НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Лабораторна робота № 2

з дисципліни

«Бази даних і засоби управління»

Виконав: студент III курсу

ФПМ групи КВ-02

Костюков Сергій Васильович

Перевірив: Павловський В. І.

Київ-2022

Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL

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

Логічна модель предметної галузі «Продаж квитків кіно»

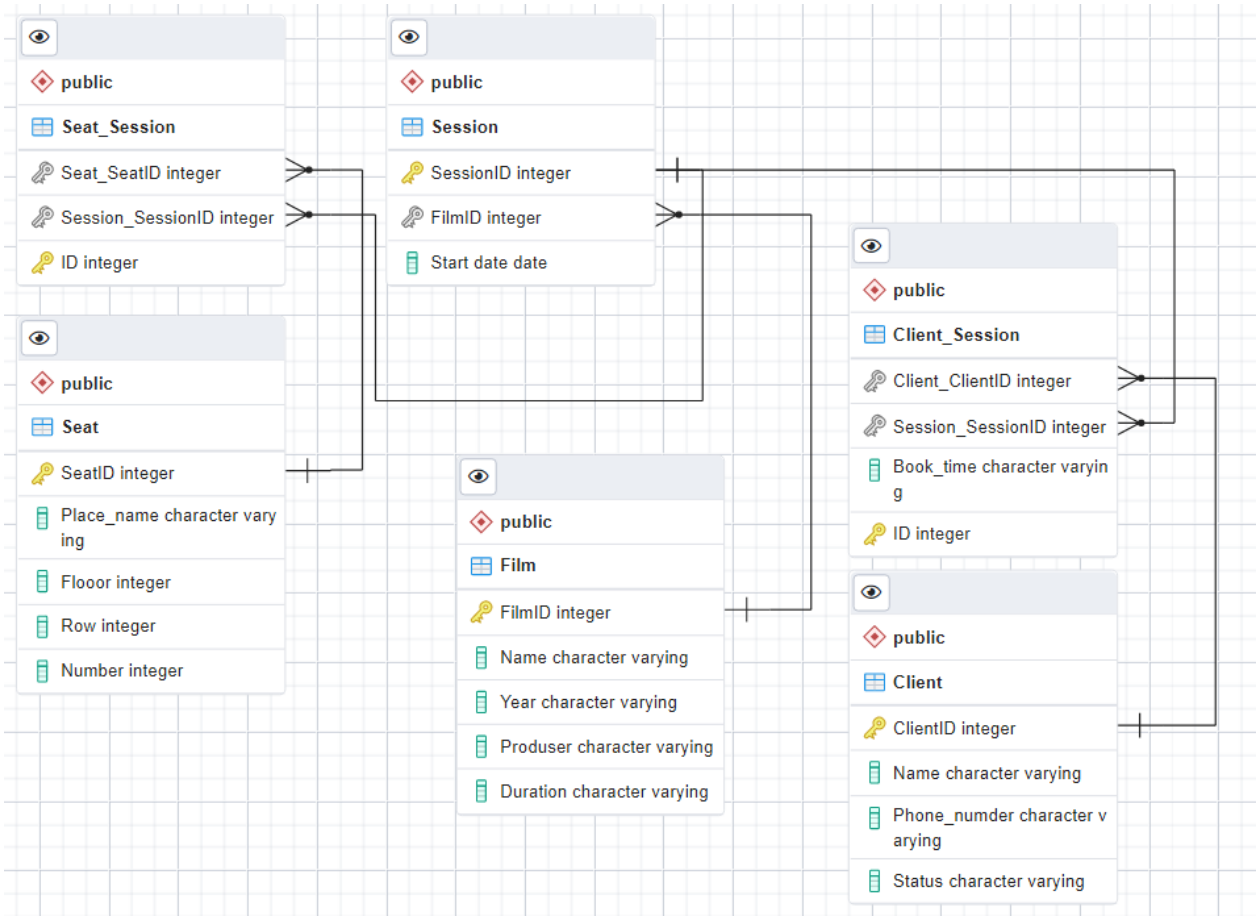


Рисунок 1 – Логічна модель предметної галузі «Продаж квитків кіно»

Середовище розробки та налаштування програмної системи

Для розробки використовувалась мова програмування Python, середовище розробки PyCharm, а також стороння бібліотека, що надає API для доступу до PostgreSQL – psycopg2.

Шаблон проектування

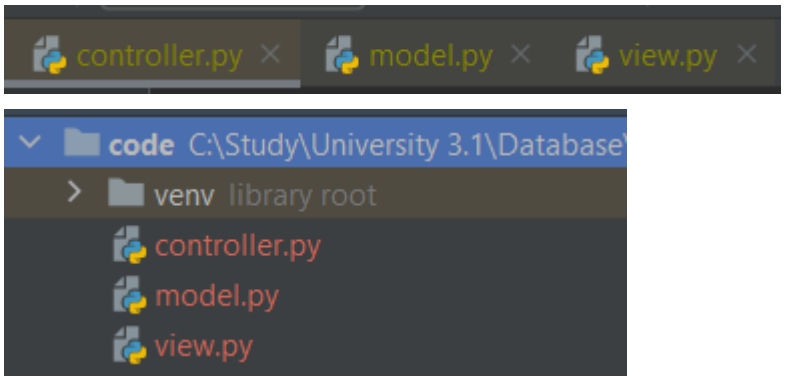
MVC(модель-подання-контролер) - Шаблон проектування, який використаний у програмі.

Model – представляє клас, що описує логіку використовуваних даних.

View – в нашому випадку консольний інтерфейс з яким буде взаємодіяти наш користувач.

Controller – представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних.

Структура програми та її опис



Програма умовно поділена на 3 модулі: файл controller.py, файл model.py, файл view.py. Класи, як видно з їх назв, повністю відповідають використаному патерну MVC.

У файлі model.py описаний клас моделі, що займається регулювання підключення до бази даних, та виконанням низькорівневих запитів до неї.

У файлі controller.py описаний інтерфейс взаємодії з користувачем, запит бажаної дії, виконання пошуку, тощо.

У файлі view.py описаний клас, що виводить результати виконання тієї чи іншої дії на екран консолі.

Структура меню програми

```
Menu
1. Getting the names of the database tables
2. Obtaining table column names and types
3. Getting table column names
4. Generate random data in the database
5. Insert data in the database
6. Edit data in the database
7. Remove data from database
8. Print data in the database table
Enter your choice:
```

Перший пункт пропонує отримання імен таблиць бази даних

Другий пункт пропонує отримання імен та типів стовпчиків таблиці

Третій пункт пропонує отримання імен стовпчиків таблиці

Четвертий пункт пропонує генерування даних в таблиці

П’ятий пункт пропонує вставку даних в таблицю

Шостий пункт пропонує оновлення даних в таблиці

Сьомий пункт пропонує видалення даних з таблиці

Восьмий пункт пропонує отримання даних таблиці

Фрагменти програм внесення, редагування та вилучення даних у БД

Фрагмент програми для внесення даних

```
    def insert_data(self, table_name, values):
    line = ''
    columns = '("'
    for key in values:
        if values[key]:
            line += '%(' + key + ')s,'
            columns += key + '","'
    columns = columns[:-3] + '")'
    self.__cursor.execute(
        sql.SQL('INSERT INTO {} {} VALUES (' + line[:-1] +
')').format(sql.Identifier(table_name),

sql.SQL(columns)),
        values)
    self.__context.commit()
```

Фрагмент програми для редагування даних

```
def change_data(self, table_name, values):
    line = ''
    condition = values.pop('condition')
    for key in values:
        if values[key]:
            line += key + '=%(' + key + ')s,'
    self.__cursor.execute(
        sql.SQL('UPDATE {} SET ' + line[:-1] + ' WHERE {}
').format(sql.Identifier(table_name), sql.SQL(condition)),
        values)
    self.__context.commit()
```

Фрагмент програми для видалення даних

```
def delete_data(self, table_name, value, cond):
    self.__cursor.execute(
        sql.SQL('DELETE FROM {} WHERE {} = {}').format(sql.Identifier(table_name),
sql.Identifier(value),
                                                       sql.SQL(cond)))
    self.__context.commit()
```

Дані фрагменти програми, які наведені вище, відповідають за функціонал додавання даних, редагування та вилучення даних у базі даних.
Взаємодія відбувається через клас Model, який займається підключенням до БД, а самі функції знаходяться у файлі Controller.

Скріншоти результатів виконання операцій вставки запису в таблицю

	SessionID [PK] integer	FilmID integer	Start_date character varying
1	1	1	10.10.2022
2	2	1	11.10.2022
3	3	2	12.10.2022

```
Menu
1. Getting the names of the database tables
2. Obtaining table column names and types
3. Getting table column names
4. Generate random data in the database
5. Insert data in the database
6. Edit data in the database
7. Remove data from database
8. Print data in the database table
Enter your choice: 5
Enter the table name: Session
Enter the colum name: SessionID FilmID Start_date
Enter the values: 4 2 11.12.2022
result:

['SessionID', 'FilmID', 'Start_date']
[(1, 1, '10.10.2022'), (2, 1, '11.10.2022'), (3, 2, '12.10.2022'), (4, 2, '11.12.2022')]
```

	SessionID [PK] integer	FilmID integer	Start_date character varying
1	1	1	10.10.2022
2	2	1	11.10.2022
3	3	2	12.10.2022
4	4	2	11.12.2022

Скріншоти результатів виконання операції редагування таблиці

	SessionID [PK] integer	FilmID integer	Start_date character varying
1	1	1	10.10.2022
2	2	1	11.10.2022
3	3	2	12.10.2022

```
Menu
1. Getting the names of the database tables
2. Obtaining table column names and types
3. Getting table column names
4. Generate random data in the database
5. Insert data in the database
6. Edit data in the database
7. Remove data from database
8. Print data in the database table
Enter your choice: 6
Enter the table name: Session
Enter the colum name: "Start_date" condition
Enter the values: 20.12.2022 "SessionID"=3
['SessionID', 'FilmID', 'Start_date']
[(1, 1, '10.10.2022'), (2, 1, '11.10.2022'), (3, 2, '20.12.2022')]
```

	SessionID [PK] integer	FilmID integer	Start_date character varying
1	1	1	10.10.2022
2	2	1	11.10.2022
3	3	2	20.12.2022

Скріншоти результатів виконання операції видалення

	SessionID [PK] integer	FilmID integer	Start_date character varying
1	1	1	10.10.2022
2	2	1	11.10.2022
3	3	2	12.10.2022
4	4	2	11.12.2022

```
Menu
1. Getting the names of the database tables
2. Obtaining table column names and types
3. Getting table column names
4. Generate random data in the database
5. Insert data in the database
6. Edit data in the database
7. Remove data from database
8. Print data in the database table
Enter your choice: 7
Enter the table name: Session
Enter the colum name: SessionID
Параметр: 4
['SessionID', 'FilmID', 'Start_date']
[(1, 1, '10.10.2022'), (2, 1, '11.10.2022'), (3, 2, '12.10.2022')]
```

	SessionID [PK] integer	FilmID integer	Start_date character varying
1	1	1	10.10.2022
2	2	1	11.10.2022
3	3	2	12.10.2022

Результат отримання назв таблиць бази даних

```
Menu
1. Getting the names of the database tables
2. Obtaining table column names and types
3. Getting table column names
4. Generate random data in the database
5. Insert data in the database
6. Edit data in the database
7. Remove data from database
8. Print data in the database table
Enter your choice: 1
Seat
Session
Film
Client
Client_Session
Seat_Session
```

Результат отримання назв таблиць бази даних

```
Menu
1. Getting the names of the database tables
2. Obtaining table column names and types
3. Getting table column names
4. Generate random data in the database
5. Insert data in the database
6. Edit data in the database
7. Remove data from database
8. Print data in the database table
Enter your choice: 2
Enter a name for the table: Client
('ClientID', 'integer')
('Name', 'character varying')
('Phone_numder', 'character varying')
('Status', 'character varying')
```

Результат отримання вмісту таблиці бази даних

```
Menu
1. Getting the names of the database tables
2. Obtaining table column names and types
3. Getting table column names
4. Generate random data in the database
5. Insert data in the database
6. Edit data in the database
7. Remove data from database
8. Print data in the database table
Enter your choice: 8
Введіть назву таблиці: Client
['ClientID', 'Name', 'Phone_numder', 'Status']
[(3, 'Oleg', '+380594753072', 'standart'), (2, 'Vlad', '+380947548371', 'vip'), (1, 'Mila', '+380947846719', 'standart')
```

Текст програми

Файл controller.py

```
import view
import model
import time

is_end = 0

model = model.DbModel()

while is_end == 0:
    view.hello()

    choice = input("Enter your choice: ")

    match choice:
        case "1":
            mas = model.get_table_names()
            view.show(mas)
            time.sleep(2)
        case "2":
            table = input("Enter a name for the table: ")
            mas = model.get_column_types(table)
            view.show(mas)
            time.sleep(2)
        case "3":
            table = input("Enter a name for the table: ")
            mas = model.get_column_names(table)
            view.show(mas)
            time.sleep(2)
        case "4":
            table = input("Enter the table name: ")
            count = input("Enter count: ")
            model.generate_data(table, count)
            mas = model.get_table_data(table)
            view.show(mas)
            time.sleep(2)
        case "5":
            table = input("Enter the table name: ")
            columns = input("Enter the colum name: ").split(' ')
            val = input("Enter the values: ").split(' ')
            values = {key: value for (key, value) in zip(columns, val)}
            model.insert_data(table, values)
            print("result:\n")
            mas = model.get_table_data(table)
            view.show(mas)
            time.sleep(2)
        case "6":
            table = input("Enter the table name: ")
            columns = input("Enter the colum name: ").split(' ')
            val = input("Enter the values: ").split(' ')
            values = {key: value for (key, value) in zip(columns, val)}
            model.change_data(table, values)
            mas = model.get_table_data(table)
            view.show(mas)
            time.sleep(2)
        case "7":
            table = input("Enter the table name: ")
            column = input("Enter the colum name: ")
            param = input("Параметр: ")
            model.delete_data(table, column, param)
```

```
            mas = model.get_table_data(table)
            view.show(mas)
            time.sleep(2)
        case "8":
            table = input("Enter the table name: ")
            mas = model.get_table_data(table)
            view.show(mas)
            time.sleep(2)
        case _:
            is_end = 1
            print("End")
```

Файл model.py

```
import psycopg2
from psycopg2 import sql


class DbModel:

    def __init__(self):
        self.host = "localhost"
        self.database = "cinema_tickets"
        self.user = "postgres"
        self.password = "123"
        try:
            self.__context = psycopg2.connect(host=self.host,
database=self.database, user=self.user,
                                              password=self.password)
            self.__cursor = self.__context.cursor()
            self.__table_names = None
        except Exception as _ex:
            print("[INFO] Error while working with PostgreSQL", _ex)

    def __del__(self):
        self.__cursor.close()
        self.__context.close()

    def clear_transaction(self):
        self.__context.rollback()

    def get_table_names(self):
        if self.__table_names is None:
            self.__cursor.execute("""SELECT table_name
                          FROM information_schema.tables
                          WHERE table_schema = 'public'""")
            self.__table_names = [table[0] for table in self.__cursor]
        return self.__table_names

    def get_column_types(self, table_name):
        self.__cursor.execute("""SELECT column_name, data_type
            FROM information_schema.columns
           WHERE table_schema = 'public' AND table_name = %s
           ORDER BY table_schema, table_name""", (table_name,))
        return self.__cursor.fetchall()

    def get_column_names(self, table_name):
        self.__cursor.execute("""
            SELECT column_name FROM information_schema.columns
            WHERE table_schema = 'public' AND table_name = %s
            ORDER BY table_schema, table_name""", (table_name,))
        return [x[0] for x in self.__cursor.fetchall()]
```

```
    def get_foreign_key_info(self, table_name):
        self.__cursor.execute("""
           SELECT kcu.column_name, ccu.table_name AS
                  foreign_table_name,
                  ccu.column_name AS foreign_column_name
           FROM information_schema.table_constraints AS tc
              JOIN information_schema.key_column_usage AS kcu
                 ON tc.constraint_name = kcu.constraint_name
                 AND tc.table_schema = kcu.table_schema
              JOIN information_schema.constraint_column_usage AS ccu
                 ON ccu.constraint name = tc.constraint name
                 AND ccu.table_schema = tc.table_schema
           WHERE tc.constraint_type = 'FOREIGN KEY' AND
                        tc.table_name=%s;""", (table_name,))
        return self.__cursor.fetchall()

    def get_table_data(self, table_name):
        id_column = self.get_column_types(table_name)[0][0]
        cursor = self.__cursor
        try:
            cursor.execute(
                sql.SQL('SELECT * FROM {}').format(sql.Identifier(table_name),
sql.SQL(id_column)))
        except Exception as _ex:
            print("[INFO] Error while working with PostgreSQL", _ex)
        return ([col.name for col in cursor.description], cursor.fetchall())

    def insert_data(self, table_name, values):
        line = ''
        columns = '("'
        for key in values:
            if values[key]:
                line += '%(' + key + ')s,'
                columns += key + '","'
        columns = columns[:-3] + '")'
        self.__cursor.execute(
            sql.SQL('INSERT INTO {} {} VALUES (' + line[:-1] +
')').format(sql.Identifier(table_name),

sql.SQL(columns)),
            values)
        self.__context.commit()

    def generate_data(self, table_name, count):
        types = self.get_column_types(table_name)
        fk_array = self.get_foreign_key_info(table_name)
        select_subquery = ""
        insert_query = 'INSERT INTO "' + table_name + '" ("'
        for i in range(1, len(types)):
            t = types[i]
            name = t[0]
            type = t[1]
            fk = [x for x in fk_array if x[0] == name]
            if fk:
                select_subquery += ('(SELECT "{}" FROM "{}" ORDER BY RANDOM(), ser
LIMIT 1)'.format(fk[0][2], fk[0][1]))
            elif type == 'integer':
                select_subquery += 'trunc(random()*100)::INT'
            elif type == 'character varying':
                select_subquery += 'chr(trunc(65 + random()*25)::INT) ||
chr(trunc(65 + random()*25)::INT)'
            elif type == 'date':
```

```
                select_subquery += """ date(timestamp '2014.01.10' +
                    random() *
                    (timestamp '2020.01.20' - timestamp '2014.01.10'))"""
            elif type == 'time without time zone':
                select_subquery += "time '00:00:00' + DATE_TRUNC('second',RANDOM()
* time '24:00:00')"
            else:
                continue

            insert_query += name
            if i != len(types) - 1:
                select subquery += ','
                insert_query += '","'
            else:
                insert_query += '") '

        self.__cursor.execute(
            insert_query + "SELECT " + select_subquery +
            " FROM generate_series(1," + str(count) + ") as ser")
        self.__context.commit()

    def change_data(self, table_name, values):
        line = ''
        condition = values.pop('condition')
        for key in values:
            if values[key]:
                line += key + '=%(' + key + ')s,'
        self.__cursor.execute(
            sql.SQL('UPDATE {} SET ' + line[:-1] + ' WHERE {}
').format(sql.Identifier(table_name), sql.SQL(condition)),
            values)
        self.__context.commit()

    def delete_data(self, table_name, value, cond):
        self.__cursor.execute(
            sql.SQL('DELETE FROM {} WHERE {} =
{}').format(sql.Identifier(table_name), sql.Identifier(value),
                                                                sql.SQL(cond)))
        self.__context.commit()
```

Файл view.py

```
def hello():
    print("Menu\n"
          "1. Getting the names of the database tables\n"
          "2. Obtaining table column names and types\n"
          "3. Getting table column names\n"
          "4. Generate random data in the database\n"
          "5. Insert data in the database\n"
          "6. Edit data in the database\n"
          "7. Remove data from database\n"
          "8. Print data in the database table")


def show(mas):
    for element in mas:
        print(element)
```