SELECT

o.customer\_id,

COUNT(o.order\_id) AS total\_orders

FROM

orders o

GROUP BY

o.customer\_id;

END;

$$ LANGUAGE plpgsql;

6.6-- Запит 6: Використання предиката ALL або ANY

CREATE OR REPLACE FUNCTION query6()

RETURNS TABLE (

customer\_id INT,

name VARCHAR

) AS $$

BEGIN

RETURN QUERY

SELECT

c.customer\_id,

c.name

FROM

customers c

WHERE

c.customer\_id = (SELECT o.customer\_id FROM orders o GROUP BY o.customer\_id ORDER BY COUNT(o.order\_id) DESC LIMIT 1);

END;

$$ LANGUAGE plpgsql;

6.7-- Запит 7: Корельований підзапит

CREATE OR REPLACE FUNCTION query7()

RETURNS TABLE (

customer\_id INT,

name VARCHAR,

total\_orders BIGINT

) AS $$

BEGIN

RETURN QUERY

SELECT

c.customer\_id,

c.name,

(SELECT COUNT(\*) FROM orders o WHERE o.customer\_id = c.customer\_id) AS total\_orders

FROM

customers c;

END;

$$ LANGUAGE plpgsql;

6.8-- Запит 8.1: Запит на заперечення (LEFT JOIN)

CREATE OR REPLACE FUNCTION query8\_1()

RETURNS TABLE (

customer\_id INT,

name VARCHAR

) AS $$

BEGIN

RETURN QUERY

SELECT

c.customer\_id,

c.name

FROM

customers c

LEFT JOIN

orders o ON c.customer\_id = o.customer\_id

WHERE

o.order\_id IS NULL;

END;

$$ LANGUAGE plpgsql;

6.9-- Запит 8.2: Запит на заперечення (IN)

CREATE OR REPLACE FUNCTION query8\_2()

RETURNS TABLE (

customer\_id INT,

name VARCHAR

) AS $$

BEGIN

RETURN QUERY

SELECT

c.customer\_id,

c.name

FROM

customers c

WHERE

c.customer\_id NOT IN (SELECT o.customer\_id FROM orders o);

END;

$$ LANGUAGE plpgsql;

6.10-- Запит 8.3: Запит на заперечення (EXISTS)

CREATE OR REPLACE FUNCTION query8\_3()

RETURNS TABLE (

customer\_id INT,

name VARCHAR

) AS $$

BEGIN

RETURN QUERY

SELECT

c.customer\_id,

c.name

FROM

customers c

WHERE

NOT EXISTS (SELECT 1 FROM orders o WHERE o.customer\_id = c.customer\_id);

END;

$$ LANGUAGE plpgsql;

6.11— Запит 9: Операція об'єднання UNION із включенням коментарю в кожен рядок

CREATE OR REPLACE FUNCTION query9()

RETURNS TABLE (

customer\_id INT,

name VARCHAR,

comment VARCHAR

) AS $$

BEGIN

RETURN QUERY

SELECT

c.customer\_id,

c.name,

'Має максимальну кількість замовлень' AS comment

FROM

customers c

WHERE

c.customer\_id = (SELECT o.customer\_id FROM orders o GROUP BY o.customer\_id ORDER BY COUNT(o.order\_id) DESC LIMIT 1)

UNION

SELECT

c.customer\_id,

c.name,

'Не має в цей час замовлень' AS comment

FROM

customers c

WHERE

c.customer\_id NOT IN (SELECT o.customer\_id FROM orders o);

END;

$$ LANGUAGE plpgsql;

**7 НАПИСАННЯ КОДУ**

Весь код знаходиться у файлі «Додаток.py»

Додаток:

import tkinter as tk # Імпорт бібліотеки для створення GUI

from tkinter import ttk, messagebox # Імпорт необхідних модулів з tkinter

import psycopg2 # Імпорт бібліотеки для роботи з PostgreSQL

# Функція для підключення до бази даних

def connect\_db():

return psycopg2.connect(dbname="KR", user="danilbarabas", password="090105", host="localhost", port="5432")

# Основний клас програми

class Application(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.title("Система управління продажами")

self.geometry("400x300")

self.create\_widgets() # Створення віджетів

# Функція для обробки входу користувача

def login(self):

conn, role, login, password = connect\_db(), self.user\_role.get(), self.user\_login.get(), self.user\_password.get()

cursor = conn.cursor()

table = "administrators" if role == "admin" else "customers"

cursor.execute(f"SELECT \* FROM {table} WHERE login=%s AND password=%s", (login, password))

user = cursor.fetchone()

conn.close()

if user:

self.destroy()

AdminInterface() if role == "admin" else ClientInterface()

else:

messagebox.showerror("Помилка", "Неправильний логін або пароль")

**8 ТЕСТУВАННЯ**

8.1Тестування окремих методів класу з застосуванням методів білого ящику

Чек-аркуш для розуміння загального процесу тестування методів класу:

Таблиця 8.1— Чек-аркуш тестування методів класу 1

| **№** | **Дія** | **Очікування** | **Результат** | **Коментар** |
| --- | --- | --- | --- | --- |
| 1 | Ініціалізація обʼєкту  класу | Обʼєкт успішно  Ствopений | Успіх | Обʼєкт класу Product  створений без помилок |
| 2 | Виклик методу  get\_product\_details | Повертає правильні дані | Успіх | Дані продукту відповідають очікуваним |
| 3 | Виклик методу  set\_product\_details | Дані успішно оновлені | Успіх | Дані продукту успішно оновлені |
| 4 | Перевірка оновлених  даних | Повертає оновлені дані | Успіх | Оновлені дані продукту відповідають очікуваним |

Тест-кейси для тестування окремих методів класу Product:

Таблиця 8.2— Тест-кейси окремих методів класу Product

| **№** | **Дія** | **Очікування** | **Результат** | **Коментар** |
| --- | --- | --- | --- | --- |
| 1 | Створення продукту з правильними даними | Продукт успішно  cтворений | Успіх | Продукт з правильними  Даними створений |
| 2 | Отримання деталей продукту | Повертає правильні дані продукту | Успіх | Дані продукту відповідають очікуваним |
| 3 | Оновлення деталей продукту | Дані продукту успішно оновлені | Успіх | Дані продукту успішно оновлені |
| 4 | Перевірка оновлених  даних продукту | Повертає оновлені дані  продукту | Успіх | Оновлені дані продукту відповідають очікуваним |

Драйвери та заглушки для тестування окремих методів класу Product:

Драйвер для тестування методу set\_product\_details  
def test\_set\_product\_details():

product = Product(1, "Product1", 100.0, 2023, "Performer1")

product.set\_product\_details("Product2", 200.0, 2024, "Performer2")

details = product.get\_product\_details()

assert details['name'] == "Product2"

assert details['price'] == 200.0

assert details['release\_year'] == 2024

assert details['performer'] == "Performer2"

test\_set\_product\_details()

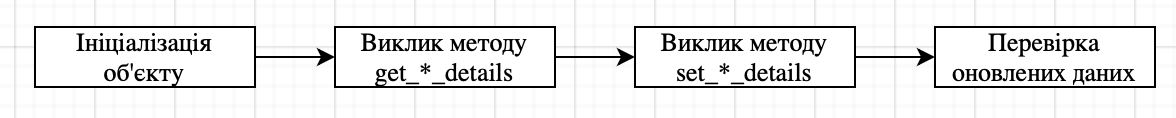
Схема тестування окремих методів класу:

Рисунок 8.1— Схема тестування класів 1

8.2 Тестування окремих класів з застосуванням методів сірого ящику

Чек-аркуш для розуміння загального процесу тестування класів:

Таблиця 8.1— Чек-аркуш тестування методів класу 2

| **№** | **Дія** | **Очікування** | **Результат** | **Коментар** |
| --- | --- | --- | --- | --- |
| 1 | Ініціалізація класу | Обʼєкт успішно  створений | Успіх | Обʼєкт класу Order  створений без помилок |
| 2 | Виклик методів класу | Методи виконуються без помилок | Успіх | Всі методи  виконуються коректно |
| 3 | Перевірка даних після виклику методів | Повертає очікувані дані | Успіх | Дані після виклику методів відповідають очікуваним |

Тест-кейси для тестування класу Order:

Таблиця 8.2— Тест-кейси окремих методів класу Order

| **№** | **Дія** | **Очікування** | **Результат** | **Коментар** |
| --- | --- | --- | --- | --- |
| 1 | Створення замовлення з правильними даними | Замовлення успішно  cтворений | Успіх | Замовлення з правильними  Даними створений |
| 2 | Отримання деталей замовлення | Повертає правильні дані замовлення | Успіх | Дані замовлення відповідають очікуваним |
| 3 | Оновлення деталей замовлення | Дані замовлення успішно оновлені | Успіх | Дані замовлення успішно оновлені |
| 4 | Перевірка оновлених  даних замовлення | Повертає оновлені дані  Замовлення | Успіх | Оновлені дані замовлення відповідають очікуваним |

Драйвери та заглушки для тестування класу Order

Заглушка для тестування класу Order:

class MockDatabase:

def \_\_init\_\_(self):

self.data = {}

def insert(self, table, record):

if table not in self.data:

self.data[table] = []

self.data[table].append(record)

def fetch(self, table):

return self.data.get(table, [])

def test\_create\_order():

mock\_db = MockDatabase()

order = Order(1, 1, 1, 2, "2024-06-10")

mock\_db.insert('orders', order.get\_order\_details())

assert len(mock\_db.fetch('orders')) == 1

test\_create\_order()

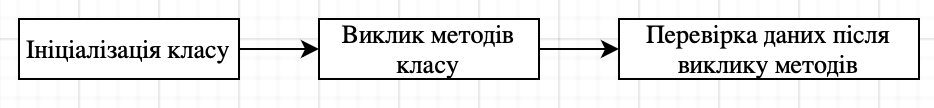
Схема тестування класів:

Рисунок 8.1— Схема тестування класів 2

8.3 Тестування програми із застосуванням методу чорного ящику

Чек-аркуш для розуміння загального процесу тестування програми:

Таблиця 8.1— Чек-аркуш загального процесу тестування програми

| **№** | **Дія** | **Очікування** | **Результат** | **Коментар** |
| --- | --- | --- | --- | --- |
| 1 | Запис програми | Програма успішно запускається | Успіх | Програма запущена без помилок |
| 2 | Логін користувача | Успішний вхід в  систему | Успіх | Логін користувача успішний |
| 3 | Виконання основних  Функцій GUI | Функції виконуються  без помилок | Успіх | Основні функції GUI  виконуються коректно |
| 4 | Виконання SQL запитів  через GUI | Запити повертають  очікувані результати | Успіх | SQL запити  виконуються і повертають очікувані  результати |

Тест-кейси для тестування GUI:

Таблиця 8.2— Тест-кейси тестування GUI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N°** | **Дія** | **Очікуваний результат** | **Результат** | **Коментар** |
| 1 | Логін як адміністратор | Вхід в систему як адміністратор | Успішно | Вхід в систему як адміністратор успішний |
| 2 | Перегляд замовлень | Відображаються всі  замовлення | Успішно | Всі замовлення відображаються  коректно |
|  | Додавання нового замовлення | Нове замовлення  успішно додане | Успішно | Нове замовлення  додане успішно |
| 4 | Видалення замовлення | Замовлення успішно  видалене | Успішно | Замовлення видалене  успішно |

Драйвери та заглушки для тестування GUI

Драйвер для тестування логіну:

import unittest

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

class TestGUI(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Chrome()

def test\_login(self):

driver = self.driver

driver.get("http://localhost:5000") # URL вашої програми

role = driver.find\_element\_by\_name("role")

role.send\_keys("admin")

login = driver.find\_element\_by\_name("login")

login.send\_keys("admin")

password = driver.find\_element\_by\_name("password")

password.send\_keys("admin")

password.send\_keys(Keys.RETURN)

self.assertIn("Адміністратор", driver.title)

def tearDown(self):

self.driver.close()

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

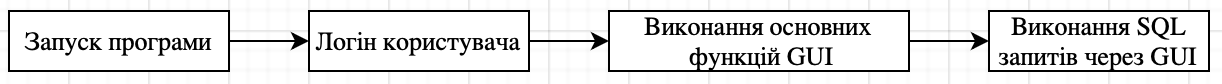
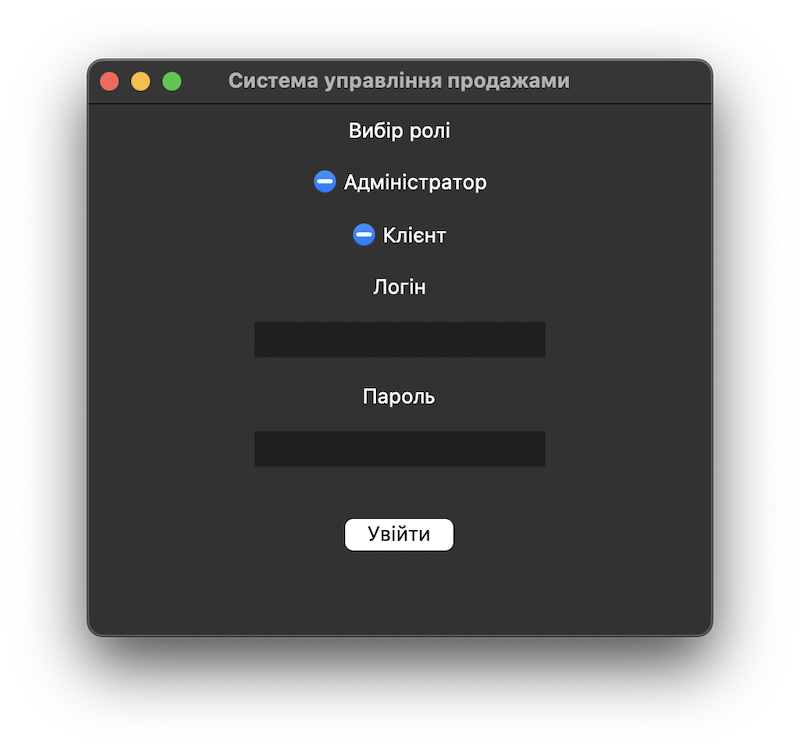
Схема тестування GUI

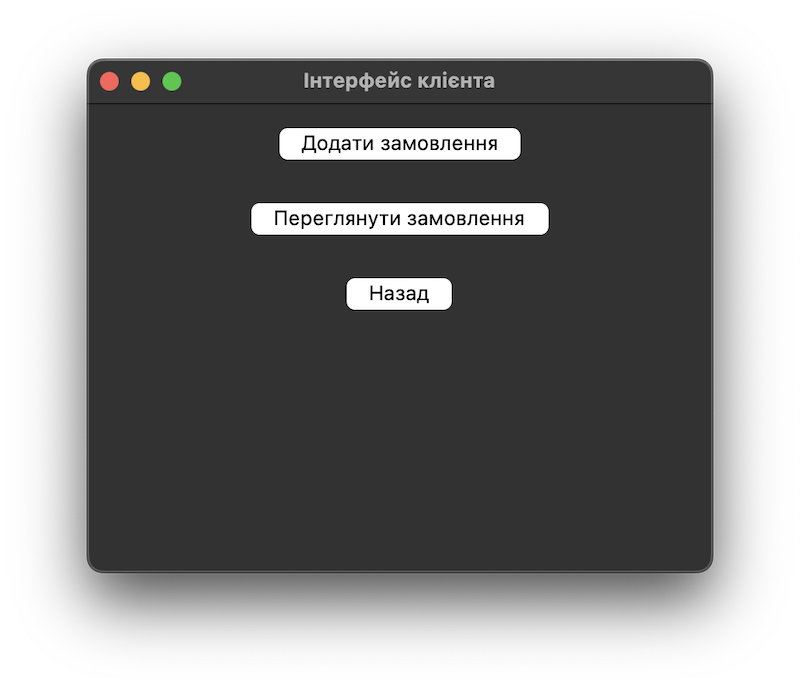
Рисунок 8.1— Схема тестування класів 3

**9 СКРІНШОТИ ДОДАТКУ**

****

1. Вікно заходу в аккаунт:

Рисунок9.1—Вікно заходу в аккаунт



2) Головне вікно клієнта:

Рисунок9.2—Головне вікно клієнта

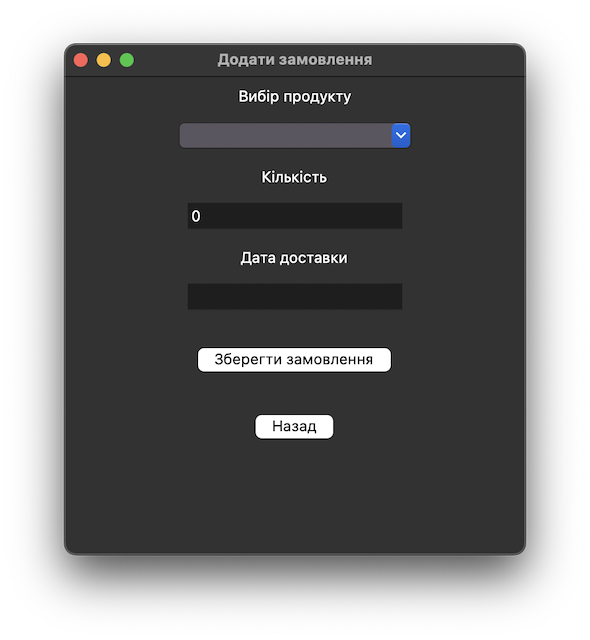
2.1) Вікно додавання замовлення:  


Рисунок9.3—Вікно додавання замовлення

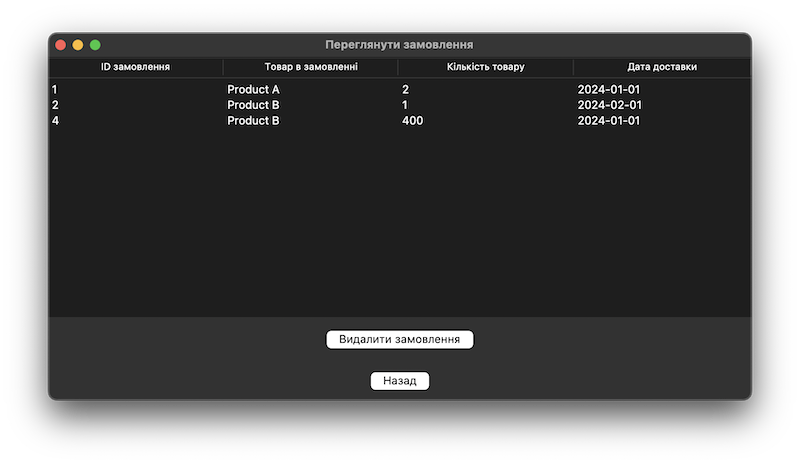
2.2) Вікно перегляду замовлень:

Рисунок9.4—Вікно перегляду замовлень

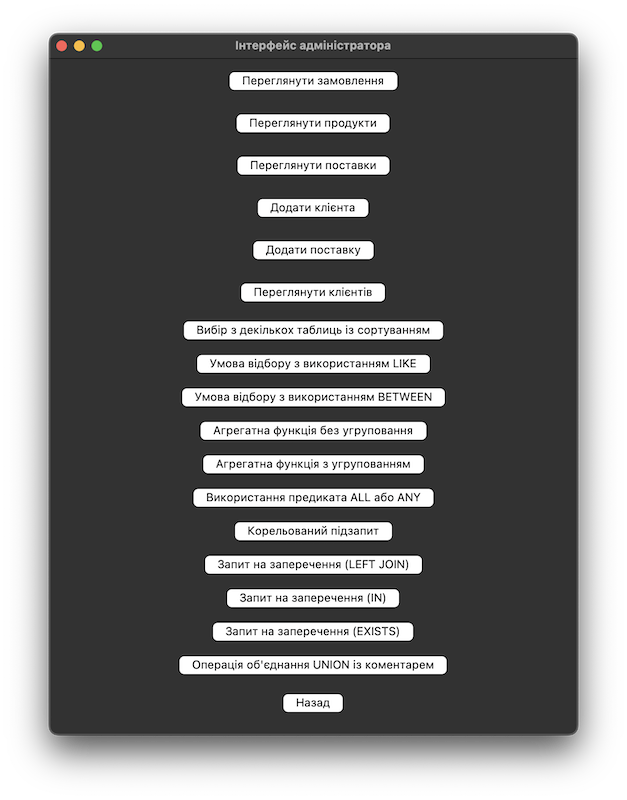
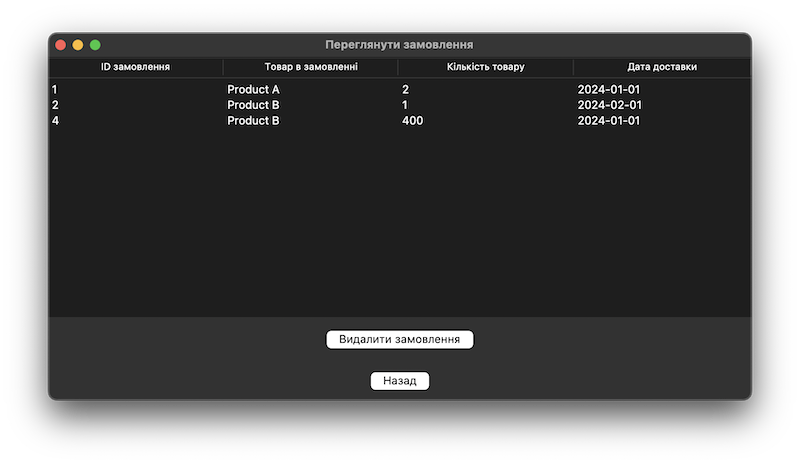
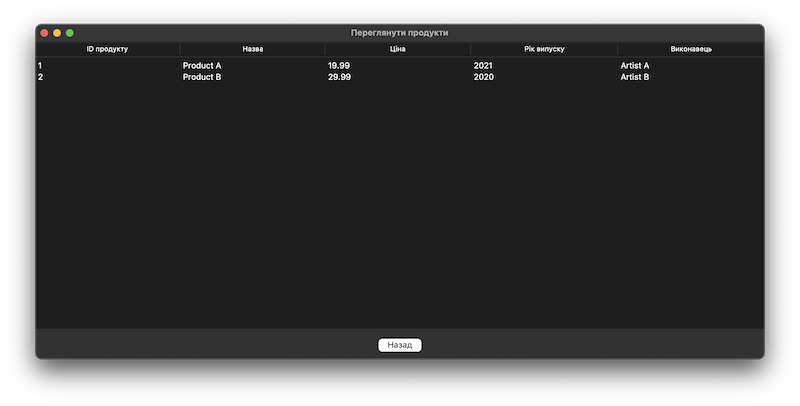
3) Головне вікно адміністратора:

Рисунок9.5—Головне вікно адміністратора



3.1) Вікно для перегляду продуктів, замовлень, поставки, клієнтів:

Рисунок9.6—Вікно перегляду замовлень



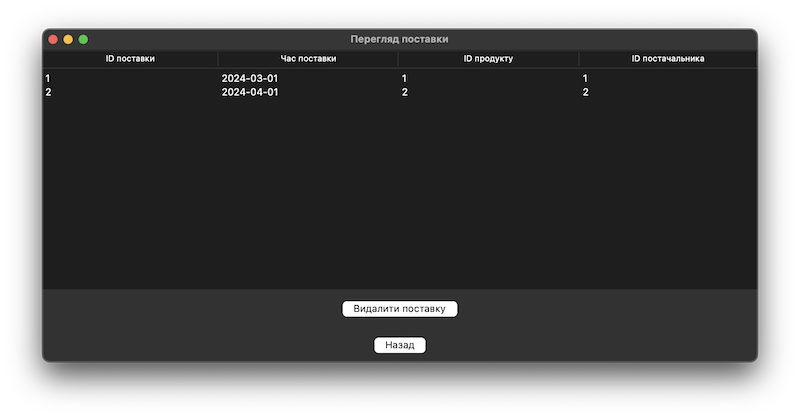
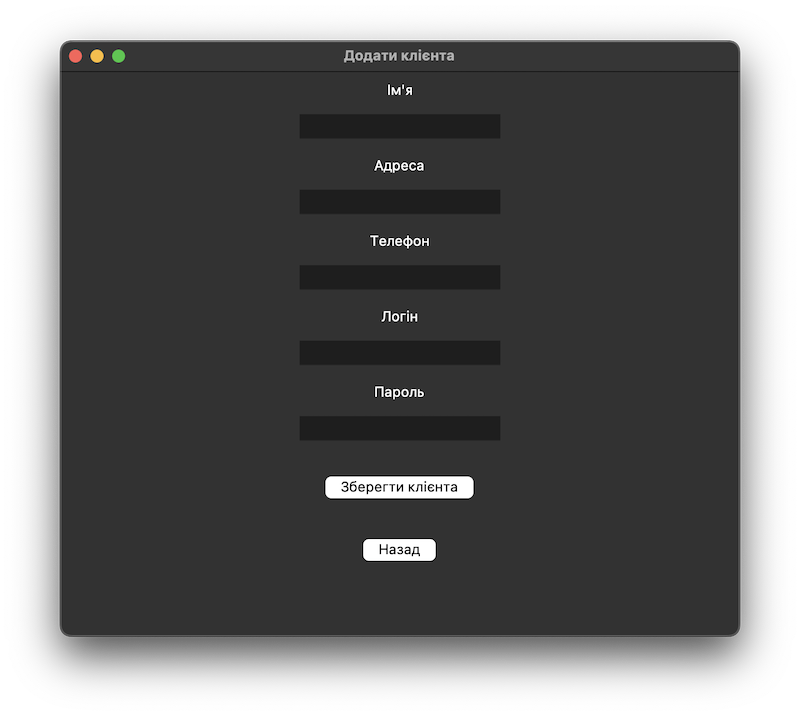
Рисунок9.7—вікно перегляду продуктів

Рисунок9.8— Вікно перегляду поставок

3.2) Вікно для додавання поставки, клієнта:

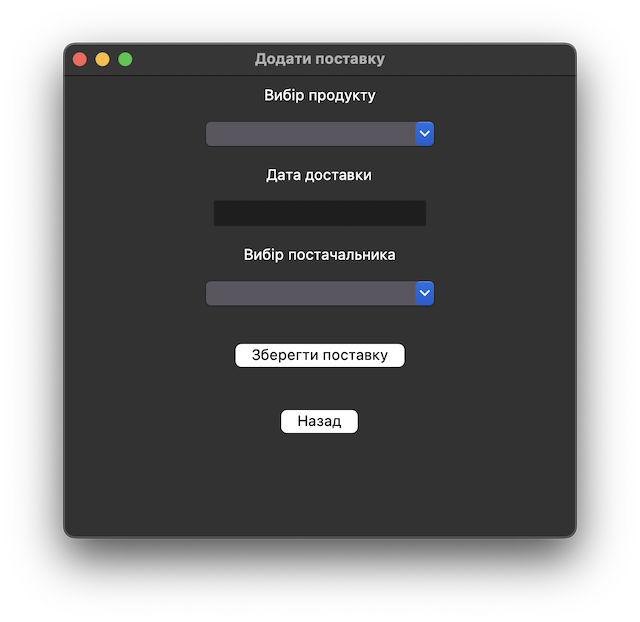


Рисунок9.9—Вікно додавання клієнта

Рисунок9.10—Вікно додавання поставки