МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп’ютерних систем

Лабораторна робота № 2

з дисципліни «Бази даних і засоби управління»

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

Виконала: Денисенко Олександра
Студентка групи КВ-91
Перевірив: Павловський В.І.

Київ 2021

Лабораторна робота №2

Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL

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

Загальне завдання роботи полягає у наступному:

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

Посилання на репозиторій GitHub з вихідним кодом: https://github.com/OleksandraDenysenko/Data-Base.git

Логічна модель предметної області «Школа»

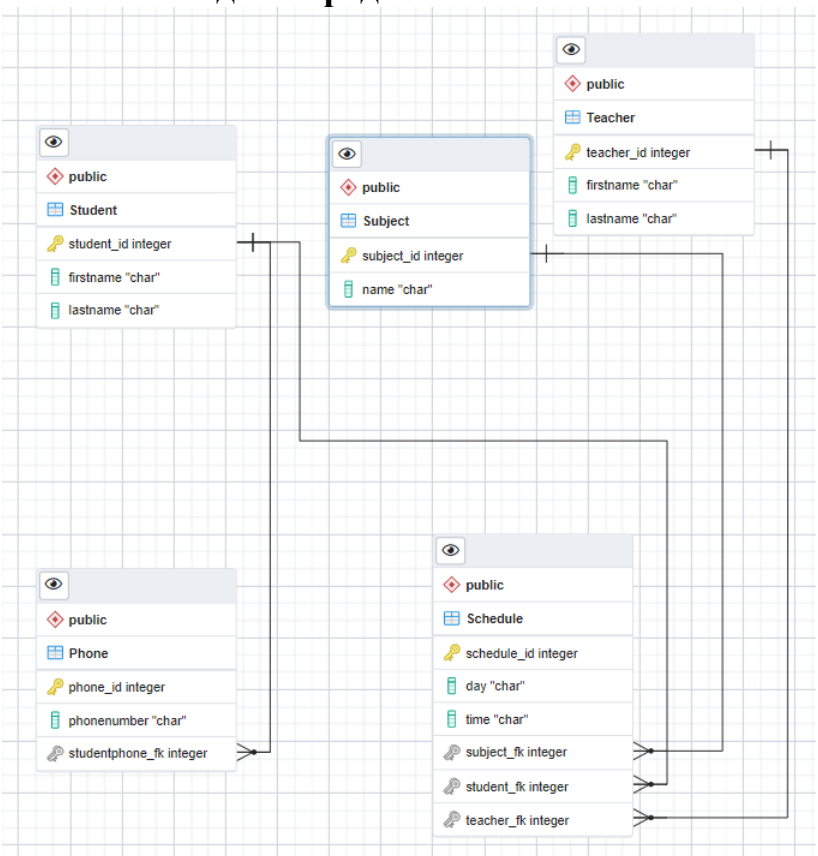

Рисунок 1 – Логічна модель предметної області «Школа»

Зміни у порівнянні з першою лабораторною роботою відсутні.

Середовище розробки та налаштування підключення до бази даних

Для виконання лабораторної роботи використовувалась мова програмування Python та середовище розробки PyCharm 2021.2.1.

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

Шаблон проектування

MVC - Шаблон проектування, який використаний у програмі.

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

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

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

Структура програми та її опис

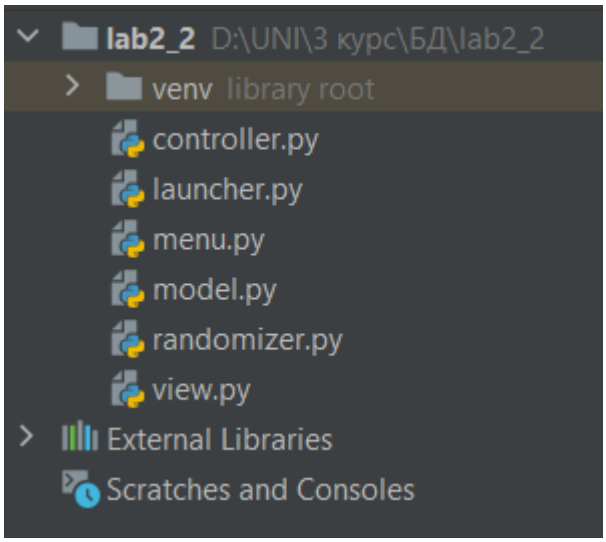

Рисунок 2 – структура програми

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

5. model.py – виконує складні операції з базою даних з конкретними таблицями БД.
6. randomizer.py – реалізовує функцію внесення випадкових даних до БД.

Структура меню програми

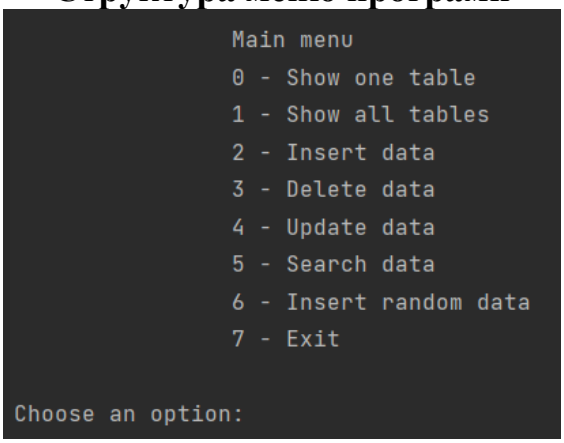

Рисунок 3 – Початкове меню програми

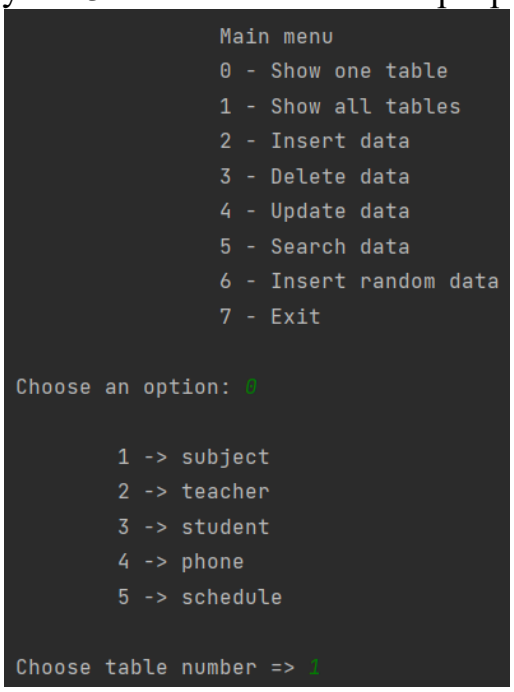

Рисунок 4 – Меню для вибору таблиці для виведення даних

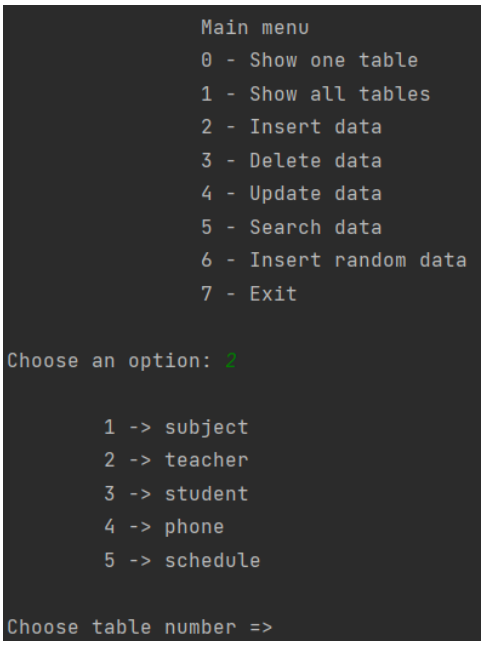

Рисунок 5 – Меню для вибору таблиці для внесення даних

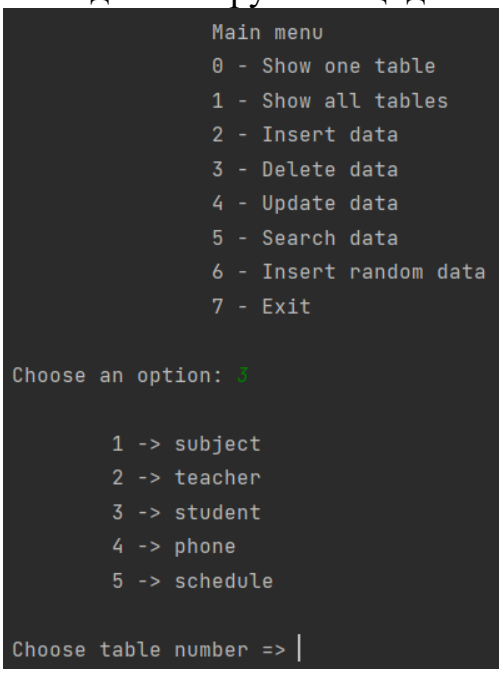

Рисунок 6 – Меню для вибору таблиці для видалення даних

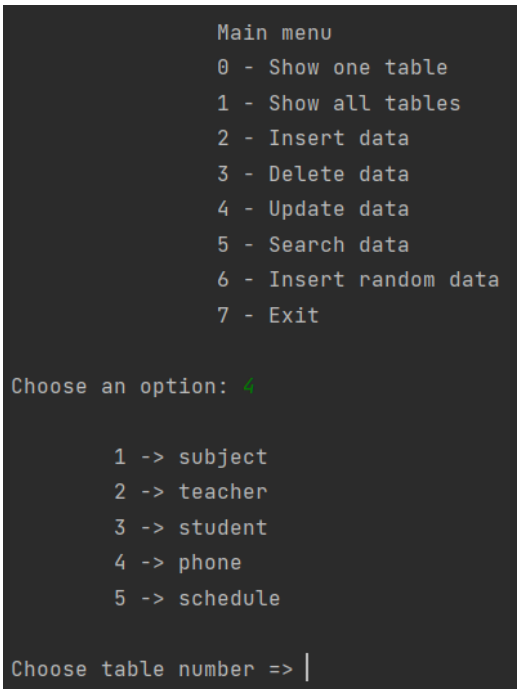

Рисунок 7 – Меню для вибору таблиці для редагування даних

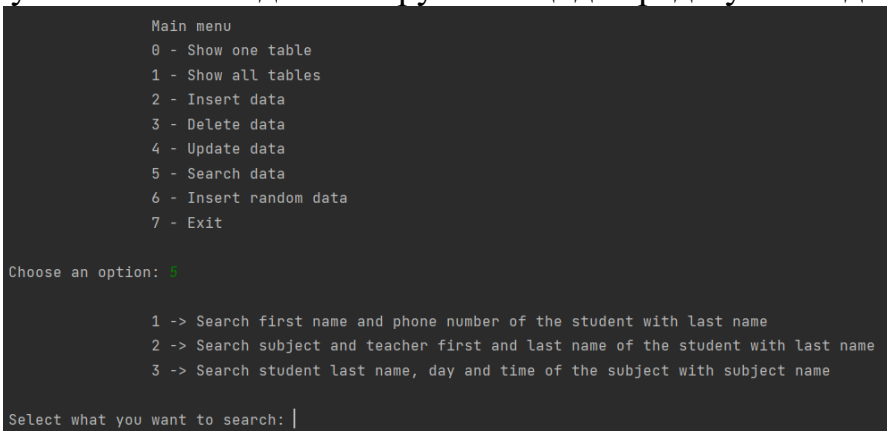

Рисунок 8 – Меню для пошуку даних

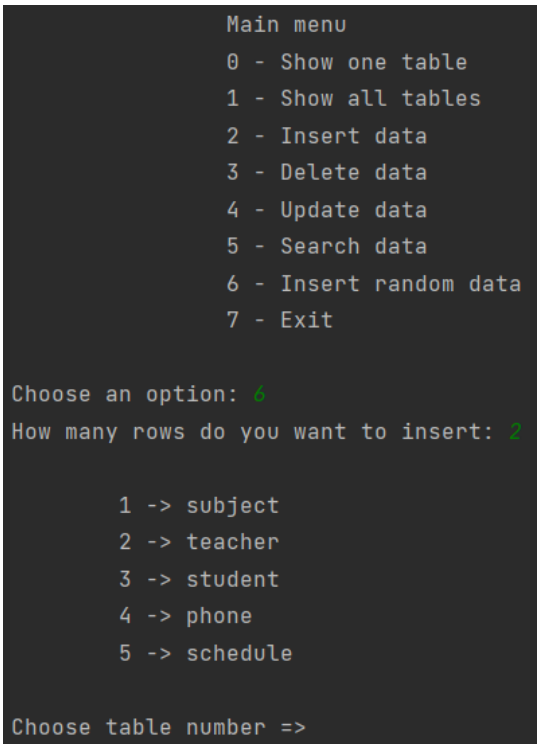

Рисунок 9 – Меню для внесення випадкових даних

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

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

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

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

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

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

Шостий пункт («5») пропонує пошук за атрибутами з декількох таблиць (Рисунок 8). Користувач обирає, який запит він хоче виконувати, після чого вводить дані для пошуку. Після введення атрибуту користувач може продовжити

пошук або повернутися до головного меню і вивести таблицю за допомогою опції «0» або вивести всі таблиці за допомогою опції «1».

Сьомий пункт («6») пропонує внесення випадкових даних (Рисунок 9). Користувач обирає, в яку саме таблицю потрібно внести дані та скільки рядків даних він хоче додати. Після цього відбувається внесення даних у відповідну таблицю.

Восьмий пункт («7») пропонує вихід з програми. Закривається з'єднання з таблицею бази даних і програма завершується.

Лістинг фрагментів програми для внесення даних

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

```
@staticmethod
def insert():
    connection = controller.connection()
    cursor = connection.cursor()
    go_on = True
    while go_on:
        View.list()
        table = controller.validtable()
        if table == 1:
            id = controller.validate_input_items("subject_id")
            name = controller.validate_input_items("name")
            cursor.execute(f"""select * from public.subject where "subject_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                controller.message("ID already exists")
            else:
                cursor.execute(f"""INSERT INTO "subject" ("subject_id", "name")
                               VALUES ('{id}', '{name}')""")
                View.complete_message("subject_id", id, "subject", "inserted")
            go_on = False
        elif table == 2:
            id = controller.validate_input_items("teacher_id")
            firstname = controller.validate_input_items("firstname")
            lastname = controller.validate_input_items("lastname")
            cursor.execute(f"""select * from public.teacher where "teacher_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                controller.message("ID already exists")
            else:
                cursor.execute(f"""INSERT INTO "teacher" ("teacher_id",
"firstname", "lastname")
                VALUES ('{id}', '{firstname}', '{lastname}')""")
                View.complete_message("teacher_id", id, "teacher", "inserted")
            go_on = False
        elif table == 3:
```

```
            id = controller.validate_input_items("student_id")
            firstname = controller.validate_input_items("firstname")
            lastname = controller.validate_input_items("lastname")
            cursor.execute(f"""select * from public.student where "student_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                controller.message("ID already exists")
            else:
                cursor.execute(f"""INSERT INTO "student" ("student_id",
"firstname", "lastname")
                VALUES ('{id}', '{firstname}', '{lastname}')""")
                View.complete_message("student_id", id, "student", "inserted")
            go_on = False
        elif table == 4:
            id = controller.validate_input_items("phone_id")
            studentphone_fk = controller.validate_input_items("student_id")
            phonenumber = controller.validate_input_items("phonenumber")
            cursor.execute(f"""select * from public.phone where "phone_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                controller.message("ID already exists")
            else:
                cursor.execute(f"""select * from public.student where
"student_id" = '{studentphone_fk}' """)
                records = cursor.fetchone()
                if records is None:
                    controller.message("The student with this ID doesn't exist")
                else:
                    cursor.execute(f"""INSERT INTO "phone" ("phone_id",
"phonenumber", "studentphone_fk")
                    VALUES ('{id}', '{phonenumber}', '{studentphone fk}')""")
                    View.complete_message("phone_id", id, "phone", "inserted")
            go_on = False
        elif table == 5:
            id = controller.validate_input_items("schedule_id")
            day = controller.validate_input_items("day")
            time = controller.validate_input_items("time")
            scstudent = controller.validate_input_items("student_id")
            scteacher = controller.validate_input_items("teacher_id")
            scsubject = controller.validate_input_items("subject_id")
            cursor.execute(f"""select * from public.schedule where "schedule_id"
= '{id}' """)
            records = cursor.fetchone()
            if records is not None:
                controller.message("ID already exists")
            else:
                cursor.execute(f"""select * from public.student where
"student_id" = '{scstudent}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""select * from public.teacher where
"teacher_id" = '{scteacher}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""select * from public.subject where
"subject_id" = '{scsubject}' """)
                        records = cursor.fetchone()
                        if records is not None:
                            cursor.execute(f"""INSERT INTO "schedule"
("schedule_id", "day", "time", "student_fk",
                            "teacher_fk", "subject_fk") VALUES ('{id}', '{day}',
```

```
'{time}', '{scstudent}',
                                '{scteacher}', '{scsubject}')""")
                                View.complete_message("schedule_id", id, "schedule",
"inserted")
                            else:
                                controller.message("The subject with this ID doesn't
exist")
                        else:
                            controller.message("The teacher with this ID doesn't
exist")
                    else:
                        controller.message("The student with this ID doesn't exist")
                go_on = False
            else:
                print('Please, enter valid value')
        connection.commit()
        cursor.close()
        controller.disconnect(connection)
```

Лістинг фрагментів програми для вилучення даних

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

```
@staticmethod
def delete():
    connection = controller.connection()
    cursor = connection.cursor()
    go_on = True
    while go_on:
        View.list()
        table = controller.validtable()
        if table == 1:
            id = controller.validate_input_items("subject_id")
            cursor.execute(f"""select * from public.subject where "subject_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                cursor.execute(f"""select * from public.schedule where
"subject_fk" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.schedule WHERE
"subject_fk" = '{id}' """)
                    View.complete_message("subject_fk", id, "schedule",
"deleted")
                cursor.execute(f"""DELETE FROM public.subject WHERE "subject_id"
= '{id}'""")
                View.complete_message("subject_id", id, "subject", "deleted")
            else:
                controller.message("No ID found")
            go_on = False
```

```
        elif table == 2:
            id = controller.validate_input_items("teacher_id")
            cursor.execute(f"""select * from public.teacher where "teacher_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                cursor.execute(f"""select * from public.schedule where
"teacher_fk" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.schedule WHERE
"teacher_fk" = '{id}' """)
                    View.complete_message("teacher_fk", id, "schedule",
"deleted")
                cursor.execute(f"""DELETE FROM public.teacher WHERE "teacher_id"
= '{id}'""")
                View.complete_message("teacher_id", id, "teacher", "deleted")
            else:
                controller.message("No ID found")
            go_on = False
        elif table == 3:
            id = controller.validate_input_items("student_id")
            cursor.execute(f"""select * from public.student where "student_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                cursor.execute(f"""select * from public.phone where
"studentphone_fk" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.phone WHERE
"studentphone_fk" = '{id}'""")
                    View.complete message("studentphone fk", id, "phone",
"deleted")
                cursor.execute(f"""select * from public.schedule where
"student_fk" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.schedule WHERE
"student_fk" = '{id}'""")
                    View.complete_message("student_fk", id, "schedule",
"deleted")
                cursor.execute(f"""DELETE FROM public.student WHERE "student_id"
= '{id}'""")
                View.complete_message("student_id", id, "student", "deleted")
            else:
                controller.message("No ID found")
            go_on = False
        elif table == 4:
            id = controller.validate_input_items("phone_id")
            cursor.execute(f"""select * from public.phone where "phone_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                cursor.execute(f"""DELETE FROM public.phone WHERE "phone_id" =
'{id}'""")
                View.complete_message("phone_id", id, "phone", "deleted")
            else:
                controller.message("No ID found")
            go_on = False
        elif table == 5:
            id = controller.validate_input_items("schedule_id")
            cursor.execute(f"""select * from public.schedule where "schedule_id"
```

```
= '{id}' """)
            records = cursor.fetchone()
            if records is not None:
                cursor.execute(f"""DELETE FROM public.schedule WHERE
"schedule_id" = '{id}'""")
                View.complete_message("schedule_id", id, "schedule", "deleted")
            else:
                controller.message("No ID found")
            go_on = False
        else:
            print("Please, enter valid number")
        connection.commit()
        cursor.close()
        controller.disconnect(connection)
```

Лістинг фрагментів програми для редагування даних

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

```
@staticmethod
def update():
    connection = controller.connection()
    cursor = connection.cursor()
    go_on = True
    while go_on:
        View.list()
        table = controller.validtable()
        if table == 1:
            id = controller.validate_input_items("subject_id")
            cursor.execute(f"""select * from public.subject where "subject_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                value = controller.validate_input_items("name")
                cursor.execute(f"""UPDATE public.subject set 'name' = '{value}'
where "subject_id" = '{id}' """)
                View.complete_message("subject_id", id, "subject", "updated")
                go_on = False
            else:
                controller.message("No subject with this ID found")

        elif table == 2:
            id = controller.validate_input_items("teacher_id")
            cursor.execute(f"""select * from public.teacher where "teacher_id" =
{id} """)
            records = cursor.fetchone()
            if records is not None:
                View.columns(2)
                continue_update = True
```

```
                while continue_update:
                    attr = input("Choose a number of column to update")
                    if attr == '1':
                        value = controller.validate_input_items("firstname")
                        attribute = "firstname"
                        continue_update = False
                    elif attr == '2':
                        value = controller.validate_input_items("lastname")
                        attribute = "lastname"
                        continue_update = False
                    else:
                        controller.message("Please, enter valid number")
                cursor.execute(f"""UPDATE "teacher" set {attribute} = '{value}'
where "teacher_id" = '{id}' """)
                View.complete_message("teacher_id", id, "teacher", "updated")
                go_on = False
                pass
            else:
                controller.message("No teacher with this ID found")
        elif table == 3:
            id = controller.validate_input_items("student_id")
            cursor.execute(f"""select * from public.student where "student_id" =
{id} """)
            records = cursor.fetchone()
            if records is not None:
                View.columns(3)
                continue_update = True
                while continue_update:
                    attr = input("Choose a number of column to update: ")
                    if attr == '1':
                        value = controller.validate_input_items("firstname")
                        attribute = "firstname"
                        continue update = False
                    elif attr == '2':
                        value = controller.validate_input_items("lastname")
                        attribute = "lastname"
                        continue_update = False
                    else:
                        controller.message("Please, enter valid number")
                cursor.execute(f"""UPDATE "student" set {attribute} = '{value}'
where "student_id" = '{id}' """)
                View.complete_message("student_id", id, "student", "updated")
                go_on = False
                pass
            else:
                controller.message("No student with this ID found")
        elif table == 4:
            id = controller.validate_input_items("phone_id")
            cursor.execute(f"""select * from public.phone where "phone_id" =
{id} """)
            records = cursor.fetchone()
            if records is not None:
                View.columns(4)
                continue_update = True
                while continue_update:
                    attr = input("Choose a number of column to update: ")
                    if attr == '1':
                        value = controller.validate_input_items("phonenumber")
                        attribute = "phonenumber"
                        continue_update = False
                    elif attr == '2':
                        value = controller.validate_input_items("student_id")
                        attribute = "studentphone_fk"
```

```
                        continue_update = False
                    else:
                        controller.message("Please, enter valid number: ")
                cursor.execute(f"""UPDATE "phone" set {attribute} = '{value}'
where "phone_id" = '{id}' """)
                View.complete_message("phone_id", id, "phone", "updated")
                go_on = False
                pass
            else:
                controller.message("No phone with this ID found")
        elif table == 5:
            id = controller.validate_input_items("schedule_id")
            cursor.execute(f"""select * from public.schedule where "schedule_id"
= {id} """)
            records = cursor.fetchone()
            if records is not None:
                View.columns(5)
                continue_update = True
                while continue_update:
                    attr = input("Choose a number of column to update ")
                    if attr == '1':
                        value = controller.validate_input_items("day")
                        attribute = "day"
                        continue_update = False
                    elif attr == '2':
                        value = controller.validate_input_items("time")
                        attribute = "time"
                        continue_update = False
                    elif attr == '3':
                        value = controller.validate_input_items("subject_id")
                        attribute = "subject_fk"
                        continue_update = False
                    elif attr == '4':
                        value = controller.validate_input_items("student_id")
                        attribute = "student_fk"
                        continue_update = False
                    elif attr == '5':
                        value = controller.validate_input_items("teacher_id")
                        attribute = "teacher_fk"
                        continue_update = False
                    else:
                        controller.message("Please, enter valid number")
                cursor.execute(f"""UPDATE "schedule" set {attribute} = '{value}'
where "schedule_id" = '{id}' """)
                View.complete_message("schedule_id", id, "schedule", "updated")
                go_on = False
                pass
            else:
                controller.message("No schedule with this ID found")
        else:
            controller.message("Please, enter valid number")
    connection.commit()
    cursor.close()
    controller.disconnect(connection)
    pass
```

Лістинг фрагментів програми для пошуку даних

Користувачеві пропонується обрати варіант пошуку: знайти ім’я та номер телефону учня за його прізвищем; знайти назву предмета, прізвище та ім’я вчителя за прізвищем учня; знайти день тижня та час, коли проводиться певний

предмет, а також учня, який на ньому навчається, за назвою предмета. Також розраховується час запиту кожного пошуку в мілісекундах.

Пошук імені та номера телефону за прізвищем учня

Запит цього пошуку реалізовано за допомогою функції search1. Користувач вводить прізвище учня, після чого виводиться таблиця із знайденим іменем та номером телефону учня, а також час виконання операції пошуку в мілісекундах. Якщо запис із заданим прізвищем не існує, користувачу буде виведено повідомлення про помилку і запропоновано або продовжити пошук, або повернутися до головного меню.

```
@staticmethod
def search1():
    connection = controller.connection()
    cursor = connection.cursor()
    lastname = controller.validate_input_items("lastname")
    cursor.execute(f"""select * from public.student where "lastname" =
'{lastname}' """)
    records = cursor.fetchone()
    if records is not None:
        search = f"""select "lastname", "firstname", "phonenumber" from
                (select c."lastname", c."firstname", p."phonenumber" from
public.phone
                p left join public.student c on c."student_id" =
p."studentphone_fk"
                where c."lastname" LIKE '{lastname}' group by c."lastname",
c."firstname", p."phonenumber")
                as foo"""
    else:
        controller.message("No student with this last name found")
    start = int(time.time() * 1000)
    cursor.execute(search)
    print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
    records = cursor.fetchall()
    cursor.close()
    return records
```

Пошук назви предмета, прізвища та імені вчителя за прізвищем учня

Запит цього пошуку реалізовано за допомогою функції search2. Користувач вводить прізвище учня, після чого виводиться таблиця із знайденим предметом, прізвищем та іменем вчителя, а також час виконання операції пошуку в мілісекундах. Якщо запис із заданим прізвищем не існує, користувачу буде виведено повідомлення про помилку і запропоновано або продовжити пошук, або повернутися до головного меню.

```
@staticmethod
def search2():
    connection = controller.connection()
    cursor = connection.cursor()
    lastname = controller.validate_input_items("lastname")
    cursor.execute(f"""select * from public.student where "lastname" =
'{lastname}' """)
    records = cursor.fetchone()
    if records is not None:
        search = f"""select c."lastname", s."name", t."firstname", t."lastname"
from public.schedule p inner
        join public.subject s on s."subject_id" = p."subject_fk" inner join
public.teacher t on
```

```
        t."teacher_id" = p."teacher_fk" inner join public.student c on
c."student_id" = p."student_fk"
        where c."lastname" LIKE '{lastname}' group by c."lastname",
        s."name", t."firstname", t."lastname" """
    else:
        controller.message("No student with this last name found")
    start = int(time.time() * 1000)
    cursor.execute(search)
    print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
    records = cursor.fetchall()
    cursor.close()
    return records
```

Пошук дня тижня та часу, коли проводиться певний предмет, а також учня, який на ньому навчається, за назвою предмета

Запит цього пошуку реалізовано за допомогою функції search3. Користувач вводить назву предмета, після чого виводиться таблиця із днем тижня та часом, коли проводиться даний предмет, прізвище учня, який на ньому навчається, а також час виконання операції пошуку в мілісекундах. Якщо запис із заданим прізвищем не існує, користувачу буде виведено повідомлення про помилку і запропоновано або продовжити пошук, або повернутися до головного меню.

```
@staticmethod
def search3():
    connection = controller.connection()
    cursor = connection.cursor()
    name = controller.validate_input_items("name")
    cursor.execute(f"""select * from public.subject where "name" = '{name}' """)
    records = cursor.fetchone()
    if records is not None:
        search = f"""select s."name", c."lastname", p."day", p."time" from
public.schedule p inner join
        public.subject s on s."subject_id" = p."subject_fk" inner join
public.student c on
        c."student_id" = p."student_fk" where s."name" LIKE '{name}' group by
s."name", c."lastname", p."day",
        p."time" """
    else:
        controller.message("No subject with this name found")
    start = int(time.time() * 1000)
    cursor.execute(search)
    print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
    records = cursor.fetchall()
    cursor.close()
    return records
```

Лістинг фрагментів програми для внесення випадкових даних

Внесення випадкових даних реалізовано за допомогою функції random у файлі randomizer.py. Спочатку користувач має обрати, скільки рядків даних він хоче додати. Після цього обирається таблиця, в яку будуть вноситися дані. Є контроль зовнішніх та внутрішніх ключів, що робить їх генерацію обмеженою в рамках можливих. Коли дані успішно внесені, висвічується відповідне повідомлення. Користувач може або продовжити внесення випадкових даних, або повернутися до головного меню.

```
@staticmethod
def random(table, count):
    connection = controller.connection()
    cursor = connection.cursor()
    go_on = True
    while go_on:
        if table == 1:
            for i in range(count):
                cursor.execute("""insert into public.subject
                                  select (SELECT MAX("subject_id")+1 FROM
public.subject),
                                  array_to_string(ARRAY(SELECT chr((97 +
round(random()*25)) :: integer)
                                  FROM
                                  generate_series(1, FLOOR(RANDOM()*(15-5)+5)::
integer)), '')""")
            go_on = False
        elif table == 2:
            for i in range(count):
                cursor.execute("""insert into public.teacher
                select (SELECT MAX("teacher_id")+1 FROM public.teacher),
                array_to_string(ARRAY(SELECT chr((97 + round(random()*25)) ::
integer)
                FROM
                generate_series(1, FLOOR(RANDOM()*(15-5)+5):: integer)), ''),
                array_to_string(ARRAY(SELECT chr((97 + round(random()*25)) ::
integer)
                FROM
                generate_series(1, FLOOR(RANDOM()*(15-5)+5):: integer)), '')""")
            go_on = False
        elif table == 3:
            for i in range(count):
                cursor.execute("""insert into public.student
                                    select (SELECT MAX("student_id")+1 FROM
public.student),
                                    array_to_string(ARRAY(SELECT chr((97 +
round(random()*25)) :: integer)
                                    FROM
                                    generate_series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), ''),
                                    array_to_string(ARRAY(SELECT chr((97 +
round(random()*25)) :: integer)
                                    FROM
                                    generate_series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), '')""")
            go_on = False
        elif table == 4:
            for i in range(count):
                cursor.execute("""insert into public.phone
                                    select (SELECT MAX("phone_id")+1 FROM
public.phone),
                                    array_to_string(ARRAY(SELECT chr((48 +
round(random()*9)) :: integer)
                                    FROM
                                    generate_series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), ''),
                                    (SELECT "student_id" FROM public.student
LIMIT 1 OFFSET (round(random() *
                                    ((SELECT COUNT("student_id") FROM
public.student)-1))))""")
            go_on = False
        elif table == 5:
            for i in range(count):
```

```
                cursor.execute("""insert into public.schedule
                                  select (SELECT MAX("schedule_id")+1 FROM
public.schedule),
                                  array_to_string(ARRAY(SELECT chr((97 +
round(random()*25)) :: integer)
                                  FROM
                                  generate_series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), ''),
                                  array_to_string(ARRAY(select
chr(trunc(48+random()*6)::int) ||
                                  chr(trunc(48+random()*10)::int) || ':' ||
chr(trunc(48+random()*6)::int) ||
                                  chr(trunc(48+random()*10)::int)), ''),
                                  (SELECT "subject_id" FROM public.subject
LIMIT 1 OFFSET (round(random() *
                                  ((SELECT COUNT("subject_id") FROM
public.subject)-1)))),
                                  (SELECT "student_id" FROM public.student
LIMIT 1 OFFSET (round(random() *
                                  ((SELECT COUNT("student_id") FROM
public.student)-1)))),
                                  (SELECT "teacher_id" FROM public.teacher
LIMIT 1 OFFSET (round(random() *
                                  ((SELECT COUNT("teacher_id") FROM
public.teacher)-1))))
                                  """)
            go_on = False
        else:
            controller.message("Please, enter valid number: ")
    print("inserted random data successfully")
    connection.commit()
    cursor.close()
    controller.disconnect(connection)
```

Лістинг модуля "Model" згідно із шаблоном MVC

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

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

model.py

```
import controller
import time
from view import View
from pandas import DataFrame
from tabulate import tabulate
from numpy import array
```

```
tables = {
    1: 'subject',
    2: 'teacher',
    3: 'student',
    4: 'phone',
    5: 'schedule',
}


class Model:
    @staticmethod
    def display_query(rows, headers):
        df = DataFrame([array(el) for el in rows], columns=array(headers))
        print(tabulate(df, headers="keys", tablefmt="pretty", showindex=False))

    @staticmethod
    def show_table(table):
        connection = controller.connection()
        cursor = connection.cursor()
        print(f"Request: SELECT * FROM {table}")
        print(f"\033[1m {table} \033[0m")
        cursor.execute(f"""SELECT * from public.{table}""")
        records = cursor.fetchall()
        cursor.close()
        return records

    @staticmethod
    def insert():
        connection = controller.connection()
        cursor = connection.cursor()
        go_on = True
        while go on:
            View.list()
            table = controller.validtable()
            if table == 1:
                id = controller.validate_input_items("subject_id")
                name = controller.validate_input_items("name")
                cursor.execute(f"""select * from public.subject where
"subject_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""INSERT INTO "subject" ("subject_id",
"name")
                                       VALUES ('{id}', '{name}')""")
                    View.complete_message("subject_id", id, "subject",
"inserted")
                go_on = False
            elif table == 2:
                id = controller.validate_input_items("teacher_id")
                firstname = controller.validate_input_items("firstname")
                lastname = controller.validate_input_items("lastname")
                cursor.execute(f"""select * from public.teacher where
"teacher_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""INSERT INTO "teacher" ("teacher_id",
"firstname", "lastname")
                    VALUES ('{id}', '{firstname}', '{lastname}')""")
```

```
                    View.complete_message("teacher_id", id, "teacher",
"inserted")
                go_on = False
            elif table == 3:
                id = controller.validate_input_items("student_id")
                firstname = controller.validate_input_items("firstname")
                lastname = controller.validate_input_items("lastname")
                cursor.execute(f"""select * from public.student where
"student_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""INSERT INTO "student" ("student_id",
"firstname", "lastname")
                    VALUES ('{id}', '{firstname}', '{lastname}')""")
                    View.complete_message("student_id", id, "student",
"inserted")
                go_on = False
            elif table == 4:
                id = controller.validate_input_items("phone_id")
                studentphone_fk = controller.validate_input_items("student_id")
                phonenumber = controller.validate_input_items("phonenumber")
                cursor.execute(f"""select * from public.phone where "phone_id" =
'{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""select * from public.student where
"student_id" = '{studentphone_fk}' """)
                    records = cursor.fetchone()
                    if records is None:
                        controller.message("The student with this ID doesn't
exist")
                    else:
                        cursor.execute(f"""INSERT INTO "phone" ("phone_id",
"phonenumber", "studentphone_fk")
                        VALUES ('{id}', '{phonenumber}',
'{studentphone_fk}')""")
                        View.complete_message("phone_id", id, "phone",
"inserted")
                go_on = False
            elif table == 5:
                id = controller.validate_input_items("schedule_id")
                day = controller.validate_input_items("day")
                time = controller.validate_input_items("time")
                scstudent = controller.validate_input_items("student_id")
                scteacher = controller.validate_input_items("teacher_id")
                scsubject = controller.validate_input_items("subject_id")
                cursor.execute(f"""select * from public.schedule where
"schedule_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""select * from public.student where
"student_id" = '{scstudent}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""select * from public.teacher where
"teacher_id" = '{scteacher}' """)
                        records = cursor.fetchone()
```

```
                        if records is not None:
                            cursor.execute(f"""select * from public.subject
where "subject_id" = '{scsubject}' """)
                            records = cursor.fetchone()
                            if records is not None:
                                cursor.execute(f"""INSERT INTO "schedule"
("schedule_id", "day", "time", "student_fk",
                                "teacher_fk", "subject_fk") VALUES ('{id}',
'{day}', '{time}', '{scstudent}',
                                '{scteacher}', '{scsubject}')""")
                                View.complete_message("schedule_id", id,
"schedule", "inserted")
                            else:
                                controller.message("The subject with this ID
doesn't exist")
                        else:
                            controller.message("The teacher with this ID doesn't
exist")
                    else:
                        controller.message("The student with this ID doesn't
exist")
                go_on = False
            else:
                print('Please, enter valid value')
        connection.commit()
        cursor.close()
        controller.disconnect(connection)

    @staticmethod
    def delete():
        connection = controller.connection()
        cursor = connection.cursor()
        go_on = True
        while go_on:
            View.list()
            table = controller.validtable()
            if table == 1:
                id = controller.validate_input_items("subject_id")
                cursor.execute(f"""select * from public.subject where
"subject_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""select * from public.schedule where
"subject_fk" = '{id}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""DELETE FROM public.schedule WHERE
"subject_fk" = '{id}' """)
                        View.complete_message("subject_fk", id, "schedule",
"deleted")
                    cursor.execute(f"""DELETE FROM public.subject WHERE
"subject_id" = '{id}'""")
                    View.complete_message("subject_id", id, "subject",
"deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            elif table == 2:
                id = controller.validate_input_items("teacher_id")
                cursor.execute(f"""select * from public.teacher where
"teacher_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
```

```
                    cursor.execute(f"""select * from public.schedule where
"teacher_fk" = '{id}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""DELETE FROM public.schedule WHERE
"teacher_fk" = '{id}' """)
                        View.complete_message("teacher_fk", id, "schedule",
"deleted")
                    cursor.execute(f"""DELETE FROM public.teacher WHERE
"teacher_id" = '{id}'""")
                    View.complete_message("teacher_id", id, "teacher",
"deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            elif table == 3:
                id = controller.validate_input_items("student_id")
                cursor.execute(f"""select * from public.student where
"student_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""select * from public.phone where
"studentphone_fk" = '{id}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""DELETE FROM public.phone WHERE
"studentphone_fk" = '{id}'""")
                        View.complete_message("studentphone_fk", id, "phone",
"deleted")
                    cursor.execute(f"""select * from public.schedule where
"student_fk" = '{id}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""DELETE FROM public.schedule WHERE
"student_fk" = '{id}'""")
                        View.complete_message("student_fk", id, "schedule",
"deleted")
                    cursor.execute(f"""DELETE FROM public.student WHERE
"student_id" = '{id}'""")
                    View.complete_message("student_id", id, "student",
"deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            elif table == 4:
                id = controller.validate_input_items("phone_id")
                cursor.execute(f"""select * from public.phone where "phone_id" =
'{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.phone WHERE "phone_id"
= '{id}'""")
                    View.complete_message("phone_id", id, "phone", "deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            elif table == 5:
                id = controller.validate_input_items("schedule_id")
                cursor.execute(f"""select * from public.schedule where
"schedule_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.schedule WHERE
```

```
"schedule_id" = '{id}'""")
                    View.complete_message("schedule_id", id, "schedule",
"deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            else:
                print("Please, enter valid number")
            connection.commit()
            cursor.close()
            controller.disconnect(connection)
            pass

    @staticmethod
    def update():
        connection = controller.connection()
        cursor = connection.cursor()
        go_on = True
        while go_on:
            View.list()
            table = controller.validtable()
            if table == 1:
                id = controller.validate_input_items("subject_id")
                cursor.execute(f"""select * from public.subject where
"subject_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    value = controller.validate_input_items("name")
                    cursor.execute(f"""UPDATE public.subject set 'name' =
'{value}' where "subject_id" = '{id}' """)
                    View.complete_message("subject_id", id, "subject",
"updated")
                    go on = False
                else:
                    controller.message("No subject with this ID found")

            elif table == 2:
                id = controller.validate_input_items("teacher_id")
                cursor.execute(f"""select * from public.teacher where
"teacher_id" = {id} """)
                records = cursor.fetchone()
                if records is not None:
                    View.columns(2)
                    continue_update = True
                    while continue_update:
                        attr = input("Choose a number of column to update")
                        if attr == '1':
                            value = controller.validate_input_items("firstname")
                            attribute = "firstname"
                            continue_update = False
                        elif attr == '2':
                            value = controller.validate_input_items("lastname")
                            attribute = "lastname"
                            continue_update = False
                        else:
                            controller.message("Please, enter valid number")
                    cursor.execute(f"""UPDATE "teacher" set {attribute} =
'{value}' where "teacher_id" = '{id}' """)
                    View.complete_message("teacher_id", id, "teacher",
"updated")
                    go_on = False
                    pass
                else:
```

```
                    controller.message("No teacher with this ID found")
            elif table == 3:
                id = controller.validate_input_items("student_id")
                cursor.execute(f"""select * from public.student where
"student_id" = {id} """)
                records = cursor.fetchone()
                if records is not None:
                    View.columns(3)
                    continue_update = True
                    while continue_update:
                        attr = input("Choose a number of column to update: ")
                        if attr == '1':
                            value = controller.validate_input_items("firstname")
                            attribute = "firstname"
                            continue_update = False
                        elif attr == '2':
                            value = controller.validate_input_items("lastname")
                            attribute = "lastname"
                            continue_update = False
                        else:
                            controller.message("Please, enter valid number")
                    cursor.execute(f"""UPDATE "student" set {attribute} =
'{value}' where "student_id" = '{id}' """)
                    View.complete_message("student_id", id, "student",
"updated")
                    go_on = False
                    pass
                else:
                    controller.message("No student with this ID found")
            elif table == 4:
                id = controller.validate_input_items("phone_id")
                cursor.execute(f"""select * from public.phone where "phone_id" =
{id} """)
                records = cursor.fetchone()
                if records is not None:
                    View.columns(4)
                    continue_update = True
                    while continue_update:
                        attr = input("Choose a number of column to update: ")
                        if attr == '1':
                            value =
controller.validate_input_items("phonenumber")
                            attribute = "phonenumber"
                            continue_update = False
                        elif attr == '2':
                            value =
controller.validate_input_items("student_id")
                            attribute = "studentphone_fk"
                            continue_update = False
                        else:
                            controller.message("Please, enter valid number: ")
                    cursor.execute(f"""UPDATE "phone" set {attribute} =
'{value}' where "phone_id" = '{id}' """)
                    View.complete_message("phone_id", id, "phone", "updated")
                    go_on = False
                    pass
                else:
                    controller.message("No phone with this ID found")
            elif table == 5:
                id = controller.validate_input_items("schedule_id")
                cursor.execute(f"""select * from public.schedule where
"schedule_id" = {id} """)
                records = cursor.fetchone()
```

```
                if records is not None:
                    View.columns(5)
                    continue_update = True
                    while continue_update:
                        attr = input("Choose a number of column to update ")
                        if attr == '1':
                            value = controller.validate_input_items("day")
                            attribute = "day"
                            continue_update = False
                        elif attr == '2':
                            value = controller.validate_input_items("time")
                            attribute = "time"
                            continue_update = False
                        elif attr == '3':
                            value =
controller.validate_input_items("subject_id")
                            attribute = "subject_fk"
                            continue_update = False
                        elif attr == '4':
                            value =
controller.validate_input_items("student_id")
                            attribute = "student_fk"
                            continue_update = False
                        elif attr == '5':
                            value =
controller.validate_input_items("teacher_id")
                            attribute = "teacher_fk"
                            continue_update = False
                        else:
                            controller.message("Please, enter valid number")
                    cursor.execute(f"""UPDATE "schedule" set {attribute} =
'{value}' where "schedule_id" = '{id}' """)
                    View.complete message("schedule id", id, "schedule",
"updated")
                    go_on = False
                    pass
                else:
                    controller.message("No schedule with this ID found")
            else:
                controller.message("Please, enter valid number")
        connection.commit()
        cursor.close()
        controller.disconnect(connection)
        pass

    @staticmethod
    def search1():
        connection = controller.connection()
        cursor = connection.cursor()
        lastname = controller.validate_input_items("lastname")
        cursor.execute(f"""select * from public.student where "lastname" =
'{lastname}' """)
        records = cursor.fetchone()
        if records is not None:
            search = f"""select "lastname", "firstname", "phonenumber" from
                    (select c."lastname", c."firstname", p."phonenumber" from
public.phone
                    p left join public.student c on c."student_id" =
p."studentphone_fk"
                    where c."lastname" LIKE '{lastname}' group by c."lastname",
c."firstname", p."phonenumber")
                    as foo"""
        else:
```

```
            controller.message("No student with this last name found")
        start = int(time.time() * 1000)
        cursor.execute(search)
        print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
        records = cursor.fetchall()
        cursor.close()
        return records

    @staticmethod
    def search2():
        connection = controller.connection()
        cursor = connection.cursor()
        lastname = controller.validate_input_items("lastname")
        cursor.execute(f"""select * from public.student where "lastname" =
'{lastname}' """)
        records = cursor.fetchone()
        if records is not None:
            search = f"""select c."lastname", s."name", t."firstname",
t."lastname" from public.schedule p inner
            join public.subject s on s."subject_id" = p."subject_fk" inner join
public.teacher t on
            t."teacher_id" = p."teacher_fk" inner join public.student c on
c."student_id" = p."student_fk"
            where c."lastname" LIKE '{lastname}' group by c."lastname",
            s."name", t."firstname", t."lastname" """
        else:
            controller.message("No student with this last name found")
        start = int(time.time() * 1000)
        cursor.execute(search)
        print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
        records = cursor.fetchall()
        cursor.close()
        return records

    @staticmethod
    def search3():
        connection = controller.connection()
        cursor = connection.cursor()
        name = controller.validate_input_items("name")
        cursor.execute(f"""select * from public.subject where "name" = '{name}'
""")
        records = cursor.fetchone()
        if records is not None:
            search = f"""select s."name", c."lastname", p."day", p."time" from
public.schedule p inner join
            public.subject s on s."subject_id" = p."subject_fk" inner join
public.student c on
            c."student_id" = p."student_fk" where s."name" LIKE '{name}' group
by s."name", c."lastname", p."day",
            p."time" """
        else:
            controller.message("No subject with this name found")
        start = int(time.time() * 1000)
        cursor.execute(search)
        print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
        records = cursor.fetchall()
        cursor.close()
        return records
```

Лістинги та скріншоти результатів виконання операції вилучення

Вилучення даних реалізовано у файлі model.py за допомогою функції delete. Користувач обирає таблицю, у якій хоче вилучити дані, а також ідентифікатор рядка, який потрібно видалити (Рисунок 6).

Якщо обрана перша таблиця «subject», пропонується ввести subject_id – ідентфікатор предмета, рядок з даними якого буде видалено. Якщо в інших таблицях є дані, що залежать від даного ідентифікатора subject_id, дані видаляються і в цих таблицях також. Про успішне видалення свідчить відповідне повідомлення. Якщо введений ідентифікатор не існує, виводиться повідомлення про помилку.

Лістинг фрагменту програми з видаленням даних у таблиці subject:

```
@staticmethod
def delete():
    connection = controller.connection()
    cursor = connection.cursor()
    go_on = True
    while go_on:
        View.list()
        table = controller.validtable()
        if table == 1:
            id = controller.validate_input_items("subject_id")
            cursor.execute(f"""select * from public.subject where "subject_id" =
'{id}' """)
            records = cursor.fetchone()
            if records is not None:
                cursor.execute(f"""select * from public.schedule where
"subject_fk" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.schedule WHERE
"subject_fk" = '{id}' """)
                    View.complete_message("subject_fk", id, "schedule",
"deleted")
                cursor.execute(f"""DELETE FROM public.subject WHERE "subject_id"
= '{id}'""")
                View.complete_message("subject_id", id, "subject", "deleted")
            else:
                controller.message("No ID found")
            go_on = False
```

Скріншоти результатів виконання операції вилучення з таблиці «subject»:

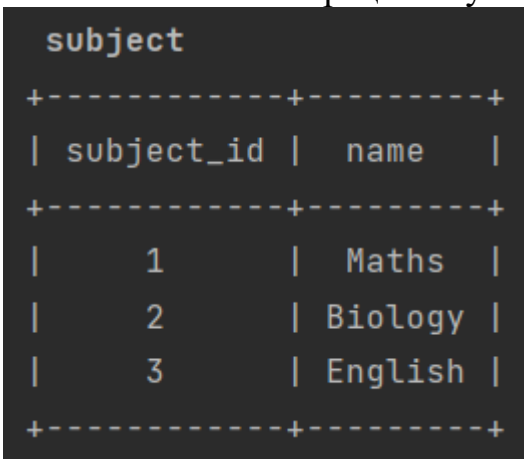

Рисунок 10 – Батьківська таблиця «subject» до вилучення даних

```
 schedule
+-------------+---------+-------+------------+------------+------------+
| schedule_id |   day   | time  | student_fk | teacher_fk | subject_fk |
+-------------+---------+-------+------------+------------+------------+
|      1      | Monday  | 14:30 |     1      |     1      |     1      |
|      2      | Tuesday | 13:00 |     2      |     2      |     2      |
+-------------+---------+-------+------------+------------+------------+
```

Рисунок 11 – Дочірня таблиця «schedule» до вилучення даних

```
Choose table number => 1
Enter subject_id: 1
The row with 'subject_fk' = '1' in table 'schedule' was deleted successfully.
The row with 'subject_id' = '1' in table 'subject' was deleted successfully.
```

Рисунок 12 – Повідомлення про успішне видалення даних з батьківської та дочірньої таблиць

```
 subject
+------------+---------+
| subject_id |  name   |
+------------+---------+
|     2      | Biology |
|     3      | English |
+------------+---------+
```

Рисунок 13 – Вміст батьківської таблиці «subject» після виконання операції вилучення даних

```
 schedule
+-------------+---------+-------+------------+------------+------------+
| schedule_id |   day   | time  | student_fk | teacher_fk | subject_fk |
+-------------+---------+-------+------------+------------+------------+
|      2      | Tuesday | 13:00 |     2      |     2      |     2      |
+-------------+---------+-------+------------+------------+------------+
```

Рисунок 14 – Вміст дочірньої таблиці «schedule» після виконання операції вилучення даних

```
Choose table number => 1
Enter subject_id: 10
No ID found
```

Рисунок 15 – Повідомлення про помилку (відсутній рядок із введеним ідентифікатором)

Якщо обрана друга таблиця «teacher», пропонується ввести teacher_id – ідентифікатор вчителя, рядок з даними якого буде видалено. Якщо в інших таблицях є дані, що залежать від даного ідентифікатора teacher_id, дані видаляються і в цих таблицях також. Про успішне видалення свідчить відповідне

повідомлення. Якщо введений ідентифікатор не існує, виводиться повідомлення про помилку.

Лістинг фрагменту програми з видаленням даних у таблиці teacher:

```
elif table == 2:
    id = controller.validate_input_items("teacher_id")
    cursor.execute(f"""select * from public.teacher where "teacher_id" = '{id}'
""")
    records = cursor.fetchone()
    if records is not None:
        cursor.execute(f"""select * from public.schedule where "teacher_fk" =
'{id}' """)
        records = cursor.fetchone()
        if records is not None:
            cursor.execute(f"""DELETE FROM public.schedule WHERE "teacher_fk" =
'{id}' """)
            View.complete_message("teacher_fk", id, "schedule", "deleted")
        cursor.execute(f"""DELETE FROM public.teacher WHERE "teacher_id" =
'{id}'""")
        View.complete_message("teacher_id", id, "teacher", "deleted")
    else:
        controller.message("No ID found")
    go_on = False
```

Скріншоти результатів виконання операції вилучення з таблиці «teacher»:

teacher

teacher_id	firstname	lastname
1	Viktoria	Ivanova
2	Margarita	Makohon

Рисунок 16 – Батьківська таблиці «teacher» до вилучення даних

schedule

schedule_id	day	time	student_fk	teacher_fk	subject_fk
3	Wednesdau	15:00	3	1	3
4	Thursday	10:00	2	2	2

Рисунок 17 – Дочірня таблиця «schedule» до вилучення даних

```
Choose table number => 2
Enter teacher_id: 2
The row with 'teacher_fk' = '2' in table 'schedule' was deleted successfully.
The row with 'teacher_id' = '2' in table 'teacher' was deleted successfully.
```

Рисунок 18 – Повідомлення про успішне видалення даних з обох таблиць

```
 teacher
+------------+-----------+----------+
| teacher_id | firstname | lastname |
+------------+-----------+----------+
|     1      | Viktoria  | Ivanova  |
+------------+-----------+----------+
```

Рисунок 19 – Батьківська таблиця «teacher» після виконання операції вилучення даних

```
 schedule
+-------------+-----------+-------+------------+------------+------------+
| schedule_id |    day    | time  | student_fk | teacher_fk | subject_fk |
+-------------+-----------+-------+------------+------------+------------+
|      3      | Wednesdau | 15:00 |     3      |     1      |     3      |
+-------------+-----------+-------+------------+------------+------------+
```

Рисунок 20 – Дочірня таблиця «schedule» після виконання операції вилучення даних

```
Choose table number => 2
Enter teacher_id: 6
No ID found
```

Рисунок 21 – Повідомлення про помилку (відсутній рядок із введеним ідентифікатором)

Якщо обрана третя таблиця «student», пропонується ввести student_id – ідентифікатор учня, рядок з даними якого буде видалено. Якщо в інших таблицях є дані, що залежать від даного ідентифікатора student_id, дані видаляються і в цих таблицях також. Про успішне видалення свідчить відповідне повідомлення. Якщо введений ідентифікатор не існує, виводиться повідомлення про помилку.

Лістинг фрагменту програми з видаленням даних у таблиці student:

```
elif table == 3:
    id = controller.validate_input_items("student_id")
    cursor.execute(f"""select * from public.student where "student_id" = '{id}'
""")
    records = cursor.fetchone()
    if records is not None:
        cursor.execute(f"""select * from public.phone where "studentphone_fk" =
'{id}' """)
        records = cursor.fetchone()
        if records is not None:
            cursor.execute(f"""DELETE FROM public.phone WHERE "studentphone_fk"
= '{id}'""")
            View.complete_message("studentphone_fk", id, "phone", "deleted")
        cursor.execute(f"""select * from public.schedule where "student_fk" =
'{id}' """)
        records = cursor.fetchone()
        if records is not None:
            cursor.execute(f"""DELETE FROM public.schedule WHERE "student_fk" =
'{id}'""")
```

```
            View.complete_message("student_fk", id, "schedule", "deleted")
        cursor.execute(f"""DELETE FROM public.student WHERE "student_id" =
'{id}'""")
        View.complete_message("student_id", id, "student", "deleted")
    else:
        controller.message("No ID found")
    go_on = False
```

Скріншоти результатів виконання операції вилучення:

student

student_id	firstname	lastname
2	Natalia	Li
3	Andriyan	Anton
4	Baryshev	Oleksiy
5	Antonenko	Oleksandra

Рисунок 22 – Батьківська таблиця «student» до вилучення даних

phone

phone_id	phonenumber	studentphone_fk
3	+380509239464	2
4	+380663436545	4

Рисунок 23 – Дочірня таблиця «phone» до вилучення даних

schedule

schedule_id	day	time	student_fk	teacher_fk	subject_fk
3	Wednesdau	15:00	3	1	3
4	Friday	13:35	4	1	2

Рисунок 24 – Дочірня таблиця «schedule» до вилучення даних

```
Choose table number => 3
Enter student_id: 4
The row with 'studentphone_fk' = '4' in table 'phone' was deleted successfully.
The row with 'student_fk' = '4' in table 'schedule' was deleted successfully.
The row with 'student_id' = '4' in table 'student' was deleted successfully.
```

Рисунок 25 – Повідомлення про успішне видалення даних з трьох таблиць

student

student_id	firstname	lastname
2	Natalia	Li
3	Andriyan	Anton
5	Antonenko	Oleksandra

Рисунок 26 – Батьківська таблиця «student» після виконання операції вилучення даних

phone

phone_id	phonenumber	studentphone_fk
3	+380509239464	2

Рисунок 27 – Дочірня таблиця «phone» після виконання операції вилучення даних

schedule

schedule_id	day	time	student_fk	teacher_fk	subject_fk
3	Wednesdau	15:00	3	1	3

Рисунок 28 – Дочірня таблиця «schedule» після виконання операції вилучення даних

```
Choose table number => 3
Enter student_id: 10
No ID found
```

Рисунок 29 – Повідомлення про помилку (відсутній введений ідентифікатор)

Якщо обрана четверта таблиця «phone», пропонується ввести phone_id – ідентифікатор телефону, рядок з даними якого буде видалено. Про успішне видалення свідчить відповідне повідомлення. Якщо введений ідентифікатор не існує, виводиться повідомлення про помилку.

Лістинг фрагменту програми з видаленням даних у таблиці phone:

```
elif table == 4:
    id = controller.validate_input_items("phone_id")
    cursor.execute(f"""select * from public.phone where "phone_id" = '{id}' """)
    records = cursor.fetchone()
    if records is not None:
        cursor.execute(f"""DELETE FROM public.phone WHERE "phone_id" =
```

```
'{id}'""")
        View.complete_message("phone_id", id, "phone", "deleted")
    else:
        controller.message("No ID found")
    go_on = False
```

Скріншоти результатів виконання операції вилучення:

phone

phone_id	phonenumber	studentphone_fk
3	+380509239464	2
4	+380666710498	3

Рисунок 30 – Таблиця «phone» до вилучення даних

```
Choose table number => 4
Enter phone_id: 4
The row with 'phone_id' = '4' in table 'phone' was deleted successfully.
```

Рисунок 31 – Повідомлення про успішне видалення даних з таблиці

phone

phone_id	phonenumber	studentphone_fk
3	+380509239464	2

Рисунок 32 – Таблиця «phone» після виконання операції вилучення даних

Якщо обрана п’ята таблиця «schedule», пропонується ввести schedule_id – ідентифікатор учня, рядок з даними якого буде видалено. Про успішне видалення свідчить відповідне повідомлення. Якщо введений ідентифікатор не існує, виводиться повідомлення про помилку.

Лістинг фрагменту програми з видаленням даних у таблиці schedule:

```
elif table == 5:
    id = controller.validate_input_items("schedule_id")
    cursor.execute(f"""select * from public.schedule where "schedule_id" =
'{id}' """)
    records = cursor.fetchone()
    if records is not None:
        cursor.execute(f"""DELETE FROM public.schedule WHERE "schedule_id" =
'{id}'""")
        View.complete_message("schedule_id", id, "schedule", "deleted")
    else:
        controller.message("No ID found")
    go_on = False
else:
    print("Please, enter valid number")
```

Скріншоти результатів виконання операції вилучення:

```
 schedule
+-------------+-----------+-------+------------+------------+------------+
| schedule_id |    day    | time  | student_fk | teacher_fk | subject_fk |
+-------------+-----------+-------+------------+------------+------------+
|      3      | Wednesdau | 15:00 |     3      |     1      |     3      |
|      4      |  Monday   | 14:25 |     5      |     1      |     2      |
|      5      | Thursday  | 16:40 |     5      |     1      |     3      |
+-------------+-----------+-------+------------+------------+------------+
```

Рисунок 33 – Таблиця «schedule» до вилучення даних

```
Choose table number => 5
Enter schedule_id: 5
The row with 'schedule_id' = '5' in table 'schedule' was deleted successfully.
```

Рисунок 34 – Повідомлення про успішне вилучення даних

```
 schedule
+-------------+-----------+-------+------------+------------+------------+
| schedule_id |    day    | time  | student_fk | teacher_fk | subject_fk |
+-------------+-----------+-------+------------+------------+------------+
|      3      | Wednesdau | 15:00 |     3      |     1      |     3      |
|      4      |  Monday   | 14:25 |     5      |     1      |     2      |
+-------------+-----------+-------+------------+------------+------------+
```

Рисунок 35 – Таблиця «schedule» після виконання операції вилучення даних

```
Choose table number => 5
Enter schedule_id: 15
No ID found
```

Рисунок 36 – Повідомлення про помилку (відсутній введений ідентифікатор)

Лістинги та скріншоти результатів виконання операції вставки запису в дочірню таблицю

Вставка даних реалізовано у файлі model.py за допомогою функції insert. Користувач обирає таблицю, у яку хоче вставити дані, а також вводить необхідні значення (Рисунок 5). Є валідація введених даних на коректність типу.

Однією з дочірніх таблиць є таблиця phone, яка пов’язує учня («student») і номер його телефону. Якщо під час виконання операції вставки буде введено зовнішній ключ studentphone_fk, що відповідає неіснуючому первинному ключу в батьківській таблиці «student», буде виведено повідомлення про помилку. Якщо такий первинний ключ існує, то буде виведено повідомлення про успішне виконання операції вставки.

Лістинг фрагменту програми із внесенням даних у таблицю phone:

```
elif table == 4:
    id = controller.validate_input_items("phone_id")
    studentphone_fk = controller.validate_input_items("student_id")
    phonenumber = controller.validate_input_items("phonenumber")
    cursor.execute(f"""select * from public.phone where "phone_id" = '{id}' """)
    records = cursor.fetchone()
    if records is not None:
```

```
        controller.message("ID already exists")
    else:
        cursor.execute(f"""select * from public.student where "student_id" =
'{studentphone_fk}' """)
        records = cursor.fetchone()
        if records is None:
            controller.message("The student with this ID doesn't exist")
        else:
            cursor.execute(f"""INSERT INTO "phone" ("phone_id", "phonenumber",
"studentphone_fk")
            VALUES ('{id}', '{phonenumber}', '{studentphone_fk}')""")
            View.complete_message("phone_id", id, "phone", "inserted")
    go_on = False
```

Скріншоти результатів виконання операції вставки в таблицю phone:

```
 phone
+----------+---------------+-----------------+
| phone_id |  phonenumber  | studentphone_fk |
+----------+---------------+-----------------+
|    3     | +380509239464 |        2        |
+----------+---------------+-----------------+
```

Рисунок 37 – Таблиця «phone» до виконання операції вставки даних

```
Choose table number => 4
Enter phone_id: 4
Enter student_id: 5
Enter phone_number: 0932250503
The row with 'phone_id' = '4' in table 'phone' was inserted successfully.
```

Рисунок 38 – Повідомлення про успішне виконання операції вставки

```
Choose table number => 4
Enter phone_id: 5
Enter student_id: 10
Enter phone_number: 0509347565
The student with this ID doesn't exist
```

Рисунок 39 – Повідомлення про помилку (посилання на неіснуючий ідентифікатор учня)

Другою дочірньою таблицею є таблиця «schedule», яка пов’язує предмет (таблиця «subject»), його день та час, учня (таблиця «student») і вчителя (таблиця «teacher»). Якщо під час виконання операції вставки буде введено зовнішні ключі student_fk, teacher_fk, subject_fk що відповідають неіснуючим первинним ключам в батьківських таблицях відповідно «student», «teacher», «subject», буде виведено повідомлення про помилку. Якщо такі первинні ключі існують, то буде виведено повідомлення про успішне виконання операції вставки.

Лістинг фрагменту програми із внесенням даних у таблицю schedule:

```
elif table == 5:
    id = controller.validate_input_items("schedule_id")
```

```
    day = controller.validate_input_items("day")
    time = controller.validate_input_items("time")
    scstudent = controller.validate_input_items("student_id")
    scteacher = controller.validate_input_items("teacher_id")
    scsubject = controller.validate_input_items("subject_id")
    cursor.execute(f"""select * from public.schedule where "schedule_id" =
'{id}' """)
    records = cursor.fetchone()
    if records is not None:
        controller.message("ID already exists")
    else:
        cursor.execute(f"""select * from public.student where "student_id" =
'{scstudent}' """)
        records = cursor.fetchone()
        if records is not None:
            cursor.execute(f"""select * from public.teacher where "teacher_id" =
'{scteacher}' """)
            records = cursor.fetchone()
            if records is not None:
                cursor.execute(f"""select * from public.subject where
"subject_id" = '{scsubject}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""INSERT INTO "schedule" ("schedule_id",
"day", "time", "student_fk",
                    "teacher_fk", "subject_fk") VALUES ('{id}', '{day}',
'{time}', '{scstudent}',
                    '{scteacher}', '{scsubject}')""")
                    View.complete_message("schedule_id", id, "schedule",
"inserted")
                else:
                    controller.message("The subject with this ID doesn't exist")
            else:
                controller.message("The teacher with this ID doesn't exist")
        else:
            controller.message("The student with this ID doesn't exist")
    go_on = False
```

Скріншоти результатів виконання операції вставки в таблицю schedule:

schedule

schedule_id	day	time	student_fk	teacher_fk	subject_fk
3	Wednesdau	15:00	3	1	3
4	Monday	14:25	5	1	2

Рисунок 40 – Таблиця «schedule» до виконання операції вставки

```
Choose table number => 5
Enter schedule_id: 5
Enter day: Wednesday
Enter time: 17:30
Enter student_id: 2
Enter teacher_id: 1
Enter subject_id: 3
The row with 'schedule_id' = '5' in table 'schedule' was inserted successfully.
```

Рисунок 41 – Повідомлення про успішне виконання операції вставки даних

schedule

schedule_id	day	time	student_fk	teacher_fk	subject_fk
3	Wednesdau	15:00	3	1	3
4	Monday	14:25	5	1	2
5	Wednesday	17:30	2	1	3

Рисунок 42 – Таблиця «schedule» після виконання операції вставки даних

```
Choose table number => 5
Enter schedule_id: 6
Enter day: Tuesday
Enter time: 12:50
Enter student_id: 7
Enter teacher_id: 10
Enter subject_id: 6
The student with this ID doesn't exist
```

Рисунок 43 – Повідомлення про помилку (посилання на неіснуючий ідентифікатор учня)

Посилання для навігації по тексту програми

Текст програми

launcher.py

```
import controller
from menu import Menu


connection = controller.connection()
cursor = connection.cursor()
Menu.menu()
cursor.close()
connection.close()
print("PostgreSQL connection is closed")
```

controller.py

```
import psycopg2
from view import View


def connection():
    return psycopg2.connect(
        user="postgres",
        password="1111",
        host="localhost",
        port="5432",
        database="postgres",
    )


def disconnect(connection):
    connection.commit()
    connection.close()


def message(text):
    return print(text)


def validtable():
    incorrect = True
    while incorrect:
        table = input('Choose table number => ')
        if table.isdigit():
            table = int(table)
```

```
            if 1 <= table <= 5:
                incorrect = False
            else:
                print('Incorrect input, try again.')
        else:
            print('Incorrect input, try again.')
    return table


def validate_input_items(name):
    if name == "id":
        value = View.enter_item("id")
        if value.isdecimal():
            return value
    elif name == "subject_id":
        value = View.enter_item("subject_id")
        if value.isdecimal():
            return value
    elif name == "teacher_id":
        value = View.enter_item("teacher_id")
        if value.isdecimal():
            return value
        else:
            message("enter only number")
            validate_input_items("teacher_id")
    elif name == "phone_id":
        value = View.enter_item("phone_id")
        if value.isdecimal():
            return value
    elif name == "schedule_id":
        value = View.enter_item("schedule_id")
        if value.isdecimal():
            return value
    elif name == "student_id":
        value = View.enter_item("student_id")
        if value.isdecimal():
            return value
    elif name == "name":
        value = View.enter_item("name")
        if value.isalpha() is False:
            return validate_input_items(name)
        return value
    elif name == "firstname":
        value = View.enter_item("firstname")
        if value.isalpha() is False:
            return validate_input_items(name)
        return value
    elif name == "lastname":
        value = View.enter_item("lastname")
        if value.isalpha() is False:
            return validate_input_items(name)
        return value
    elif name == "phonenumber":
        value = View.enter_item("phone_number")
        if value.isdigit() is False:
            print("Enter like in example: 0998889999")
            return validate_input_items(name)
        else:
            if len(value) != 10:
                print("Phone should have 10 numbers")
                return validate_input_items(name)
            return str("+38" + value)
    elif name == "day":
```

```
        value = View.enter_item("day")
        if value.isalpha() is False:
            return validate_input_items(name)
        return value
    elif name == "time":
        value = View.enter_item("time")
        li = list(value.split(":"))
        li_hour = li[0]
        li_min = li[1]
        if li_hour.isdecimal() and li_min.isdecimal():
            if 20 > int(li_hour) > 7 and -1 < int(li_min) < 60:
                li = [li_hour, li_min]
                return ':'.join(li)
            else:
                message("Please, enter hours from 7 to 20 and minutes from 0 to
59")
                return validate_input_items(name)
        else:
            message("Please, enter only digits with : between")
```

model.py

```
import controller
import time
from view import View
from pandas import DataFrame
from tabulate import tabulate
from numpy import array


tables = {
    1: 'subject',
    2: 'teacher',
    3: 'student',
    4: 'phone',
    5: 'schedule',
}


class Model:
    @staticmethod
    def display_query(rows, headers):
        df = DataFrame([array(el) for el in rows], columns=array(headers))
        print(tabulate(df, headers="keys", tablefmt="pretty", showindex=False))

    @staticmethod
    def show_table(table):
        connection = controller.connection()
        cursor = connection.cursor()
        print(f"Request: SELECT * FROM {table}")
        print(f"\033[1m {table} \033[0m")
        cursor.execute(f"""SELECT * from public.{table}""")
        records = cursor.fetchall()
        cursor.close()
        return records

    @staticmethod
    def insert():
        connection = controller.connection()
```

```
        cursor = connection.cursor()
        go_on = True
        while go_on:
            View.list()
            table = controller.validtable()
            if table == 1:
                id = controller.validate_input_items("subject_id")
                name = controller.validate_input_items("name")
                cursor.execute(f"""select * from public.subject where
"subject_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""INSERT INTO "subject" ("subject_id",
"name")
                                       VALUES ('{id}', '{name}')""")
                    View.complete_message("subject_id", id, "subject",
"inserted")
                go_on = False
            elif table == 2:
                id = controller.validate_input_items("teacher_id")
                firstname = controller.validate_input_items("firstname")
                lastname = controller.validate_input_items("lastname")
                cursor.execute(f"""select * from public.teacher where
"teacher_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""INSERT INTO "teacher" ("teacher_id",
"firstname", "lastname")
                    VALUES ('{id}', '{firstname}', '{lastname}')""")
                    View.complete_message("teacher_id", id, "teacher",
"inserted")
                go_on = False
            elif table == 3:
                id = controller.validate_input_items("student_id")
                firstname = controller.validate_input_items("firstname")
                lastname = controller.validate_input_items("lastname")
                cursor.execute(f"""select * from public.student where
"student_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""INSERT INTO "student" ("student_id",
"firstname", "lastname")
                    VALUES ('{id}', '{firstname}', '{lastname}')""")
                    View.complete_message("student_id", id, "student",
"inserted")
                go_on = False
            elif table == 4:
                id = controller.validate_input_items("phone_id")
                studentphone_fk = controller.validate_input_items("student_id")
                phonenumber = controller.validate_input_items("phonenumber")
                cursor.execute(f"""select * from public.phone where "phone_id" =
'{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""select * from public.student where
```

```
"student_id" = '{studentphone_fk}' """)
                    records = cursor.fetchone()
                    if records is None:
                        controller.message("The student with this ID doesn't
exist")
                    else:
                        cursor.execute(f"""INSERT INTO "phone" ("phone_id",
"phonenumber", "studentphone_fk")
                        VALUES ('{id}', '{phonenumber}',
'{studentphone_fk}')""")
                        View.complete_message("phone_id", id, "phone",
"inserted")
                go_on = False
            elif table == 5:
                id = controller.validate_input_items("schedule_id")
                day = controller.validate_input_items("day")
                time = controller.validate_input_items("time")
                scstudent = controller.validate_input_items("student_id")
                scteacher = controller.validate_input_items("teacher_id")
                scsubject = controller.validate_input_items("subject_id")
                cursor.execute(f"""select * from public.schedule where
"schedule_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    controller.message("ID already exists")
                else:
                    cursor.execute(f"""select * from public.student where
"student_id" = '{scstudent}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""select * from public.teacher where
"teacher_id" = '{scteacher}' """)
                        records = cursor.fetchone()
                        if records is not None:
                            cursor.execute(f"""select * from public.subject
where "subject_id" = '{scsubject}' """)
                            records = cursor.fetchone()
                            if records is not None:
                                cursor.execute(f"""INSERT INTO "schedule"
("schedule_id", "day", "time", "student_fk",
                                "teacher_fk", "subject_fk") VALUES ('{id}',
'{day}', '{time}', '{scstudent}',
                                '{scteacher}', '{scsubject}')""")
                                View.complete_message("schedule_id", id,
"schedule", "inserted")
                            else:
                                controller.message("The subject with this ID
doesn't exist")
                        else:
                            controller.message("The teacher with this ID doesn't
exist")
                    else:
                        controller.message("The student with this ID doesn't
exist")
                go_on = False
            else:
                print('Please, enter valid value')
        connection.commit()
        cursor.close()
        controller.disconnect(connection)

    @staticmethod
    def delete():
```

```
        connection = controller.connection()
        cursor = connection.cursor()
        go_on = True
        while go_on:
            View.list()
            table = controller.validtable()
            if table == 1:
                id = controller.validate_input_items("subject_id")
                cursor.execute(f"""select * from public.subject where
"subject_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""select * from public.schedule where
"subject_fk" = '{id}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""DELETE FROM public.schedule WHERE
"subject_fk" = '{id}' """)
                        View.complete_message("subject_fk", id, "schedule",
"deleted")
                    cursor.execute(f"""DELETE FROM public.subject WHERE
"subject_id" = '{id}'""")
                    View.complete_message("subject_id", id, "subject",
"deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            elif table == 2:
                id = controller.validate_input_items("teacher_id")
                cursor.execute(f"""select * from public.teacher where
"teacher_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""select * from public.schedule where
"teacher_fk" = '{id}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""DELETE FROM public.schedule WHERE
"teacher_fk" = '{id}' """)
                        View.complete_message("teacher_fk", id, "schedule",
"deleted")
                    cursor.execute(f"""DELETE FROM public.teacher WHERE
"teacher_id" = '{id}'""")
                    View.complete_message("teacher_id", id, "teacher",
"deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            elif table == 3:
                id = controller.validate_input_items("student_id")
                cursor.execute(f"""select * from public.student where
"student_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""select * from public.phone where
"studentphone_fk" = '{id}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""DELETE FROM public.phone WHERE
"studentphone_fk" = '{id}'""")
                        View.complete_message("studentphone_fk", id, "phone",
"deleted")
                    cursor.execute(f"""select * from public.schedule where
```

```
"student_fk" = '{id}' """)
                    records = cursor.fetchone()
                    if records is not None:
                        cursor.execute(f"""DELETE FROM public.schedule WHERE
"student_fk" = '{id}'""")
                        View.complete_message("student_fk", id, "schedule",
"deleted")
                    cursor.execute(f"""DELETE FROM public.student WHERE
"student_id" = '{id}'""")
                    View.complete_message("student_id", id, "student",
"deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            elif table == 4:
                id = controller.validate_input_items("phone_id")
                cursor.execute(f"""select * from public.phone where "phone_id" =
'{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.phone WHERE "phone_id"
= '{id}'""")
                    View.complete_message("phone_id", id, "phone", "deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            elif table == 5:
                id = controller.validate_input_items("schedule_id")
                cursor.execute(f"""select * from public.schedule where
"schedule_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    cursor.execute(f"""DELETE FROM public.schedule WHERE
"schedule_id" = '{id}'""")
                    View.complete_message("schedule_id", id, "schedule",
"deleted")
                else:
                    controller.message("No ID found")
                go_on = False
            else:
                print("Please, enter valid number")
            connection.commit()
            cursor.close()
            controller.disconnect(connection)
            pass

    @staticmethod
    def update():
        connection = controller.connection()
        cursor = connection.cursor()
        go_on = True
        while go_on:
            View.list()
            table = controller.validtable()
            if table == 1:
                id = controller.validate_input_items("subject_id")
                cursor.execute(f"""select * from public.subject where
"subject_id" = '{id}' """)
                records = cursor.fetchone()
                if records is not None:
                    value = controller.validate_input_items("name")
                    cursor.execute(f"""UPDATE public.subject set 'name' =
'{value}' where "subject_id" = '{id}' """)
```

```
                    View.complete_message("subject_id", id, "subject",
"updated")
                    go_on = False
                else:
                    controller.message("No subject with this ID found")

            elif table == 2:
                id = controller.validate_input_items("teacher_id")
                cursor.execute(f"""select * from public.teacher where
"teacher_id" = {id} """)
                records = cursor.fetchone()
                if records is not None:
                    View.columns(2)
                    continue_update = True
                    while continue_update:
                        attr = input("Choose a number of column to update")
                        if attr == '1':
                            value = controller.validate_input_items("firstname")
                            attribute = "firstname"
                            continue_update = False
                        elif attr == '2':
                            value = controller.validate_input_items("lastname")
                            attribute = "lastname"
                            continue_update = False
                        else:
                            controller.message("Please, enter valid number")
                    cursor.execute(f"""UPDATE "teacher" set {attribute} =
'{value}' where "teacher_id" = '{id}' """)
                    View.complete_message("teacher_id", id, "teacher",
"updated")
                    go_on = False
                    pass
                else:
                    controller.message("No teacher with this ID found")
            elif table == 3:
                id = controller.validate_input_items("student_id")
                cursor.execute(f"""select * from public.student where
"student_id" = {id} """)
                records = cursor.fetchone()
                if records is not None:
                    View.columns(3)
                    continue_update = True
                    while continue_update:
                        attr = input("Choose a number of column to update: ")
                        if attr == '1':
                            value = controller.validate_input_items("firstname")
                            attribute = "firstname"
                            continue_update = False
                        elif attr == '2':
                            value = controller.validate_input_items("lastname")
                            attribute = "lastname"
                            continue_update = False
                        else:
                            controller.message("Please, enter valid number")
                    cursor.execute(f"""UPDATE "student" set {attribute} =
'{value}' where "student_id" = '{id}' """)
                    View.complete_message("student_id", id, "student",
"updated")
                    go_on = False
                    pass
                else:
                    controller.message("No student with this ID found")
            elif table == 4:
```

```
                id = controller.validate_input_items("phone_id")
                cursor.execute(f"""select * from public.phone where "phone_id" =
{id} """)
                records = cursor.fetchone()
                if records is not None:
                    View.columns(4)
                    continue_update = True
                    while continue_update:
                        attr = input("Choose a number of column to update: ")
                        if attr == '1':
                            value =
controller.validate_input_items("phonenumber")
                            attribute = "phonenumber"
                            continue_update = False
                        elif attr == '2':
                            value =
controller.validate_input_items("student_id")
                            attribute = "studentphone_fk"
                            continue_update = False
                        else:
                            controller.message("Please, enter valid number: ")
                    cursor.execute(f"""UPDATE "phone" set {attribute} =
'{value}' where "phone_id" = '{id}' """)
                    View.complete_message("phone_id", id, "phone", "updated")
                    go_on = False
                    pass
                else:
                    controller.message("No phone with this ID found")
            elif table == 5:
                id = controller.validate_input_items("schedule_id")
                cursor.execute(f"""select * from public.schedule where
"schedule_id" = {id} """)
                records = cursor.fetchone()
                if records is not None:
                    View.columns(5)
                    continue_update = True
                    while continue_update:
                        attr = input("Choose a number of column to update ")
                        if attr == '1':
                            value = controller.validate_input_items("day")
                            attribute = "day"
                            continue_update = False
                        elif attr == '2':
                            value = controller.validate_input_items("time")
                            attribute = "time"
                            continue_update = False
                        elif attr == '3':
                            value =
controller.validate_input_items("subject_id")
                            attribute = "subject_fk"
                            continue_update = False
                        elif attr == '4':
                            value =
controller.validate_input_items("student_id")
                            attribute = "student_fk"
                            continue_update = False
                        elif attr == '5':
                            value =
controller.validate_input_items("teacher_id")
                            attribute = "teacher_fk"
                            continue_update = False
                        else:
                            controller.message("Please, enter valid number")
```

```
                    cursor.execute(f"""UPDATE "schedule" set {attribute} =
'{value}' where "schedule_id" = '{id}' """)
                    View.complete_message("schedule_id", id, "schedule",
"updated")
                    go_on = False
                    pass
                else:
                    controller.message("No schedule with this ID found")
            else:
                controller.message("Please, enter valid number")
        connection.commit()
        cursor.close()
        controller.disconnect(connection)
        pass

    @staticmethod
    def search1():
        connection = controller.connection()
        cursor = connection.cursor()
        lastname = controller.validate_input_items("lastname")
        cursor.execute(f"""select * from public.student where "lastname" =
'{lastname}' """)
        records = cursor.fetchone()
        if records is not None:
            search = f"""select "lastname", "firstname", "phonenumber" from
                    (select c."lastname", c."firstname", p."phonenumber" from
public.phone
                    p left join public.student c on c."student_id" =
p."studentphone_fk"
                    where c."lastname" LIKE '{lastname}' group by c."lastname",
c."firstname", p."phonenumber")
                    as foo"""
        else:
            controller.message("No student with this last name found")
        start = int(time.time() * 1000)
        cursor.execute(search)
        print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
        records = cursor.fetchall()
        cursor.close()
        return records

    @staticmethod
    def search2():
        connection = controller.connection()
        cursor = connection.cursor()
        lastname = controller.validate_input_items("lastname")
        cursor.execute(f"""select * from public.student where "lastname" =
'{lastname}' """)
        records = cursor.fetchone()
        if records is not None:
            search = f"""select c."lastname", s."name", t."firstname",
t."lastname" from public.schedule p inner
            join public.subject s on s."subject_id" = p."subject_fk" inner join
public.teacher t on
            t."teacher_id" = p."teacher_fk" inner join public.student c on
c."student_id" = p."student_fk"
            where c."lastname" LIKE '{lastname}' group by c."lastname",
            s."name", t."firstname", t."lastname" """
        else:
            controller.message("No student with this last name found")
        start = int(time.time() * 1000)
        cursor.execute(search)
```

```
        print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
        records = cursor.fetchall()
        cursor.close()
        return records

    @staticmethod
    def search3():
        connection = controller.connection()
        cursor = connection.cursor()
        name = controller.validate_input_items("name")
        cursor.execute(f"""select * from public.subject where "name" = '{name}'
""")
        records = cursor.fetchone()
        if records is not None:
            search = f"""select s."name", c."lastname", p."day", p."time" from
public.schedule p inner join
            public.subject s on s."subject_id" = p."subject_fk" inner join
public.student c on
            c."student_id" = p."student_fk" where s."name" LIKE '{name}' group
by s."name", c."lastname", p."day",
            p."time" """
        else:
            controller.message("No subject with this name found")
        start = int(time.time() * 1000)
        cursor.execute(search)
        print("--- Time of search = {} ms ---".format(int((time.time() * 1000) -
start)))
        records = cursor.fetchall()
        cursor.close()
        return records
```

view.py

```
class View:
    def __init__(self, table, records):
        self.table = table
        self.records = records

    @staticmethod
    def complete_message(attribute, value, table, action):
        print(f"The row with '{attribute}' = '{value}' in table '{table}' was
{action} successfully.")

    @staticmethod
    def enter_item(item):
        data = input("Enter {}: ".format(item))
        return data

    @staticmethod
    def list():
        print('''
        1 -> subject
        2 -> teacher
        3 -> student
        4 -> phone
        5 -> schedule
```

```
        ''')

    @staticmethod
    def columns(table):
        if table == 1:
            print('''
                1 -> name
            ''')
        elif table == 2:
            print('''
                1 -> firstname
                2 -> lastname
            ''')
        elif table == 3:
            print('''
                1 -> firstname
                2 -> lastname
            ''')
        elif table == 4:
            print('''
                1 -> phone number
                2 -> student ID
            ''')
        elif table == 5:
            print('''
                1 -> day
                2 -> time
                3 -> subject ID
                4 -> student ID
                5 -> teacher Id
            ''')

    @staticmethod
    def search():
        print('''
                1 -> Search first name and phone number of the student with last
name
                2 -> Search subject and teacher first and last name of the
student with last name
                3 -> Search student last name, day and time of the subject with
subject name
                ''')
```

menu.py

```
from model import Model
from view import View
import controller
from randomizer import Randomizer


class Menu:
    @staticmethod
    def menu():
        while True:
            print('''
                Main menu
                0 - Show one table
                1 - Show all tables
```

```
                2 - Insert data
                3 - Delete data
                4 - Update data
                5 - Search data
                6 - Insert random data
                7 - Exit
                ''')
            choice = input('Choose an option: ')
            if choice == '0':
                View.list()
                table = controller.validtable()
                if table == 1:
                    Model.display_query(Model.show_table("subject"),
["subject_id", "name"])
                elif table == 2:
                    Model.display_query(Model.show_table("teacher"),
["teacher_id", "firstname", "lastname"])
                elif table == 3:
                    Model.display_query(Model.show_table("student"),
["student_id", "firstname", "lastname"])
                elif table == 4:
                    Model.display_query(Model.show_table("phone"), ["phone_id",
"phonenumber", "studentphone_fk"])
                elif table == 5:
                    Model.display_query(Model.show_table("schedule"),
                                        ["schedule_id", "day", "time",
"student_fk", "teacher_fk", "subject_fk"])
            elif choice == '1':
                Model.display_query(Model.show_table("subject"), ["subject_id",
"name"])
                Model.display_query(Model.show_table("teacher"), ["teacher_id",
"firstname", "lastname"])
                Model.display_query(Model.show_table("student"), ["student_id",
"firstname", "lastname"])
                Model.display_query(Model.show_table("phone"), ["phone_id",
"phonenumber", "studentphone_fk"])
                Model.display_query(Model.show_table("schedule"),
                                    ["schedule_id", "day", "time", "student_fk",
"teacher_fk", "subject_fk"])
            elif choice == '2':
                end_insert = False
                while not end_insert:
                    Model.insert()
                    incorrect = True
                    while incorrect:
                        answer = input('Continue working with insert? Enter Yes
or No ')
                        if answer == 'No':
                            end_insert = True
                            incorrect = False
                        elif answer == 'Yes':
                            incorrect = False
                            pass
                        else:
                            print('Please, enter Yes or No')
            elif choice == '3':
                end_delete = False
                while not end_delete:
                    Model.delete()
                    incorrect = True
                    while incorrect:
                        answer = input('Continue working with delete? Enter Yes
or No ')
```

```
                    if answer == 'No':
                        end_delete = True
                        incorrect = False
                    elif answer == 'Yes':
                        incorrect = False
                        pass
                    else:
                        print('Please, enter Yes or No ')
            elif choice == '4':
                end_update = False
                while not end_update:
                    Model.update()
                    incorrect = True
                    while incorrect:
                        answer = input('Continue working with update? Enter Yes
or No ')
                        if answer == 'No':
                            end_update = True
                            incorrect = False
                        elif answer == 'Yes':
                            incorrect = False
                            pass
                        else:
                            print('Please, enter Yes or No ')
            elif choice == '5':
                end_search = False
                View.search()
                choice = int(input("Select what you want to search: "))
                while not end_search:
                    if choice == 1:
                        Model.display_query(Model.search1(), ["Last name",
"First name", "Phone number"])
                    elif choice == 2:
                        Model.display_query(Model.search2(),
                                            ["Last name", "Subject", "First name
of teacher", "Last name of teacher"])
                    elif choice == 3:
                        Model.display_query(Model.search3(), ["Subject", "Last
name of the student",
                                                              "Day of subject",
"Time of subject"])
                    incorrect = True
                    while incorrect:
                        answer = input('Continue working with search? Enter Yes
or No ')
                        if answer == 'No':
                            end_search = True
                            incorrect = False
                        elif answer == 'Yes':
                            incorrect = False
                            pass
                        else:
                            print('Please, enter Yes or No ')
            elif choice == '6':
                end_insert = False
                count = int(input("How many rows do you want to insert: "))
                View.list()
                table = controller.validtable()
                while not end_insert:
                    Randomizer.random(table, count)
                    go_on = True
                    while go_on:
                        answer = input('Continue working with random data? Enter
```

```
Yes or No: ')
                        if answer == 'No':
                            end_insert = True
                            go_on = False
                        elif answer == 'Yes':
                            go_on = False
                            pass
                        else:
                            print('Please, enter Yes or No ')
            elif choice == '7':
                break
            else:
                print('Please, enter valid number')
```

randomizer.py

```
import controller


class Randomizer:
    @staticmethod
    def random(table, count):
        connection = controller.connection()
        cursor = connection.cursor()
        go_on = True
        while go_on:
            if table == 1:
                for i in range(count):
                    cursor.execute("""insert into public.subject
                                      select (SELECT MAX("subject_id")+1 FROM
public.subject),
                                      array_to_string(ARRAY(SELECT chr((97 +
round(random()*25)) :: integer)
                                      FROM
                                      generate_series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), '')""")
                go_on = False
            elif table == 2:
                for i in range(count):
                    cursor.execute("""insert into public.teacher
                    select (SELECT MAX("teacher_id")+1 FROM public.teacher),
                    array_to_string(ARRAY(SELECT chr((97 + round(random()*25))
:: integer)
                    FROM
                    generate_series(1, FLOOR(RANDOM()*(15-5)+5):: integer)),
''),
                    array_to_string(ARRAY(SELECT chr((97 + round(random()*25))
:: integer)
                    FROM
                    generate_series(1, FLOOR(RANDOM()*(15-5)+5):: integer)),
'')""")
                go_on = False
            elif table == 3:
                for i in range(count):
                    cursor.execute("""insert into public.student
                                      select (SELECT MAX("student_id")+1 FROM
public.student),
                                      array to string(ARRAY(SELECT chr((97 +
round(random()*25)) :: integer)
```

```
                                        FROM
                                        generate_series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), ''),
                                        array_to_string(ARRAY(SELECT chr((97 +
round(random()*25)) :: integer)
                                        FROM
                                        generate_series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), '')""")
                go_on = False
            elif table == 4:
                for i in range(count):
                    cursor.execute("""insert into public.phone
                                        select (SELECT MAX("phone_id")+1 FROM
public.phone),
                                        array_to_string(ARRAY(SELECT chr((48 +
round(random()*9)) :: integer)
                                        FROM
                                        generate_series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), ''),
                                        (SELECT "student_id" FROM public.student
LIMIT 1 OFFSET (round(random() *
                                        ((SELECT COUNT("student_id") FROM
public.student)-1))))""")
                go_on = False
            elif table == 5:
                for i in range(count):
                    cursor.execute("""insert into public.schedule
                                        select (SELECT MAX("schedule_id")+1 FROM
public.schedule),
                                        array_to_string(ARRAY(SELECT chr((97 +
round(random()*25)) :: integer)
                                        FROM
                                        generate series(1, FLOOR(RANDOM()*(15-
5)+5):: integer)), ''),
                                        array_to_string(ARRAY(select
chr(trunc(48+random()*6)::int) ||
                                        chr(trunc(48+random()*10)::int) || ':'
|| chr(trunc(48+random()*6)::int) ||
                                        chr(trunc(48+random()*10)::int)), ''),
                                        (SELECT "subject_id" FROM public.subject
LIMIT 1 OFFSET (round(random() *
                                        ((SELECT COUNT("subject_id") FROM
public.subject)-1)))),
                                        (SELECT "student_id" FROM public.student
LIMIT 1 OFFSET (round(random() *
                                        ((SELECT COUNT("student_id") FROM
public.student)-1)))),
                                        (SELECT "teacher_id" FROM public.teacher
LIMIT 1 OFFSET (round(random() *
                                        ((SELECT COUNT("teacher_id") FROM
public.teacher)-1))))
                                        """)
                go_on = False
            else:
                controller.message("Please, enter valid number: ")
        print("inserted random data successfully")
        connection.commit()
        cursor.close()
        controller.disconnect(connection)
```