МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №2

з дисципліни “ Бази даних”

тема “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”

Виконав

студент III курсу

групи КП-82

Суходольський Євгеній

Віталійович

Перевірив

“____” “_____” 20___р.

Радченко К. О.

Київ 2020

Мета роботи: здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Постановка завдання

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

Опис програми

Програма створена за патерном MVC (Model-View-Controller). Складається відповідно з модулів model , view та controller.

У модулі model реалізовані функції , що здійснюють SQL запити до Бази Даних.

У модулі view реалізовані функції виводу даних з таблиць.

У класі Controller реалізовані функції для відповідних меню та допоміжні функції.

Скріншот результатів виконання операції вилучення

Рис. 1. Початкове меню

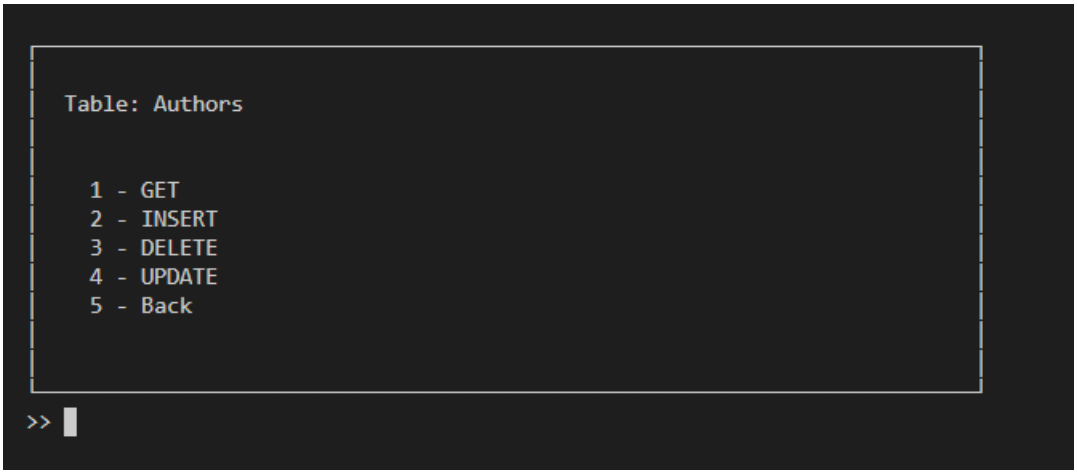
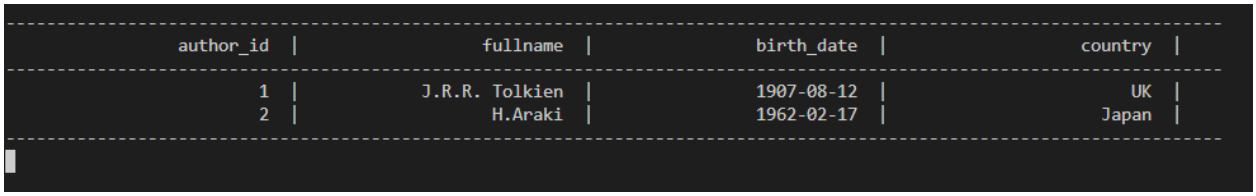


Рис. 2. Меню роботи із сутністю Authors



author_id	fullname	birth_date	country
1	J.R.R. Tolkien	1907-08-12	UK
2	H.Araki	1962-02-17	Japan

Рис. 3. Результат виконання методу GET при роботі із сутністю Authors

Рис. 4-5. Виконання методу INSERT при роботі із сутністю Authors та її успішне виконання

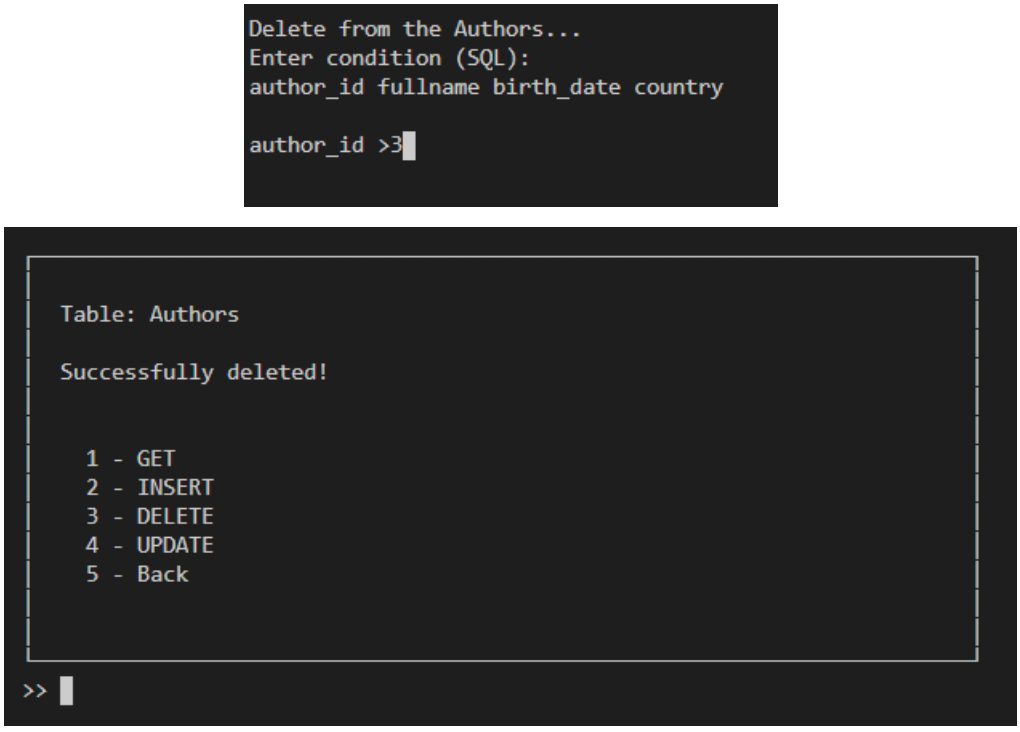


Рис. 6-7. Виконання методу DELETE при роботі із сутністю Authors та її успішне виконання

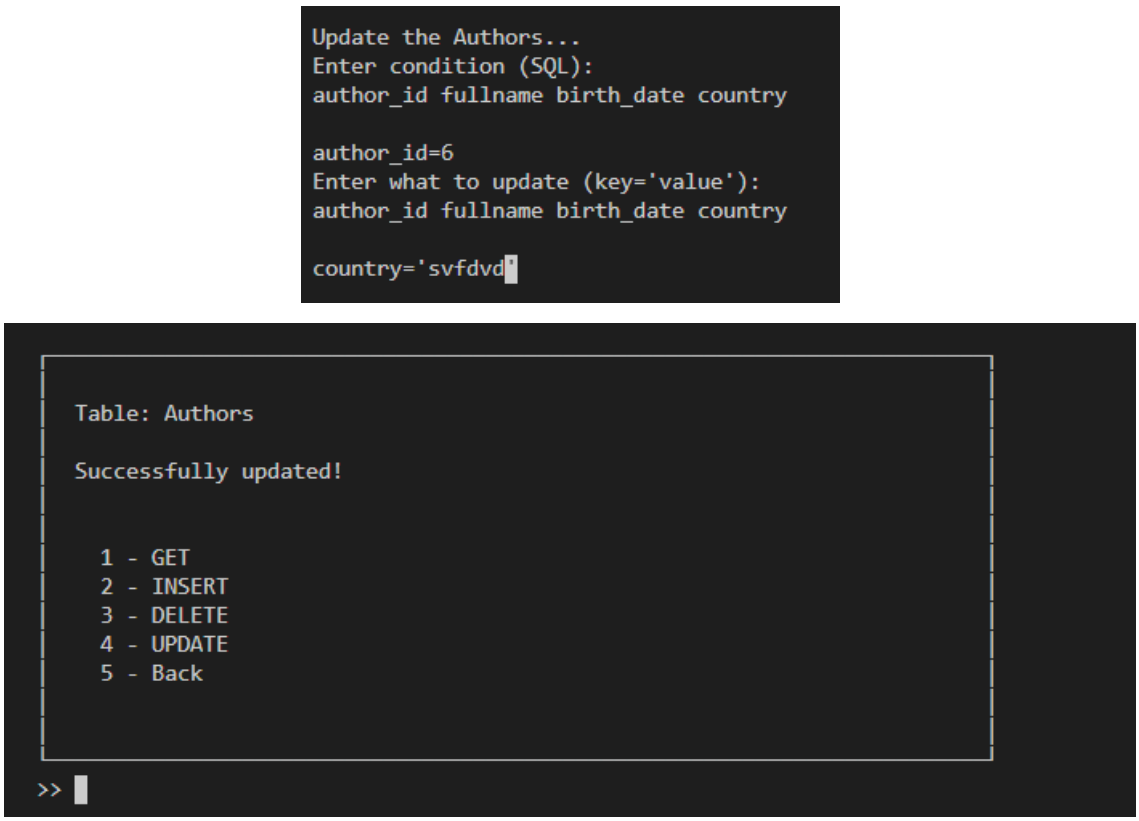
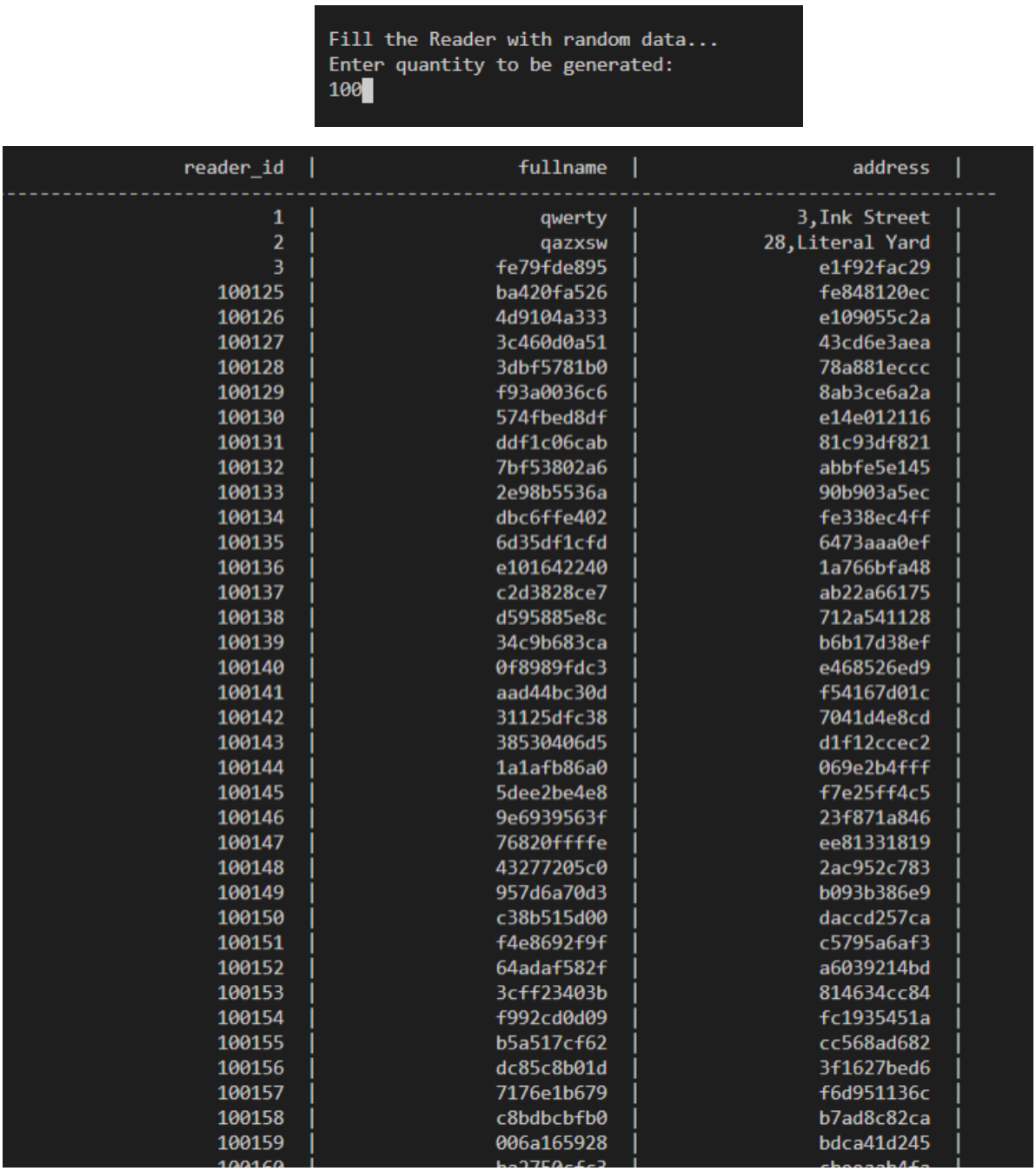


Рис. 8-9. Виконання методу UPDATE при роботі із сутністю Authors та її успішне виконання



reader_id	fullname	address
1	qwerty	3,Ink Street
2	qazxsw	28,Literal Yard
3	fe79fde895	e1f92fac29
100125	ba420fa526	fe848120ec
100126	4d9104a333	e109055c2a
100127	3c460d0a51	43cd6e3aea
100128	3dbf5781b0	78a881eccc
100129	f93a0036c6	8ab3ce6a2a
100130	574fbed8df	e14e012116
100131	ddf1c06cab	81c93df821
100132	7bf53802a6	abbfe5e145
100133	2e98b5536a	90b903a5ec
100134	dbc6ffe402	fe338ec4ff
100135	6d35df1cfd	6473aaa0ef
100136	e101642240	1a766bfa48
100137	c2d3828ce7	ab22a66175
100138	d595885e8c	712a541128
100139	34c9b683ca	b6b17d38ef
100140	0f8989fdc3	e468526ed9
100141	aad44bc30d	f54167d01c
100142	31125dfc38	7041d4e8cd
100143	38530406d5	d1f12ccec2
100144	1a1afb86a0	069e2b4fff
100145	5dee2be4e8	f7e25ff4c5
100146	9e6939563f	23f871a846
100147	76820ffffe	ee81331819
100148	43277205c0	2ac952c783
100149	957d6a70d3	b093b386e9
100150	c38b515d00	daccd257ca
100151	f4e8692f9f	c5795a6af3
100152	64adaf582f	a6039214bd
100153	3cff23403b	814634cc84
100154	f992cd0d09	fc1935451a
100155	b5a517cf62	cc568ad682
100156	dc85c8b01d	3f1627bed6
100157	7176e1b679	f6d951136c
100158	c8bdbcbfb0	b7ad8c82ca
100159	006a165928	bdca41d245

Рис. 10-11. Заповнення таблиці із сутністю Readers «рандомізованими даними» та вивід частини таблиці

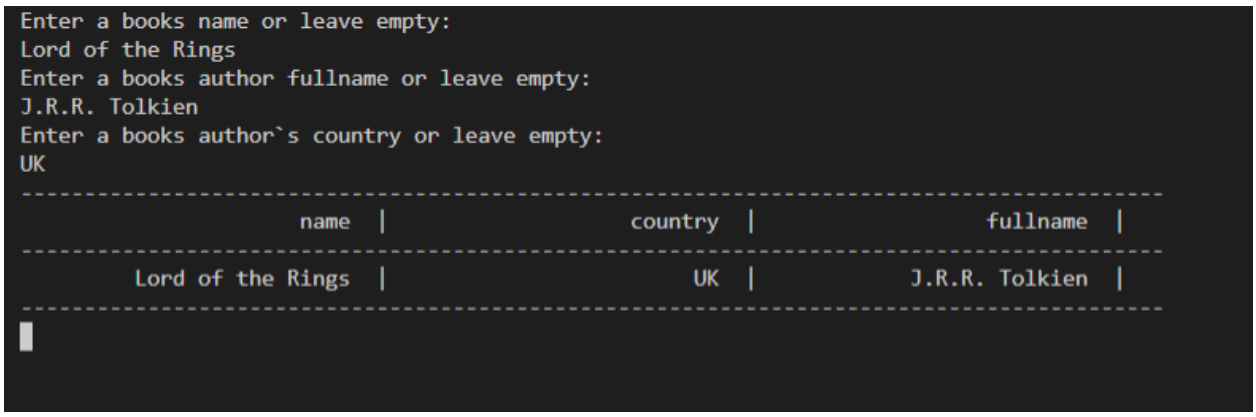


Рис. 12. Введення пошукового запиту та результат його виконання

Лістинг програми

Model.py

```
import os
import psycopg2

class Model:
    def __init__(self):
        try:
            self.connection = psycopg2.connect(host="localhost", port="5432"
, database='Library', user='postgres', password='qwerty21qwerty')
            self.cursor = self.connection.cursor()
        except (Exception, psycopg2.Error) as error:
            print('Error while connecting to PostgreSQL', error)

    def get_column_names(self):
        return [d[0] for d in self.cursor.description]

    def create_data_base(self):
        f = open('library.txt', 'r')
        self.cursor.execute(f.read())
        self.connection.commit()

    def get(self, table_name, param, arg):
        try:
            query = f'SELECT {param} FROM public."{table_name}"'
            if arg:
                query += ' WHERE ' + arg
            self.cursor.execute(query)
            return self.get_column_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def insert(self, table_name, columns, values):
        try:
            query = f'INSERT INTO public. "{table_name}" ({columns}) VALUES
({values});'
            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def delete(self, table_name, condition):
        try:
            query = f'DELETE FROM public. "{table_name}" WHERE {condition};'
            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def update(self, table_name, value, new_value):
        try:
```

```
            query = f'UPDATE public. "{table_name}" SET {new_value} WHERE {v
alue}'
            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def filter_books_table(self, name, author_fullname, author_country):
        query = f"""
        SELECT b.name, a.country, a.fullname
        FROM public."Books" b
        INNER JOIN public."Authors_books" l ON b.book_id=l.book_id
        INNER JOIN public."Authors" a ON l.author_id=a.author_id
        """
        if name:
            query += f'\nWHERE name LIKE\'%{name}%\''
        if author_fullname and not name:
            query += f'\nWHERE author_fullname LIKE \'%{author_fullname}%\''
        elif author_fullname:
            query += f' AND author_fullname LIKE \'%{author_fullname}%\''
        if author_country and not name and not author_fullname:
            query += f'\nWHERE author_country LIKE \'%{author_country}%\''
        elif author_country:
            query += f' AND author_country LIKE \'%{author_country}%\''
        try:
            self.cursor.execute(query)
            return self.get_column_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def fill_readers_table_with_random_data(self, quantity):
        sql = f"""
        CREATE OR REPLACE FUNCTION randomReaders()
            RETURNS void AS $$
        DECLARE
            step integer := 1;
        BEGIN
            LOOP EXIT WHEN step > {quantity};
                INSERT INTO public."Readers" (fullname, address)
                VALUES (
                    substring(md5(random()::text), 1, 10),
                    substring(md5(random()::text), 1, 10)
                );
                step := step + 1;
            END LOOP;
        END;
        $$ LANGUAGE PLPGSQL;
        SELECT randomReaders();
        """
        try:
            self.cursor.execute(sql)
        finally:
```

```
            self.connection.commit()
```

View.py

```
class View:
    def print_data(self, data):
        columns, rows = data
        line_length = 30 * len(columns)

        self.print_separator(line_length)
        self.print_row(columns)
        self.print_separator(line_length)

        for row in rows:
            self.print_row(row)

        self.print_separator(line_length)

    def print_row(self, row):
        for column in row:
            print(str(column).rjust(26, ' ') + '  |', end='')
        print('')

    def print_separator(self, length):
        print('-' * length)
```

Controller.py

```
from consolemenu import SelectionMenu
from view import View
from model import Model

TABLES_NAMES = ['Authors', 'Books', 'Authors_books', 'Readers',
'Subscriptions', 'Book_subscription_links']
TABLES = {
    'Authors':['author_id', 'fullname', 'birth_date', 'country'],
    'Books':['book_id', 'name', 'publish_date', 'quantity'],
    'Authors_books':['id', 'author_id', 'book_id'],
    'Readers':['reader_id', 'fullname', 'address'],
    'Subscriptions':['subscription_id', 'start_date', 'end_date',
'reader_id'],
    'Book_subscription_links':['id', 'book_id', 'subscription_id']
}

def get_input(table_name=''):
    if table_name:
```

```
        print(' '.join(TABLES[table_name]), end='\n\n')
    return input()

def get_insert_input(table_name):
    print(' '.join(TABLES[table_name]), end='\n\n')
    return input(), input()

def press_enter():
    input()

class Controller:
    def __init__(self):
        self.model = Model()
        self.view = View()

    def show_start_menu(self, input=''):
        selection_menu = SelectionMenu(
            TABLES_NAMES + ['Find books by name or author`s fullname',
            'Fill the Readers (random)'],
            title = 'Input:',
            subtitle=input)

        selection_menu.show()
        option = selection_menu.selected_option
        if option < len(TABLES_NAMES):
            table_name = TABLES_NAMES[option]
            self.show_entity_menu(table_name)
        elif option == 6:
            self.filter_books()
        elif option == 7:
            self.random_data_for_readers_table()
        else:
            print('')

    def show_entity_menu(self, table_name, input=''):
        options = ['GET', 'INSERT', 'DELETE', 'UPDATE']
        methods = [self.get, self.insert, self.delete, self.update]

        selection_menu = SelectionMenu(options,
        f'Table: {table_name}',
        exit_option_text = 'Back',
        subtitle = input)
        selection_menu.show()

        try:
            method = methods[selection_menu.selected_option]
            method(table_name)
        except IndexError:
            self.show_start_menu()

    def get(self, table_name):
```

```
        try:
            data = self.model.get(table_name, '*', '')
            self.view.print_data(data)
            press_enter()
            self.show_entity_menu(table_name)
        except Exception as error:
            self.show_entity_menu(table_name, str(error))

    def insert(self, table_name):
        try:
            print(f'Insert into the {table_name}...\nEnter columns (column1)
- enter - and values (\'value1\') divided with commas.')
            columns, values = get_insert_input(table_name)
            self.model.insert(table_name, columns, values)
            self.show_entity_menu(table_name, 'Successfully inserted!')
        except Exception as error:
            self.show_entity_menu(table_name, str(error))

    def delete(self, table_name):
        try:
            print(f'Delete from the {table_name}...\nEnter condition
(SQL):')
            condition = get_input(table_name)
            self.model.delete(table_name, condition)
            self.show_entity_menu(table_name, 'Successfully deleted!')
        except Exception as error:
            self.show_entity_menu(table_name, str(error))

    def update(self, table_name):
        try:
            print(f'Update the {table_name}...\nEnter condition (SQL):')
            condition = get_input(table_name)
            print('Enter what to update (key=\'value\'):')
            to_update = get_input(table_name)
            self.model.update(table_name, condition, to_update)
            self.show_entity_menu(table_name, 'Successfully updated!')
        except Exception as error:
            self.show_entity_menu(table_name, str(error))

    def filter_books(self):
        try:
            print('Enter a books name or leave empty:')
            books_name = get_input()
            print('Enter a books author fullname or leave empty:')
            books_author_fullname = get_input()
            print('Enter a books author`s country or leave empty:')
            author_country = get_input()
            data = self.model.filter_books_table(books_name,
books_author_fullname, author_country)
            self.view.print_data(data)
            press_enter()
```

```
            self.show_start_menu()
        except Exception as error:
            self.show_start_menu(str(error))

    def random_data_for_readers_table(self):
        try:
            print('Fill the Reader with random data...\nEnter quantity to be
generated:')
            readers_quantiy = get_input()
            self.model.fill_readers_table_with_random_data(readers_quantiy)
            self.show_start_menu('Successfully generated random Readers!')
        except Exception as error:
            self.show_start_menu(str(error))
```

Main.py

```
from controller import Controller

Controller().show_start_menu()
```