

Національний технічний університет України  
“Київський політехнічний інститут імені Ігоря Сікорського”  
Кафедра інженерії програмного забезпечення в енергетиці

# ***КУРСОВА РОБОТА***

з дисципліни: «Бази даних»

тема: "Інформаційна система: Магазин гаджетів"

Керівник – Дацюк О.А.	Виконав Оніщук М.І.
Допущено до захисту	Студент 2-го курсу
_____ 2024 р.	Групи ТВ-22
Захищено з оцінкою	залікова книжка № ТВ-22
_____	

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

## КУРСОВА РОБОТА

з дисципліни «Бази даних»

(назва дисципліни)

на тему: «Інформаційна система: Магазин гаджетів»

Студента групи ТВ-22

спеціальність 121 «Інженерія програмного забезпечення»

освітня програма Інженерія програмного забезпечення  
інтелектуальних кібер-фізичних систем в енергетиці

Оніщук М.І.

(прізвище та ініціали)

Керівник старший викладач Дацюк О.А.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка \_\_\_\_\_

Кількість балів: \_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_

Члени комісії

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2024 рік

**Національний технічний університет України**  
**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет Навчально-науковий інститут атомної та теплової енергетики  
( повна назва)

Кафедра інженерії програмного забезпечення в енергетиці  
(повна назва)

рівень вищої освіти перший (бакалаврський)

спеціальність 121 «Інженерія програмного забезпечення»  
(шифр і назва)

освітня програма Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці  
(назва)

**З А В Д А Н Н Я**  
**НА КУРСОВУ РОБОТУ СТУДЕНТУ**  
Оніщуку Максимові Івановичу  
(прізвище, ім'я, по батькові)

- Тема роботи «Інформаційна система: магазин гаджетів»  
керівник курсової роботи – Дацюк Оксана Антонівна, старший викладач.  
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
- Строк подання студентом роботи – 2 січня 2024 р.
- Вихідні дані до проекту (роботи): мова SQL.
- Зміст пояснювальної записки курсової роботи (перелік питань, які потрібно розробити) – Розробити інформаційну систему для керування магазином гаджетів
- Перелік графічного матеріалу –
- Дата видачі завдання – 24 вересня 2023 р.

**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів виконання курсової роботи	Строк виконання етапів роботи	Примітка
1	Затвердження теми роботи	17.09.2023	
2	Аналіз інформаційної системи	15.10.2023	
3	Створення бази даних та розробка системи	25.11.2023	
4	Програмна реалізація задачі	28.12.2023	
5	Написання розрахунково-пояснювальної записки	02.01.2024	

**Студент**

\_\_\_\_\_  
( підпис )

**Оніщук М.І.**  
(прізвище та ініціали)

**Керівник курсової роботи**

\_\_\_\_\_  
( підпис )

**Дацюк О.А.**  
(прізвище та ініціали)

## **АНОТАЦІЯ**

У даній курсовій роботі було розроблено систему для автоматизації магазину гаджетів. Основним функціоналом якої є додавання, редагування, видалення, пошук інформації про покупців, продавців, виробників, товарів та замовлення. Дана система дозволяє вести облік товарів та замовлень, контролювати продавців, які працюють на магазин, переглядати історію замовлень та покупців даного магазину.

Для створення даного програмного продукту було використано мови SQL та Python.

## **ANNOTATION**

In this term paper, we have developed a system for automating a gadget store. The main functionality of which is to add, edit, delete, search for information about buyers, sellers, manufacturers, products and orders. This system allows you to keep records of goods and orders, control sellers who work for the store, view the history of orders and customers of this store.

SQL and Python languages were used to create this software product.

## Зміст

ВСТУП.....	5
1. ОПИС ІНФОРМАЦІЙНОЇ СИСТЕМИ .....	6
2. МОДЕЛЬ БАЗИ ДАНИХ .....	8
2.1 Концептуальна модель бази даних .....	8
2.2 ER-діаграма бази даних.....	10
2.3 Використані програмні інструменти .....	10
3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ТА ІНТЕРФЕЙСУ КОРИСТУВАЧА	12
3.1 Структура проекту та код.....	12
3.2 Індивідуальний внесок .....	14
4. ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА.....	19
ВИСНОВОК .....	24
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	25
ДОДАТОК .....	26

## ВСТУП

На сьогоднішній день інформаційні технології є необхідною складовою як окремої людини так і всього людства. Постійно створюються нові та вдосконалюються старі. Базы даних є невід'ємною її частиною, яка забезпечує зберігання великої кількості даних у вигляді різноманітних таблиць. Основною метою бази даних є працювати з великою кількістю інформації шляхом зберігання, видалення та управління даними. Сьогодні у всесвітній павутині Інтернету існує багато динамічних веб-сайтів, які обробляються через бази даних. Наприклад, модель, яка перевіряє наявність номерів у готелі. Це приклад динамічного веб-сайту, який використовує базу даних. Сучасні бази даних управляються системою управління базами даних (СУБД). Наразі існує багато систем для управління базами даних, найвідомішими з яких є MySQL, PostgreSQL, Server, Oracle та інші. SQL або мова структурованих запитів використовується для роботи з даними, що зберігаються в базі даних. SQL залежить від реляційної алгебри та кортежного реляційного числення.[1]

Об'єктом дослідження є магазин гаджетів, який займається продажем ноутбуків, комп'ютерів та подібних товарів.

Метою курсової роботи є проектування та розробка інформаційно-пошукової системи для нашого підприємства з метою підвищення продуктивності його роботи.

# 1. ОПИС ІНФОРМАЦІЙНОЇ СИСТЕМИ

Наша система описує роботу з магазином гаджетів від імені адміністратора. Вона розділяється на 5 підпрограм яка дозволяє виконувати наступні дії над інформацією з бази даних:

- Перегляд – можливість виводу інформації у вигляді таблиць
- Пошук – можливість фільтрування інформації за певними ознаками з подальшим виводом у таблиці
- Створення – можливість додавання нового об'єкта користувачем з заданням усіх необхідних значень
- Редагування – корегування та оновлення інформації про наявні об'єкти
- Видалення – можливість вилучення повної інформації про певний об'єкт у базі даних
- Сортування – можливість форматування порядку виводу інформації за певною ознакою
- Формування звіту – можливість документування усієї таблиці, або певного уривку з бази даних у файл формату Excel.

Також користувач має доступ до інформації про наступні пункти які містяться у кожному розділі про себе:

- Товари – даний пункт описує певний продукт та міститься в собі наступні поля:
  - Код товару
  - Код виробника
  - Ціна
  - Дата виробництва
  - Тип товару
  - Опис товару
  - Кількість на складі
- Виробники – цей розділ містить в собі такі колонки:

- Код виробника
- Назва
- Адреса
- Номер телефону
- Електронна пошта
- Покупці – ця таблиця є досить короткою та містить в собі лише 4 значення:
  - Код покупця
  - Ім'я
  - Прізвище
  - По-батькові
- Продавці – даний пункт має такі поля:
  - Код продавця
  - Ім'я
  - Прізвище
  - Зарплата
  - Номер телефону
- Замовлення – дана таблиця описує придбання певним покупцем якогось товару та містить наступні колонки:
  - Номер замовлення
  - Код покупця
  - Код товару
  - Код продавця
  - Дата замовлення
  - Чи оплачений товар?



## 2. МОДЕЛЬ БАЗИ ДАНИХ

### 2.1 Концептуальна модель бази даних

Концептуальна модель бази даних є абстрактним представленням даних та їх взаємозв'язків у певній предметній області. Вона описує сутності (entities) та їх взаємозв'язки, але не звертає увагу на технічні деталі реалізації у конкретній базі даних.

Основні аспекти концептуальної моделі бази даних:

- **Сутності (Entities):** Сутність представляє об'єкт або поняття в предметній області, для якої створюється база даних. Наприклад, якщо ми створюємо базу даних для управління бібліотекою, то «Книга» та «Читач» можуть бути сутностями.
- **Відносини (Relationships):** Відносини вказують на взаємозв'язки між сутностями. Наприклад, в бібліотеці книга може бути взята у читача, тобто існує відношення між «Книга» та «Читач».
- **Атрибути (Attributes):** Атрибути — це властивості чи характеристики сутностей. Наприклад, атрибутами книги можуть бути «Назва», «Автор», «Рік видання» і т. д.
- **Ключі (Keys):** Ключі ідентифікують унікальний запис в таблиці. Кожна сутність має один або декілька ключів. Наприклад, у таблиці «Книги» ключем може бути ISBN.
- **Кардинальність (Cardinality):** Кардинальність визначає кількість сутностей, що беруть участь у відношенні. Вона вказує, чи є відношення один до одного, один до багатьох або багато до багатьох.

Дана концептуальна модель бази даних складається з п'яти таблиць які пов'язані між собою:

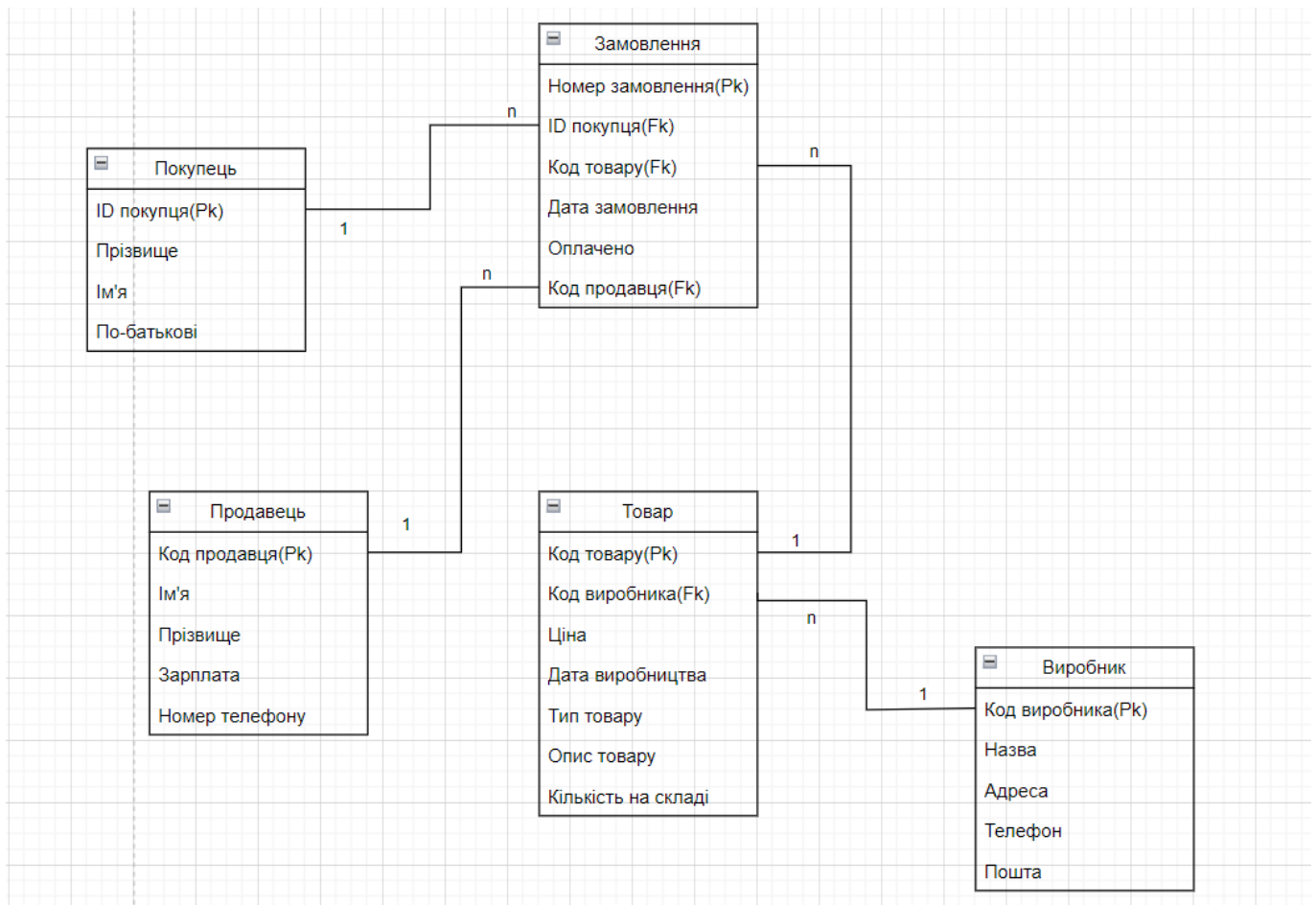


Рисунок 2.1 – Концептуальна модель бази даних

## 2.2 ER-діаграма бази даних

Модель ER розшифровується як модель Entity-Relationship. Це модель даних високого рівня. Ця модель використовується для визначення елементів даних і зв'язку для визначеної системи.

Він розробляє концептуальний дизайн бази даних. Він також розробляє дуже простий і зручний для проектування перегляд даних.

У моделюванні ER структура бази даних зображується як діаграма, яка називається діаграмою сутності-зв'язку.[2]

В даній діаграмі описуються таблиці бази даних і зв'язки між ними:

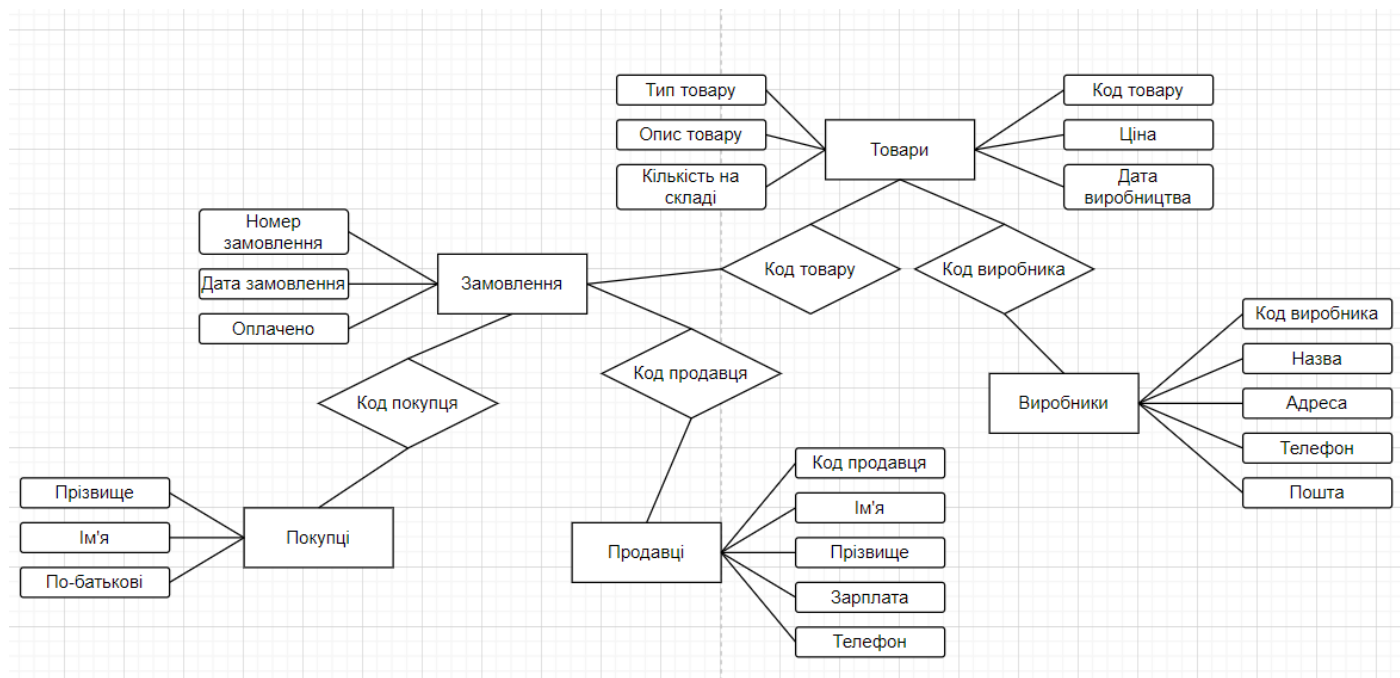


Рисунок 2.2 – ER-діаграма бази даних

## 2.3 Використані програмні інструменти

При створенні нашої інформаційної системи було використано наступні інструменти:

- СКБД MySQL та середовище розробки DataGrip – для створення та налагодження самої бази даних

- Мова програмування Python – використано для програмування користувацького інтерфейсу
- Середовище розробки PyCharm – тут розроблявся програмний код

## 3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

### 3.1 Структура проекту та код

Загалом наш проект складається з одного файлу(модуля) в який було імпортовано наступні модулі:

1. Tkinter та customtkinter – бібліотеки для створення просто користувацького інтерфейсу
2. Random – бібліотека для генерування випадкових значень
3. mysql.connector – модуль для роботи з базами даних в мові python
4. pandas – бібліотека для роботи з масивами та базами даних та файлами
5. datetime – бібліотека для роботи з датою та часом

Дані бібліотека надалі будуть використовуватись у нашому програмному коді.

Крім того у нашому проекті міститься наступні сім класів:

- AdminWindow – даний клас описує невелике стартове меню користувача, в якому можна вибрати з чим саме хоче працювати
- Window – цей клас є батьківським, який описує загальний вигляд усіх дочірніх вікон, які створюються під конкретну якусь таблицю нашої бази даних
- ProductsWin – дочірній клас від Window, у якому розписані функції виводу, пошуку, створення, видалення, редагування, сортування та формування звіту у форматі Excel для таблиці продуктів
- OrdersWin – також дочірній клас, але створений для таблиці замовлень.
- VendorsWin – такий самий але зроблений під виробників
- CustomersWin – такий самий, але для покупців
- SellersWin – такий самий, але відповідно створений під продавців

- **Searcher** – клас у якому прописані алгоритми для пошуку у базі даних певних текстових, числових або часових значень враховуючи обробку різноманітних помилок.

Ось тут на прикладі вікна продуктів було показано функцію виведення списку продуктів, яка за допомогою **pandas** отримує формує масив даних **DataFrame** та передається у батьківський метод який виводить у таблицю користувача:

```
def all(self, **kwargs):
    self.__result = pd.read_sql( sql: """
        SELECT
        p.ProductID,
        v.name AS VendorName,
        p.price,
        p.ProdDate,
        p.type,
        p.description,
        p.quantity
    FROM
        products p
    LEFT JOIN
        vendors v ON p.vendorID = v.VendorID;""", con=admin)
    super().all(self.__result)
```

Рисунок 3.1 – Вивід списку продуктів

Ось наведено функцію для сформування звіту у форматі **Excel**, яка отримує від користувача назву файлу та створює його:

```
def to_doc(self, my_input):
    name = self.__name_doc.get()
    my_input.to_excel(name+".xlsx")
    messagebox.showinfo( title: "showinfo", message: "Звіт записано")
```

3.2 – Формування звіту

## 3.2 Індивідуальний внесок

Моїм індивідуальним завданням було створення функції пошуку, редагування, створення та видалення зі створенням для цього відповідним інтерфейсом для користувача.

Ось наведено функцію для пошуку певних значень які мають задану числову характеристику у вказаному діапазоні. Ця функція приймає на вхід 2 значення для задання діапазону, DataFrame, у якому повинні відсіюватись рядки, назва поля та текст помилка яка буде виводитись у окремому вікні:

```
7 usages
@staticmethod
def search_diapazone(column: str, from_val: int | float | str, to_val: int | float | str, result: pd.DataFrame, error_mes: str):
    if not (from_val == "" or to_val == ""):
        try:
            from_val = int(from_val)
            to_val = int(to_val)
            if from_val > to_val:
                messagebox.showwarning(title="showwarning", error_mes)
            else:
                result = result[
                    (result[column] >= from_val) & (result[column] <= to_val)]
        except ValueError:
            messagebox.showwarning(title="showwarning", error_mes)
    return result
```

Рисунок 3.7 – Пошук за заданим діапазоном

Далі показано функцію пошуку значень за вказаним діапазоном дат, яка працює схожим чином як із числами:

```
3 usages
@staticmethod
def search_date_diapazone(column: str, from_val: str, to_val: str, result: pd.DataFrame, error_mes: str):
    if not (from_val == "" or to_val == ""):
        try:
            from_val = datetime.strptime(from_val, _format: "%Y-%m-%d")
            to_val = datetime.strptime(to_val, _format: "%Y-%m-%d")
            if from_val > to_val:
                messagebox.showwarning(title="showwarning", error_mes)
            else:
                result = result[(result[column].apply(lambda x: datetime.strptime(str(x), _format: "%Y-%m-%d")) >= from_val)
                                & (result[column].apply(lambda x: datetime.strptime(str(x), _format: "%Y-%m-%d")) <= to_val)]
        except ValueError:
            messagebox.showwarning(title="showwarning", error_mes)
    return result
```

Рисунок 3.8 – Пошук за діапазоном дат

І наступні функцію були створені для пошуку значень за вказаними текстовими змінними з клавіатури користувача та списку відповідно. Різниця між ними є досить малою. Тільки у другій значення повинно строго дорівнювати шуканому, тоді як у першому задане значення повинно міститись у шуканих значеннях:

```
14 usages
@staticmethod
def search_text_having_none(column: str, text: str, result: pd.DataFrame):
    if not text == "":
        if text == "-":
            result = result[(result[column].apply(lambda x: str(x) == "None"))]
        else:
            result = result[(result[column].apply(lambda x: text in str(x)))]
    return result

2 usages
@staticmethod
def search_text_from_list(column: str, text: str, result: pd.DataFrame):
    if not text == "":
        if text == "-":
            result = result[(result[column].apply(lambda x: str(x) == "None"))]
        else:
            result = result[(result[column].apply(lambda x: str(x) == text))]
    return result
```

Рисунок 3.9 – Пошук за текстовими значеннями

В наступному скріншоті показано на прикладі вікна покупців функцію пошуку значень таблиці за заданими критеріями, яка отримує спочатку увесь список, а потім відсіює значення за допомогою вищезгаданого класу Searcher та вказаних функцій:

```
def search_g():
    search = Searcher()

    self.__result = pd.read_sql( sql: """
        SELECT *
        FROM customers;""", con=admin)

    from_id = from_kod.get()
    to_id = to_kod.get()
    self.__result = search.search_diapazone( column: "CustomerID", from_id, to_id, self.__result,
        error_mes: "Направильно вказано діапазон кодів(умову не зараховано)")

    name = name_in.get()
    self.__result = search.search_text_having_none( column: "name", name, self.__result)

    surname = surname_in.get()
    self.__result = search.search_text_having_none( column: "surname", surname, self.__result)

    lastname = lastname_in.get()
    self.__result = search.search_text_having_none( column: "lastName", lastname, self.__result)

    self.clear_table()
    for index, row in self.__result.iterrows():
        values = tuple(row.values)
        self._Window__table.insert("", tk.END, values=values)
```

Рисунок 3.10 – Пошук покупців



Далі наведено рисунки функції створення перевірки наявності необхідних значень та створення списку необхідних значень з подальшим створенням нового продавця у нашій таблиці, якщо хоча б одне значення буде не вказане, або вказане, але неправильно програма викличе вікно з помилкою :

```
ven_id = random.randint(a: 30, b: 100000000)
name = name_in.get()
adress = adress_in.get()
phone = phone_in.get()
email = email_in.get()

list_args = [ven_id]
```

```
if not name == "":
    list_args.append(str(name))
else:
    messagebox.showwarning(title: "Warning", message: "Не вказано назви")
    return

if not adress == "":
    list_args.append(str(adress))
else:
    messagebox.showwarning(title: "Warning", message: "Не вказано адреси")
    return

if not phone == "":
    try:
        int(phone)
        list_args.append(str(phone))
    except ValueError:
        messagebox.showwarning(title: "Warning", message: "Неправильно вказано номер телефону")
        return
else:
    messagebox.showwarning(title: "Warning", message: "Не вказано номеру телефону")
    return

if not email == "":
    list_args.append(str(email))
else:
    messagebox.showwarning(title: "Warning", message: "Не вказано пошти")
    return
```

```
list_args = tuple(list_args)

self._Window__cursor.execute("INSERT INTO vendors VALUES " + str(list_args) + ";")
admin.commit()
```

Рисунки 3.11, 3.12, 3.13 – створення продавця

Далі на скріншоті наведено приклад функції редагування даних виробника, яка перевіряє які поля користувач змінив та за допомогою функції execute редагує дані в базі даних:

```
def edit(self):
    p_id = self._Window__edit_en.get()
    all_id = pd.read_sql(sql="SELECT VendorID FROM vendors", con=admin)
    all_id = all_id["VendorID"].tolist()
    try:
        p_id = int(p_id)
    except ValueError:
        messagebox.showwarning(title="warning", message="Неправильно введено код виробника")
        return

    if p_id not in all_id:
        messagebox.showwarning(title="Warning", message="Немає такого виробника")
        return
```

```
def edit_q():
    name = name_in.get()
    address = address_in.get()
    phone = phone_in.get()
    email = email_in.get()

    if not name == "":
        self._Window__cursor.execute("UPDATE vendors SET name = " + str(name) + " WHERE VendorID = " + str(p_id) + ";" )
        admin.commit()

    if not address == "":
        self._Window__cursor.execute(
            "UPDATE vendors SET address = " + str(address) + " WHERE VendorID = " + str(p_id) + ";" )
        admin.commit()

    try:
        phone = int(phone)
        if not phone == "":
            self._Window__cursor.execute("UPDATE vendors SET phoneNumber = " + str(phone) + " WHERE VendorID = " + str(p_id) + ";" )
            admin.commit()
    except ValueError:
        messagebox.showwarning(title="Warning", message="Неправильно введено номер")

    if not email == "":
        self._Window__cursor.execute("UPDATE vendors SET email = " + str(email) + " WHERE VendorID = " + str(p_id) + ";" )
        admin.commit()
```

Рисунки 3.14, 3.15 – редагування даних виробника

Наступні два скріншоти показує функцію видалення замовлення, перший який є шаблонним батьківським, а другий з класу OrdersWin:

```
def delete_selected_item(self, pageid, page):
    res = self.__delete_en.get()
    try:
        self.__cursor.execute("DELETE FROM "+page+" WHERE "+pageid+" = " + res + ";")
        admin.commit()
    except mysql.connector.errors.ProgrammingError:
        return
```

```
def delete_selected_item(self, **kwargs):  
    super().delete_selected_item( pageid: "OrderID", page: "orders")
```

Рисунки 3.16, 3.17 – видалення замовлення

## 4. ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

При запуску нашої системи користувач побачить меню для вибору таблиці з якою буде працювати, виглядає вона наступним чином:

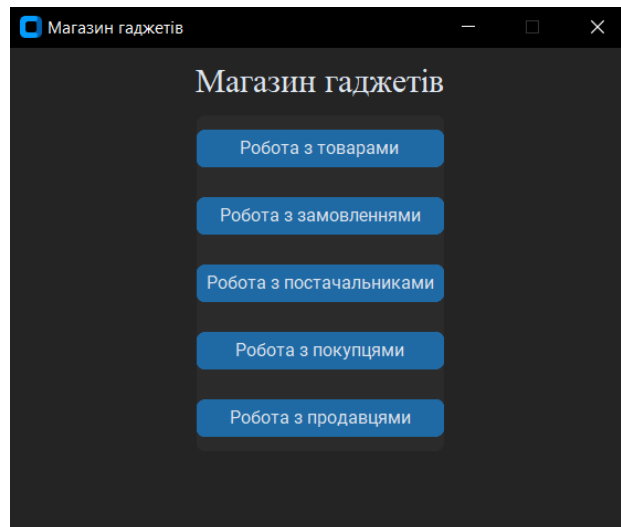


Рисунок 4.1 – Головне меню

Далі ми будемо переглядати вивід на прикладі товарів, так як вікна для різних таблиць мають лише невеликі відмінності. Отже після натискання кнопки «Робота з товарами» користувач бачить наступне вікно:

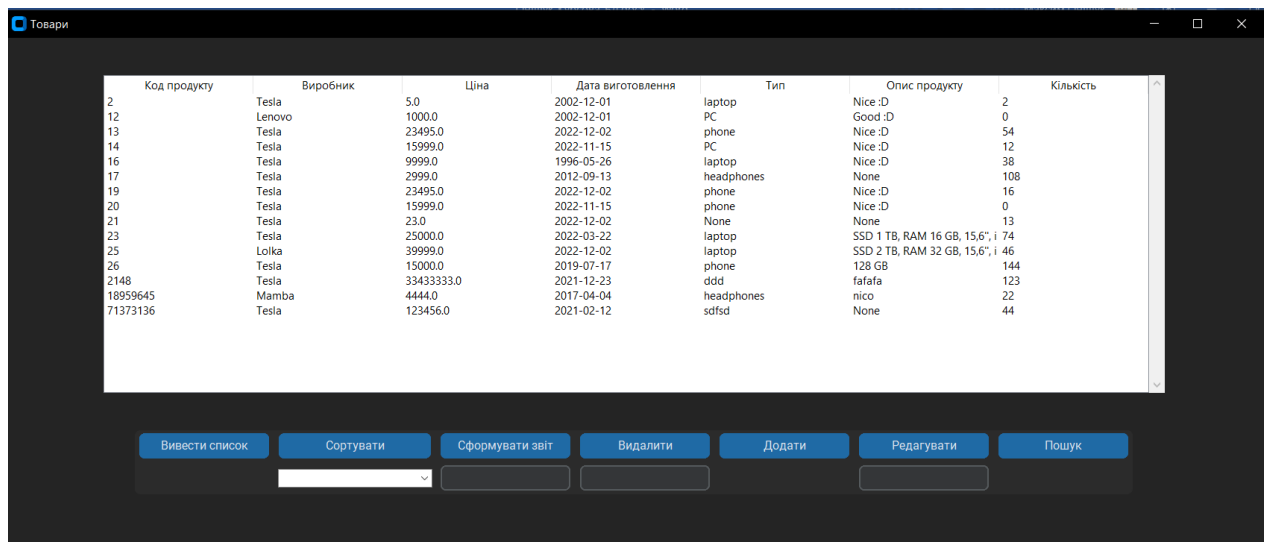


Рисунок 4.2 – Вікно для роботи з товарами

Перш за все користувач може відсортувати дану таблицю на свій лад, вибравши параметри за яким він хоче провести сортування. На рисунку 3.5 зображено відсортований вивід товарів за датою виготовлення:

Код продукту	Виробник	Ціна	Дата виготовлення	Тип	Опис продукту	Кількість
16	Tesla	9999.0	1996-05-26	laptop	Nice :D	38
2	Tesla	5.0	2002-12-01	laptop	Nice :D	2
12	Lenovo	1000.0	2002-12-01	PC	Good :D	0
17	Tesla	2999.0	2012-09-13	headphones	None	108
18959645	Mamba	4444.0	2017-04-04	headphones	nico	22
26	Tesla	15000.0	2019-07-17	phone	128 GB	144
71373136	Tesla	123456.0	2021-02-12	sd fsd	None	44
2148	Tesla	33433333.0	2021-12-23	ddd	fafafa	123
23	Tesla	25000.0	2022-03-22	laptop	SSD 1 TB, RAM 16 GB, 15,6", i 74	74
14	Tesla	15999.0	2022-11-15	PC	Nice :D	12
20	Tesla	15999.0	2022-11-15	phone	Nice :D	0
13	Tesla	23495.0	2022-12-02	phone	Nice :D	54
19	Tesla	23495.0	2022-12-02	phone	Nice :D	16
21	Tesla	23.0	2022-12-02	None	None	13
25	Lolka	39999.0	2022-12-02	laptop	SSD 2 TB, RAM 32 GB, 15,6", i 46	46

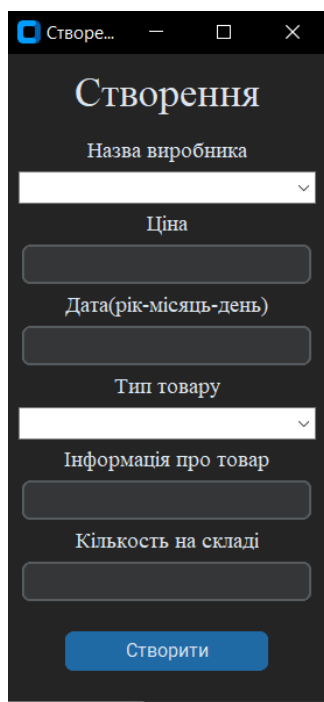
Рисунок 4.3 – Відсортований вивід

Далі користувач може побачити кнопку «Сформувати звіт», натиснувши відразу яку нічого не станеться, тому що користувач скоріш за все не ввів назву файлу в який буде зберігатись звіт, який зображено на рисунку 4.4. У цей файл зберігається таблиця яка зображена у вікні програми, зберігаючи навіть сортований порядок за певним полем:

	A	B	C	D	E	F	G	H	I
		ProductID	VendorName	price	ProdDate	type	description	quantity	
0		16	Tesla	9999	1996-05-26	laptop	Nice :D	38	
1		2	Tesla	5	2002-12-01	laptop	Nice :D	2	
2		12	Lenovo	1000	2002-12-01	PC	Good :D	0	
3		17	Tesla	2999	2012-09-13	headphones		108	
4		18959645	Mamba	4444	2017-04-04	headphon	nico	22	
5		26	Tesla	15000	2019-07-17	phone	128 GB	144	
6		71373136	Tesla	123456	2021-02-12	sd fsd	None	44	
7		2148	Tesla	33433333	2021-12-23	ddd	fafafa	123	
8		23	Tesla	25000	2022-03-22	laptop	SSD 1 TB, RAM 16 GB, 15,6", i 74	74	
9		14	Tesla	15999	2022-11-15	PC	Nice :D	12	
10		20	Tesla	15999	2022-11-15	phone	Nice :D	0	
11		13	Tesla	23495	2022-12-02	phone	Nice :D	54	
12		19	Tesla	23495	2022-12-02	phone	Nice :D	16	
13		21	Tesla	23	2022-12-02			13	
14		25	Lolka	39999	2022-12-02	laptop	SSD 2 TB, RAM 32 GB, 15,6", i 46	46	

Рисунок 4.4 – Сформований звіт

Якщо користувач захоче створити новий продукт, то він може натиснути кнопку «Додати» та з'явиться наступне вікно, в якому користувач повинен вказати значення у кожному полі:



The screenshot shows a mobile application window titled "Створення" (Creation) with a dark theme. The window contains the following elements from top to bottom:

- A title bar with a blue square icon, the text "Створе...", and standard window controls (minimize, maximize, close).
- A title "Створення" in a large, light-colored font.
- A label "Назва виробника" (Manufacturer name) above a white input field with a dropdown arrow on the right.
- A label "Ціна" (Price) above a dark gray input field.
- A label "Дата(рік-місяць-день)" (Date (year-month-day)) above a dark gray input field.
- A label "Тип товару" (Product type) above a white input field with a dropdown arrow on the right.
- A label "Інформація про товар" (Product information) above a dark gray input field.
- A label "Кількість на складі" (Quantity in stock) above a dark gray input field.
- A blue button with the text "Створити" (Create) at the bottom.

Рисунок 4.5 – Створення товару

Натиснувши кнопку «Редагувати» користувач отримає наступне вікно, яке є дуже схожим на попереднє, але на відміну від створення користувач може вказувати лише деяка поля які він хоче змінити:

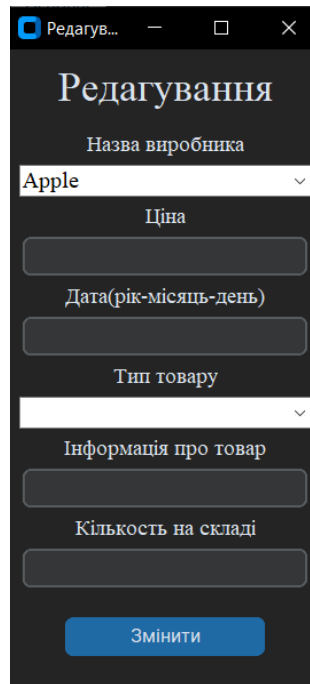


Рисунок 4.6– Редагування інформації про товар

Якщо користувач захоче знайти всі товари які мають якусь певну ознаку він може натиснути кнопку «Пошук» та отримає наступне вікно, після чого введе дані для пошуку:

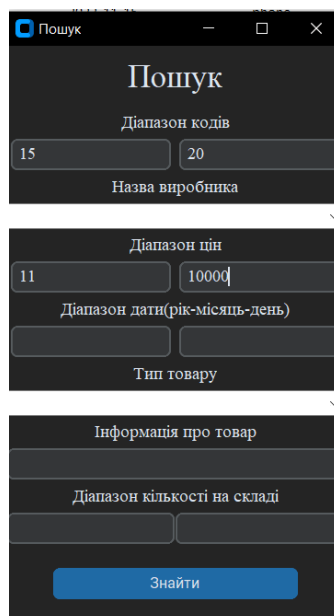
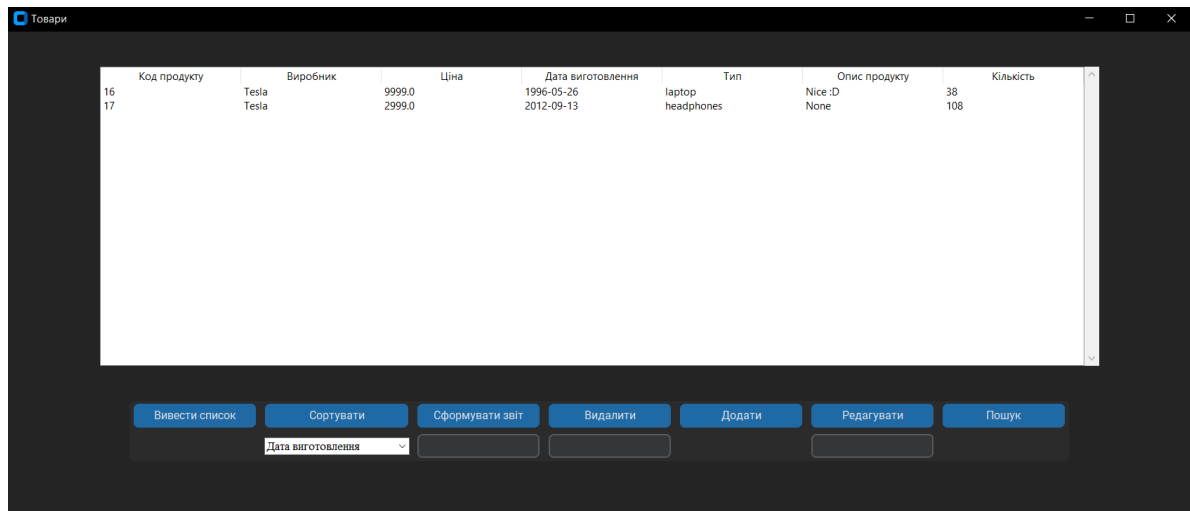


Рисунок 4.7– Пошук товарів

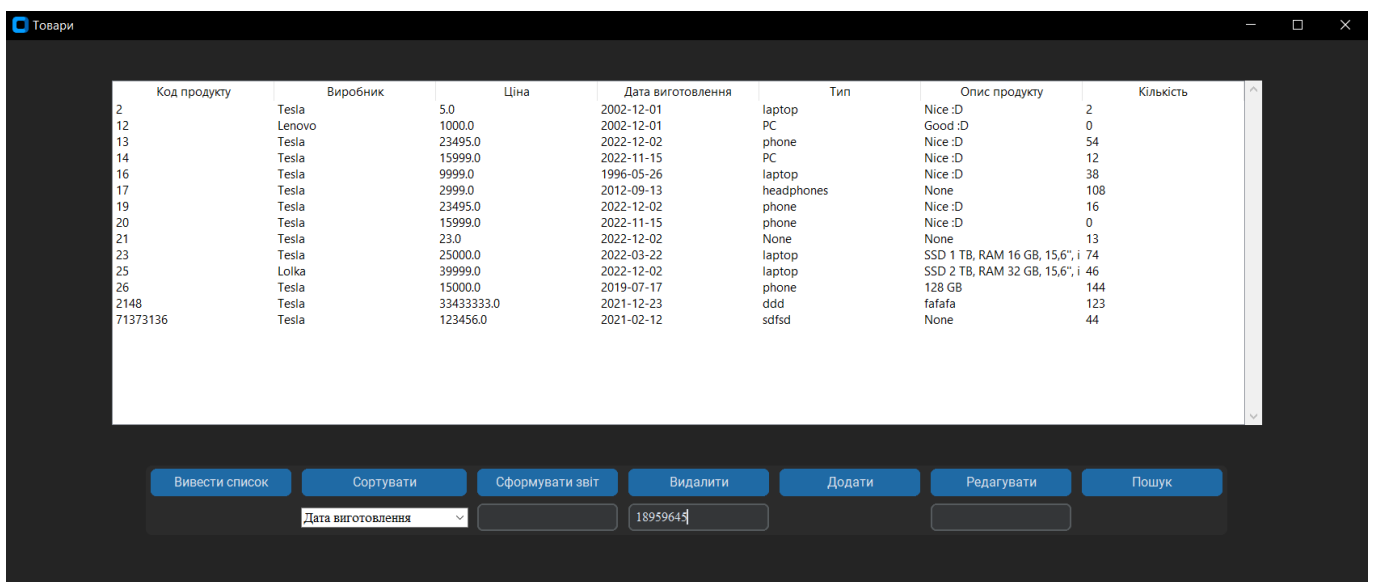
Натиснувши кнопку «Знайти» таблиця виведе результати, які відповідають заданим критеріям користувача:



Код продукту	Виробник	Ціна	Дата виготовлення	Тип	Опис продукту	Кількість
16	Tesla	9999.0	1996-05-26	laptop	Nice :D	38
17	Tesla	2999.0	2012-09-13	headphones	None	108

Рисунок 4.8 – Результат пошуку

Користувач може видалити певний рядок таблиці вказавши код та натиснувши кнопку «Видалити», але якщо користувач захоче видалити товар який задіяний у іншій таблиці, то рядок не видалиться, щоб зробити це користувач повинен змінити або видалити рядки у інших таблицях де задіяний цей об'єкт. Приклад успішного видалення:



Код продукту	Виробник	Ціна	Дата виготовлення	Тип	Опис продукту	Кількість
2	Tesla	5.0	2002-12-01	laptop	Nice :D	2
12	Lenovo	1000.0	2002-12-01	PC	Good :D	0
13	Tesla	23495.0	2022-12-02	phone	Nice :D	54
14	Tesla	15999.0	2022-11-15	PC	Nice :D	12
16	Tesla	9999.0	1996-05-26	laptop	Nice :D	38
17	Tesla	2999.0	2012-09-13	headphones	None	108
19	Tesla	23495.0	2022-12-02	phone	Nice :D	16
20	Tesla	15999.0	2022-11-15	phone	Nice :D	0
21	Tesla	23.0	2022-12-02	None	None	13
23	Tesla	25000.0	2022-03-22	laptop	SSD 1 TB, RAM 16 GB, 15,6", i 74	74
25	Lolka	39999.0	2022-12-02	laptop	SSD 2 TB, RAM 32 GB, 15,6", i 46	46
26	Tesla	15000.0	2019-07-17	phone	128 GB	144
2148	Tesla	33433333.0	2021-12-23	ddd	fafafa	123
71373136	Tesla	123456.0	2021-02-12	sdfsd	None	44

Рисунок 4.9 – видалення товару

Після завершення роботи користувач може натиснути на хрестик у правому верхньому кутку головного меню і тоді всі вікна закриються відразу.



## ВИСНОВОК

Під час виконання курсової роботи була спроектована та реалізована база даних та інформаційна система. Для взаємодії з базою даних було використано модуль `mysql.connector` в мові Python. Для нашої інформаційної системи під назвою «Магазин гаджетів» було створено відповідно ER-діаграму та концептуальну модель бази даних.

Створення системи для управління магазином гаджетів та використання баз даних в цьому контексті дозволяє організувати та взаємодіяти з великим обсягом інформації, пов'язаної із продажами, клієнтами, товарами та іншими аспектами бізнесу. В цілому, використання баз даних в системах управління магазином гаджетів є ключовим аспектом для забезпечення ефективного та організованого функціонування бізнесу, що дозволяє зосередитися на стратегічному розвитку та задоволенні потреб клієнтів.

Розроблена система дозволяє:

- Перегляд – виведення інформації з бази даних у програмі
- Пошук – виведення інформації лише про ті об'єкти які підпадають під задані користувачем критерії
- Створення – створення та додавання нового об'єкта та інформації про нього у базу даних
- Редагування – зміна певної інформації про наявний об'єкт
- Видалення – вилучення наявного об'єкта з бази даних
- Сортування – зміна порядку виводу об'єктів певної таблиці
- Формування звіту – експортування даних з таблиці у файл формату Excel

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ЩО ТАКЕ БАЗИ ДАНИХ, ЇХ ПРИЗНАЧЕННЯ ТА ВИДИ? [Електронний ресурс] – Режим доступу: <https://futurenow.com.ua/shho-take-bazy-danyh-yih-pryznachennya-ta-vydy/>
2. JavaTpoint [Електронний ресурс] – Режим доступу: <https://www.javatpoint.com/dbms-er-model-concept>

# ДОДАТОК

Лістинг програмного модуля  
«Назва модуля»

**НТУУ "КПІ" НН ІАТЕ ІПЗЕ ТВ-22**

**Листів 7**

**Київ – 2024**

**Функція яка отримує на вхід масив даних для виводу інформації про певну таблицю бази даних**

```
def all(self, dataframe):
    self.clear_table()
    for index, row in dataframe.iterrows():
        values = tuple(row.values)
        self.__table.insert("", tk.END, values=values)
```

**Функція яка за допомогою бібліотеки pandas отримує масив даних з таблиці товарів, виконуючи запит у базу даних та повертає його у батьківську функцію виводу:**

```
def all(self, **kwargs):
    self.__result = pd.read_sql("""
        SELECT
        p.ProductID,
        v.name AS VendorName,
        p.price,
        p.ProdDate,
        p.type,
        p.description,
        p.quantity
    FROM
        products p
    LEFT JOIN
        vendors v ON p.vendorID = v.VendorID;""", con=admin)
    super().all(self.__result)
```

Функція для формування звіту:

```
def to_doc(self, my_input):
    name = self.__name_doc.get()
    my_input.to_excel(name+".xlsx")
    messagebox.showinfo("showinfo", "Звіт записано")
```

**Функція батьківського класу Window, яка отримує назву стовпця з кодом об'єкта та назви таблиці для видалення певного рядка з таблиці:**

```
def delete_selected_item(self, pageid, page):
    res = self.__delete_en.get()
```

```

try:
    self.__cursor.execute("DELETE FROM "+page+" WHERE "+pageid+" = " + res + ";")
    admin.commit()
except mysql.connector.errors.ProgrammingError:
    return

```

### Функція з класу **ProductWin** для пошуку:

```

def search_g():
    # створення об'єкта для пошуку
    search = Searcher()

    # отримання всього списку товарів
    self.__result = pd.read_sql("""
        SELECT
        p.ProductID,
        v.name AS VendorName,
        p.price,
        p.ProdDate,
        p.type,
        p.description,
        p.quantity
    FROM
        products p
    LEFT JOIN
        vendors v ON p.vendorID = v.VendorID;""", con=admin)

    # відсіювання за критеріями
    from_id = from_kod.get()
    to_id = to_kod.get()
    self.__result = search.search_diapazone("ProductID", from_id, to_id, self.__result, "Неправильно вказано
діапазон кодів(умову не зараховано)")

    vendor = vendor_in.get()
    self.__result = search.search_text_from_list("VendorName", vendor, self.__result)

    price_from = from_price.get()
    price_to = to_price.get()
    self.__result = search.search_diapazone("price", price_from, price_to, self.__result, "Неправильно вказано
діапазон цін(умову не зараховано)")

```

```

date_from = from_date.get()
date_to = to_date.get()
self.__result = search.search_date_diapazone("ProdDate", date_from, date_to, self.__result, "Направильно
вказано діапазон дат(умову не зараховано)")

type_p = type_in.get()
self.__result = search.search_text_from_list("type", type_p, self.__result)

desc = desc_in.get()
self.__result = search.search_text_having_none("description", desc, self.__result)

qua_from = from_qua.get()
qua_to = to_qua.get()
self.__result = search.search_diapazone("quantity", qua_from, qua_to, self.__result, "Неправильно
вказано діапазон кількості(умову не зараховано)")

# Вивід результату
self.clear_table()
for index, row in self.__result.iterrows():
    values = tuple(row.values)
    self._Window__table.insert("", tk.END, values=values)

```

### **Функція в класі OrdersWin для створення замовлення:**

```

def add():
    # Отримання значень введених користувачем
    order_id = random.randint(30, 1000000000)
    customer = customer_in.get()
    product = prod_in.get()
    date = date_in.get()
    ispayed = ispayed_in.get()
    seller = seller_in.get()

    # створення списку та обробка кожного значення на правильність, з подальшим додаванням у
список
    list_args = [order_id]

    if not customer == "":
        custome = pd.read_sql("SELECT CustomerID FROM customers WHERE name = " + str(customer) +
";", con=admin)

```

```

        customer = custome["CustomerID"].loc[0]
        list_args.append(str(customer))
    else:
        messagebox.showwarning("Warning", "Не вказано покупця")
        return

    if not product == "":
        produc = pd.read_sql("SELECT ProductID FROM products WHERE description = '" + str(product) +
";", con=admin)
        product = produc["ProductID"].loc[0]
        list_args.append(str(product))
    else:
        messagebox.showwarning("Warning", "Не вказано товару")
        return

    if not ispayed == "":
        if ispayed == "Так":
            list_args.append(1)
        elif ispayed == "Hi":
            list_args.append(0)
    else:
        messagebox.showwarning("Warning", "Не вказано оплату")
        return

    if not date == "":
        try:
            datetime.strptime(date, "%Y-%m-%d")
            list_args.append(str(date))
        except ValueError:
            messagebox.showwarning("Warning", "Неправильно введено дату")
    else:
        messagebox.showwarning("Warning", "Не вказано дати")
        return

    if not seller == "":
        selle = pd.read_sql("SELECT sellerID FROM sellers WHERE name = '" + str(seller) + "'", con=admin)
        seller = selle["sellerID"].loc[0]
        list_args.append(str(seller))

```

else:

```
    messagebox.showwarning("Warning", "Не вказано продавця")
```

```
    return
```

#У разі коли всі значення введено коректно список значень перетворюється у кортеж та за допомогою курсора додається у базу даних

```
list_args = tuple(list_args)
```

```
self._Window__cursor.execute("INSERT INTO orders VALUES " + str(list_args) + ";")
```

```
admin.commit()
```

## Функція редагування у класі CustomersWin:

```
def edit(self):
```

# отримання значення коду від користувача та перевірка чи існує такий покупець і чи провально введений сам код

```
    p_id = self._Window__edit_en.get()
```

```
    all_id = pd.read_sql("SELECT CustomerID FROM customers", con=admin)
```

```
    all_id = all_id["CustomerID"].tolist()
```

```
    try:
```

```
        p_id = int(p_id)
```

```
    except ValueError:
```

```
        messagebox.showwarning("warning", "Неправильно введено код покупця")
```

```
        return
```

```
    if p_id not in all_id:
```

```
        messagebox.showwarning("Warning", "Немає такого покупця")
```

```
        return
```

# Створення віджетів вікна редагування

```
edit_win = ctk.CTk()
```

```
edit_win.title("Редагування")
```

```
top = ctk.CTkLabel(edit_win, text="Редагування", font=("Times New Roman", 28))
```

```
top.grid(stick="ew", pady=10)
```

```
name_label = ctk.CTkLabel(edit_win, text="Ім'я покупця", font=("Times New Roman", 15))
```

```
name_label.grid(stick="ew")
```

```
name_in = ctk.CTkEntry(edit_win, font=("Times new Roman", 15))
```

```
name_in.grid(stick="ew", padx=10)
```



```

surname_label = ctk.CTkLabel(edit_win, text="Прізвище покупця", font=("Times New Roman", 15))
surname_label.grid(stick="ew")
surname_in = ctk.CTkEntry(edit_win, font=("Times new Roman", 15))
surname_in.grid(stick="ew", padx=10)

```

```

lastname_label = ctk.CTkLabel(edit_win, text="По-батькові покупця", font=("Times New Roman", 15))
lastname_label.grid(stick="ew")
lastname_in = ctk.CTkEntry(edit_win, font=("Times new Roman", 15))
lastname_in.grid(stick="ew", padx=10)

```

```

def edit_q():
    # Зчитування значень користувача
    name = name_in.get()
    surname = surname_in.get()
    lastname = lastname_in.get()
    # Перевірка значень та редагування
    if not name == "":
        self._Window__cursor.execute(
            "UPDATE customers SET name = " + str(name) + " WHERE CustomerID = " + str(p_id) + ";"
        )
        admin.commit()

    if not surname == "":
        self._Window__cursor.execute(
            "UPDATE customers SET surname = " + str(surname) + " WHERE CustomerID = " + str(p_id) + ";"
        )
        admin.commit()

    if not lastname == "":
        self._Window__cursor.execute(
            "UPDATE customers SET lastName = " + str(lastname) + " WHERE CustomerID = " + str(p_id) +
            ";"
        )
        admin.commit()

```

### **Функція сортування в класі SellersWin:**

```

def sort(self):
    # Зчитування типу сортування
    type_s = self._Window__type_of_sort.get()
    sort_type = None
    match type_s:

```

```

case "Код продавця":
    sort_type = "sellerID"
case "Ім'я":
    sort_type = "name"
case "Прізвище":
    sort_type = "surname"
case "Зарплата":
    sort_type = "salary"
# Отримати масив відсортованих даних з бази
if sort_type:
    self.__result = pd.read_sql("""
        SELECT *
        FROM sellers
        ORDER BY "" + sort_type + ""," con=admin)

# Виведення результату
self.clear_table()
for index, row in self.__result.iterrows():
    values = tuple(row.values)
    self._Window__table.insert("", tk.END, values=values)

```