

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп’ютерних систем

**Лабораторна робота № 2**

з дисципліни «Бази даних і засоби управління»

«**Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL**»

Виконала: Костюченко А. В.

Студентка групи КВ-03

**Київ 2023**

**Лабораторна робота №2**

Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

Посилання на репозиторій GitHub з вихідним кодом: <https://github.com/reginakasteen/data-bases/tree/main/data_bases>

**Логічна модель предметної області «Готель»**

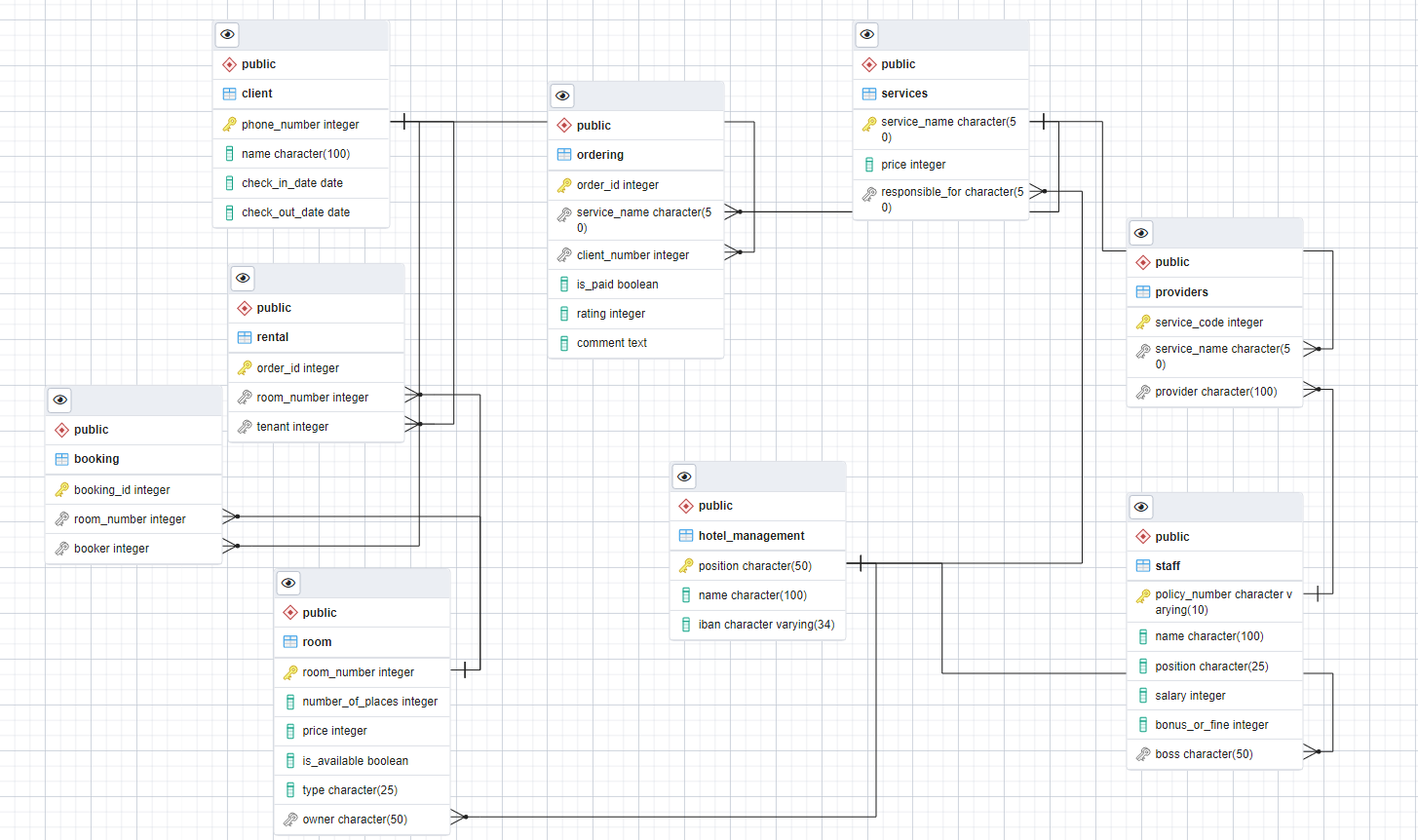


Рисунок 1 – Логічна модель предметної області «Готель»

**Середовище розробки та налаштування підключення до бази даних**

Для виконання лабораторної роботи використовувалась мова програмування Python та середовище розробки PyCharm

Для підключення до серверу бази даних PostgreSQL використано сторонню бібліотеку psycorg2.

**Шаблон проектування**

MVC - Шаблон проектування, який використаний у програмі.

Model – представляє клас, що описує логіку використовуваних даних. Клас реалізований у файлі model.py.

View – консольний інтерфейс для взаємодії з користувачем. Відповідає за введення/виведення даних. У моїй програмі це реалізовано за допомогою двох файлів: view.py (клас View) і menu.py (клас Menu).

Controller – представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує введені користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання. У програмі це реалізовано у файлі controller.py.

**Структура програми та її опис**

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – структура програми

1. launcher.py – точка входу в програму, запускає початковий інтерфейс.
2. view.py – файл, що відповідає за функціонал виведення даних і повідомлень для користувача.
3. menu.py – файл, що реалізовує меню для взаємодії з користувачем, приймає введені дані від користувача і передає їх у контролер.
4. controller.py – виконує підключення до бази даних, обробляє введені користувачем дані, подає відповідну команду до model.py.
5. model.py – виконує складні операції з базою даних з конкретними таблицями БД.

**Структура меню програми**

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Початкове меню програми

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Меню для вибору таблиці для виведення даних

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Меню для вибору таблиці для внесення даних

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Меню для вибору таблиці для видалення даних

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Меню для вибору таблиці для редагування даних

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Меню для пошуку даних

Меню для користувача складається з cеми пунктів (Рисунок 3).

Перший пункт («0») пропонує виведення однієї таблиці за вибором (Рисунок 4). Перед виведенням даних, користувач обирає, яку саме таблицю потрібно вивести. Після цього на екрані виводяться всі рядки і стовпчики таблиці БД.

Другий пункт («1») пропонує виведення всіх таблиць. Послідовно виводяться усі таблиці БД, після чого користувач знову повертається до головного меню і може обрати нову опцію для взаємодії з таблицями бази даних.

Третій пункт («2») пропонує внесення даних (Рисунок 5). Перед тим, як ввести дані, потрібно вибрати, для якої таблиці буде відбуватися внесення. Тому користувач вводить номер, що відповідає певній таблиці. Після цього користувач вводить дані для кожного атрибуту.

Четвертий пункт («3») пропонує видалення даних (Рисунок 6). Перед тим, як видалити дані, потрібно вибрати, для якої таблиці буде відбуватися видалення. Тому користувач вводить номер, що відповідає певній таблиці. Після цього користувач вводить ідентифікатор рядка, який потрібно видалити. Потім відбувається видалення даних відповідного рядка.

П’яий пункт («4») пропонує редагування даних (Рисунок 7). Перед тим, як редагувати дані, потрібно вибрати, для якої таблиці буде відбуватися редагування. Тому користувач вводить номер, що відповідає певній таблиці. Після цього користувач обирає, які саме дані редагувати, вводить нові дані, які записуються в таблицю.

Шостий пункт («5») пропонує пошук за атрибутами з декількох таблиць (Рисунок 8). Користувач обирає, який запит він хоче виконувати, після чого вводить дані для пошуку. Після введення атрибуту користувач може продовжити пошук або повернутися до головного меню і вивести таблицю за допомогою опції «0» або вивести всі таблиці за допомогою опції «1».

Восьмий пункт («6») пропонує вихід з програми. Закривається з’єднання з таблицею бази даних і програма завершується.

Внесення рандомних даних не реалізовано.

**Лістинг фрагментів програми для внесення даних**

@staticmethod  
def insert():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 go\_on = True  
 while go\_on:  
 View.list()  
 table = controller.validtable()  
 if table == 1:  
 phone\_number = controller.validate\_input\_items("phone\_number")  
 name = controller.validate\_input\_items("name")  
 check\_in = controller.validate\_input\_items("check\_in")  
 check\_out = controller.validate\_input\_items("check\_out")  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("Person already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "client" ("phone\_number", "name", "check\_in\_date", "check\_out\_date")   
 VALUES ('{phone\_number}', '{name}', '{check\_in}', '{check\_out}')""")  
 View.complete\_message("phone\_number", phone\_number, "client", "inserted")  
 go\_on = False  
 elif table == 2:  
 position = controller.validate\_input\_items("position")  
 name = controller.validate\_input\_items("name")  
 iban = controller.validate\_input\_items("iban")  
 cursor.execute(f"""select \* from public.hotel\_management where "position" = '{position}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "hotel\_management" ("position", "name", "iban")   
 VALUES ('{position}', '{name}', '{iban}')""")  
 View.complete\_message("position", position, "hotel\_management", "inserted")  
 go\_on = False  
 elif table == 3:  
 room = controller.validate\_input\_items("room")  
 number\_of = controller.validate\_input\_items("number")  
 price = controller.validate\_input\_items("price")  
 is\_available = controller.validate\_input\_items("is\_available")  
 type = controller.validate\_input\_items("type")  
 owner = controller.validate\_input\_items("owner")  
 cursor.execute(f"""select \* from public.room where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "room" ("room\_number", "number\_of\_places", "price", "is\_available", "type", "owner")   
 VALUES ('{room}', '{number\_of}', '{price}', '{is\_available}', '{type}', '{owner}')""")  
 View.complete\_message("room", room, "room", "inserted")  
 go\_on = False  
 elif table == 4:  
 name = controller.validate\_input\_items("name")  
 responsible = controller.validate\_input\_items("responsible")  
 price = controller.validate\_input\_items("price")  
 cursor.execute(f"""select \* from public.services where "service\_name" = '{name}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "services" ("service\_name", "price", "responsible\_for")   
 VALUES ('{name}', '{price}', '{responsible}')""")  
 View.complete\_message("phone\_id", id, "phone", "inserted")  
 go\_on = False  
 elif table == 5:  
 policy = controller.validate\_input\_items("policy")  
 name = controller.validate\_input\_items("name")  
 position = controller.validate\_input\_items("position")  
 salary = controller.validate\_input\_items("salary")  
 bonus = controller.validate\_input\_items("bonus")  
 boss = controller.validate\_input\_items("boss")  
 cursor.execute(f"""select \* from public.staff where "policy\_number" = '{policy}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "staff" ("policy\_number", "name", "position", "salary", "bonus\_or\_fine", "boss")   
 VALUES ('{policy}', '{name}', '{position}', '{salary}', '{bonus}', '{boss}')""")  
 View.complete\_message("policy\_number", policy, "staff", "inserted")  
 go\_on = False  
 elif table == 6:  
 booking = controller.validate\_input\_items("rent\_id")  
 booker = controller.validate\_input\_items("phone\_number")  
 room = controller.validate\_input\_items("room")  
 cursor.execute(f"""select \* from public.booking where "booking\_id" = '{booking}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{booker}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.room where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""INSERT INTO "booking" ("booking\_id", "booker", "room\_number")   
 VALUES ('{booking}', '{booker}', '{room}')""")  
 View.complete\_message("booking\_ID", booking, "booking", "inserted")  
 else:  
 controller.message("The room with this ID doesn't exist")  
 else:  
 controller.message("The client with this ID doesn't exist")  
 go\_on = False  
 elif table == 7:  
 rent = controller.validate\_input\_items("rent\_id")  
 tenant = controller.validate\_input\_items("phone\_number")  
 room = controller.validate\_input\_items("room")  
 cursor.execute(f"""select \* from public.rental where "order\_id" = '{rent}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{tenant}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.room where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""INSERT INTO "rental" ("order\_id", "tenant", "room\_number")   
 VALUES ('{rent}', '{tenant}', '{room}')""")  
 View.complete\_message("order\_ID", rent, "rental", "inserted")  
 else:  
 controller.message("The room with this ID doesn't exist")  
 else:  
 controller.message("The client with this ID doesn't exist")  
 go\_on = False  
 elif table == 8:  
 order = controller.validate\_input\_items("service\_code")  
 service = controller.validate\_input\_items("name")  
 client = controller.validate\_input\_items("phone\_number")  
 is\_paid = controller.validate\_input\_items("is\_paid")  
 rating = controller.validate\_input\_items("reting")  
 comment = controller.validate\_input\_items("comment")  
 cursor.execute(f"""select \* from public.ordering where "order\_id" = '{order}'""")  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""select \* from public.services where "service\_name" = '{service}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{client}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""INSERT INTO "ordering"   
 ("order\_id", "service\_name", "client\_number", "is\_paid", "rating", "comment")   
 VALUES ('{order}', '{service}', '{client}', '{is\_paid}', '{rating}', '{comment}')""")  
 View.complete\_message("order\_ID", order, "ordering", "inserted")  
 else:  
 controller.message("The client with this ID doesn't exist")  
 else:  
 controller.message("The service with this ID doesn't exist")  
 go\_on = False  
 elif table == 9:  
 code = controller.validate\_input\_items("sevice\_code")  
 service = controller.validate\_input\_items("name")  
 provider = controller.validate\_input\_items("policy")  
 cursor.execute(f"""select \* from public.providers where "service\_code" = '{code}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""select \* from public.staff where "policy\_number" = '{provider}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.services where "service\_name" = '{service}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""INSERT INTO "providers" ("service\_code", "service\_name", "provider")   
 VALUES ('{code}', '{service}', '{provider}')""")  
 View.complete\_message("service\_code", code, "providers", "inserted")  
 else:  
 controller.message("The service with this ID doesn't exist")  
 else:  
 controller.message("The worker with this ID doesn't exist")  
 go\_on = False  
 else:  
 print('Please, enter valid value')  
 connection.commit()  
 cursor.close()  
 controller.disconnect(connection)

Внесення даних відбувається за допомогою функції insert. Функція вносить дані відповідно до обраної користувачем таблиці в меню. Є контроль введених даних на коректність типу у функції validate\_input\_items у файлі controller.py (лістинг надано нижче у звіті). У разі помилки повертає повну інформацію про помилку і повертає користувача до внесення даних. У разі успішного внесення даних користувачеві виводиться відповідне повідомлення і пропонується вибір щодо подальшого внесення даних. Якщо користувач не хоче вносити дані, він повертається до головного меню.

**Лістинг фрагментів програми для вилучення даних**

Вилучення даних відбувається за допомогою функції delete. Користувачем обирається необхідна таблиця, з якої потрібно видалити дані. Є перевірка на коректність введення номеру таблиці. Далі користувачеві пропонується ввести номер ідентифікатора рядка, який потрібно видалити. Якщо рядок із таким ідентифікатором відсутній, користувачу буде виведено відповідне повідомлення і буде запропоновано або продовжити видалення, або перейти до головного меню. У разі успішного видалення рядка користувач побачить відповідне повідомлення, а також зможу обрати: продовжувати видалення або повернутися до головного меню. Якщо ідентифікатор рядка, що потрібно видалити, є зовнішнім ключем у іншій таблиці, рядок з іншої таблиці також буде видалено, про це буде свідчити виведене повідомлення.

@staticmethod  
def delete():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 go\_on = True  
 while go\_on:  
 View.list()  
 table = controller.validtable()  
 if table == 1:  
 phone\_number = controller.validate\_input\_items("phone\_number")  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.booking where "booker" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.booking WHERE "booker" = '{phone\_number}' """)  
 View.complete\_message("booker", phone\_number, "booking", "deleted")  
 cursor.execute(f"""select \* from public.rental where "tenant" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.rental WHERE "tenant" = '{phone\_number}' """)  
 View.complete\_message("tenant", phone\_number, "rental", "deleted")  
 cursor.execute(f"""select \* from public.ordering where "client\_number" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.ordering WHERE "client\_number" = '{phone\_number}' """)  
 View.complete\_message("client\_number", phone\_number, "ordering", "deleted")  
 cursor.execute(f"""DELETE FROM public.client WHERE "phone\_number" = '{phone\_number}'""")  
 View.complete\_message("phone\_number", phone\_number, "client", "deleted")  
 else:  
 controller.message("No person found")  
 go\_on = False  
 elif table == 2:  
 position = controller.validate\_input\_items("position")  
 cursor.execute(f"""select \* from public.hotel\_management where "position" = '{position}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.room where "owner" = '{position}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.room WHERE "owner" = '{position}'""")  
 View.complete\_message("owner", position, "room", "deleted")  
 cursor.execute(f"""select \* from public.staff where "boss" = '{position}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.staff WHERE "boss" = '{position}'""")  
 View.complete\_message("boss", position, "staff", "deleted")  
 cursor.execute(f"""DELETE FROM public.hotel\_management WHERE "position" = '{position}'""")  
 View.complete\_message("position", position, "hotel\_management", "deleted")  
 else:  
 controller.message("No person found")  
 go\_on = False  
 elif table == 3:  
 room = controller.validate\_input\_items("room")  
 cursor.execute(f"""select \* from public.room where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.booking where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.booking WHERE "room\_number" = '{room}'""")  
 View.complete\_message("roon\_number", room, "booking", "deleted")  
 cursor.execute(f"""select \* from public.rental where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.rental WHERE "room\_number" = '{room}'""")  
 View.complete\_message("room\_number", room, "rental", "deleted")  
 cursor.execute(f"""DELETE FROM public.services WHERE "service\_name" = '{room}'""")  
 View.complete\_message("room\_number", room, "room", "deleted")  
 else:  
 controller.message("No room found")  
 go\_on = False  
 elif table == 4:  
 name = controller.validate\_input\_items("name")  
 cursor.execute(f"""select \* from public.services where "service\_name" = '{name}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.ordering where "service\_name" = '{name}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.ordering WHERE "service\_name" = '{name}'""")  
 View.complete\_message("service\_name", name, "ordering", "deleted")  
 cursor.execute(f"""select \* from public.providers where "service\_name" = '{name}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.providers WHERE "service\_name" = '{name}'""")  
 View.complete\_message("service\_name", name, "providers", "deleted")  
 cursor.execute(f"""DELETE FROM public.services WHERE "service\_name" = '{name}'""")  
 View.complete\_message("service\_name", name, "services", "deleted")  
 else:  
 controller.message("No service found")  
 go\_on = False  
 elif table == 5:  
 policy = controller.validate\_input\_items("policy")  
 cursor.execute(f"""select \* from public.staff where "policy\_number" = '{policy}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.providers where "service\_name" = '{policy}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.providers WHERE "service\_name" = '{policy}'""")  
 View.complete\_message("provider", policy, "providers", "deleted")  
 cursor.execute(f"""DELETE FROM public.staff WHERE "policy\_number" = '{policy}'""")  
 View.complete\_message("policy\_number", policy, "staff", "deleted")  
 else:  
 controller.message("No person found")  
 go\_on = False  
 elif table == 6:  
 rent\_id = controller.validate\_input\_items("rent\_id")  
 cursor.execute(f"""select \* from public.booking where "booking\_id" = '{rent\_id}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.booking WHERE "booking\_id" = '{rent\_id}'""")  
 View.complete\_message("booking\_id", rent\_id, "booking", "deleted")  
 else:  
 controller.message("No booking found")  
 go\_on = False  
 elif table == 7:  
 rent\_id = controller.validate\_input\_items("rent\_id")  
 cursor.execute(f"""select \* from public.rental where "order\_id" = '{rent\_id}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.rental WHERE "order\_id" = '{rent\_id}'""")  
 View.complete\_message("order\_id", rent\_id, "rental", "deleted")  
 else:  
 controller.message("No rent found")  
 go\_on = False  
 elif table == 8:  
 order\_id = controller.validate\_input\_items("order")  
 cursor.execute(f"""select \* from public.ordering where "order\_id" = '{order\_id}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.ordering WHERE "order\_id" = '{order\_id}'""")  
 View.complete\_message("schedule\_id", id, "schedule", "deleted")  
 else:  
 controller.message("No order found")  
 go\_on = False  
 elif table == 9:  
 code = controller.validate\_input\_items("service\_code")  
 cursor.execute(f"""select \* from public.providers where "service\_code" = '{code}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.providers WHERE "service\_code" = '{code}'""")  
 View.complete\_message("service\_code", code, "providers", "deleted")  
 else:  
 controller.message("No service found")  
 go\_on = False  
 else:  
 print("Please, enter valid number")  
 connection.commit()  
 cursor.close()  
 controller.disconnect(connection)  
 pass

**Лістинг фрагментів програми для редагування даних**

Редагування даних відбувається за допомогою функції update. Користувачем обирається необхідна таблиця, у якій потрібно відредагувати дані. Є перевірка на коректність введення номеру таблиці. Далі користувачеві потрібно ввести номер ідентифікатора рядка, який потрібно відредагувати. Якщо такого ідентифікатора не існує, буде виведено відповідне повідомлення про помилку і запропоновано або продовжити редагування, або повернутися до головного меню. Якщо введений номер ідентифікатора існує, користувач може обрати, значення якого саме атрибуту він хоче відредагувати. Також є перевірка введених даних на коректність типу. У разі некоректних даних буде виведено повідомлення про помилку. Після успішного редагування користувач побачить відповідне повідомлення і зможе або продовжити редагування, або повернутися до головного меню.

@staticmethod  
def update():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 go\_on = True  
 while go\_on:  
 View.list()  
 table = controller.validtable()  
 if table == 1:  
 phone\_number = controller.validate\_input\_items("phone\_number")  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 continue\_update = True  
 if records is not None:  
 View.columns(1)  
 while continue\_update:  
 attr = input("Choose a number of column to update")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("check\_in")  
 attribute = "check\_in\_date"  
 continue\_update = False  
 elif attr == '3':  
 value = controller.validate\_input\_items("check\_out")  
 attribute = "check\_out\_date"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number")  
 cursor.execute(f"""UPDATE public.client set {attribute} = '{value}' where "phone\_number" = '{phone\_number}' """)  
 View.complete\_message("phone\_nomber", phone\_number, "client", "updated")  
 go\_on = False  
 else:  
 controller.message("No client with this ID found")  
 elif table == 2:  
 position = controller.validate\_input\_items("position")  
 cursor.execute(f"""select \* from public.hotel\_management where "position" = {position} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(2)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("iban")  
 attribute = "iban"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number")  
 cursor.execute(f"""UPDATE "hotel\_management" set {attribute} = '{value}' where "position" = '{id}' """)  
 View.complete\_message("position", position, "hotel\_management", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No boss with this ID found")  
 elif table == 3:  
 room = controller.validate\_input\_items("room")  
 cursor.execute(f"""select \* from public.room where "room\_number" = {room} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(3)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("number")  
 attribute = "number\_of\_places"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("price")  
 attribute = "price"  
 continue\_update = False  
 elif attr == '3':  
 value = controller.validate\_input\_items("is\_available")  
 attribute = "is\_available"  
 continue\_update = False  
 elif attr == '4':  
 value = controller.validate\_input\_items("type")  
 attribute = "type"  
 continue\_update = False  
 elif attr == '5':  
 value = controller.validate\_input\_items("owner")  
 attribute = "owner"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number")  
 cursor.execute(f"""UPDATE "room" set {attribute} = '{value}' where "room\_number" = '{room}' """)  
 View.complete\_message("room\_number", room, "room", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No room with this ID found")  
 elif table == 4:  
 name = controller.validate\_input\_items("name")  
 cursor.execute(f"""select \* from public.services where "service\_name" = {name} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(4)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("price")  
 attribute = "price"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("responsible")  
 attribute = "responsible\_for"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "services" set {attribute} = '{value}' where "service\_name" = '{name}' """)  
 View.complete\_message("service\_name", name, "services", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No service with this ID found")  
 elif table == 5:  
 policy = controller.validate\_input\_items("policy")  
 cursor.execute(f"""select \* from public.staff where "policy\_number" = {policy} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(5)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update ")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("position")  
 attribute = "position"  
 continue\_update = False  
 elif attr == '3':  
 value = controller.validate\_input\_items("salary")  
 attribute = "salary"  
 continue\_update = False  
 elif attr == '4':  
 value = controller.validate\_input\_items("bonus")  
 attribute = "bonus\_or\_fine"  
 continue\_update = False  
 elif attr == '5':  
 value = controller.validate\_input\_items("boss")  
 attribute = "boss"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number")  
 cursor.execute(f"""UPDATE "staff" set {attribute} = '{value}' where "policy\_number" = '{policy}' """)  
 View.complete\_message("policy\_number", policy, "staff", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No worker with this ID found")  
 elif table == 6:  
 rent\_id = controller.validate\_input\_items("rent\_id")  
 cursor.execute(f"""select \* from public.booking where "booking\_id" = {rent\_id} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(6)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("room")  
 attribute = "room\_number"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("phone\_number")  
 attribute = "booker"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "booking" set {attribute} = '{value}' where "booking\_id" = '{rent\_id}' """)  
 View.complete\_message("booking\_id", rent\_id, "booking", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No booking with this ID found")  
 elif table == 7:  
 rent\_id = controller.validate\_input\_items("rent\_id")  
 cursor.execute(f"""select \* from public.rental where "order\_id" = {rent\_id} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(7)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("room")  
 attribute = "room\_number"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("phone\_number")  
 attribute = "tenant"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "rental" set {attribute} = '{value}' where "order\_id" = '{rent\_id}' """)  
 View.complete\_message("order\_id", rent\_id, "rental", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No rent with this ID found")  
 elif table == 8:  
 id = controller.validate\_input\_items("order")  
 cursor.execute(f"""select \* from public.ordering where "order\_id" = {id} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(8)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "service\_name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("phone\_number")  
 attribute = "client\_number"  
 continue\_update = False  
 elif attr == '3':  
 value = controller.validate\_input\_items("is\_paid")  
 attribute = "is\_paid"  
 continue\_update = False  
 elif attr == '4':  
 value = controller.validate\_input\_items("rating")  
 attribute = "rating"  
 continue\_update = False  
 elif attr == '5':  
 value = controller.validate\_input\_items("comment")  
 attribute = "comment"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "ordering" set {attribute} = '{value}' where "order\_id" = '{id}' """)  
 View.complete\_message("order\_id", id, "ordering", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No order with this ID found")  
 elif table == 9:  
 id = controller.validate\_input\_items("order")  
 cursor.execute(f"""select \* from public.providers where "service\_code" = {id} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(9)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "service\_name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("policy")  
 attribute = "provider"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "providers" set {attribute} = '{value}' where "service\_code" = '{id}' """)  
 View.complete\_message("service\_code", id, "providers", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No order with this ID found")  
 else:  
 controller.message("Please, enter valid number")  
 connection.commit()  
 cursor.close()  
 controller.disconnect(connection)  
 pass

**Лістинг фрагментів програми для пошуку даних**

Користувачеві пропонується обрати варіант пошуку. Також розраховується час запиту кожного пошуку в мілісекундах.

*Пошук бронювання та номеру телефону за номером клієнта*

Запит цього пошуку реалізовано за допомогою функції search1. Користувач вводить ім’я клієнта, після чого виводиться таблиця із знайденим іменем та іншими даними, а також час виконання операції пошуку в мілісекундах.

@staticmethod  
def search1():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 name = controller.validate\_input\_items("name")  
 cursor.execute(f"""select \* from public.client where "name" = '{name}' """)  
 records = cursor.fetchone()  
 #if records is not None:  
 search = f"""select "name", "phone\_number", "booking\_id" from (select c."name", c."phone\_number", b."booking\_id"  
 from "client" c left join "booking" b on b."booker" = c."phone\_number"  
 where c."name" LIKE '{name}' group by c."name", c."phone\_number", b."booking\_id") as foo"""  
 start = int(time.time() \* 1000)  
 cursor.execute(search)  
 print("--- Time of search = {} ms ---".format(int((time.time() \* 1000) - start)))  
 records = cursor.fetchall()  
 cursor.close()  
 return records

*Пошук кімнат, які знаходяться у розпорядженні директора, та їхні типи за іменем директора*

Запит цього пошуку реалізовано за допомогою функції search2. Користувач вводить ім’я директора, після чого виводиться таблиця із знайденими приміщеннями, їх типами та іменем директора, а також час виконання операції пошуку в мілісекундах.

@staticmethod  
def search2():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 name = controller.validate\_input\_items("name")  
 cursor.execute(f"""select \* from public.hotel\_management where "name" = '{name}' """)  
 records = cursor.fetchone()  
 search = f"""select "name", "room\_number", "type" from (select h."name", r."room\_number", r."type"  
 from "hotel\_management" h left join "room" r on r."owner" = h."position"  
 where h."name" LIKE '{name}' group by h."name", r."room\_number", r."type") as foo"""  
 start = int(time.time() \* 1000)  
 cursor.execute(search)  
 print("--- Time of search = {} ms ---".format(int((time.time() \* 1000) - start)))  
 records = cursor.fetchall()  
 cursor.close()  
 return records

*Пошук послуг, які коштують дорожче певної суми, їні назви та директор, котрий контролює якість*

Запит цього пошуку реалізовано за допомогою функції search3. Користувач вводить ціну послуги, після чого виводиться таблиця із списком послуг, ціна яких строго вище введеної суми, а також час виконання операції пошуку в мілісекундах.

@staticmethod  
def search3():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 price = controller.validate\_input\_items("price")  
 cursor.execute(f"""select \* from public.services where "price" = '{price}' """)  
 records = cursor.fetchone()  
 search = f"""select "price", "service\_name", "name" from (select s."price", s."service\_name", h."name"  
 from "services" s left join "hotel\_management" h on h."position" = s."responsible\_for"  
 where s."price" > '{price}' group by s."price", s."service\_name", h."name") as foo"""  
 start = int(time.time() \* 1000)  
 cursor.execute(search)  
 print("--- Time of search = {} ms ---".format(int((time.time() \* 1000) - start)))  
 records = cursor.fetchall()  
 cursor.close()  
 return records

**Лістинг модуля "Model" згідно із шаблоном MVC**

Модуль Model реалізовано у файлі model.py. Усі основні операції внесення, редагування, видалення та виведення даних реалізовано у класі Model. Також файл містить словник (перелік) усіх таблиць, яким відповідає певний номер. Він передаватиметься як параметр для безпосередніх дій над таблицею БД.

У класі Model реалізовано наступні функції: display\_query – приймає рядки і назви стовпців таблиці, далі виводить дані у вигляді таблиці; show\_table – приймає назву таблиці як параметр і передає всі дані в display\_query для виведення; insert – реалізує внесення даних (пояснення надано у пункті з відповідним лістингом вище); delete – реалізує видалення даних (пояснення надано у пункті з відповідним лістингом вище); update – реалізує редагування даних (пояснення надано у пункті з відповідним лістингом вище); search1, search2, search3 – реалізують 3 варіанти пошуку (пояснення надано у пункті з відповідним лістингом вище).

**Результати роботи програми**

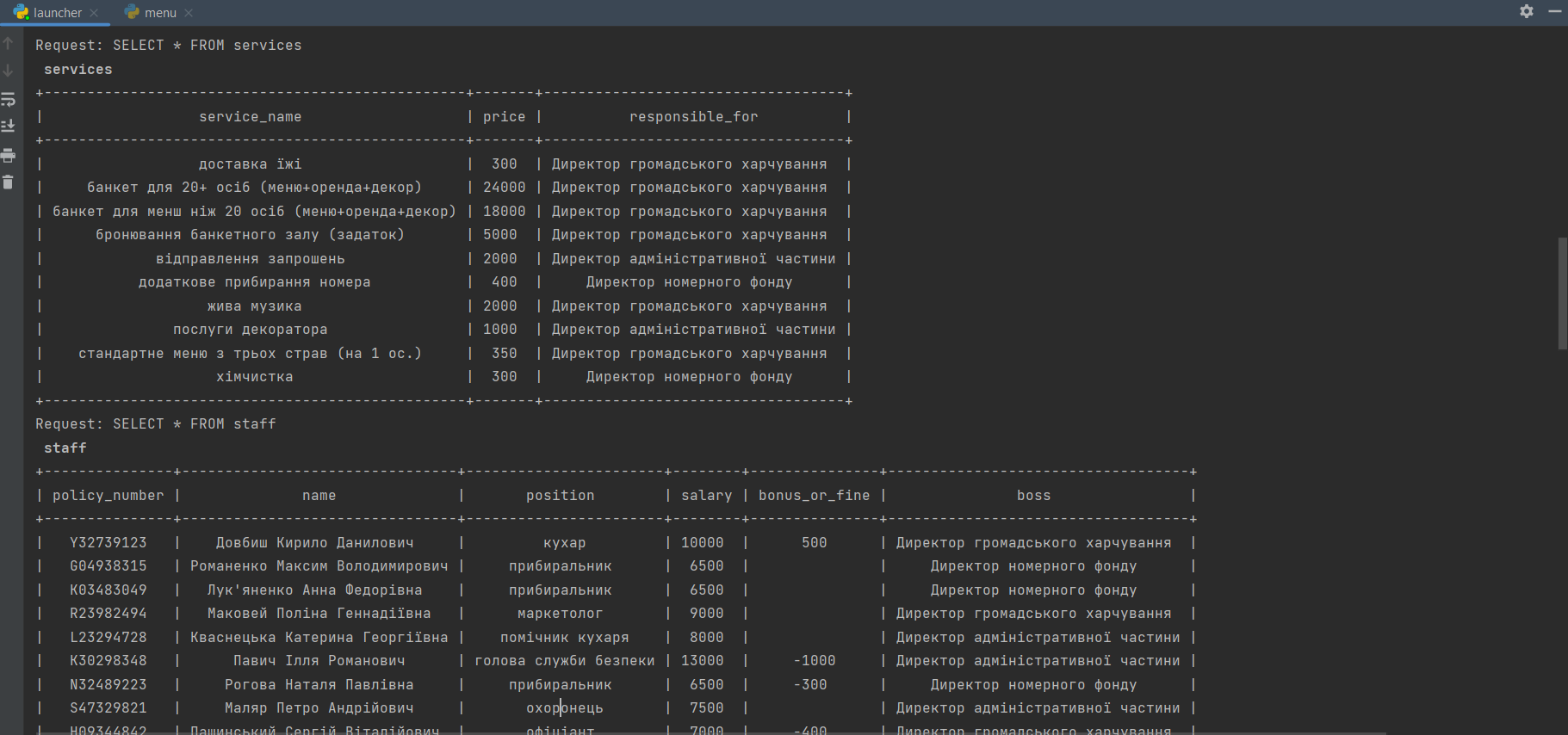
Виведення таблиць першим пунктом

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

Виведення однієї таблиці

Изображение выглядит как текст

Автоматически созданное описание

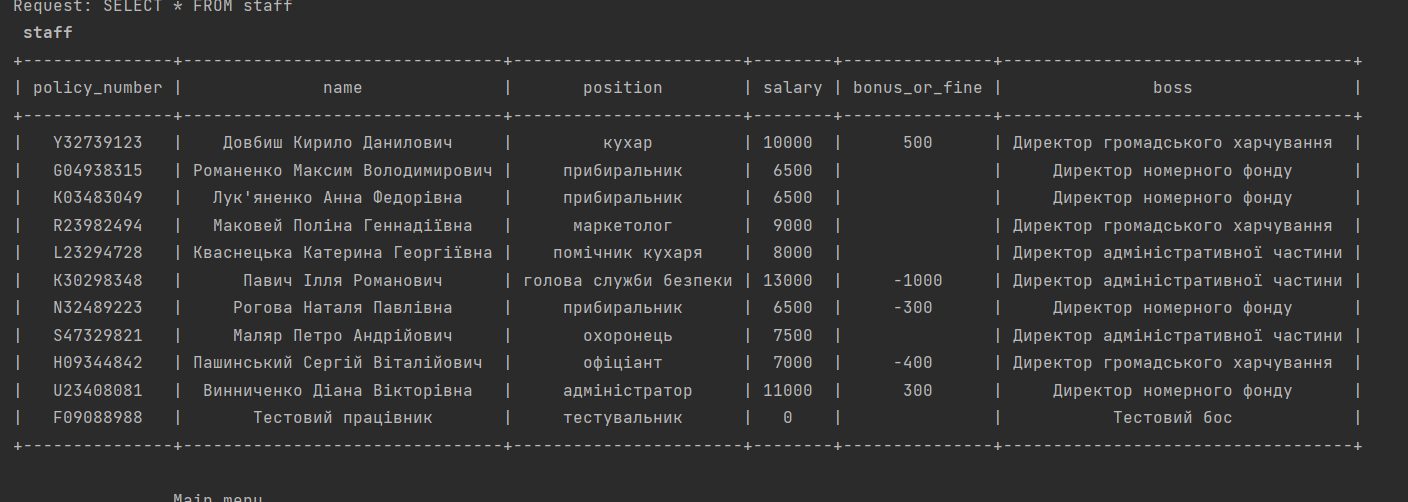
Демонстрація видалення даних. Додамо нового тестового директора в таблицю hotel\_management, а також нового працівника та кімнату, якими він керує, у відповідні таблиці.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

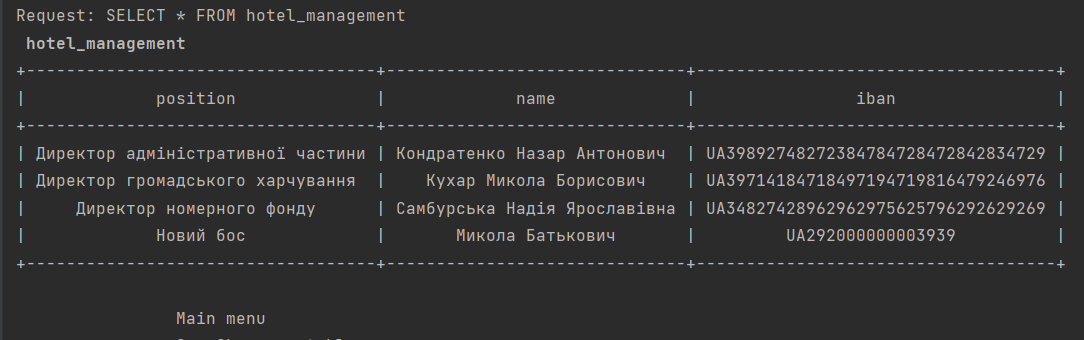


Проведемо вилучення тестового боса із таблиці hotel\_management.

Изображение выглядит как текст

Автоматически созданное описание

Тестовий бос пішов із всіх таблиць



Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Додамо нового боса за допомогою функції insert. (Тестового боса на той момент ще не видаляли)

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Додамо новий номер з типом “унікальний”:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Спробуємо знайти для оновлення даних клієнта з неіснуючим номером телефону. Як бачимо, програма його не знаходить, при тому вимагає введення номеру певної довжини.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Спробуємо оновити дату в’їзду пані Вишнецької за її номером:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Для коректного запису в базу даних, контролер фільтрує пунктуаційні знаки в даті, що вводиться та замінює їх рискою:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Оновимо інформацію про виконавця замовлення в таблиці providers:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Тепер замовлення номер 231 виконує працівник з полісом R23982494:

Изображение выглядит как текст

Автоматически созданное описание

Реалізуємо функції пошуку. Знайдемо номер телефону та бронювання пані Вишнецької:

Изображение выглядит как текст

Автоматически созданное описание

Пошук працює коректно

Изображение выглядит как текст

Автоматически созданное описание

Знайдемо кімнати, які знаходяться у власності пана Кухаря:

Изображение выглядит как текст

Автоматически созданное описание

Пан Кухар дійсно директор громадського харчування і керує банкетними залами та терасою:

Изображение выглядит как текст

Автоматически созданное описание

Знайдемо в готелі послуги, що дорожче 1000 грн, та людей, які їх контролюють:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Текст програми мовою Python**

**menu.py**

from model import Model  
from view import View  
import controller  
  
  
class Menu:  
 @staticmethod  
 def menu():  
 while True:  
 print('''  
 Main menu  
 0 - Show one table  
 1 - Show all tables  
 2 - Insert data  
 3 - Delete data  
 4 - Update data  
 5 - Search data  
 6 - Exit  
 ''')  
 choice = input('Choose an option: ')  
 if choice == '0':  
 View.list()  
 table = controller.validtable()  
 if table == 1:  
 Model.display\_query(Model.show\_table("client"),  
 ["phone\_number", "name", "check\_in\_date", "check\_out\_date"])  
 elif table == 2:  
 Model.display\_query(Model.show\_table("hotel\_management"), ["position", "name", "iban"])  
 elif table == 3:  
 Model.display\_query(Model.show\_table("room"),  
 ["room\_number", "number\_of\_places", "price", "is\_available", "type", "owner"])  
 elif table == 4:  
 Model.display\_query(Model.show\_table("services"), ["service\_name", "price", "responsible\_for"])  
 elif table == 5:  
 Model.display\_query(Model.show\_table("staff"),  
 ["policy\_number", "name", "position", "salary", "bonus\_or\_fine", "boss"])  
 elif table == 6:  
 Model.display\_query(Model.show\_table("booking"),  
 ["booking\_id", "room\_number", "booker"])  
 elif table == 7:  
 Model.display\_query(Model.show\_table("rental"),  
 ["order\_id", "room\_number", "tenant"])  
 elif table == 8:  
 Model.display\_query(Model.show\_table("ordering"),  
 ["order\_id", "service\_name", "client\_number", "is\_paid", "rating", "comment"])  
 elif table == 9:  
 Model.display\_query(Model.show\_table("providers"),  
 ["service\_code", "service\_name", "provider"])  
 elif choice == '1':  
 Model.display\_query(Model.show\_table("client"),  
 ["phone\_number", "name", "check\_in\_date", "check\_out\_date"])  
 Model.display\_query(Model.show\_table("hotel\_management"), ["position", "name", "iban"])  
 Model.display\_query(Model.show\_table("room"),  
 ["room\_number", "number\_of\_places", "price", "is\_available", "type", "owner"])  
 Model.display\_query(Model.show\_table("services"), ["service\_name", "price", "responsible\_for"])  
 Model.display\_query(Model.show\_table("staff"),  
 ["policy\_number", "name", "position", "salary", "bonus\_or\_fine", "boss"])  
 Model.display\_query(Model.show\_table("booking"),  
 ["booking\_id", "room\_number", "booker"])  
 Model.display\_query(Model.show\_table("rental"),  
 ["order\_id", "room\_number", "tenant"])  
 Model.display\_query(Model.show\_table("ordering"),  
 ["order\_id", "service\_name", "client\_number", "is\_paid", "rating", "comment"])  
 Model.display\_query(Model.show\_table("providers"),  
 ["service\_code", "service\_name", "provider"])  
  
 elif choice == '2':  
 end\_insert = False  
 while not end\_insert:  
 Model.insert()  
 incorrect = True  
 while incorrect:  
 answer = input('Continue working with insert? Enter Yes or No ')  
 if answer == 'No':  
 end\_insert = True  
 incorrect = False  
 elif answer == 'Yes':  
 incorrect = False  
 pass  
 else:  
 print('Please, enter Yes or No')  
 elif choice == '3':  
 end\_delete = False  
 while not end\_delete:  
 Model.delete()  
 incorrect = True  
 while incorrect:  
 answer = input('Continue working with delete? Enter Yes or No ')  
 if answer == 'No':  
 end\_delete = True  
 incorrect = False  
 elif answer == 'Yes':  
 incorrect = False  
 pass  
 else:  
 print('Please, enter Yes or No ')  
 elif choice == '4':  
 end\_update = False  
 while not end\_update:  
 Model.update()  
 incorrect = True  
 while incorrect:  
 answer = input('Continue working with update? Enter Yes or No ')  
 if answer == 'No':  
 end\_update = True  
 incorrect = False  
 elif answer == 'Yes':  
 incorrect = False  
 pass  
 else:  
 print('Please, enter Yes or No ')  
 elif choice == '5':  
 end\_search = False  
 View.search()  
 choice = int(input("Select what you want to search: "))  
 while not end\_search:  
 if choice == 1:  
 Model.display\_query(Model.search1(), ["Name", "Number", "Booking"])  
 elif choice == 2:  
 Model.display\_query(Model.search2(), ["Name", "Room", "Type"])  
 elif choice == 3:  
 Model.display\_query(Model.search3(), ["Price", "Service", "Responsible for"])  
 incorrect = True  
 while incorrect:  
 answer = input('Continue working with search? Enter Yes or No ')  
 if answer == 'No':  
 end\_search = True  
 incorrect = False  
 elif answer == 'Yes':  
 incorrect = False  
 pass  
 else:  
 print('Please, enter Yes or No ')  
 elif choice == '6':  
 break  
 else:  
 print('Please, enter valid number')

**main.py**

from view import Menu  
  
Menu.mainmenu()  
print("PostgreSQL connection is closed")

**launcher.py**

import controller  
from menu import Menu  
  
  
connection = controller.connection()  
cursor = connection.cursor()  
Menu.menu()  
cursor.close()  
connection.close()  
print("PostgreSQL connection is closed")  
5

**controller.py**

import psycopg2  
import re  
from view import View  
  
  
  
def connection():  
 return psycopg2.connect(  
 user="postgres",  
 password="ametyst7",  
 host="localhost",  
 port="5432",  
 database="hotel",  
 )  
  
  
def disconnect(connection):  
 connection.commit()  
 connection.close()  
  
def is\_digit(n):  
 try:  
 int(n)  
 return True  
 except ValueError:  
 return False  
  
  
def message(text):  
 return print(text)  
  
  
def validtable():  
 incorrect = True  
 while incorrect:  
 table = input('Choose table number => ')  
 if table.isdigit():  
 table = int(table)  
 if 1 <= table <= 9:  
 incorrect = False  
 else:  
 print('Incorrect input, try again.')  
 else:  
 print('Incorrect input, try again.')  
 return table  
  
  
def validate\_input\_items(name):  
 if name == "price":  
 value = View.enter\_item("price")  
 if value.isdecimal():  
 return value  
 if name == "salary":  
 value = View.enter\_item("salary")  
 if value.isdecimal():  
 return value  
 elif name == "iban":  
 value = View.enter\_item("iban")  
 if value.isalnum():  
 return value  
 elif name == "is\_available":  
 message("enter true or false")  
 value = View.enter\_item("is the room available or not")  
 #if (value == "true" or value == "false"):  
 return value  
 elif name == "is\_paid":  
 message("enter true or false")  
 value = View.enter\_item("is the order paid or not")  
 return value  
 elif name == "comment":  
 value = View.enter\_item("comment")  
 return value  
 elif name == "rating":  
 value = View.enter\_item("rate")  
 return value  
 elif name == "room":  
 value = View.enter\_item("room number")  
 if value.isdecimal():  
 return value  
 else:  
 message("enter 1 <= number <= 999")  
 validate\_input\_items("room number")  
 elif name == "order":  
 value = View.enter\_item("order number")  
 if value.isdecimal():  
 return value  
 else:  
 message("enter 1 <= number <= 999")  
 validate\_input\_items("room number")  
 elif name == "service\_code":  
 value = View.enter\_item("order number")  
 if value.isdecimal():  
 return value  
 else:  
 message("enter 1 <= number <= 999")  
 validate\_input\_items("service code")  
 elif name == "number":  
 value = View.enter\_item("number of places")  
 if value.isdecimal():  
 return value  
 elif name == "rent\_id":  
 value = View.enter\_item("rental/booking ID")  
 if (len(value) == 0):  
 return value  
 else:  
 message("enter 1000 <= number <= 9999")  
 validate\_input\_items("rental/booking ID")  
 elif name == "name":  
 value = View.enter\_item("name")  
 if (len(value) == 0):  
 return validate\_input\_items(name)  
 return value  
 elif name == "responsible":  
 value = View.enter\_item("responsible for the service's quality")  
 if (len(value) == 0):  
 return validate\_input\_items(name)  
 return value  
 elif name == "type":  
 value = View.enter\_item("type of room")  
 if (len(value) == 0):  
 return validate\_input\_items(name)  
 return value  
 elif name == "position":  
 value = View.enter\_item("position")  
 if (len(value) == 0):  
 return validate\_input\_items(name)  
 return value  
 elif name == "boss":  
 value = View.enter\_item("boss position")  
 if (len(value) == 0):  
 return validate\_input\_items(name)  
 return value  
 elif name == "owner":  
 value = View.enter\_item("room's owner")  
 if (len(value) == 0):  
 return validate\_input\_items(name)  
 return value  
 elif name == "policy":  
 value = View.enter\_item("policy number")  
 if (len(value) != 9):  
 print("Enter like in example: A12345678")  
 return validate\_input\_items(name)  
 return value  
 elif name == "check\_in":  
 value = View.enter\_item("check in date")  
 res = re.sub(r"[/\\,.]", "-", re.search(r"(\d+.\*?\d+.\*?\d+)", value).group(1))  
 return res  
 elif name == "check\_out":  
 value = View.enter\_item("check out date")  
 res = re.sub(r"[/\\.]", "-", re.search(r"(\d+.\*?\d+.\*?\d+)", value).group(1))  
 return res  
 elif name == "bonus":  
 value = View.enter\_item("bonus or fine value")  
 return value  
 elif name == "phone\_number":  
 value = View.enter\_item("phone number")  
 if value.isdigit() is False:  
 print("Enter like in example: 998889999")  
 return validate\_input\_items(name)  
 else:  
 if len(value) != 9:  
 print("Phone should have 9 numbers")  
 return validate\_input\_items(name)  
 return value

**view.py**

class View:  
 def \_\_init\_\_(self, table, records):  
 self.table = table  
 self.records = records  
  
 @staticmethod  
 def complete\_message(attribute, value, table, action):  
 print(f"The row with '{attribute}' = '{value}' in table '{table}' was {action} successfully.")  
  
 @staticmethod  
 def enter\_item(item):  
 data = input("Enter {}: ".format(item))  
 return data  
  
 @staticmethod  
 def list():  
 print('''  
 1 -> client  
 2 -> hotel management  
 3 -> room  
 4 -> services  
 5 -> staff  
 6 -> booking  
 7 -> rental  
 8 -> ordering  
 9 -> providers  
 ''')  
  
 @staticmethod  
 def columns(table):  
 if table == 1:  
 print('''  
 1 -> name  
 2 -> check in date  
 3 -> check out date  
 ''')  
 elif table == 2:  
 print('''  
 1 -> name  
 2 -> iban  
 ''')  
 elif table == 3:  
 print('''  
 1 -> number of places  
 2 -> price   
 3 -> is available  
 4 -> type   
 5 -> owner  
 ''')  
 elif table == 4:  
 print('''  
 1 -> price  
 2 -> responsible for  
 ''')  
 elif table == 5:  
 print('''  
 1 -> name  
 2 -> position  
 3 -> salary  
 4 -> bonus or fine  
 5 -> boss  
 ''')  
 elif table == 6:  
 print('''  
 1 -> room number  
 2 -> booker  
 ''')  
 elif table == 7:  
 print('''  
 1 -> room number  
 2 -> tenant  
 ''')  
 elif table == 8:  
 print('''  
 1 -> service name  
 2 -> client number  
 3 -> is paid  
 4 -> rating  
 5 -> comment  
 ''')  
 elif table == 9:  
 print('''  
 1 -> service name  
 2 -> provider  
 ''')  
  
 @staticmethod  
 def search():  
 print('''  
 1 -> Search phone number and booking orders of the client with name  
 2 -> Search rooms of the manager and types of them with the name  
 3 -> Search service name and who is responsible for it with service's price  
 ''')

**model.py**

import controller  
import time  
from view import View  
from pandas import DataFrame  
from tabulate import tabulate  
from numpy import array  
  
  
tables = {  
 1: 'client',  
 2: 'hotel management',  
 3: 'room',  
 4: 'services',  
 5: 'staff',  
 6: 'booking',  
 7: 'rental',  
 8: 'ordering',  
 9: 'providers',  
}  
  
  
class Model:  
 @staticmethod  
 def display\_query(rows, headers):  
 df = DataFrame([array(el) for el in rows], columns=array(headers))  
 print(tabulate(df, headers="keys", tablefmt="pretty", showindex=False))  
  
 @staticmethod  
 def show\_table(table):  
 connection = controller.connection()  
 cursor = connection.cursor()  
 print(f"Request: SELECT \* FROM {table}")  
 print(f"\033[1m {table} \033[0m")  
 cursor.execute(f"""SELECT \* from public.{table}""")  
 records = cursor.fetchall()  
 cursor.close()  
 return records  
  
 @staticmethod  
 def insert():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 go\_on = True  
 while go\_on:  
 View.list()  
 table = controller.validtable()  
 if table == 1:  
 phone\_number = controller.validate\_input\_items("phone\_number")  
 name = controller.validate\_input\_items("name")  
 check\_in = controller.validate\_input\_items("check\_in")  
 check\_out = controller.validate\_input\_items("check\_out")  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("Person already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "client" ("phone\_number", "name", "check\_in\_date", "check\_out\_date")   
 VALUES ('{phone\_number}', '{name}', '{check\_in}', '{check\_out}')""")  
 View.complete\_message("phone\_number", phone\_number, "client", "inserted")  
 go\_on = False  
 elif table == 2:  
 position = controller.validate\_input\_items("position")  
 name = controller.validate\_input\_items("name")  
 iban = controller.validate\_input\_items("iban")  
 cursor.execute(f"""select \* from public.hotel\_management where "position" = '{position}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "hotel\_management" ("position", "name", "iban")   
 VALUES ('{position}', '{name}', '{iban}')""")  
 View.complete\_message("position", position, "hotel\_management", "inserted")  
 go\_on = False  
 elif table == 3:  
 room = controller.validate\_input\_items("room")  
 number\_of = controller.validate\_input\_items("number")  
 price = controller.validate\_input\_items("price")  
 is\_available = controller.validate\_input\_items("is\_available")  
 type = controller.validate\_input\_items("type")  
 owner = controller.validate\_input\_items("owner")  
 cursor.execute(f"""select \* from public.room where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "room" ("room\_number", "number\_of\_places", "price", "is\_available", "type", "owner")   
 VALUES ('{room}', '{number\_of}', '{price}', '{is\_available}', '{type}', '{owner}')""")  
 View.complete\_message("room", room, "room", "inserted")  
 go\_on = False  
 elif table == 4:  
 name = controller.validate\_input\_items("name")  
 responsible = controller.validate\_input\_items("responsible")  
 price = controller.validate\_input\_items("price")  
 cursor.execute(f"""select \* from public.services where "service\_name" = '{name}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "services" ("service\_name", "price", "responsible\_for")   
 VALUES ('{name}', '{price}', '{responsible}')""")  
 View.complete\_message("phone\_id", id, "phone", "inserted")  
 go\_on = False  
 elif table == 5:  
 policy = controller.validate\_input\_items("policy")  
 name = controller.validate\_input\_items("name")  
 position = controller.validate\_input\_items("position")  
 salary = controller.validate\_input\_items("salary")  
 bonus = controller.validate\_input\_items("bonus")  
 boss = controller.validate\_input\_items("boss")  
 cursor.execute(f"""select \* from public.staff where "policy\_number" = '{policy}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""INSERT INTO "staff" ("policy\_number", "name", "position", "salary", "bonus\_or\_fine", "boss")   
 VALUES ('{policy}', '{name}', '{position}', '{salary}', '{bonus}', '{boss}')""")  
 View.complete\_message("policy\_number", policy, "staff", "inserted")  
 go\_on = False  
 elif table == 6:  
 booking = controller.validate\_input\_items("rent\_id")  
 booker = controller.validate\_input\_items("phone\_number")  
 room = controller.validate\_input\_items("room")  
 cursor.execute(f"""select \* from public.booking where "booking\_id" = '{booking}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{booker}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.room where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""INSERT INTO "booking" ("booking\_id", "booker", "room\_number")   
 VALUES ('{booking}', '{booker}', '{room}')""")  
 View.complete\_message("booking\_ID", booking, "booking", "inserted")  
 else:  
 controller.message("The room with this ID doesn't exist")  
 else:  
 controller.message("The client with this ID doesn't exist")  
 go\_on = False  
 elif table == 7:  
 rent = controller.validate\_input\_items("rent\_id")  
 tenant = controller.validate\_input\_items("phone\_number")  
 room = controller.validate\_input\_items("room")  
 cursor.execute(f"""select \* from public.rental where "order\_id" = '{rent}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{tenant}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.room where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""INSERT INTO "rental" ("order\_id", "tenant", "room\_number")   
 VALUES ('{rent}', '{tenant}', '{room}')""")  
 View.complete\_message("order\_ID", rent, "rental", "inserted")  
 else:  
 controller.message("The room with this ID doesn't exist")  
 else:  
 controller.message("The client with this ID doesn't exist")  
 go\_on = False  
 elif table == 8:  
 order = controller.validate\_input\_items("service\_code")  
 service = controller.validate\_input\_items("name")  
 client = controller.validate\_input\_items("phone\_number")  
 is\_paid = controller.validate\_input\_items("is\_paid")  
 rating = controller.validate\_input\_items("reting")  
 comment = controller.validate\_input\_items("comment")  
 cursor.execute(f"""select \* from public.ordering where "order\_id" = '{order}'""")  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""select \* from public.services where "service\_name" = '{service}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{client}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""INSERT INTO "ordering"   
 ("order\_id", "service\_name", "client\_number", "is\_paid", "rating", "comment")   
 VALUES ('{order}', '{service}', '{client}', '{is\_paid}', '{rating}', '{comment}')""")  
 View.complete\_message("order\_ID", order, "ordering", "inserted")  
 else:  
 controller.message("The client with this ID doesn't exist")  
 else:  
 controller.message("The service with this ID doesn't exist")  
 go\_on = False  
 elif table == 9:  
 code = controller.validate\_input\_items("sevice\_code")  
 service = controller.validate\_input\_items("name")  
 provider = controller.validate\_input\_items("policy")  
 cursor.execute(f"""select \* from public.providers where "service\_code" = '{code}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 controller.message("ID already exists")  
 else:  
 cursor.execute(f"""select \* from public.staff where "policy\_number" = '{provider}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.services where "service\_name" = '{service}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""INSERT INTO "providers" ("service\_code", "service\_name", "provider")   
 VALUES ('{code}', '{service}', '{provider}')""")  
 View.complete\_message("service\_code", code, "providers", "inserted")  
 else:  
 controller.message("The service with this ID doesn't exist")  
 else:  
 controller.message("The worker with this ID doesn't exist")  
 go\_on = False  
 else:  
 print('Please, enter valid value')  
 connection.commit()  
 cursor.close()  
 controller.disconnect(connection)  
  
 @staticmethod  
 def delete():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 go\_on = True  
 while go\_on:  
 View.list()  
 table = controller.validtable()  
 if table == 1:  
 phone\_number = controller.validate\_input\_items("phone\_number")  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.booking where "booker" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.booking WHERE "booker" = '{phone\_number}' """)  
 View.complete\_message("booker", phone\_number, "booking", "deleted")  
 cursor.execute(f"""select \* from public.rental where "tenant" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.rental WHERE "tenant" = '{phone\_number}' """)  
 View.complete\_message("tenant", phone\_number, "rental", "deleted")  
 cursor.execute(f"""select \* from public.ordering where "client\_number" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.ordering WHERE "client\_number" = '{phone\_number}' """)  
 View.complete\_message("client\_number", phone\_number, "ordering", "deleted")  
 cursor.execute(f"""DELETE FROM public.client WHERE "phone\_number" = '{phone\_number}'""")  
 View.complete\_message("phone\_number", phone\_number, "client", "deleted")  
 else:  
 controller.message("No person found")  
 go\_on = False  
 elif table == 2:  
 position = controller.validate\_input\_items("position")  
 cursor.execute(f"""select \* from public.hotel\_management where "position" = '{position}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.room where "owner" = '{position}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.room WHERE "owner" = '{position}'""")  
 View.complete\_message("owner", position, "room", "deleted")  
 cursor.execute(f"""select \* from public.staff where "boss" = '{position}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.staff WHERE "boss" = '{position}'""")  
 View.complete\_message("boss", position, "staff", "deleted")  
 cursor.execute(f"""DELETE FROM public.hotel\_management WHERE "position" = '{position}'""")  
 View.complete\_message("position", position, "hotel\_management", "deleted")  
 else:  
 controller.message("No person found")  
 go\_on = False  
 elif table == 3:  
 room = controller.validate\_input\_items("room")  
 cursor.execute(f"""select \* from public.room where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.booking where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.booking WHERE "room\_number" = '{room}'""")  
 View.complete\_message("roon\_number", room, "booking", "deleted")  
 cursor.execute(f"""select \* from public.rental where "room\_number" = '{room}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.rental WHERE "room\_number" = '{room}'""")  
 View.complete\_message("room\_number", room, "rental", "deleted")  
 cursor.execute(f"""DELETE FROM public.services WHERE "service\_name" = '{room}'""")  
 View.complete\_message("room\_number", room, "room", "deleted")  
 else:  
 controller.message("No room found")  
 go\_on = False  
 elif table == 4:  
 name = controller.validate\_input\_items("name")  
 cursor.execute(f"""select \* from public.services where "service\_name" = '{name}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.ordering where "service\_name" = '{name}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.ordering WHERE "service\_name" = '{name}'""")  
 View.complete\_message("service\_name", name, "ordering", "deleted")  
 cursor.execute(f"""select \* from public.providers where "service\_name" = '{name}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.providers WHERE "service\_name" = '{name}'""")  
 View.complete\_message("service\_name", name, "providers", "deleted")  
 cursor.execute(f"""DELETE FROM public.services WHERE "service\_name" = '{name}'""")  
 View.complete\_message("service\_name", name, "services", "deleted")  
 else:  
 controller.message("No service found")  
 go\_on = False  
 elif table == 5:  
 policy = controller.validate\_input\_items("policy")  
 cursor.execute(f"""select \* from public.staff where "policy\_number" = '{policy}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""select \* from public.providers where "service\_name" = '{policy}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.providers WHERE "service\_name" = '{policy}'""")  
 View.complete\_message("provider", policy, "providers", "deleted")  
 cursor.execute(f"""DELETE FROM public.staff WHERE "policy\_number" = '{policy}'""")  
 View.complete\_message("policy\_number", policy, "staff", "deleted")  
 else:  
 controller.message("No person found")  
 go\_on = False  
 elif table == 6:  
 rent\_id = controller.validate\_input\_items("rent\_id")  
 cursor.execute(f"""select \* from public.booking where "booking\_id" = '{rent\_id}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.booking WHERE "booking\_id" = '{rent\_id}'""")  
 View.complete\_message("booking\_id", rent\_id, "booking", "deleted")  
 else:  
 controller.message("No booking found")  
 go\_on = False  
 elif table == 7:  
 rent\_id = controller.validate\_input\_items("rent\_id")  
 cursor.execute(f"""select \* from public.rental where "order\_id" = '{rent\_id}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.rental WHERE "order\_id" = '{rent\_id}'""")  
 View.complete\_message("order\_id", rent\_id, "rental", "deleted")  
 else:  
 controller.message("No rent found")  
 go\_on = False  
 elif table == 8:  
 order\_id = controller.validate\_input\_items("order")  
 cursor.execute(f"""select \* from public.ordering where "order\_id" = '{order\_id}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.ordering WHERE "order\_id" = '{order\_id}'""")  
 View.complete\_message("schedule\_id", id, "schedule", "deleted")  
 else:  
 controller.message("No order found")  
 go\_on = False  
 elif table == 9:  
 code = controller.validate\_input\_items("service\_code")  
 cursor.execute(f"""select \* from public.providers where "service\_code" = '{code}' """)  
 records = cursor.fetchone()  
 if records is not None:  
 cursor.execute(f"""DELETE FROM public.providers WHERE "service\_code" = '{code}'""")  
 View.complete\_message("service\_code", code, "providers", "deleted")  
 else:  
 controller.message("No service found")  
 go\_on = False  
 else:  
 print("Please, enter valid number")  
 connection.commit()  
 cursor.close()  
 controller.disconnect(connection)  
 pass  
  
 @staticmethod  
 def update():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 go\_on = True  
 while go\_on:  
 View.list()  
 table = controller.validtable()  
 if table == 1:  
 phone\_number = controller.validate\_input\_items("phone\_number")  
 cursor.execute(f"""select \* from public.client where "phone\_number" = '{phone\_number}' """)  
 records = cursor.fetchone()  
 continue\_update = True  
 if records is not None:  
 View.columns(1)  
 while continue\_update:  
 attr = input("Choose a number of column to update")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("check\_in")  
 attribute = "check\_in\_date"  
 continue\_update = False  
 elif attr == '3':  
 value = controller.validate\_input\_items("check\_out")  
 attribute = "check\_out\_date"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number")  
 cursor.execute(f"""UPDATE public.client set {attribute} = '{value}' where "phone\_number" = '{phone\_number}' """)  
 View.complete\_message("phone\_nomber", phone\_number, "client", "updated")  
 go\_on = False  
 else:  
 controller.message("No client with this ID found")  
 elif table == 2:  
 position = controller.validate\_input\_items("position")  
 cursor.execute(f"""select \* from public.hotel\_management where "position" = {position} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(2)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("iban")  
 attribute = "iban"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number")  
 cursor.execute(f"""UPDATE "hotel\_management" set {attribute} = '{value}' where "position" = '{id}' """)  
 View.complete\_message("position", position, "hotel\_management", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No boss with this ID found")  
 elif table == 3:  
 room = controller.validate\_input\_items("room")  
 cursor.execute(f"""select \* from public.room where "room\_number" = {room} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(3)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("number")  
 attribute = "number\_of\_places"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("price")  
 attribute = "price"  
 continue\_update = False  
 elif attr == '3':  
 value = controller.validate\_input\_items("is\_available")  
 attribute = "is\_available"  
 continue\_update = False  
 elif attr == '4':  
 value = controller.validate\_input\_items("type")  
 attribute = "type"  
 continue\_update = False  
 elif attr == '5':  
 value = controller.validate\_input\_items("owner")  
 attribute = "owner"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number")  
 cursor.execute(f"""UPDATE "room" set {attribute} = '{value}' where "room\_number" = '{room}' """)  
 View.complete\_message("room\_number", room, "room", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No room with this ID found")  
 elif table == 4:  
 name = controller.validate\_input\_items("name")  
 cursor.execute(f"""select \* from public.services where "service\_name" = {name} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(4)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("price")  
 attribute = "price"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("responsible")  
 attribute = "responsible\_for"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "services" set {attribute} = '{value}' where "service\_name" = '{name}' """)  
 View.complete\_message("service\_name", name, "services", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No service with this ID found")  
 elif table == 5:  
 policy = controller.validate\_input\_items("policy")  
 cursor.execute(f"""select \* from public.staff where "policy\_number" = {policy} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(5)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update ")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("position")  
 attribute = "position"  
 continue\_update = False  
 elif attr == '3':  
 value = controller.validate\_input\_items("salary")  
 attribute = "salary"  
 continue\_update = False  
 elif attr == '4':  
 value = controller.validate\_input\_items("bonus")  
 attribute = "bonus\_or\_fine"  
 continue\_update = False  
 elif attr == '5':  
 value = controller.validate\_input\_items("boss")  
 attribute = "boss"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number")  
 cursor.execute(f"""UPDATE "staff" set {attribute} = '{value}' where "policy\_number" = '{policy}' """)  
 View.complete\_message("policy\_number", policy, "staff", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No worker with this ID found")  
 elif table == 6:  
 rent\_id = controller.validate\_input\_items("rent\_id")  
 cursor.execute(f"""select \* from public.booking where "booking\_id" = {rent\_id} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(6)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("room")  
 attribute = "room\_number"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("phone\_number")  
 attribute = "booker"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "booking" set {attribute} = '{value}' where "booking\_id" = '{rent\_id}' """)  
 View.complete\_message("booking\_id", rent\_id, "booking", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No booking with this ID found")  
 elif table == 7:  
 rent\_id = controller.validate\_input\_items("rent\_id")  
 cursor.execute(f"""select \* from public.rental where "order\_id" = {rent\_id} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(7)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("room")  
 attribute = "room\_number"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("phone\_number")  
 attribute = "tenant"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "rental" set {attribute} = '{value}' where "order\_id" = '{rent\_id}' """)  
 View.complete\_message("order\_id", rent\_id, "rental", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No rent with this ID found")  
 elif table == 8:  
 id = controller.validate\_input\_items("order")  
 cursor.execute(f"""select \* from public.ordering where "order\_id" = {id} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(8)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "service\_name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("phone\_number")  
 attribute = "client\_number"  
 continue\_update = False  
 elif attr == '3':  
 value = controller.validate\_input\_items("is\_paid")  
 attribute = "is\_paid"  
 continue\_update = False  
 elif attr == '4':  
 value = controller.validate\_input\_items("rating")  
 attribute = "rating"  
 continue\_update = False  
 elif attr == '5':  
 value = controller.validate\_input\_items("comment")  
 attribute = "comment"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "ordering" set {attribute} = '{value}' where "order\_id" = '{id}' """)  
 View.complete\_message("order\_id", id, "ordering", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No order with this ID found")  
 elif table == 9:  
 id = controller.validate\_input\_items("order")  
 cursor.execute(f"""select \* from public.providers where "service\_code" = {id} """)  
 records = cursor.fetchone()  
 if records is not None:  
 View.columns(9)  
 continue\_update = True  
 while continue\_update:  
 attr = input("Choose a number of column to update: ")  
 if attr == '1':  
 value = controller.validate\_input\_items("name")  
 attribute = "service\_name"  
 continue\_update = False  
 elif attr == '2':  
 value = controller.validate\_input\_items("policy")  
 attribute = "provider"  
 continue\_update = False  
 else:  
 controller.message("Please, enter valid number: ")  
 cursor.execute(f"""UPDATE "providers" set {attribute} = '{value}' where "service\_code" = '{id}' """)  
 View.complete\_message("service\_code", id, "providers", "updated")  
 go\_on = False  
 pass  
 else:  
 controller.message("No order with this ID found")  
 else:  
 controller.message("Please, enter valid number")  
 connection.commit()  
 cursor.close()  
 controller.disconnect(connection)  
 pass  
  
 @staticmethod  
 def search1():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 name = controller.validate\_input\_items("name")  
 cursor.execute(f"""select \* from public.client where "name" = '{name}' """)  
 records = cursor.fetchone()  
 #if records is not None:  
 search = f"""select "name", "phone\_number", "booking\_id" from (select c."name", c."phone\_number", b."booking\_id"  
 from "client" c left join "booking" b on b."booker" = c."phone\_number"  
 where c."name" LIKE '{name}' group by c."name", c."phone\_number", b."booking\_id") as foo"""  
 start = int(time.time() \* 1000)  
 cursor.execute(search)  
 print("--- Time of search = {} ms ---".format(int((time.time() \* 1000) - start)))  
 records = cursor.fetchall()  
 cursor.close()  
 return records  
  
 @staticmethod  
 def search2():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 name = controller.validate\_input\_items("name")  
 cursor.execute(f"""select \* from public.hotel\_management where "name" = '{name}' """)  
 records = cursor.fetchone()  
 search = f"""select "name", "room\_number", "type" from (select h."name", r."room\_number", r."type"  
 from "hotel\_management" h left join "room" r on r."owner" = h."position"  
 where h."name" LIKE '{name}' group by h."name", r."room\_number", r."type") as foo"""  
 start = int(time.time() \* 1000)  
 cursor.execute(search)  
 print("--- Time of search = {} ms ---".format(int((time.time() \* 1000) - start)))  
 records = cursor.fetchall()  
 cursor.close()  
 return records  
  
  
 @staticmethod  
 def search3():  
 connection = controller.connection()  
 cursor = connection.cursor()  
 price = controller.validate\_input\_items("price")  
 cursor.execute(f"""select \* from public.services where "price" = '{price}' """)  
 records = cursor.fetchone()  
 search = f"""select "price", "service\_name", "name" from (select s."price", s."service\_name", h."name"  
 from "services" s left join "hotel\_management" h on h."position" = s."responsible\_for"  
 where s."price" > '{price}' group by s."price", s."service\_name", h."name") as foo"""  
 start = int(time.time() \* 1000)  
 cursor.execute(search)  
 print("--- Time of search = {} ms ---".format(int((time.time() \* 1000) - start)))  
 records = cursor.fetchall()  
 cursor.close()  
 return records