

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 С.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 ВИМОГ	5
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
8.1 Тестування окремих методів класу з застосуванням методів білого ящика	35
8.2 Тестування окремих класів з застосуванням методів сірого ящика	36
8.3 Тестування програми із застосуванням методу чорного ящика	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.
- Магазины: 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:

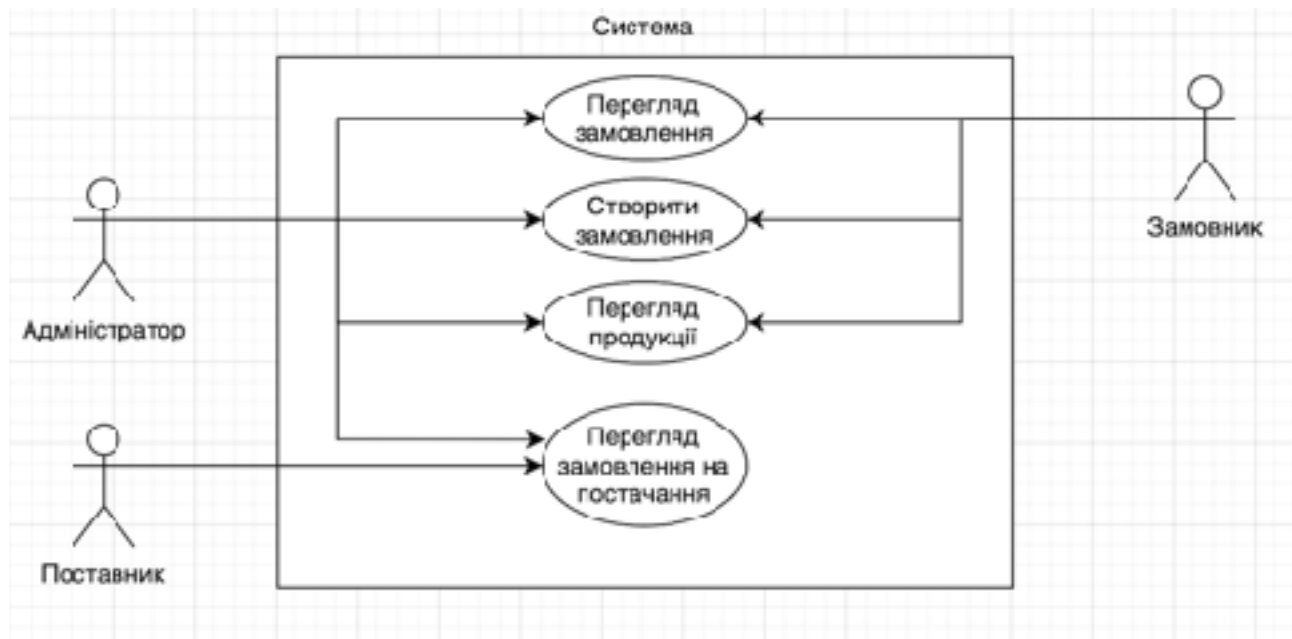


Рисунок 1.1— Діаграма прецедентів

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 доставки
- 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 варіантів використання

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:

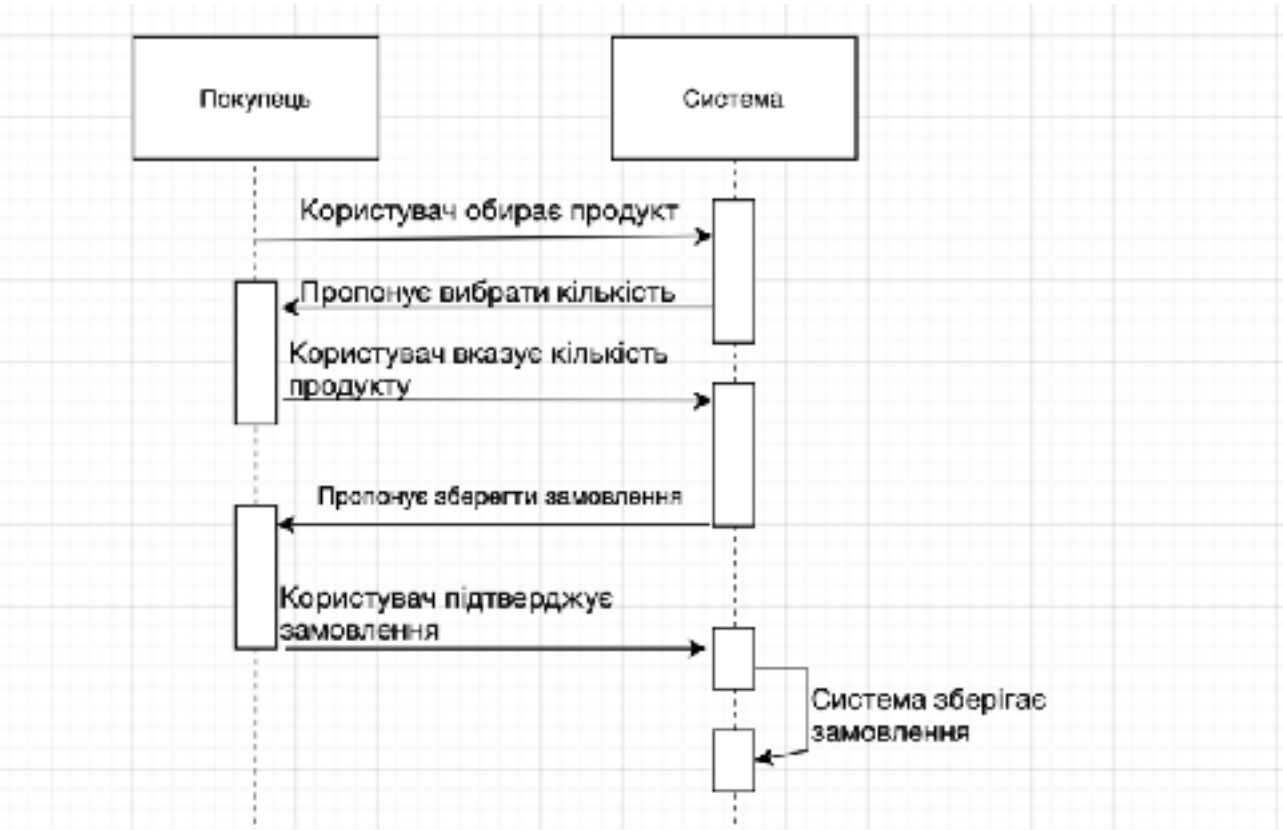


Рисунок 3.1— Діаграма послідовностей

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":

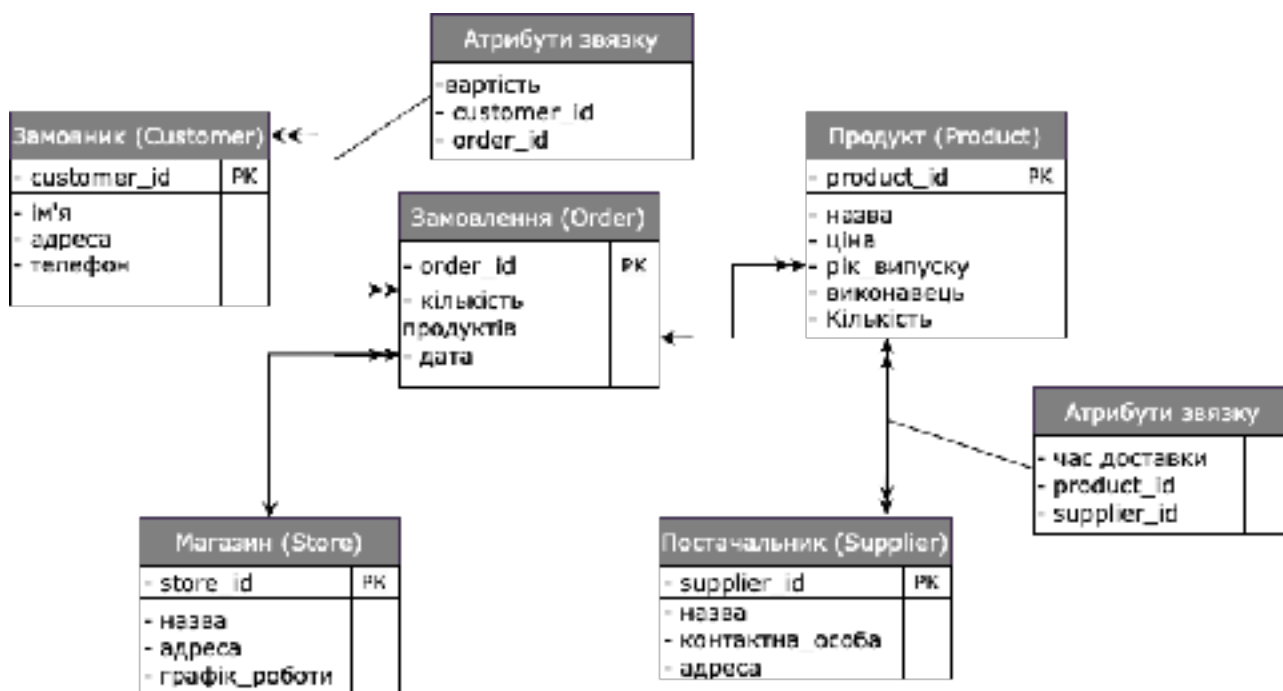


Рисунок 3.1— Діаграма "сутність-зв'язок"

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 $$
BEGIN
    RETURN QUERY
    SELECT
        c.customer_id,
        c.name,
        c.address
    FROM
        customers c
    WHERE
        c.name LIKE 'A%';
END;
$$ 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: Корельованийий 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.ру»

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

8.1 Тестування окремих методів класу з застосуванням методів білого ящика

Чек-аркуш для розуміння загального процесу тестування методів класу:

Таблиця 8.1— Чек-аркуш тестування методів класу 1

№	Дія	Очікування	Результат	Коментар
1	Ініціалізація об'єкту класу	Об'єкт успішно Створений	Успіх	Об'єкт класу Product створений без помилок
2	Виклик методу get_product_details	Повертає правильні дані	Успіх	Дані продукту відповідають очікуваням
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 очікуваням

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 створений	Успіх	Продукт з правильними Даними створений
2	Отримання деталей продукту	Повертає правильні дані продукту	Успіх	Дані продукту відповідають очікуваням
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 очікуваням

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:

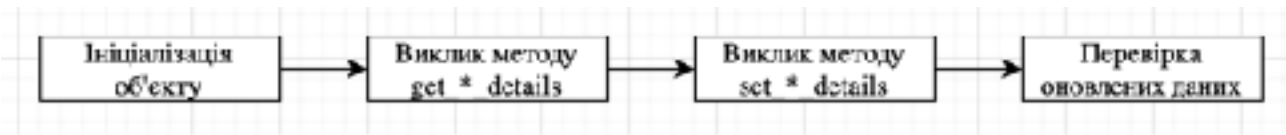


Рисунок 8.1— Схема тестування класів 1

8.2 Тестування окремих класів з застосуванням методів сірого ящика

Чек-аркуш для розуміння загального процесу тестування класів:

Таблиця 8.1— Чек-аркуш тестування методів класу 2

№	Дія	Очікування	Результат	Коментар
1	Ініціалізація класу	Об'єкт успішно створений	Успіх	Об'єкт 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 створений	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:

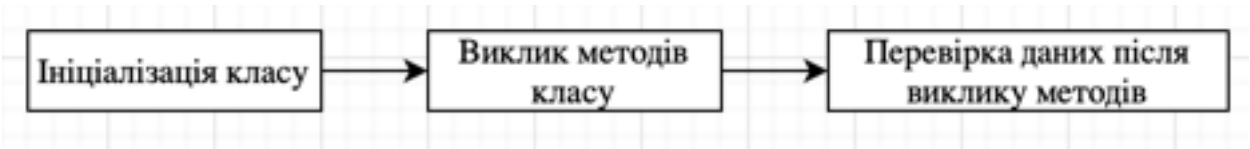


Рисунок 8.1— Схема тестування класів 2

8.3 Тестування програми із застосуванням методу чорного ящика

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

№	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
	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

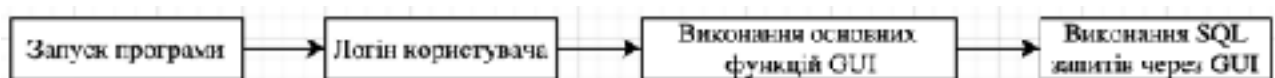


Рисунок 8.1— Схема тестування класів 3

9 СКРІНШОТИ ДОДАТКУ

1) Вікно заходу в акаунт:

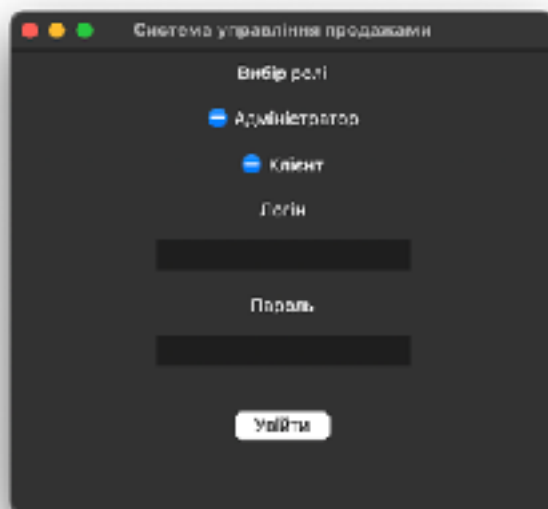


Рисунок9.1—Вікно заходу в акаунт

2) Головне вікно клієнта:

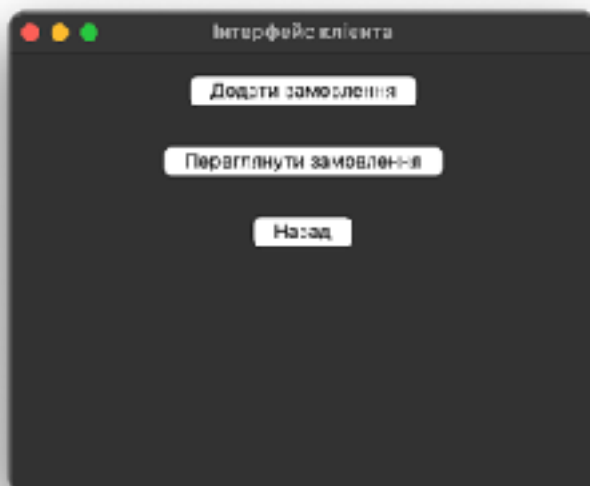


Рисунок9.2—Головне вікно клієнта

2.1) Вікно додавання замовлення:

Додати замовлення

Вибір продукту

Кількість

0

Дата доставки

Зберегти замовлення

Назад

Рисунок9.3—Вікно додавання замовлення

2.2) Вікно перегляду замовлень:

Переглянути замовлення

ID замовлення	Товар в замовленні	Кількість товару	Дата доставки
1	Product A	2	2024-01-01
2	Product B	1	2024-02-01
4	Product B	400	2024-01-01

Видалити замовлення

Назад

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

ID продукту	Назва	Ціна	Рік випуску	Категорія
1	Product A	10.99	2020	Animal
2	Product B	20.99	2021	Animal

Нова

Рисунок9.7—вікно перегляду продуктів

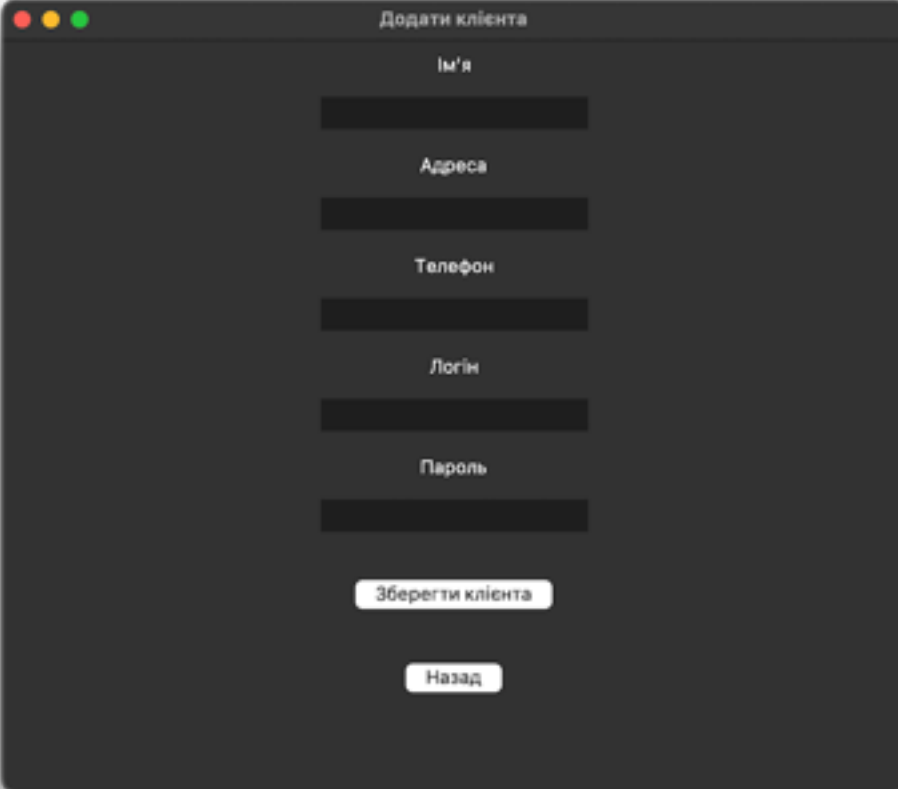
ID поставки	Час поставки	ID продукту	ID постачальника
1	2024-03-01	1	1
2	2024-04-01	2	2

Видалити поставку

Нова

Рисунок9.8— Вікно перегляду поставок

3.2) Вікно для додавання поставки, клієнта:



Додати клієнта

Ім'я

Адреса

Телефон

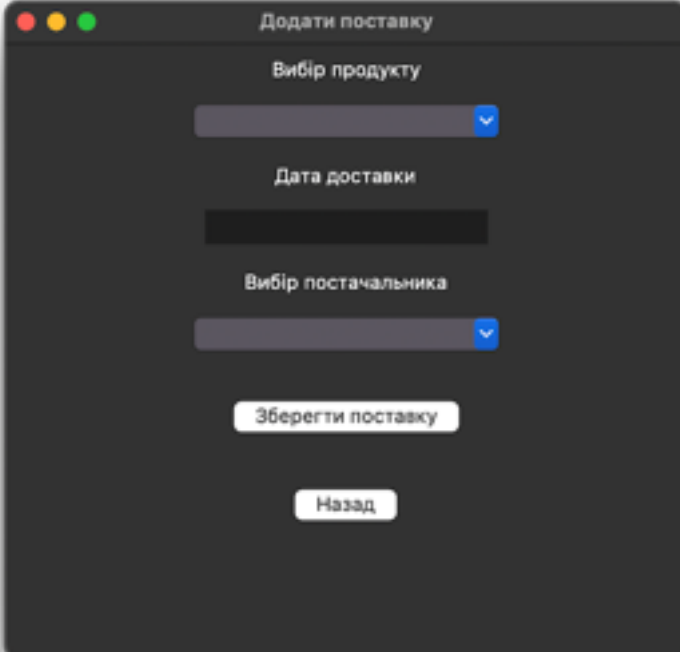
Логін

Пароль

Зберегти клієнта

Назад

Рисунок9.9—Вікно додавання клієнта



Додати поставку

Вибір продукту

Дата доставки

Вибір постачальника

Зберегти поставку

Назад

Рисунок9.10—Вікно додавання поставки