НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих

комп’ютерних систем

Лабораторна робота № 2

З дисципліни

«Бази даних та засоби управління»

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

Виконав студент III курсу

ФПМ групи КВ-03

Галицький Данило

Київ 2022

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).

Варіант :

- блог (користувачі, дописи, коментарі, реакції);

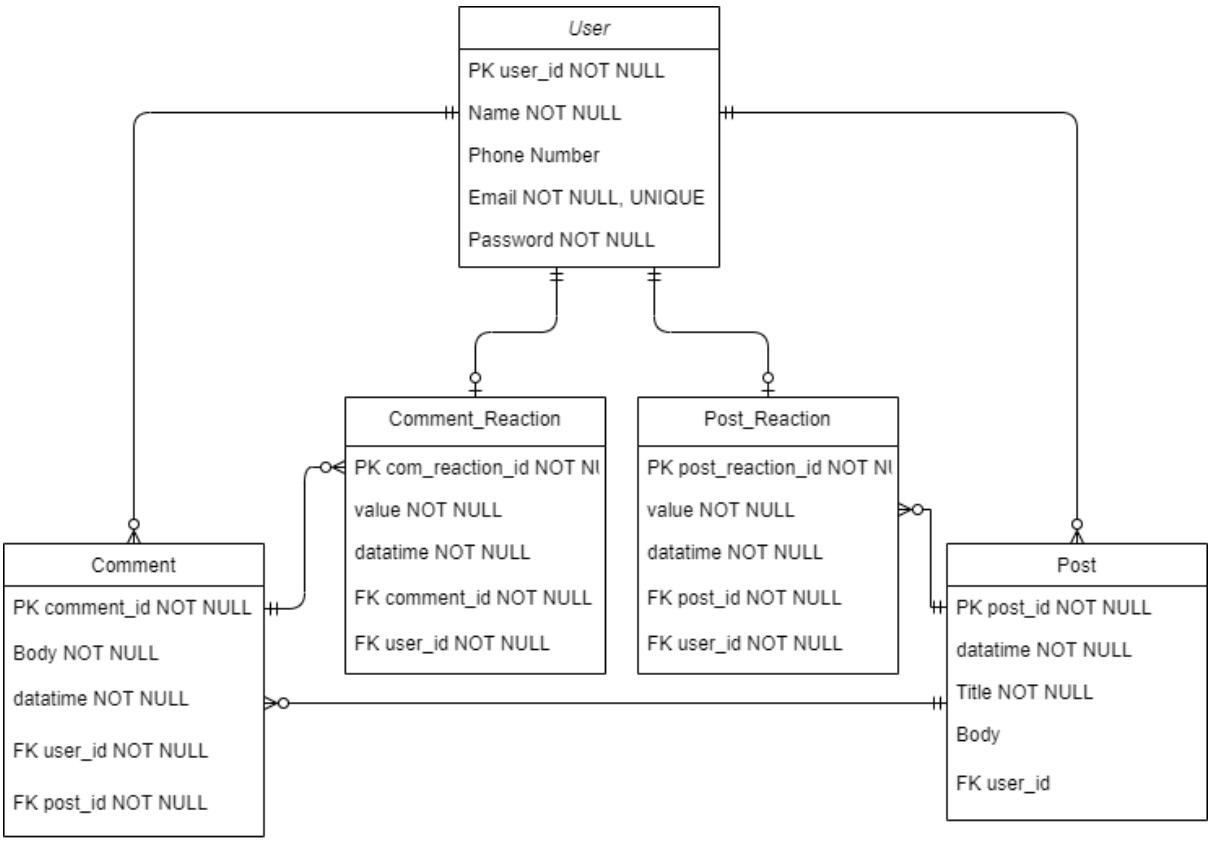
User - сутність User може створювати безліч Posts або жодного. Також User може залишити Comments. Ще User може відреагувати на певний Post або Comment. Атрибути: PK user_id, name, U email, phone number, password.

Post - у цій сутності завжди є один User, також може мати багато Comments та Post_reactions. Атрибути: PK post_id, datetime, title, body, FK user_id.

Post_reaction - ця сутність завжди має одного User, який може відреагувати один раз на один Post. Атрибути: PK post_reaction_id, value, datetime, FK user_id, FK post_id.

Comment - сутність має того, хто коментує, тобто User та що саме коментує, тобто Post. Та ще ця сутність може мати безліч Comment_reactions. PK comment_id, datetime, body, FK user_id, FK post_id.

Comment_reaction - ця сутність завжди має одного User, який може відреагувати один раз на один Comment. Атрибути: PK com_reaction_id, value, datetime, FK user_id, FK comment_id.



URL репозиторію з вихідним кодом

https://github.com/DanilGalytskyy/DB_lab2.git

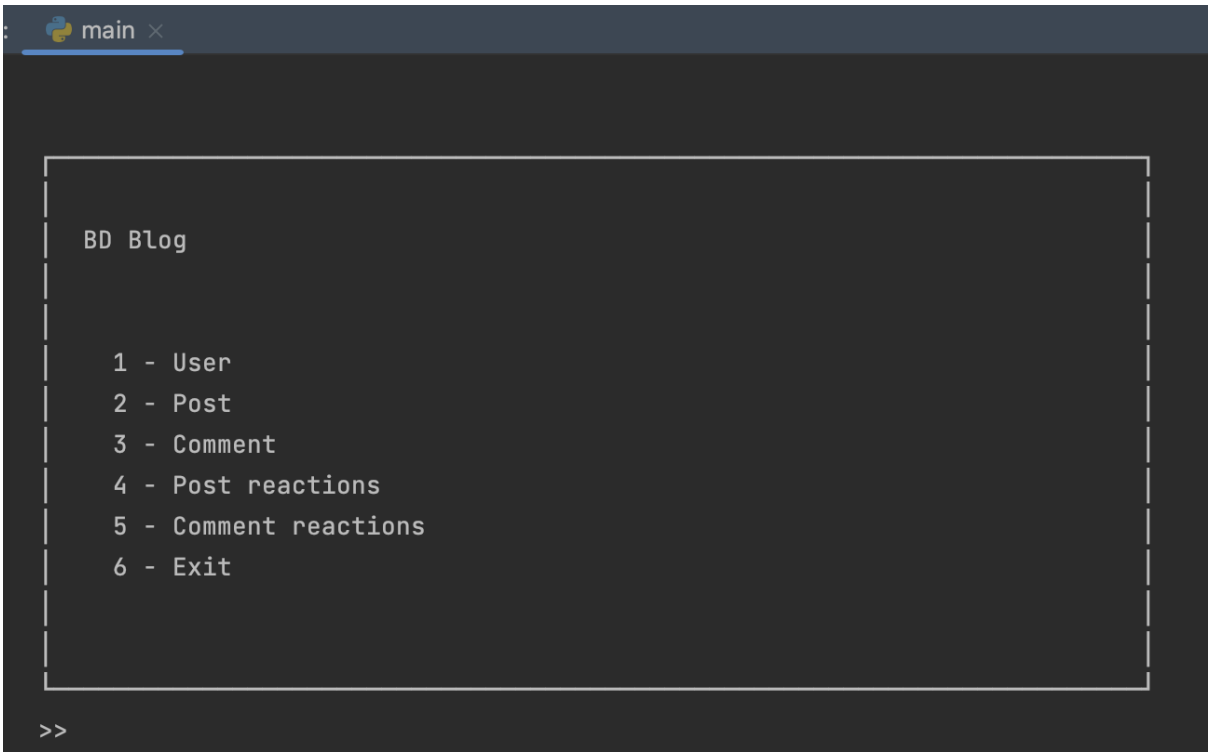
- назва мови програмування:

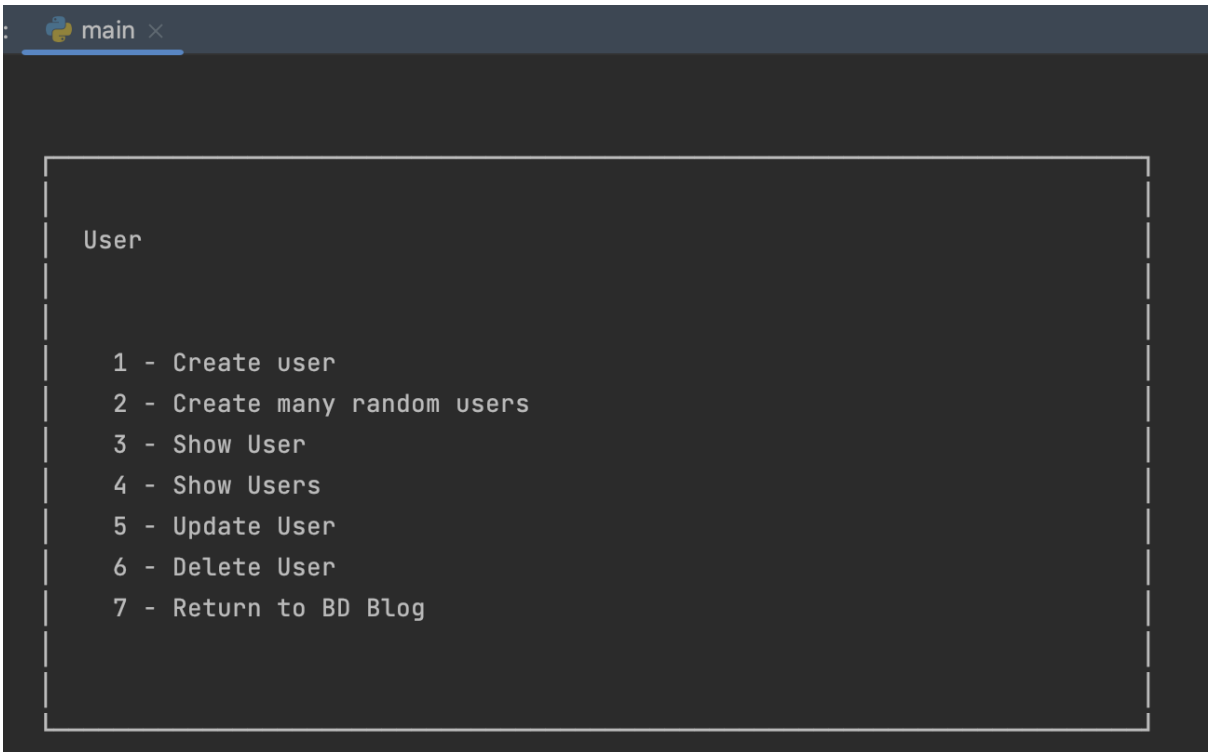
  Python

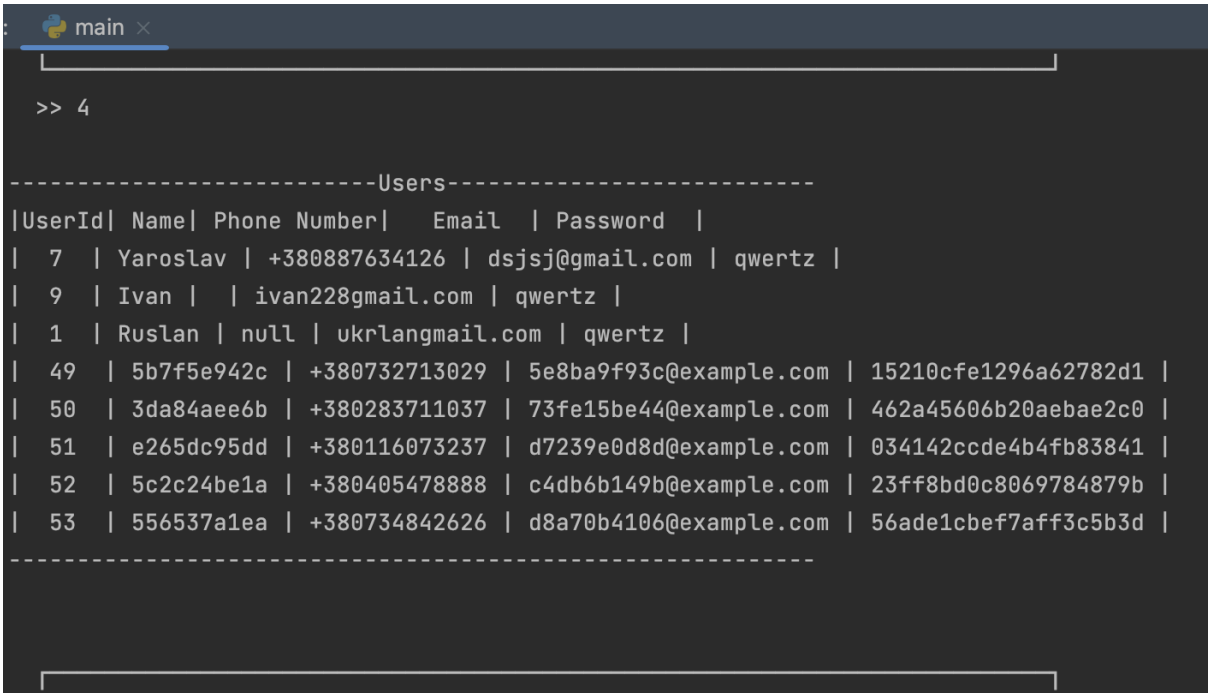
- бібліотеки, що були використані

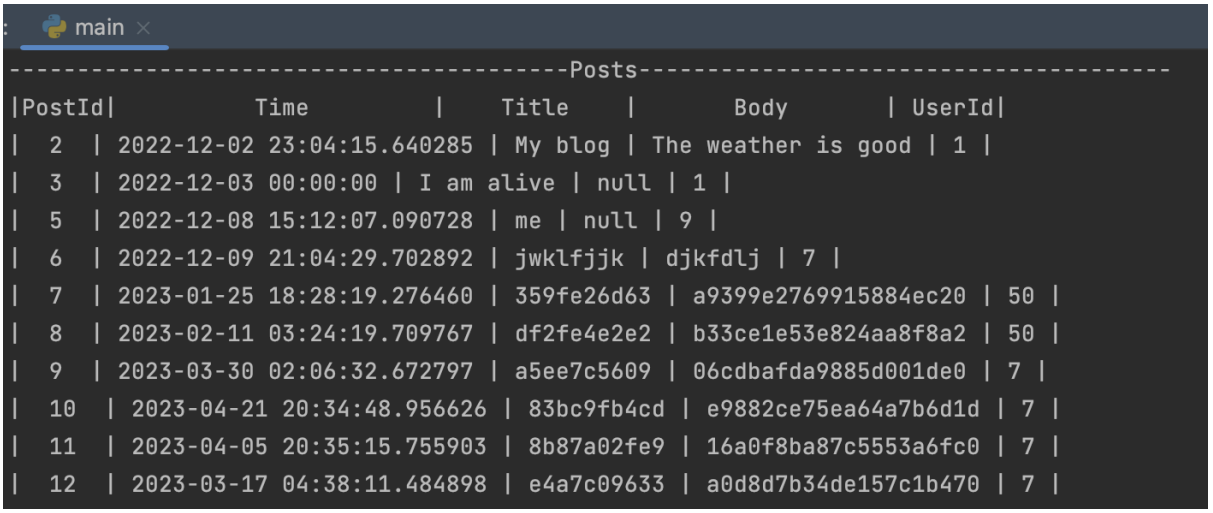
  psycopg2, consolemenu, timeit

Деталізоване завдання №1









Якщо ми видаляємо рядок з батьківської таблиці, то отримаємо наступне повідомлення:

А тепер создаємо пост з неіснуючим юзером і отримаємо наступне повідомлення:

```
┌──────────────────────────────────────────────────────────┐
│                                                          │
│  Post                                                    │
│                                                          │
│                                                          │
│    1 - Show Post                                         │
│    2 - Show Posts                                        │
│    3 - Show Posts by UserName                            │
│    4 - Create Post                                       │
│    5 - Create random Posts                               │
│    6 - Update Post                                       │
│    7 - Delete Post                                       │
│    8 - Return to BD Blog                                 │
│                                                          │
│                                                          │
└──────────────────────────────────────────────────────────┘
  >> 4

Enter post date: now()
Enter post title: Check
Enter post body: lab
Enter user_id: 10
Error with post inserting insert or update on table "Post" violates foreign key constraint "Post_user_id_fkey"
DETAIL:  Key (user_id)=(10) is not present in table "User".
```

Деталізоване завдання №2

```
main ×
┌──────────────────────────────────────────────────────────┐
└──────────────────────────────────────────────────────────┘

  >> 4

--------------------------Users--------------------------
|UserId| Name| Phone Number|   Email   | Password  |
|  7   | Yaroslav | +380887634126 | dsjsj@gmail.com | qwertz |
|  9   | Ivan |   | ivan228gmail.com | qwertz |
|  1   | Ruslan | null | ukrlangmail.com | qwertz |
|  49   | 5b7f5e942c | +380732713029 | 5e8ba9f93c@example.com | 15210cfe1296a62782d1 |
|  50   | 3da84aee6b | +380283711037 | 73fe15be44@example.com | 462a45606b20aebae2c0 |
|  51   | e265dc95dd | +380116073237 | d7239e0d8d@example.com | 034142ccde4b4fb83841 |
|  52   | 5c2c24be1a | +380405478888 | c4db6b149b@example.com | 23ff8bd0c8069784879b |
|  53   | 556537a1ea | +380734842626 | d8a70b4106@example.com | 56ade1cbef7aff3c5b3d |
-----------------------------------------------------------

┌──────────────────────────────────────────────────────────┐
```

Тепер создаємо 4 рандомних юзера:

```
┌──────────────────────────────────────────────────────────┐
│                                                          │
│  User                                                    │
│                                                          │
│                                                          │
│    1 - Create user                                       │
│    2 - Create many random users                          │
│    3 - Show User                                         │
│    4 - Show Users                                        │
│    5 - Update User                                       │
│    6 - Delete User                                       │
│    7 - Return to BD Blog                                 │
│                                                          │
│                                                          │
└──────────────────────────────────────────────────────────┘

  >> 4

--------------------------Users--------------------------
|UserId| Name| Phone Number|   Email  | Password  |
|  7  | Yaroslav | +380887634126 | dsjsj@gmail.com | qwertz |
|  9  | Ivan |  | ivan228gmail.com | qwertz |
|  1  | Ruslan | null | ukrlangmail.com | qwertz |
|  49  | 5b7f5e942c | +380732713029 | 5e8ba9f93c@example.com | 15210cfe1296a62782d1 |
|  50  | 3da84aee6b | +380283711037 | 73fe15be44@example.com | 462a45606b20aebae2c0 |
|  51  | e265dc95dd | +380116073237 | d7239e0d8d@example.com | 034142ccde4b4fb83841 |
|  52  | 5c2c24be1a | +380405478888 | c4db6b149b@example.com | 23ff8bd0c8069784879b |
|  53  | 556537a1ea | +380734842626 | d8a70b4106@example.com | 56ade1cbef7aff3c5b3d |
|  55  | 415bba4ad6 | +380765516042 | f08d453d7f@example.com | 088774ab9bff26c189ce |
|  56  | 082d111de0 | +380962938344 | 89731c6966@example.com | be61e8f2f0e534c760b9 |
|  57  | c4e4fe3857 | +380880052861 | 456846dbc8@example.com | b646558b15cdcdc696d7 |
|  58  | 4aa3671c28 | +380945905987 | fc989a5b5b@example.com | f3ca88aab7efa5288081 |
-----------------------------------------------------------
```

SQL запит має наступний вигляд:

```
38
      Danylo
39    def insert_many_random_users(connection, count):
40        try:
41            cursor = connection.cursor()
42            cursor.execute("insert into public.\"User\" (name, phone_number, email, password)"
43                           "select left(md5(random()::text), 10),"
44                           "concat('+380',trunc(random()*1000000000)::int),"
45                           "concat(left(md5(random()::text), 10),'@example.com'), left(md5(random()::text), 20)"
46                           "FROM generate_series(1, {})".format(count))
47            connection.commit()
48            cursor.close()
49        except(Exception, psycopg2.Error) as error:
50            print("Error with user inserting", error)
51
52
```

Теж саме робимо з постом, додаємо 5 рандомних постів:

```
----------------------------------------Posts----------------------------------------
|PostId|          Time          |    Title    |       Body       | UserId|
|  2   | 2022-12-02 23:04:15.640285 | My blog | The weather is good | 1 |
|  3   | 2022-12-03 00:00:00 | I am alive | null | 1 |
|  5   | 2022-12-08 15:12:07.090728 | me | null | 9 |
|  6   | 2022-12-09 21:04:29.702892 | jwklfjjk | djkfdlj | 7 |
|  7   | 2023-01-25 18:28:19.276460 | 359fe26d63 | a9399e2769915884ec20 | 50 |
|  8   | 2023-02-11 03:24:19.709767 | df2fe4e2e2 | b33ce1e53e824aa8f8a2 | 50 |
|  9   | 2023-03-30 02:06:32.672797 | a5ee7c5609 | 06cdbafda9885d001de0 | 7 |
|  10   | 2023-04-21 20:34:48.956626 | 83bc9fb4cd | e9882ce75ea64a7b6d1d | 7 |
|  11   | 2023-04-05 20:35:15.755903 | 8b87a02fe9 | 16a0f8ba87c5553a6fc0 | 7 |
|  12   | 2023-03-17 04:38:11.484898 | e4a7c09633 | a0d8d7b34de157c1b470 | 7 |
|  13   | 2023-02-04 23:36:19.044202 | 13bd3d5a81 | 568c1a117f0ee827d395 | 7 |
```

SQL запит має наступний вигляд:

```
Danylo
def insert_many_random_posts(connection, userId, count):
    try:
        cursor = connection.cursor()
        cursor.execute("insert into public.\"Post\" (datetime, title, body, user_id)"
                       "select NOW() + (random() * (NOW()+'90 days' - NOW())) + '30 days', "
                       "left(md5(random()::text), 10), left(md5(random()::text), 20), {} "
                       "from generate_series(1, {})".format(userId, count))
        connection.commit()
        cursor.close()
    except(Exception, psycopg2.Error) as error:
        print("Error with post inserting", error)

```

Теж саме робимо з комментом, додаємо 5 рандомних комментів:

```
-------------------------------------Comments-------------------------------------
|commentId|   Body   |           datetime            | userId| postId|
|  1   | agree | 2022-12-08 16:10:34.294681 | 9 | 2 |
|  2   | I like it | 2022-12-09 21:14:02.121305 | 7 | 2 |
|  8   | 1b7c2bd0e2 | 2023-03-16 05:03:43.557775 | 50 | 17 |
|  9   | 23fe164616 | 2023-03-04 17:02:09.180554 | 50 | 17 |
|  10   | c416bc6ab0 | 2023-04-03 08:43:48.709613 | 9 | 24 |
|  11   | 2f0fa009ab | 2023-03-09 21:24:51.274228 | 9 | 24 |
|  12   | 33a2912303 | 2023-03-25 00:50:26.113812 | 9 | 24 |
|  13   | e2a5ec9b66 | 2023-02-19 08:58:48.139705 | 52 | 14 |
|  14   | 3c05bb69fe | 2023-04-19 07:28:40.090230 | 52 | 14 |
|  15   | 0d577beb60 | 2023-04-04 18:55:14.641906 | 52 | 14 |
----------------------------------------------------------------------------------
```

SQL запит має наступний вигляд:

```python
Danylo
def insert_many_random_comments(connection, userId, postId, count):
    try:
        cursor = connection.cursor()
        cursor.execute("insert into public.\"Comment\" (body, datetime, user_id, post_id)"
                       "select left(md5(random()::text), 10),"
                       "NOW() + (random() * (NOW()+'90 days' - NOW())) + '30 days',"
                       "{}, {} from generate_series(1, {})".format(userId, postId, count))
        connection.commit()
        cursor.close()
    except(Exception, psycopg2.Error) as error:
        print("Error with comment inserting", error)
```

Деталізоване завдання №3

Зробимо пошук за ім’ям:

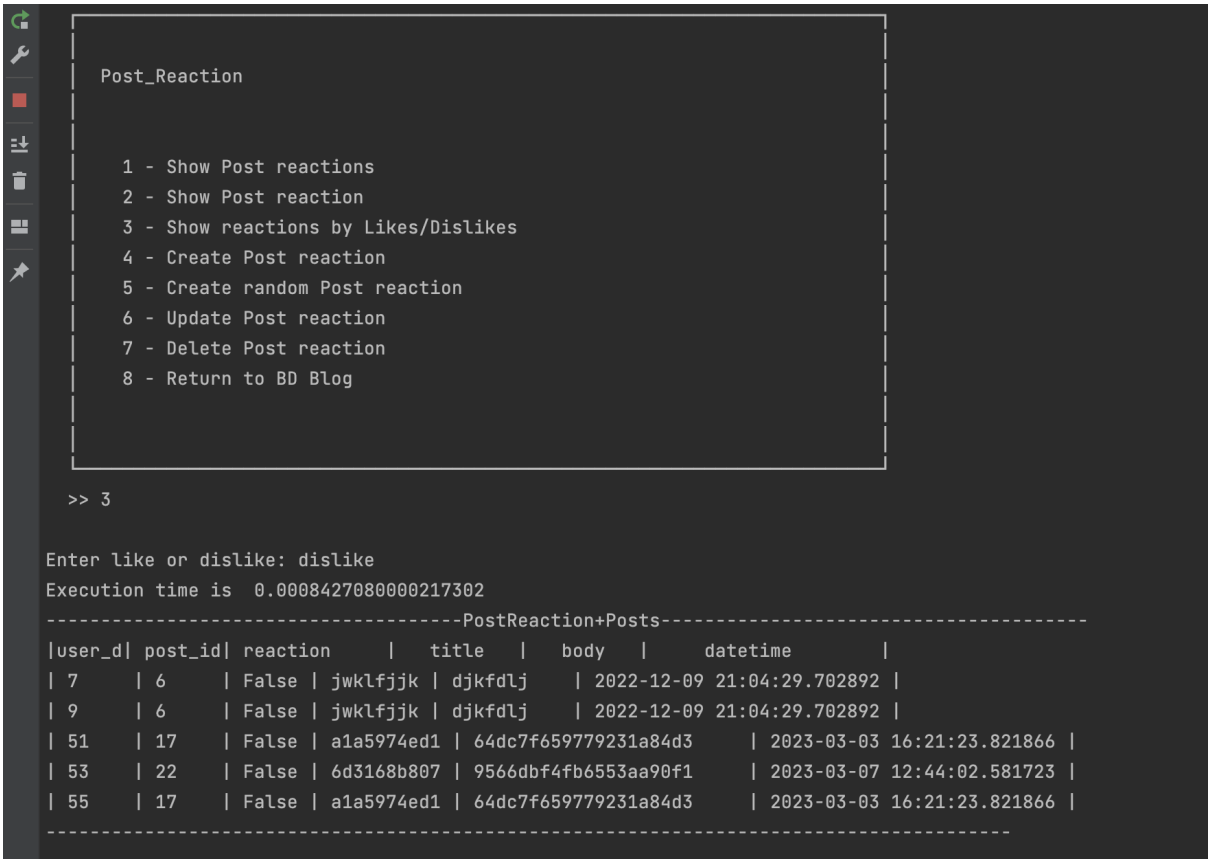
```
Post

   1 - Show Post
   2 - Show Posts
   3 - Show Posts by UserName
   4 - Create Post
   5 - Create random Posts
   6 - Update Post
   7 - Delete Post
   8 - Return to BD Blog

>> 3

Enter name of user Ivan
Execution time is  0.0013205000000198197
-------------------------------------Users+Posts-------------------------------------
|post_d| User_id| name     |   title   |   body   |     datetime        |
| 5     | 9     | Ivan | me | null      | 2022-12-08 15:12:07.090728 |
-------------------------------------------------------------------------------------
```

SQL запит має наступний вигляд:

```
22
        Danylo
23      @staticmethod
24      def findItemName(connection, name):
25          cursor = connection.cursor()
26          cursor.execute("SELECT post_id, public.\"User\".user_id, public.\"User\".name, public.\"Post\
27                         "public.\"Post\".body, public.\"Post\".datetime "
28                         "FROM public.\"Post\" INNER JOIN public.\"User\" ON "
29                         "public.\"Post\".user_id=public.\"User\".user_id"
30                         " WHERE public.\"User\".name LIKE '%{}%';".format(name))
31          value = cursor.fetchall()
32          return value
33
```

Зробимо пошук за лайкам або дизлайкам постів :

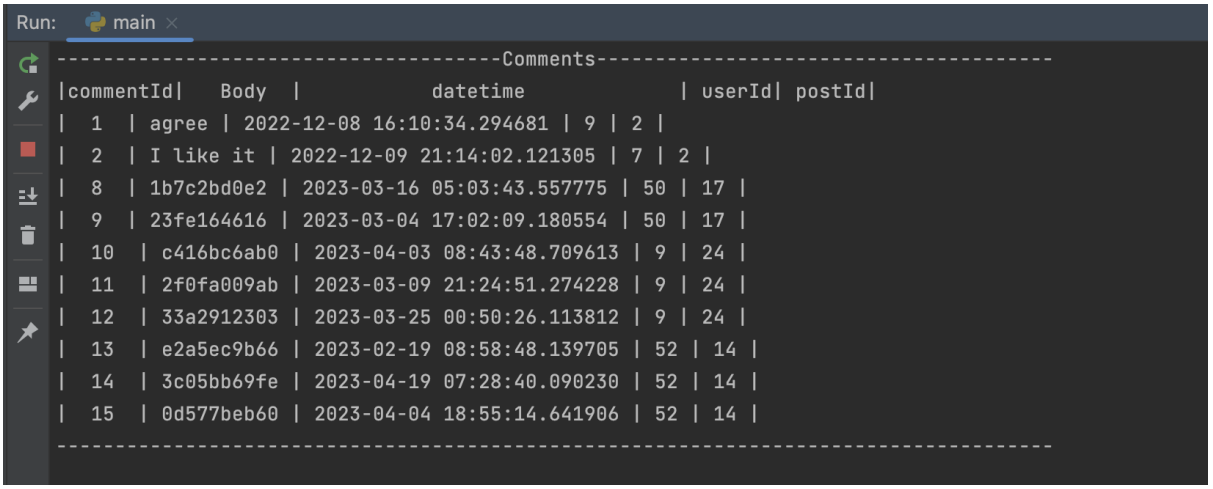
```
┌──────────────────────────────────────────────────────────┐
│                                                          │
│  Post_Reaction                                           │
│                                                          │
│                                                          │
│    1 - Show Post reactions                               │
│    2 - Show Post reaction                                │
│    3 - Show reactions by Likes/Dislikes                  │
│    4 - Create Post reaction                              │
│    5 - Create random Post reaction                       │
│    6 - Update Post reaction                              │
│    7 - Delete Post reaction                              │
│    8 - Return to BD Blog                                 │
│                                                          │
│                                                          │
└──────────────────────────────────────────────────────────┘

  >> 3

Enter like or dislike: like
Execution time is  0.00236683400000004
-------------------------------------PostReaction+Posts-------------------------------------
|user_d| post_id| reaction     |   title   |   body   |     datetime        |
| 9     | 5     | True | me | null      | 2022-12-08 15:12:07.090728 |
| 9     | 15    | True | 5778549f0b | ff7fd6394d4e628c9e75      | 2023-02-03 10:53:31.155413 |
| 51    | 15    | True | 5778549f0b | ff7fd6394d4e628c9e75      | 2023-02-03 10:53:31.155413 |
| 51    | 16    | True | 53b8984376 | 296da52fde45e9114a58      | 2023-02-06 04:27:14.969455 |
| 53    | 24    | True | c4caf229fe | d40288d39396254fc045      | 2023-02-07 20:19:56.419835 |
| 52    | 21    | True | e5f75370b8 | 5e617aae880da2d299e6      | 2023-03-26 18:20:13.366895 |
| 53    | 20    | True | 637c7a36b9 | ca1797f3b4e7c15a7087      | 2023-02-06 03:05:35.550188 |
-------------------------------------------------------------------------------------------
```

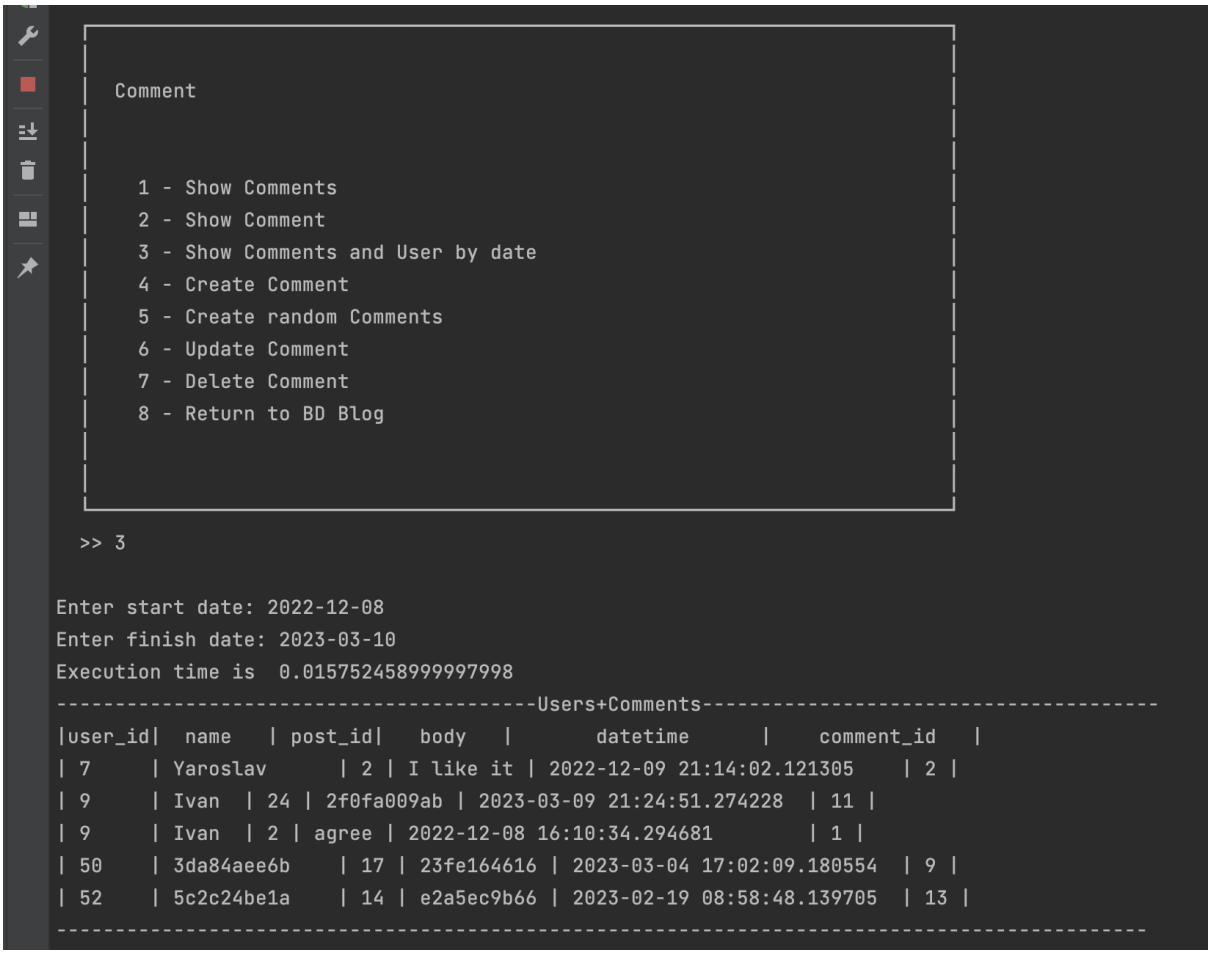


SQL запит має наступний вигляд:

```
Danylo
@staticmethod
def findPostReactions(connection, reaction):
    cursor = connection.cursor()
    start = timer()
    cursor.execute("SELECT public.\"Post_Reaction\".user_id, public.\"Post\".post_id, value, "
                   "public.\"Post\".title, "
                   "public.\"Post\".body, public.\"Post\".datetime "
                   "FROM public.\"Post_Reaction\" INNER JOIN public.\"Post\" ON "
                   "public.\"Post_Reaction\".post_id=public.\"Post\".post_id "
                   "WHERE  value = {};".format(reaction))
    value = cursor.fetchall()
    end = timer()
    print('Execution time is ', end - start)
    return value
```

А тепер зробимо пошук за певним періодом коментарів та хто їх залишив :





SQL запит має наступний вигляд:

```
    Danylo
@staticmethod
def findRowBetweenDates(connection, first, second):
    cursor = connection.cursor()
    start = timer()
    cursor.execute("SELECT public.\"User\".user_id, public.\"User\".name, public.\"Comment\".post_id, body, "
                   "public.\"Comment\".datetime, comment_id "
                   "FROM public.\"Comment\" INNER JOIN public.\"User\" ON "
                   "public.\"Comment\".user_id=public.\"User\".user_id "
                   "WHERE  datetime BETWEEN to_date('{}','YYYY-MM-DD') "
                   "AND to_date('{}','YYYY-MM-DD'); ".format(first, second))
    value = cursor.fetchall()
    end = timer()
    print('Execution time is ', end - start)
    return value
```

Деталізоване завдання №4

Model надає дані та методи роботи з ними: запити до бази даних, перевірка на коректність. Модель не залежить від подання (не знає як дані візуалізувати) та контролера (не має точок взаємодії з користувачем), просто надаючи доступ до даних та управління ними. Model будується таким чином, щоб відповідати на запити, змінюючи свій стан, при цьому може бути вбудоване повідомлення "спостерігачів".

Файл model.py:

```
import psycopg2
import queriesSQL as SQL
import inspect


class Model(object):

    def __init__(self):
        self._connection = self.connect_to_db()

    @property
```

```
    def connection(self):
        return self._connection

    @staticmethod
    def connect_to_db():
        connection = psycopg2.connect(host="localhost", port="5432",
user="postgres", password="qwerty123")
        return connection

    def disconnect_from_db(self):
        if self.connection is not None:
            self.connection.close()


class ModelUser(Model):

    def __init__(self):
        super().__init__()
        self._tableName = "User"

    @property
    def tableName(self):
        return self._tableName

    def read_user(self, userId):
```

```
        return SQL.select_one(self.connection, self._tableName, userId)

    def read_users(self):
        return SQL.select_all(self.connection, self._tableName)

    def create_user(self, name, phone_number, email, password):
        return SQL.insert_one_user(self.connection, name, phone_number, email,
password)

    def create_many_users(self, count):
        return SQL.insert_many_random_users(self.connection, count)

    def reform_user(self, user_id, name, phone_number, email, password):
        return SQL.update_one_user(self.connection, user_id, name,
phone_number, email, password)

    def remove_user(self, userId):
        return SQL.delete_one(self.connection, self._tableName, userId)


class ModelPost(Model):
    def __init__(self):
        super().__init__()
        self._tableName = "Post"
```

```
    @property
    def tableName(self):
        return self._tableName

    def read_posts(self):
        return SQL.select_all(self.connection, self._tableName)

    def read_post(self, postId):
        return SQL.select_one(self.connection, self._tableName, postId)

    def read_users_posts(self, name):
        return inspect.Inspect.findItemName(self.connection, name)

    def create_post(self, datetime, title, body, user_id):
        return SQL.insert_one_post(self.connection, datetime, title, body, user_id)

    def create_many_posts(self, count):
        try:
            userId = inspect.Inspect.findExistRow(self.connection, "User")
            assert inspect.Inspect.findExistingId(self.connection, "User", userId),
'\033[91m item id isn\'t exist ' \
                                                    '\033[0m '
            SQL.insert_many_random_posts(self.connection, userId, count)
            return True
        except Exception as err:
```

```
            print(err)
            return False

    def reform_post(self, post_id, datetime, title, body, user_id):
        return SQL.update_one_post(self.connection, post_id, datetime, title,
body, user_id)

    def remove_post(self, postId):
        return SQL.delete_one(self.connection, self._tableName, postId)


class ModelComment(Model):
    def __init__(self):
        super().__init__()
        self._tableName = "Comment"

    @property
    def tableName(self):
        return self._tableName

    def read_comment(self, commentId):
        return SQL.select_one(self.connection, self._tableName, commentId)

    def read_comments(self):
        return SQL.select_all(self.connection, self._tableName)
```

```
    def read_comment_byDates(self, first, second=None):
        return inspect.Inspect.findRowBetweenDates(self.connection, first, second)


    def create_comment(self, body, datetime, user_id, post_id):
        return SQL.insert_one_comment(self.connection, body, datetime, user_id, post_id)


    def create_many_comments(self, count):
        try:
            userId = inspect.Inspect.findExistRow(self.connection, "User")
            postId = inspect.Inspect.findExistRow(self.connection, "Post")
            assert inspect.Inspect.findExistingId(self.connection, "User", userId), '\033[91m item id isn\'t exist ' \
                                                                                    '\033[0m '
            assert inspect.Inspect.findExistingId(self.connection, "Post", postId), '\033[91m item id isn\'t exist ' \
                                                                                    '\033[0m '
            SQL.insert_many_random_comments(self.connection, userId, postId, count)
            return True
        except Exception as err:
            print(err)
            return False
```

```
    def reform_comment(self, comment_id, body, datetime, user_id, post_id):
        return SQL.update_one_comment(self.connection, comment_id, body, datetime, user_id, post_id)

    def remove_comment(self, commentId):
        return SQL.delete_one(self.connection, self._tableName, commentId)


class ModelPost_Reaction(Model):
    def __init__(self):
        super().__init__()
        self._tableName = "Post_Reaction"

    @property
    def tableName(self):
        return self._tableName

    def read_post_reactions(self):
        return SQL.select_all(self.connection, self._tableName)

    def read_post_reaction(self, reactionId):
        return SQL.select_one(self.connection, self._tableName, reactionId)

    def read_reactions_ByLikeDislike(self, reaction):
        return inspect.Inspect.findPostReactions(self.connection, reaction)
```

```
    def create_post_reaction(self, value, datetime, post_id, user_id):
        return SQL.insert_post_reaction(self.connection, value, datetime, post_id,
user_id)


    def create_random_post_reaction(self):
        try:
            userId = inspect.Inspect.findExistRow(self.connection, "User")
            postId = inspect.Inspect.findExistRow(self.connection, "Post")
            assert inspect.Inspect.findExistingId(self.connection, "User", userId),
'\033[91m item id isn\'t exist ' \
                                                  '\033[0m '
            assert inspect.Inspect.findExistingId(self.connection, "Post", postId),
'\033[91m item id isn\'t exist ' \
                                                  '\033[0m '
            SQL.insert_random_post_reaction(self.connection, postId, userId, )
            return True
        except Exception as err:
            print(err)
            return False


    def reform_post_reaction(self, post_reaction_id, value, datetime, post_id,
user_id):
        return SQL.update_one_post_reaction(self.connection, post_reaction_id,
value, datetime, post_id, user_id)
```

```
    def remove_post_reaction(self, post_reactionId):
        return SQL.delete_one(self.connection, self._tableName, post_reactionId)



class ModelComment_Reaction(Model):
    def __init__(self):
        super().__init__()
        self._tableName = "Comment_Reaction"

    @property
    def tableName(self):
        return self._tableName

    def read_comment_reactions(self):
        return SQL.select_all(self.connection, self._tableName)

    def read_comment_reaction(self, comment_reactionId):
        return SQL.select_one(self.connection, self._tableName,
comment_reactionId)

    def create_comment_reaction(self, value, datetime, comment_id, user_id):
        return SQL.insert_comment_reaction(self.connection, value, datetime,
comment_id, user_id)

    def create_random_comment_reaction(self):
```

```
        try:
            userId = inspect.Inspect.findExistRow(self.connection, "User")
            commentId = inspect.Inspect.findExistRow(self.connection,
"Comment")
            assert inspect.Inspect.findExistingId(self.connection, "User", userId),
'\033[91m item id isn\'t exist ' \
                                                  '\033[0m '
            assert inspect.Inspect.findExistingId(self.connection, "Comment",
                                                  commentId), '\033[91m item id isn\'t exist ' \
                                                  '\033[0m '
            SQL.insert_random_comment_reaction(self.connection, commentId,
userId)
            return True
        except Exception as err:
            print(err)
            return False


    def reform_comment_reaction(self, comment_reaction_id, value, datetime,
comment_id, user_id):
        return SQL.update_one_comment_reaction(self.connection,
comment_reaction_id, value, datetime, comment_id,
                                               user_id)


    def remove_comment_reaction(self, comment_reactionId):
        return SQL.delete_one(self.connection, self._tableName,
comment_reactionId)
```