МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп’ютерних технологій

ЗВІТ

про виконання ЛАБОРАТОРНОЇ РОБОТИ № 10

з дисципліни “Бази даних”

На тему: “Використання ORM та графічного інтерфейсу для роботи з базою даних транспортної фірми”

Виконав

студент 2 курсу

групи ФЕП-23

Чепара Станіслав Богданович

Перевірив доцент:

Анохін Володимир Євгенович

Львів 2025

КОД прикріпив на github: https://github.com/Stanislavwx/Databases/tree/main/lab9

КОД прикріпив на github: https://github.com/Stanislavwx/Databases/tree/main/lab9

КОД прикріпив на github:
https://github.com/Stanislavwx/Databases/tree/main/lab10

Мета роботи

Ознайомитися з принципами роботи ORM (Object-Relational Mapping) на прикладі бібліотеки SQLAlchemy, навчитися описувати структуру бази даних у вигляді класів-моделей, а також реалізувати графічний клієнт на основі Tkinter для виконання CRUD-операцій (створення, читання, оновлення, видалення) над даними транспортної фірми.

Обладнання та програмне забезпечення

• ПК/ноутбук з операційною системою Windows / Linux / macOS
• Python 3.x
• Бібліотека SQLAlchemy
• Стандартна бібліотека Tkinter
• СУБД SQLite (файл бази даних transport.db)
• Редактор коду Vim

Теоретичні відомості

ORM (Object-Relational Mapping) — це технологія, яка дозволяє працювати з реляційною базою даних через об’єктно-орієнтовану модель. Замість написання сирих SQL-запитів розробник працює з класами та об’єктами, а бібліотека ORM автоматично транслює операції над об’єктами у відповідні SQL-інструкції.

SQLAlchemy — популярна ORM-бібліотека для Python, яка надає потужні засоби для опису моделей, створення та міграції схеми бази даних, організації сесій доступу до даних та виконання запитів.

Tkinter — стандартна бібліотека Python для побудови графічних інтерфейсів користувача (GUI). Вона дозволяє створювати вікна, кнопки, таблиці, поля введення та інші елементи інтерфейсу.

У даній лабораторній роботі база даних моделює роботу транспортної фірми. Основні сутності:
• Client — клієнти (юридичні та фізичні особи);
• Driver — водії;
• Vehicle — транспортні засоби;
• Order — замовлення на перевезення;
• TripDetails — деталізація поїздки (призначення водія та транспорту, вартість,

статус);
• TripLog — фактичні дані виконання поїздки (часи відправлення/прибуття, коментарі).

Хід роботи

1. 1. Створено структуру проєкту:

- • models.py — опис ORM-моделей (Client, Driver, Vehicle, Order, TripDetails, TripLog) та функції підключення до бази даних;
  • gui.py — графічний клієнт на Tkinter, який використовує SQLAlchemy для доступу до даних;
  • transport.db — файл бази даних SQLite, який створюється автоматично при першому запуску.

2. 2. Реалізовано файл models.py з ORM-моделями.

У файлі models.py створено підключення до БД SQLite за допомогою create_engine("sqlite:///transport.db"). Оголошено базовий клас Base за допомогою declarative_base(), а також класи-моделі:
• Client — таблиця clients з полями id, client_type, name, contacts;
• Driver — таблиця drivers з полями id, full_name, license_number, phone;
• Vehicle — таблиця vehicles з полями id, reg_number, vehicle_type, capacity, description;
• Order — таблиця orders з полями id, client_id, route, departure_time, arrival_time;
• TripDetails — таблиця trip_details, яка пов’язує Order, Driver та Vehicle, містить статус та вартість;
• TripLog — таблиця trip_logs, яка зберігає фактичні часи поїздки та коментар.

3. 3. Описано зв’язки між сутностями.

За допомогою ForeignKey та relationship налаштовано такі зв’язки:
• Client 1..N Order (клієнт може мати багато замовлень);
• Order 1..1 TripDetails (кожному замовленню відповідають деталі поїздки);
• Driver 1..N TripDetails (водій може виконувати багато поїздок);
• Vehicle 1..N TripDetails (транспортний засіб може використовуватись у багатьох поїздках);
• TripDetails 1..1 TripLog (для кожної поїздки може бути один запис фактичних даних).

4. 4. Створено графічний клієнт gui.py.

У файлі gui.py створено головне вікно додатку TransportApp, яке містить вкладки (Notebook):
• Clients — робота з таблицею клієнтів;
• Drivers — робота з водіями;
• Vehicles — робота з транспортом;
• Orders — робота із замовленнями;
• Trips — робота з деталями поїздок та журналом (TripDetails + TripLog).

Кожна вкладка реалізована як окремий клас-нащадок BaseTab і містить таблицю (Treeview) для відображення даних та кнопки для виконання CRUD-операцій (додати, редагувати, видалити, оновити). Для введення та редагування даних використовуються модальні діалогові вікна (Toplevel) з полями введення.

5. 5. Реалізовано функції для додавання, оновлення та видалення записів.

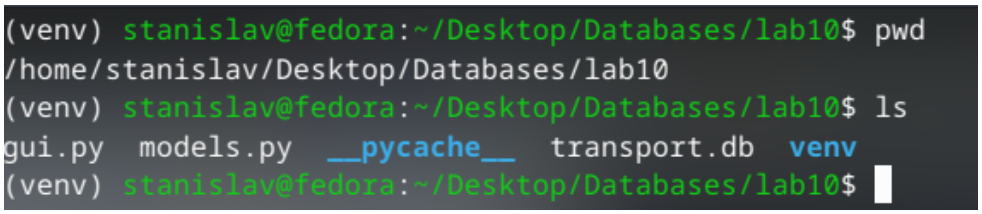
При натисканні кнопок "Додати" або "Редагувати" відкривається відповідне діалогове вікно, де користувач вводить або змінює дані. Після збереження інформація оновлюється у базі даних за допомогою сесії SQLAlchemy (session.add(), session.commit()), а таблиця у вікні перезавантажується.

Кнопка "Видалити" викликає підтвердження (messagebox.askyesno), після чого запис видаляється з БД та інтерфейс оновлюється.

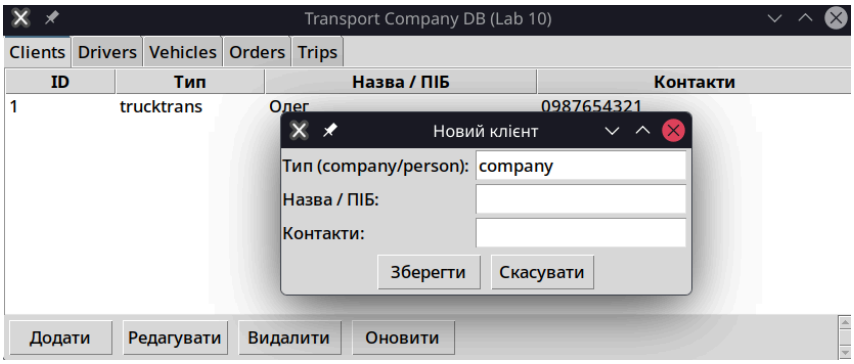
6. 6. Налаштовано формат введення дат та роботи з журналом поїздок.

Для полів дати та часу використовується формат YYYY-MM-DD HH:MM. Перед збереженням значення, введені користувачем у текстові поля, перетворюються на об’єкти datetime за допомогою функції datetime.strptime. У вкладці Trips при збереженні даних поїздки автоматично створюється або оновлюється пов'язаний запис у таблиці TripLog.

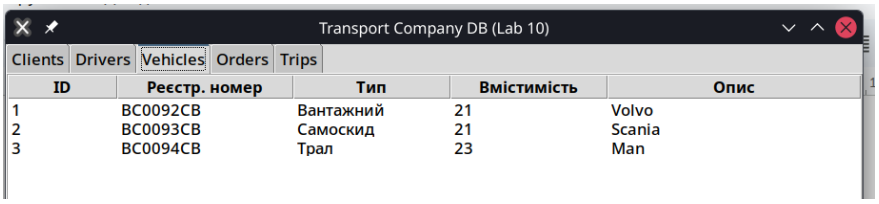
Скріншоти виконання роботи
структура проєкту у файловій системі (models.py, gui.py, transport.db).



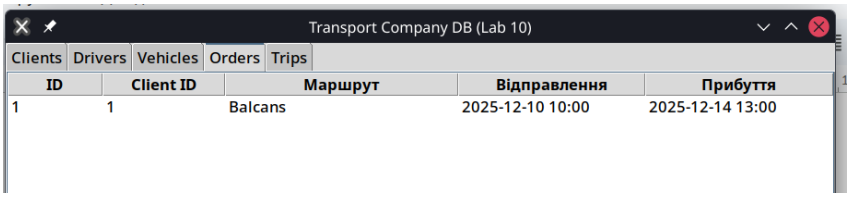
приклад додавання або редагування клієнта (діалогове вікно).



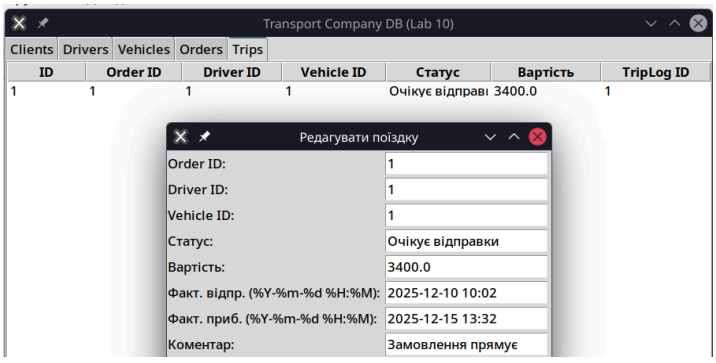
вкладка Drivers або Vehicles з відображенням записів.



вкладка Orders з прикладом введення дати та маршруту



вкладка Trips з прикладом введення TripDetails та TripLog.



Висновки

У ході виконання лабораторної роботи було розглянуто підхід ORM для взаємодії з реляційною базою даних на прикладі бібліотеки SQLAlchemy. Було описано структуру бази даних транспортної фірми у вигляді класів-моделей, налаштовано зв’язки між сутностями та організовано зручний доступ до даних через сесію ORM.

Додатково реалізовано графічний клієнт на основі Tkinter, який забезпечує виконання CRUD-операцій над основними таблицями (Clients, Drivers, Vehicles, Orders, Trips). Такий підхід дозволяє абстрагуватися від деталей SQL, спростити роботу з даними та покращити читаність і підтримуваність коду застосунку.