НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Розрахунково-графічна робота

з дисципліни

«Бази даних та засоби управління»

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

Виконала: студентка групи КВ-11
Петрук Ольга

Telegram: @olyaaaaaaaaaaaaaaaaa

Київ – 2023

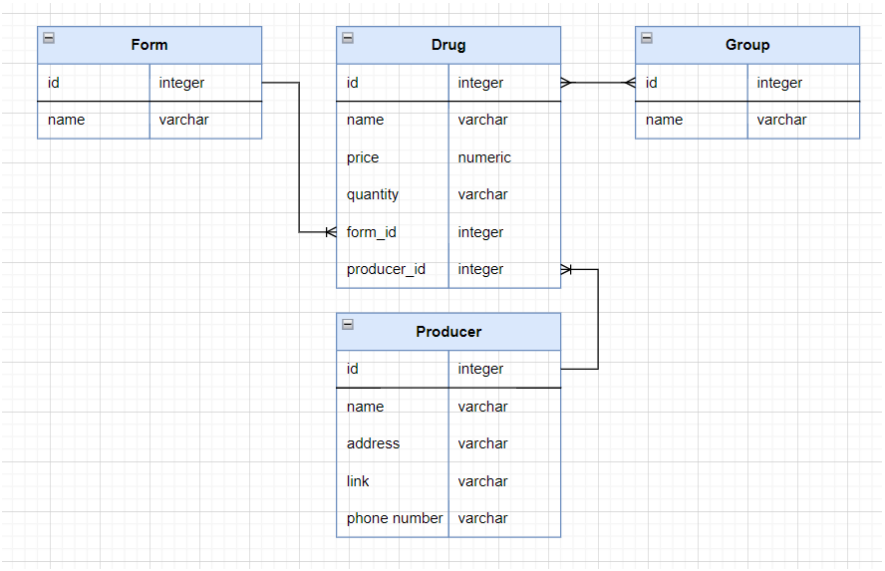
Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції перегляду, 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).

URL репозиторію з вихідним кодом: https://github.com/kebabgirl/db

Скріншот розробленої моделі «сутність-зв’язок» предметної галузі «Довідник медичних препаратів»:



Сутності з описом призначення:

Предметна галузь «Електронний довідник медичних препаратів» включає в себе 4 сутності, кожна сутність містить декілька атрибутів:

1. Drug (drug_id, name, price, quantity, form_id, producer_id).
2. Group (group_id, name).
3. Form (form_id, name).
4. Producer (producer_id, name, address, link, phone_number).

Сутність Drug описує медичні препарати, які є в довіднику. Кожен препарат містить свій унікальний id, назву, ціну, обсяг, id своєї форми та id свого виробника.

Сутність Group відповідає за групи медичних препаратів. У кожної групи є свій унікальний id та назва.

Сутність Form відповідає за форми медичних препаратів. У кожної форми є свій унікальний id та назва.

Сутність Producer описує виробників медичних препаратів. У кожного виробника є свій унікальний id, назва, адреса, посилання та номер телефону. У виробника може бути багато медичних препаратів, але для кожного препарату він є єдиним.

Схема бази даних у графічному вигляді:

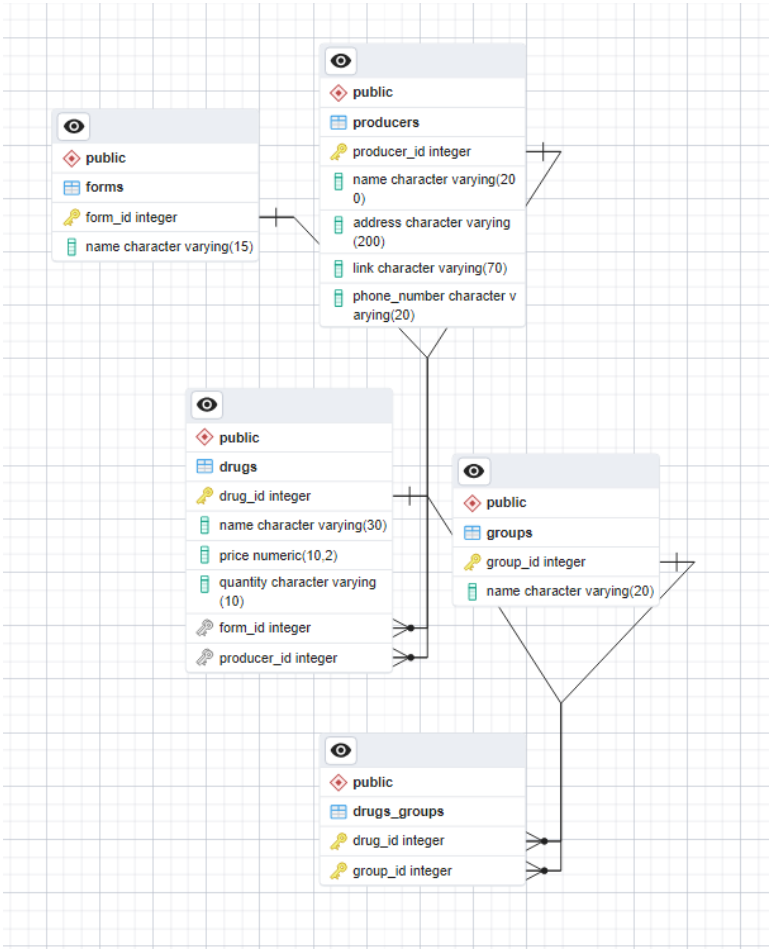


Схема меню користувача з описом функціональності кожного пункту

```
Меню:
1. Додати рядок
2. Генерування «рандомізованих» даних (тільки для таблиці "Drugs")
3. Показати таблицю
4. Редагувати рядок
5. Видалити рядок
6. Пошук
7. Вихід
Виберіть пункт:
```

Рис. 1 – скріншот меню програми

```
Таблиці:
1. Drugs
2. Producers
3. Forms
4. Groups
5. Drugs Groups
6. Повернутися до меню
Оберіть потрібну таблицю:
```

Рис. 2 – скріншот списку таблиць

Меню складається з 7 пунктів (Рис. 1):

1. **Додавання рядка.**
   Використовується при потребі записати дані в таблицю. Після вибору цього пункту, потрібно обрати таблицю (Рис. 2), для якої буде виконана ця операція. Після вибору таблиці, користувач має ввести дані для кожного атрибуту таблиці, щоб додати новий рядок. При

успішному запиті, користувач отримає повідомлення: "Drug added successfully!", а в разі будь-яких помилок - "Something went wrong: {e}" з текстом помилки.

2. **Генерування «рандомізованих» даних.**
   За умовою завдання, потрібно застосувати до 1-2 таблиць, тому було обрано таблицю drugs. Ця опція зроблена для додавання певної кількості препаратів до головної таблиці «drugs». Для додавання згенерованих «рагдомізованих» препаратів, потрібно ввести число полів, яке ми хочемо додати. При успішному запиті, користувач отримає повідомлення: "Random fields added successfully!", а в разі будь-яких помилок - "Something went wrong: {e}" з текстом помилки.
3. **Показ таблиці.**
   Створений для показу таблиці. Перед виведенням, користувач обирає, яку саме таблицю потрібно вивести (Рис. 2). Після цього на екрані виводяться всі поля обраної таблиці БД.
4. **Редагування рядка.**
   Використовується для редагування полів по id у таблицях. Спочатку потрібно обрати, для якої таблиці буде відбуватися редагування (Рис. 2),. Після чого, користувач вводить id поля, яке потрібно змінити. Залишилось ввести нові дані для кожного атрибуту таблиці. При успішному запиті, користувач отримає повідомлення: "Drug updated successfully!", а в разі будь-яких помилок - "Something went wrong: {e}" з текстом помилки.
5. **Видалення рядка.**
   Опція для видалення рядку по id у таблицях. Спочатку потрібно обрати, для якої таблиці буде відбуватися видалення рядка (Рис. 2),. Після чого, користувач вводить id рядка, який потрібно видалити. При успішному запиті, користувач отримає повідомлення: "Drug deleted successfully!", а в разі будь-яких помилок - "Something went wrong: {e}" з текстом помилки.
6. **Пошук.**
   Опція, створена для пошуку за атрибутами з декількох таблиць. Пропонується 4 варіанти вибору:

```
Виберіть пункт: 6

Пошук:
1. Відповідність препарату групі
2. ТОП найдешевших препаратів
3. Кількість препаратів у кожній групі
4. Повернутися до меню
Оберіть щось:
```

3 основних та ще 1 для повернення до меню. При виборі *першого пункту*, буде показана таблиця, яка містить інформацію про препарати ("drugs") та групи ("groups"), до яких вони відносяться, при цьому враховуючи випадки, коли препарат не належить до жодної групи. При виборі *другого пункту,* користувач повинен ввести число, яке буде означати максимальну ціну препарату для сортування. У результаті буде показана таблиця з назвою препарату, формою препарату та ціною, де ціна менше введеного користувачем значення, відсортована за ціною у зростаючому порядку. При виборі *третього пункту,* буде показана таблиця з назвою групи та кількістю препаратів у кожній групі, включаючи групи, в яких кількість препаратів може бути 0.

7. **Вихід.**
   Завершує виконання програми.

Для розробки було використано:

Середовище для відлагодження SQL-запитів до бази даних – *PgAdmin4.*

Мова програмування – *Python 3.10.*

Середовище розробки програмного забезпечення – *PyCharm Community Edition.*

Бібліотека взаємодії з *PostgreSQL - psycopg2.*

Бібліотека для замірів часу: *time.*

Завдання 1

(для демонстрації будуть вибрані по 1-2 будь-яких таблиці для кожного пункту)

Внесення даних

Таблиця “drugs” до:

	drug_id [PK] integer	name character varying (30)	price numeric (10,2)	quantity character varying (10)	form_id integer	producer_id integer
1	100016	Drug2	1.00	1p	1	1
2	100015	IP	58.65	82	2	1
3	100014	YK	93.70	32	7	1
4	100013	KQ	1.66	96	5	4
5	100012	KS	99.73	81	3	3
6	100011	CL	36.26	13	6	2
7	100010	VH	0.15	29	2	3
8	100009	EH	29.75	93	7	2
9	100008	WP	18.75	9	6	4
10	100007	GT	77.05	79	2	3

```
Меню:
1. Додати рядок
2. Генерування «рандомізованих» даних (тільки для таблиці "Drugs")
3. Показати таблицю
4. Редагувати рядок
5. Видалити рядок
6. Пошук
7. Вихід
Виберіть пункт: 1

Таблиці:
1. Drugs
2. Producers
3. Forms
4. Groups
5. Drugs Groups
6. Повернутися до меню
Оберіть потрібну таблицю: 1

Adding drug:
Enter drug name: drug_name
Enter drug price: 555
Enter drug quantity: 555ml
Enter drug form_id: 5
Enter drug producer_id: 5
Drug added successfully!
```

Таблиця “drugs” після:

	drug_id [PK] integer	name character varying (30)	price numeric (10,2)	quantity character varying (10)	form_id integer	producer_id integer
1	100017	drug_name	555.00	555ml	5	5
2	100016	Drug2	1.00	1p	1	1
3	100015	IP	58.65	82	2	1
4	100014	YK	93.70	32	7	1
5	100013	KQ	1.66	96	5	4
6	100012	KS	99.73	81	3	3
7	100011	CL	36.26	13	6	2
8	100010	VH	0.15	29	2	3
9	100009	EH	29.75	93	7	2
10	100008	WP	18.75	9	6	4

Таблиця “forms” до:

	form_id [PK] integer	name character varying (15)
1	1	таблетки
2	2	сироп
3	3	порошок
4	4	капсули
5	5	гель
6	6	розчин
7	7	краплі
8	9	form2

```
Меню:
1. Додати рядок
2. Генерування «рандомізованих» даних (тільки для таблиці "Drugs")
3. Показати таблицю
4. Редагувати рядок
5. Видалити рядок
6. Пошук
7. Вихід
Виберіть пункт: 1

Таблиці:
1. Drugs
2. Producers
3. Forms
4. Groups
5. Drugs Groups
6. Повернутися до меню
Оберіть потрібну таблицю: 3

Adding form:
Enter form name: form_name
Form added successfully!
```

Таблиця “forms” після:

	form_id [PK] integer	name character varying (15)
1	1	таблетки
2	2	сироп
3	3	порошок
4	4	капсули
5	5	гель
6	6	розчин
7	7	краплі
8	9	form2
9	10	form_name

Перегляд даних

Таблиця “producers”:

```
Меню:
1. Додати рядок
2. Генерування «рандомізованих» даних (тільки для таблиці "Drugs")
3. Показати таблицю
4. Редагувати рядок
5. Видалити рядок
6. Пошук
7. Вихід
Виберіть пункт: 3

Таблиці:
1. Drugs
2. Producers
3. Forms
4. Groups
5. Drugs Groups
6. Повернутися до меню
Оберіть потрібну таблицю: 2

Producers:
ID: 1, Name: Реккітт Бенкізер Хелскер Інтернешнл Лімітед, Address: Тейн Роуд, Ноттінгем, Ноттінгемшир, NG90 2DB, Велика Британія., Link: None , Phone number: +44 20 1234 5678
ID: 2, Name: АТ «КИЇВСЬКИЙ ВІТАМІННИЙ ЗАВОД», Address: 04073, Україна, м. Київ, вул. Копилівська, 38, Link: www.vitamin.com.ua , Phone number: None
ID: 3, Name: Товариство з обмеженою відповідальністю «ФАРМЕКС ГРУП», Address: Україна, 08301, Київська обл., місто Бориспіль, вулиця Шевченка, будинок 100., Link: None , Phone
ID: 4, Name: КРКА, д.д., Ново место, Словенія, Address: Шмар'ешка цеста 6, 8501 Ново место, Словенія, Link: None , Phone number: None
ID: 5, Name: ГСК Консьюмер Хелскер САРЛ / GSK Consumer Healthcare SARL, Address: Рут де Летра, Ніон, 1260, Швейцарія/Route de l'Etraz, Nyon, 1260, Switzerland, Link: None , Pho
```

Таблиця "groups":

```
1. Додати рядок
2. Генерування «рандомізованих» даних (тільки для таблиці "Drugs")
3. Показати таблицю
4. Редагувати рядок
5. Видалити рядок
6. Пошук
7. Вихід
Виберіть пункт: 3

Таблиці:
1. Drugs
2. Producers
3. Forms
4. Groups
5. Drugs Groups
6. Повернутися до меню
Оберіть потрібну таблицю: 4

Groups:
ID: 1, Name: вітаміни
ID: 2, Name: антибіотики
ID: 3, Name: противірусні
ID: 4, Name: знеболюючі
ID: 5, Name: очні
ID: 6, Name: протизапальні
ID: 7, Name: жарознижуючі
ID: 8, Name: від кашлю
ID: 9, Name: від нежиті
ID: 11, Name: Group1
```

Редагування даних

Таблиця "drugs" до:

	drug_id [PK] integer	name character varying (30)	price numeric (10,2)	quantity character varying (10)	form_id integer	producer_id integer
1	100017	drug_name	555.00	555ml	5	5
2	100016	Drug2	1.00	1p	1	1
3	100015	IP	58.65	82	2	1
4	100014	YK	93.70	32	7	1
5	100013	KQ	1.66	96	5	4

```
Меню:
1. Додати рядок
2. Генерування «рандомізованих» даних (тільки для таблиці "Drugs")
3. Показати таблицю
4. Редагувати рядок
5. Видалити рядок
6. Пошук
7. Вихід
Виберіть пункт: 4

Таблиці:
1. Drugs
2. Producers
3. Forms
4. Groups
5. Drugs Groups
6. Повернутися до меню
Оберіть потрібну таблицю: 1

Updating drug:
Enter drug ID: 100017
Enter drug name: drug_name2
Enter drug price: 333
Enter drug quantity: 333ml
Enter drug form_id: 3
Enter drug producer_id: 3
Drug updated successfully!
```

Таблиця “drugs” після:

	drug_id [PK] integer	name character varying (30)	price numeric (10,2)	quantity character varying (10)	form_id integer	producer_id integer
1	100017	drug_name2	333.00	333ml	3	3
2	100016	Drug2	1.00	1p	1	1
3	100015	IP	58.65	82	2	1
4	100014	YK	93.70	32	7	1
5	100013	KQ	1.66	96	5	4

Видалення даних

Таблиця “forms” до:

	form_id [PK] integer	name character varying (15)
1	1	таблетки
2	2	сироп
3	3	порошок
4	4	капсули
5	5	гель
6	6	розчин
7	7	краплі
8	9	form2
9	10	form_name

```
Меню:
1. Додати рядок
2. Генерування «рандомізованих» даних (тільки для таблиці "Drugs")
3. Показати таблицю
4. Редагувати рядок
5. Видалити рядок
6. Пошук
7. Вихід
Виберіть пункт: 5

Таблиці:
1. Drugs
2. Producers
3. Forms
4. Groups
5. Drugs Groups
6. Повернутися до меню
Оберіть потрібну таблицю: 3

Deleting form:
Enter form ID: 10
Form deleted successfully!
```

Таблиця “forms” після:

	form_id [PK] integer	name character varying (15)
1	1	таблетки
2	2	сироп
3	3	порошок
4	4	капсули
5	5	гель
6	6	розчин
7	7	краплі
8	9	form2

Завдання 2

Генерування «рандомізованих» даних

```
Меню:
1. Додати рядок
2. Генерування «рандомізованих» даних (тільки для таблиці "Drugs")
3. Показати таблицю
4. Редагувати рядок
5. Видалити рядок
6. Пошук
7. Вихід
Виберіть пункт: 2

Таблиці:
1. Drugs
2. Producers
3. Forms
4. Groups
5. Drugs Groups
6. Повернутися до меню
Оберіть потрібну таблицю: 1

Adding random drugs:
Enter the number: 100000
```

Для генерування «рандомізованих» даних була вибрана таблиця “drugs”, в яку було додано 100000 полів:

	drug_id [PK] integer	name character varying (30)	price numeric (10,2)	quantity character varying (10)	form_id integer	producer_id integer
1	1	Нурофен	100.00	12 шт	1	1
2	2	Вітамін С зі смак. апельс.	12.99	10 шт	1	2
3	3	Парацетамол	12.75	10 шт	4	3
4	4	Новірин	150.00	20 шт	1	2
5	5	Гербіон	220.00	150 мл	2	4
6	6	Отривін	155.00	10 мл	7	5
7	7	Стрепсилс	200.25	20шт	1	1
8	8	Драг	99.00	99	5	5
9	12	PQ	11.09	9	1	4
10	13	AT	73.08	43	2	3
11	14	WL	62.99	14	4	1
12	15	TP	92.98	40	1	5
13	16	NI	93.19	70	2	3
14	17	KN	32.65	57	5	1
15	18	UU	53.78	97	6	4
16	19	IE	64.86	43	3	3
17	20	UD	27.40	12	6	2
18	21	YJ	79.30	86	4	3
19	22	AH	79.71	54	7	2
20	23	WG	90.56	79	2	1
21	24	AW	0.01	64	5	4
22	25	GI	50.33	94	6	1
23	26	HK	90.37	17	4	1
24	27	CE	33.74	37	5	5
25	28	FW	12.53	27	5	1
26	29	NA	89.88	71	6	5
27	30	KW	10.41	79	3	4
28	31	PK	46.80	65	1	1
29	32	OS	75.00	79	2	5
30	33	IM	8.04	85	7	2
31	34	DG	52.10	82	1	5
32	35	SF	23.80	13	7	3
33	36	UT	17.29	68	4	1
34	37	NV	15.81	56	3	3
35	38	MQ	7.65	57	7	2
36	39	HC	34.94	16	1	5
37	40	KM	21.40	92	5	1
38	41	PV	19.60	60	1	2
39	42	VG	84.15	8	7	3
40	43	DR	39.93	64	1	4
41	44	SW	59.15	80	3	4
42	45	PY	76.59	55	3	3
46	49	KW	10.91	90	5	3
47	50	AV	9.47	43	3	1
48	51	FD	12.09	84	6	5
49	52	SG	49.82	46	7	4
50	53	YA	50.02	13	6	1
51	54	FY	10.42	93	3	4
52	55	RL	84.52	38	2	3
53	56	PC	83.88	73	1	4
54	57	VV	58.02	90	1	2
55	58	UG	65.33	57	3	5
56	59	JQ	18.95	45	2	3
57	60	VV	73.32	92	4	2
58	61	AV	20.76	70	4	3
59	62	LP	71.73	97	6	3
60	63	FF	57.71	82	2	4
61	64	EI	26.16	58	5	3
62	65	KJ	73.81	80	2	4
63	66	UM	61.93	33	7	3
64	67	XM	19.85	89	4	2
65	68	YA	68.39	89	5	1
66	69	HH	60.96	53	7	3
67	70	SM	37.97	13	5	5
68	71	VX	82.53	97	1	4
69	72	BQ	30.89	85	7	5
70	73	NW	66.37	28	3	4
71	74	TY	1.61	46	6	5
72	75	VX	24.34	32	3	2
73	76	UT	10.25	29	5	3
74	77	EX	78.36	15	3	5
75	78	NS	16.91	49	6	4
76	79	ER	62.01	48	4	2
77	80	FO	26.67	12	7	5
78	81	WD	17.14	92	6	3
79	82	YJ	22.57	92	5	5
80	83	WN	53.62	87	6	2
81	84	DX	11.42	12	6	3
82	85	VA	92.69	87	6	5
83	86	OI	75.49	91	7	5
84	87	JX	92.08	91	4	1
85	88	OG	0.14	11	1	1
86	89	GQ	51.75	43	7	3

Для реалізації генерування «рандомізованих» даних був використаний такий SQL-запит:

```
INSERT INTO drugs (name, price, quantity, form_id, producer_id)
SELECT
  chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int),
  random() * 100,
  trunc(random() * 100),
  trunc(random() * 7) + 1,
  trunc(random() * 5) + 1
FROM generate_series(1, %s);
```

Цей запит вставляє випадкові дані в стовпці "name", "price", "quantity", "form_id" і "producer_id" таблиці "drugs". У функції generate_series, %s вказує на параметр, який необхідно передати для визначення кількості вставлених записів.

Тут:

*chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int)*: Генерує випадкову назву для препарату, що складається з 2 випадкових букв латинського алфавіту.

*random() * 100*: Генерує випадкову ціну для препарату в межах від 0 до 100.

trunc(random() * 100): Генерує випадкову кількість одиниць препарату в межах від 0 до 99 і округляє його до цілого числа.

*trunc(random() * 7) + 1*: Генерує випадковий ідентифікатор форми препарату від 1 до 7.

*trunc(random() * 5) + 1*: Генерує випадковий ідентифікатор виробника препарату від 1 до 5.

Завдання 3

Пошук даних

```
Пошук:
1. Відповідність препарату групі
2. ТОП найдешевших препаратів
3. Кількість препаратів у кожній групі
4. Повернутися до меню
Обреріть щось:
```

«Пошук» має 3 основних опції:

1. **Відповідність препарату групі.**
   При виборі цього пункту, буде сформована та показана таблиця, яка містить інформацію про препарати ("drugs") та групи ("groups"), до яких вони відносяться, при цьому враховуючи випадки, коли препарат не належить до жодної групи.

```
Drug ID: 100006, Drug name: RU, Group name: None
Drug ID: 100007, Drug name: GT, Group name: None
Drug ID: 100008, Drug name: WP, Group name: None
Drug ID: 100009, Drug name: EH, Group name: None
Drug ID: 100010, Drug name: VH, Group name: None
Drug ID: 100011, Drug name: CL, Group name: None
Drug ID: 100012, Drug name: KS, Group name: None
Drug ID: 100013, Drug name: KQ, Group name: None
Drug ID: 100014, Drug name: YK, Group name: None
Drug ID: 100015, Drug name: IP, Group name: None
Drug ID: 100016, Drug name: Drug2, Group name: None
Час виконання: 474.73 мс
```

   Так як в таблиці «drugs» існує більше 100 тис. полів, то важко побачити коректну роботу цього пошуку. Тож зробимо деякі модифікації запиту, а саме додамо LIMIT 10, щоб показати перші 10 полів:

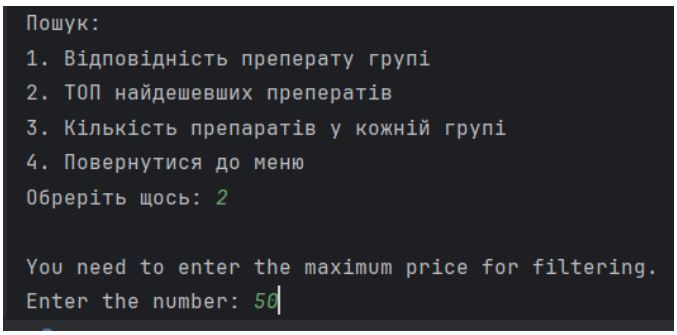
```
Drugs-Groups:
Drug ID: 1, Drug name: Нурофен, Group name: антибіотики
Drug ID: 1, Drug name: Нурофен, Group name: знеболюючі
Drug ID: 1, Drug name: Нурофен, Group name: жарознижуючі
Drug ID: 2, Drug name: Вітамін С зі смак. апельс., Group name: вітаміни
Drug ID: 3, Drug name: Парацетамол, Group name: знеболюючі
Drug ID: 3, Drug name: Парацетамол, Group name: жарознижуючі
Drug ID: 4, Drug name: Новірин, Group name: противірусні
Drug ID: 5, Drug name: Гербіон, Group name: від кашлю
Drug ID: 6, Drug name: Отривін, Group name: від нежиті
Drug ID: 7, Drug name: Стрепсилс, Group name: None
Час виконання: 7.02 мс
```

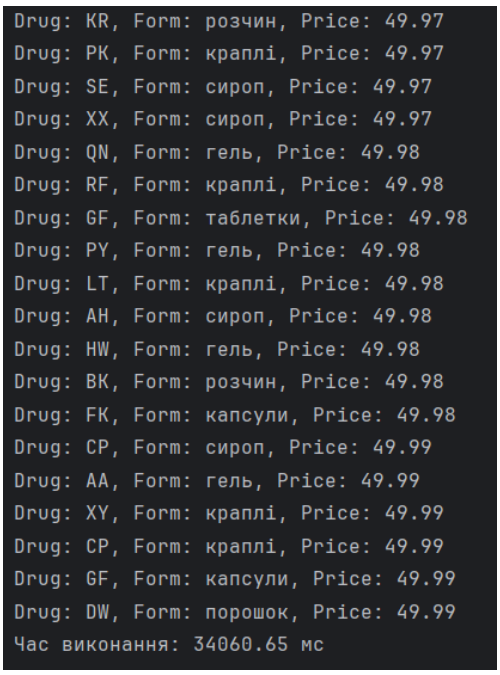
Для реалізації такого пошуку був використаний такий SQL-запит:

```
SELECT
  drugs.drug_id,
  drugs.name AS drug_name,
  groups.name AS group_name
FROM
  drugs
LEFT JOIN
  drugs_groups ON drugs_groups.drug_id = drugs.drug_id
LEFT JOIN
  groups ON drugs_groups.group_id = groups.group_id;
```

Цей запит використовує конструкцію SELECT для вибору конкретних стовпців із таблиць "drugs" і "groups". Операції LEFT JOIN використовуються для об'єднання цих таблиць за відповідними ідентифікаторами ("drug_id" і "group_id").

2. **ТОП найдешевших препаратів.**
   При виборі цього пункту, користувач повинен ввести число, яке буде означати максимальну ціну препарату для сортування. У результаті буде сформована та показана таблиця з назвою препарату, формою препарату та ціною, де ціна менше введеного користувачем значення, відсортована за ціною у зростаючому порядку.



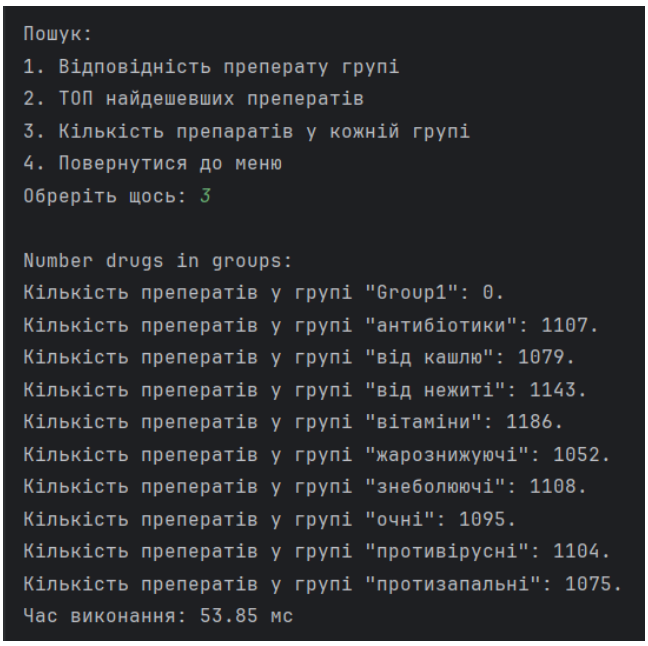


Для реалізації такого пошуку був використаний такий SQL-запит:

```
SELECT
  drugs.name AS drug_name,
  forms.name AS form_name,
  drugs.price
FROM
  drugs
JOIN
  forms ON drugs.form_id = forms.form_id
WHERE
  drugs.price < %s
ORDER BY
  drugs.price ASC;
```

Цей запит використовує оператор JOIN для об'єднання таблиць "drugs" і "forms" за умовою відповідності ідентифікаторів форми. Умова WHERE drugs.price < %s обмежує результати лише тими записами, де ціна препарату менше вказаного значення. Крім того, ORDER BY drugs.price ASC сортує результати за ціною в порядку зростання.

3. **Кількість препаратів у кожній групі.**
   При виборі цього пункту, буде сформований список з кількістю препаратів у кожній групі, включаючи групи, в яких кількість препаратів може бути 0.



Для реалізації такого пошуку був використаний такий SQL-запит:

```
SELECT
  groups.name AS group_name,
  COALESCE(COUNT(drugs.drug_id), 0) AS drug_count
FROM
  groups
LEFT JOIN
  drugs_groups ON groups.group_id = drugs_groups.group_id
LEFT JOIN
  drugs ON drugs_groups.drug_id = drugs.drug_id
GROUP BY
  groups.group_id, groups.name
ORDER BY
  group_name;
```

Основні етапи цього запиту:
LEFT JOIN таблиці "groups" з таблицею "drugs_groups" за умовою відповідності ідентифікаторів групи.
Знову LEFT JOIN з таблицею "drugs" за умовою відповідності ідентифікаторів препарату.
Використання функції COALESCE для заміщення NULL значень кількості препарату нулем.
GROUP BY для групування результатів за ідентифікатором групи та назвою групи.
ORDER BY для сортування результатів за назвою групи.

Також, після виведення даних, виведено час виконання запиту у мілісекундах.

Завдання 4

Шаблон MVC

MVC розшифровується як "модель-подання-контролер" (від англ. model-view-controller). Це спосіб організації коду, який передбачає виділення блоків, що відповідають за рішення різних завдань. Один блок відповідає за дані програми, інший відповідає за зовнішній вигляд, а третій контролює роботу програми.

Компоненти MVC:

1. Модель - цей компонент відповідає за дані, а також визначає структуру додатка.

2. Представлення - цей компонент відповідає за взаємодію з користувачем. Тобто код компонента "view" визначає зовнішній вигляд додатка і способи його використання.

3. Контролер - цей компонент відповідає за зв'язок між "model" та "view". Код компонента "controller" визначає, як програма реагує на дії користувача. По суті, це мозок MVC-додатка.

Отже, main.py – точка входу в програму, запускає початковий інтерфейс.

model.py – виконує операції з базою даних.

view.py – файл, що відповідає за функціонал виведення даних, повідомлень для користувача та реалізовує меню для взаємодії з користувачем, приймає введені дані від користувача і передає їх у контролер.

controller.py – виконує підключення до бази даних, обробляє введені користувачем дані, подає відповідну команду до model.py.

Код програми

main.py

```
from controller import Controller

if __name__ == "__main__":
    controller = Controller()
    controller.run()
```

model.py

```
import psycopg2

class Model:
    def __init__(self):
        self.conn = psycopg2.connect(
            dbname='directory',
            user='postgres',
            password='1111',
            host='localhost',
            port=3000
        )

    def add_drug(self, name, price, quantity, form_id, producer_id):
        c = self.conn.cursor()
        c.execute('INSERT INTO drugs (name, price, quantity, form_id,
producer_id) VALUES (%s, %s, %s, %s, %s)', (name, price, quantity, form_id,
producer_id))
        self.conn.commit()

    def add_producer(self, name, address, link, phone_number):
        c = self.conn.cursor()
        c.execute('INSERT INTO producers (name, address, link , phone_number)
VALUES (%s, %s, %s, %s)', (name, address, link, phone_number))
        self.conn.commit()

    def add_form(self, name):
        c = self.conn.cursor()
        c.execute('INSERT INTO forms (name) VALUES (%s)', (name,))
        self.conn.commit()

    def add_group(self, name):
        c = self.conn.cursor()
        c.execute('INSERT INTO groups (name) VALUES (%s)', (name,))
        self.conn.commit()

    def add_drug_group(self, drug_id, group_id):
        c = self.conn.cursor()
        c.execute('INSERT INTO drugs_groups (drug_id, group_id) VALUES (%s,
%s)', (drug_id, group_id))
        self.conn.commit()

    def get_all_drugs(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM drugs')
        return c.fetchall()

    def get_all_producers(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM producers')
        return c.fetchall()
```

```
    def get_all_forms(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM forms')
        return c.fetchall()

    def get_all_groups(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM groups')
        return c.fetchall()

    def get_all_drugs_groups(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM drugs_groups')
        return c.fetchall()

    def get_all_drugs_with_groups(self):
        c = self.conn.cursor()
        c.execute('SELECT drugs.drug_id, drugs.name AS drug_name, groups.name
AS group_name FROM drugs LEFT JOIN drugs_groups ON drugs_groups.drug_id =
drugs.drug_id LEFT JOIN groups ON drugs_groups.group_id = groups.group_id;')
        return c.fetchall()

    def show_sorting_by_price(self, number):
        c = self.conn.cursor()
        c.execute('SELECT drugs.name AS drug_name, forms.name AS form_name,
drugs.price FROM drugs JOIN forms ON drugs.form_id = forms.form_id WHERE
drugs.price < %s ORDER BY drugs.price ASC;', (number,))
        return c.fetchall()

    def show_number_drugs_in_groups(self):
        c = self.conn.cursor()
        c.execute('SELECT groups.name AS group_name,
COALESCE(COUNT(drugs.drug_id), 0) AS drug_count FROM groups LEFT JOIN
drugs_groups ON groups.group_id = drugs_groups.group_id LEFT JOIN drugs ON
drugs_groups.drug_id = drugs.drug_id GROUP BY groups.group_id, groups.name
ORDER BY group_name;')
        return c.fetchall()

    def update_drug(self, drug_id, name, price, quantity, form_id,
producer_id):
        c = self.conn.cursor()
        c.execute('UPDATE drugs SET name=%s, price=%s, quantity=%s,
form_id=%s, producer_id=%s WHERE drug_id=%s', (name, price, quantity,
form_id, producer_id, drug_id))
        self.conn.commit()

    def update_producer(self, producer_id, name, address, link,
phone_number):
        c = self.conn.cursor()
        c.execute('UPDATE producers SET name=%s, address=%s, link=%s,
phone_number=%s WHERE producer_id=%s', (name, address, link, phone_number,
producer_id))
        self.conn.commit()

    def update_form(self, form_id, name):
        c = self.conn.cursor()
        c.execute('UPDATE forms SET name=%s WHERE form_id=%s', (name,
form_id))
        self.conn.commit()

    def update_group(self, group_id, name):
        c = self.conn.cursor()
```

```
        c.execute('UPDATE groups SET name=%s WHERE group_id=%s', (name,
group_id))
        self.conn.commit()

    def update_drug_group(self, drug_id, group_id):
        c = self.conn.cursor()
        c.execute('UPDATE drugs_groups SET group_id = %s WHERE drug_id = %s',
(group_id, drug_id))
        self.conn.commit()

    def delete_drug(self, drug_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM drugs WHERE drug_id=%s', (drug_id,))
        self.conn.commit()

    def delete_producer(self, producer_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM producers WHERE producer_id=%s',
(producer_id,))
        self.conn.commit()

    def delete_form(self, form_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM forms WHERE form_id=%s', (form_id,))
        self.conn.commit()

    def delete_group(self, group_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM groups WHERE group_id=%s', (group_id,))
        self.conn.commit()

    def delete_drug_group(self, drug_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM drugs_groups WHERE drug_id = %s', (drug_id,))
        self.conn.commit()

    def add_random_fields(self, number):
        c = self.conn.cursor()
        c.execute(
            'INSERT INTO drugs (name, price, quantity, form_id, producer_id)
SELECT chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int), random() * 100, trunc(random() * 100), trunc(random() * 7) + 1,
trunc(random() * 5) + 1 FROM generate_series(1, %s)',
            (number,)
        )
        self.conn.commit()
```

view.py

```
class View:

    def show_menu(self):
        self.show_message("\nМеню:")
        self.show_message("1. ?"одати рядок")
        self.show_message('2. ?"енерування «рандомізованих» даних (тільки для
таблиці "Drugs")')
        self.show_message("3. Показати таблицю")
        self.show_message("4. Редагувати рядок")
        self.show_message("5. ?'идалити рядок")
        self.show_message("6. Пошук")
        self.show_message("7. ?'ихід")
        choice = input("?'иберіть пункт: ")
```

```
        return choice

    def show_tables(self):
        self.show_message("\nТаблиці:")
        self.show_message("1. Drugs")
        self.show_message("2. Producers")
        self.show_message("3. Forms")
        self.show_message("4. Groups")
        self.show_message("5. Drugs Groups")
        self.show_message("6. Повернутися до меню")
        table = input("Оберіть потрібну таблицю: ")
        return table

    def show_search(self):
        self.show_message("\nПошук:")
        self.show_message("1. Відповідність препарату групі")
        self.show_message("2. ТОП найдешевших препаратів")
        self.show_message("3. Кількість препаратів у кожній групі")
        self.show_message("4. Повернутися до меню")
        choice = input("Обреріть щось: ")
        return choice

    def show_forms(self, forms):
        print("\nForms:")
        for form in forms:
            print(f"ID: {form[0]}, Name: {form[1]}")

    def show_groups(self, groups):
        print("\nGroups:")
        for group in groups:
            print(f"ID: {group[0]}, Name: {group[1]}")

    def show_drugs_groups(self, drugs_groups):
        print("\nDrugs Groups:")
        for dg in drugs_groups:
            print(f"drug_ID: {dg[0]}, group_ID: {dg[1]}")

    def show_producers(self, producers):
        print("\nProducers:")
        for producer in producers:
            print(f"ID: {producer[0]}, Name: {producer[1]}, Address:
{producer[2]}, Link: {producer[3]} , Phone number: {producer[4]}")

    def show_drugs(self, drugs):
        print("\nDrugs:")
        for drug in drugs:
            print(f"ID: {drug[0]}, Name: {drug[1]}, Price: {drug[2]},
Quantity: {drug[3]}, Form_id: {drug[4]}, Producer_id: {drug[5]}")

    def show_drugs_with_groups(self, rows):
        print("\nDrugs-Groups:")
        for row in rows:
            print(f"Drug ID: {row[0]}, Drug name: {row[1]}, Group name:
{row[2]}")

    def show_sorting_by_price(self, rows):
        print("\nSorting by price:")
        for row in rows:
            print(f"Drug: {row[0]}, Form: {row[1]}, Price: {row[2]}")

    def show_number_drugs_in_groups(self, rows):
        print("\nNumber drugs in groups:")
        for row in rows:
            print(f'Кількість препаратів у групі "{row[0]}": {row[1]}.')
```

```
def get_drug_input(self):
    while True:
        try:
            name = input("Enter drug name: ")
            if name.strip():
                break
            else:
                print("Name cannot be empty.")
        except ValueError:
            print("It must be a string.")
    while True:
        try:
            price = input("Enter drug price: ")
            if price.strip():
                price = float(price)
                break
            else:
                print("Price cannot be empty.")
        except ValueError:
            print("It must be a number.")
    while True:
        try:
            quantity = input("Enter drug quantity: ")
            if quantity.strip():
                break
            else:
                print("Quantity cannot be empty.")
        except ValueError:
            print("It must be a string.")
    while True:
        try:
            form_id = int(input("Enter drug form_id: "))
            break
        except ValueError:
            print("Form_id must be a number.")
    while True:
        try:
            producer_id = int(input("Enter drug producer_id: "))
            break
        except ValueError:
            print("Producer_id must be a number.")
    return name, price, quantity, form_id, producer_id

def get_producer_input(self):
    while True:
        try:
            name = input("Enter producer name: ")
            if name.strip():
                break
            else:
                print("Name cannot be empty.")
        except ValueError:
            print("It must be a string.")
    while True:
        try:
            address = input("Enter producer address: ").strip()
            if address == "":
                address = None
                break
            else:
                break
        except ValueError:
            print("It must be a string.")
```

```
        while True:
            try:
                link = input("Enter producer link: ").strip()
                if link == "":
                    link = None
                    break
                else:
                    break
            except ValueError:
                print("It must be a string.")
        while True:
            try:
                phone_number = input("Enter producer phone number: ").strip()
                if phone_number == "":
                    phone_number = None
                    break
                else:
                    break
            except ValueError:
                print("It must be a string.")
        return name, address, link, phone_number

    def get_form_input(self):
        while True:
            try:
                name = input("Enter form name: ")
                if name.strip():
                    break
                else:
                    print("Name cannot be empty.")
            except ValueError:
                print("It must be a string.")
        return name

    def get_group_input(self):
        while True:
            try:
                name = input("Enter group name: ")
                if name.strip():
                    break
                else:
                    print("Name cannot be empty.")
            except ValueError:
                print("It must be a string.")
        return name

    def get_drug_group_input(self):
        while True:
            try:
                drug_id = int(input("Enter drug_id: "))
                break
            except ValueError:
                print("Drug_id must be a number.")
        while True:
            try:
                group_id = int(input("Enter group_id: "))
                break
            except ValueError:
                print("Group_id must be a number.")
        return drug_id, group_id

    def get_drug_id(self):
        while True:
            try:
```

```
                id = int(input("Enter drug ID: "))
                break
            except ValueError:
                print("It must be a number.")
        return id

    def get_producer_id(self):
        while True:
            try:
                id = int(input("Enter producer ID: "))
                break
            except ValueError:
                print("It must be a number.")
        return id

    def get_form_id(self):
        while True:
            try:
                id = int(input("Enter form ID: "))
                break
            except ValueError:
                print("It must be a number.")
        return id

    def get_group_id(self):
        while True:
            try:
                id = int(input("Enter group ID: "))
                break
            except ValueError:
                print("It must be a number.")
        return id

    def get_task_id(self):
        while True:
            try:
                id = int(input("Enter task ID: "))
                break
            except ValueError:
                print("It must be a number.")
        return id

    def show_message(self, message):
        print(message)

    def get_number(self):
        while True:
            try:
                number = int(input("Enter the number: "))
                break
            except ValueError:
                print("It must be a number.")
        return number
```

controller.py

```
import time
from model import Model
from view import View

class Controller:
    def __init__(self):
```

```
        self.model = Model()
        self.view = View()

    def run(self):
        while True:
            choice = self.view.show_menu()

            if choice == '7':
                break
            if choice == '6':
                self.process_search_option()
            elif choice in ['1', '2', '3', '4', '5']:
                self.process_menu_choice(choice)
            else:
                self.view.show_message("Wrong choice. Try again.")

    def process_menu_choice(self, choice):
        while True:
            table = self.view.show_tables()

            if table == '6':
                break

            if choice == '1':
                self.process_add_option(table)
            elif choice == '2':
                self.process_add_random_option(table)
            elif choice == '3':
                self.process_view_option(table)
            elif choice == '4':
                self.process_update_option(table)
            elif choice == '5':
                self.process_delete_option(table)

    def process_add_option(self, table):
        if table == '1':
            self.view.show_message("\nAdding drug:")
            self.add_drug()
        elif table == '2':
            self.view.show_message("\nAdding producer:")
            self.add_producer()
        elif table == '3':
            self.view.show_message("\nAdding form:")
            self.add_form()
        elif table == '4':
            self.view.show_message("\nAdding group:")
            self.add_group()
        elif table == '5':
            self.view.show_message("\nAdding drug to group:")
            self.add_drug_group()
        else:
            self.view.show_message("Wrong choice. Try again.")

    def process_add_random_option(self, table):
        if table == '1':
            self.view.show_message("\nAdding random drugs:")
            self.add_random_fields()
        else:
            self.view.show_message("Wrong choice. Try again.")

    def process_view_option(self, table):
        if table == '1':
            self.view_drugs()
        elif table == '2':
```

```
            self.view_producers()
        elif table == '3':
            self.view_forms()
        elif table == '4':
            self.view_groups()
        elif table == '5':
            self.view_drugs_groups()
        elif table == '6':
            self.view.show_menu()
        else:
            self.view.show_message("Wrong choice. Try again.")

    def process_update_option(self, table):
        if table == '1':
            self.view.show_message("\nUpdating drug:")
            self.update_drug()
        elif table == '2':
            self.view.show_message("\nUpdating producer:")
            self.update_producer()
        elif table == '3':
            self.view.show_message("\nUpdating form:")
            self.update_form()
        elif table == '4':
            self.view.show_message("\nUpdating group:")
            self.update_group()
        elif table == '5':
            self.view.show_message("\nUpdating drug to group:")
            self.update_drug_group()
        else:
            self.view.show_message("Wrong choice. Try again.")

    def process_delete_option(self, table):
        if table == '1':
            self.view.show_message("\nDeleting drug:")
            self.delete_drug()
        elif table == '2':
            self.view.show_message("\nDeleting producer:")
            self.delete_producer()
        elif table == '3':
            self.view.show_message("\nDeleting form:")
            self.delete_form()
        elif table == '4':
            self.view.show_message("\nDeleting group:")
            self.delete_group()
        elif table == '5':
            self.view.show_message("\nDeleting drug to group:")
            self.delete_drug_group()
        else:
            self.view.show_message("Wrong choice. Try again.")

    def process_search_option(self):
        option = self.view.show_search()

        if option == '1':
            start_time = time.time()
            self.show_drugs_with_groups()
            end_time = time.time()
            elapsed_time = (end_time - start_time) * 1000
            print(f"Час виконання: {elapsed_time:.2f} мс")
        elif option == '2':
            start_time = time.time()
            self.show_sorting_by_price()
            end_time = time.time()
            elapsed_time = (end_time - start_time) * 1000
```

```
                print(f"Час виконання: {elapsed_time:.2f} мс")
            elif option == '3':
                start_time = time.time()
                self.show_number_drugs_in_groups()
                end_time = time.time()
                elapsed_time = (end_time - start_time) * 1000
                print(f"Час виконання: {elapsed_time:.2f} мс")
            else:
                self.view.show_menu()

    def add_drug(self):
        try:
            name, price, quantity, form_id, producer_id =
self.view.get_drug_input()
            self.model.add_drug(name, price, quantity, form_id, producer_id)
            self.view.show_message("Drug added successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def add_producer(self):
        try:
            name, address, link, phone_number =
self.view.get_producer_input()
            self.model.add_producer(name, address, link, phone_number)
            self.view.show_message("Producer added successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def add_form(self):
        try:
            name = self.view.get_form_input()
            self.model.add_form(name)
            self.view.show_message("Form added successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def add_group(self):
        try:
            name = self.view.get_group_input()
            self.model.add_group(name)
            self.view.show_message("Group added successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def add_drug_group(self):
        try:
            drug_id, group_id = self.view.get_drug_group_input()
            self.model.add_drug_group(drug_id, group_id)
            self.view.show_message("Drug-Group added successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def view_drugs(self):
        try:
            drugs = self.model.get_all_drugs()
            self.view.show_drugs(drugs)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def view_producers(self):
        try:
            producers = self.model.get_all_producers()
            self.view.show_producers(producers)
        except Exception as e:
```

```
            self.view.show_message(f"Something went wrong: {e}")

    def view_forms(self):
        try:
            forms = self.model.get_all_forms()
            self.view.show_forms(forms)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def view_groups(self):
        try:
            groups = self.model.get_all_groups()
            self.view.show_groups(groups)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def view_drugs_groups(self):
        try:
            drugs_groups = self.model.get_all_drugs_groups()
            self.view.show_drugs_groups(drugs_groups)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def show_drugs_with_groups(self):
        try:
            rows = self.model.get_all_drugs_with_groups()
            self.view.show_drugs_with_groups(rows)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def show_sorting_by_price(self):
        try:
            self.view.show_message("\nYou need to enter the maximum price for
filtering.")
            number = self.view.get_number()
            rows = self.model.show_sorting_by_price(number)
            self.view.show_sorting_by_price(rows)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def show_number_drugs_in_groups(self):
        try:
            rows = self.model.show_number_drugs_in_groups()
            self.view.show_number_drugs_in_groups(rows)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_drug(self):
        try:
            drug_id = self.view.get_drug_id()
            name, price, quantity, form_id, producer_id =
self.view.get_drug_input()
            self.model.update_drug(drug_id, name, price, quantity, form_id,
producer_id)
            self.view.show_message("Drug updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_producer(self):
        try:
            producer_id = self.view.get_producer_id()
            name, address, link, phone_number =
self.view.get_producer_input()
```

```
            self.model.update_producer(producer_id, name, address, link,
phone_number)
            self.view.show_message("Producer updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_form(self):
        try:
            form_id = self.view.get_form_id()
            name = self.view.get_form_input()
            self.model.update_form(form_id, name)
            self.view.show_message("Form updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_group(self):
        try:
            group_id = self.view.get_group_id()
            name = self.view.get_group_input()
            self.model.update_group(group_id, name)
            self.view.show_message("Group updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_drug_group(self):
        try:
            drug_id = self.view.get_drug_id()
            group_id = self.view.get_group_id()
            self.model.update_drug_group(drug_id, group_id)
            self.view.show_message("Drug-group updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def delete_drug(self):
        try:
            drug_id = self.view.get_drug_id()
            self.model.delete_drug(drug_id)
            self.view.show_message("Drug deleted successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def delete_producer(self):
        try:
            producer_id = self.view.get_producer_id()
            self.model.delete_producer(producer_id)
            self.view.show_message("Producer deleted successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def delete_form(self):
        try:
            form_id = self.view.get_form_id()
            self.model.delete_form(form_id)
            self.view.show_message("Form deleted successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def delete_group(self):
        try:
            group_id = self.view.get_group_id()
            self.model.delete_group(group_id)
            self.view.show_message("Group deleted successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")
```

```
def delete_drug_group(self):
    try:
        group_id = self.view.get_drug_id()
        self.model.delete_drug_group(group_id)
        self.view.show_message("Drug-group deleted successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def add_random_fields(self):
    try:
        number = self.view.get_number()
        self.model.add_random_fields(number)
        self.view.show_message("Random fields added successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")
```