A black and white drawing of a building

Description automatically generated

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського» ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Розрахунково-графічна робота**

з дисципліни **Бази даних і засоби управління**

*на тему: “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”*

Виконала:

студентка ІІI курсу

групи КВ-12

Прокопенко С.П.

Перевірив:

Павловский В. І.

Київ – 2024

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

**Виконання роботи**

Нижче наведені сутності предметної області онлайн-платформи для оренди спорядження та зв’язки між сутностями обраної предметної галузі.

**Опис сутностей**

1. Користувач (User)

Атрибути: UserID, Name, Email, Number

Призначення: зберігає інформацію про користувача

1. Оренда (Rental)

Атрибути: RentalID, Duration, RentalDate, ReturnDate, UserID, Equipmen

Призначення: зберігає інформацію про оренду спорядження

1. Спорядження (Equipment)

Атрибути: EquipmentID, Name, RentalCost

Призначення: зберігає інформацію про спорядження.

**Опис зв’язків між сутностями**

Зв’язок “User” – “Rental” 1:N, де “Rental” — сутність, оскільки кожен користувач може укладати декілька оренд спорядження.

Зв’язок “Rental” — “Equipment” 1:1, так як кожна оренда має відношення лише до однієї одиниці спорядження.

Графічне подання концептуальної моделі «Сутність-зв’язок» зображено на рисунку 1.

Изображение выглядит как диаграмма, линия

Автоматически созданное описание

Рисунок 1 — ER-діаграма, побудована за нотацією Чена

За розробленої ER-діаграмою було створено схему бази даних. Її зображено на рисунку 2.

Изображение выглядит как текст, линия, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2 — Схема бази даних

**Середовище та компоненти розробки**

Для виконання завдань розрахунково-графічної роботи було використано мову програмування Python 3.11.7 і бібліотека psycopg2, інтегроване середовище розробки Visual Studio Code.

**Шаблон проектування**

В ході виконання лабораторної роботи було використано шаблон проектування MVC (Model-View-Controller), отже було написано 3 класи, а програму було розбито на 3 окремі файли (не враховуючи основний файл main.py):

1. Модель (Model) — клас, що надає дані та реагує на команди контролера. Модель не залежить від представлення і контролера, надаючи їм доступ до даних та доступ на управління ними.
2. Представлення (View) — відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі. Отримує дані від моделі і відображає їх користувачу.
3. Контролер (Controller) — клас, що інтерпретує дії користувача, сповіщаючи модель про необхідність змін. Забезпечує зв’язок між користувачем та системою.

**Структура програми**

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

**Структура меню програми**

**Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание**

**Фрагмент коду з файлу Controller.py, в якому наведено головний цикл роботи програми**

**def** run**(**self**):**

**while** **True:**

choice **=** self**.**view**.**menuShow**()**

**if** **(**choice **==** "1"**):**

self**.**outputTables**()**

**elif** **(**choice **==** "2"**):**

self**.**outputColumns**()**

**elif** **(**choice **==** "3"**):**

self**.**taskAdd**()**

**elif** **(**choice **==** "4"**):**

self**.**taskUpdate**()**

**elif** **(**choice **==** "5"**):**

self**.**taskDelete**()**

**elif** **(**choice **==** "6"**):**

self**.**taskGenerate**()**

**elif** **(**choice **==** "7"**):**

**return**

**else:**

self**.**view**.**printMessage**(**"\n" **+** "Wrong number. Please, try again."**)**

**Фрагменти кодів з файлу model.py, де наведені функції внесення, редагування, вилучення та генерації даних у базі даних**

**Функція внесення даних:**

**def** taskAdd**(**self**,** table**,** columns**,** values**):**

cursor **=** self**.**connection**.**cursor**()**

columnsString **=** ", "**.**join**(**columns**)**

placeHolders **=** ", "**.**join**(**values**)**

cursor**.**execute**(**f'INSERT INTO "{table}" ({columnsString}) VALUES ' **+** f"({placeHolders})"**)**

self**.**connection**.**commit**()**

**Функція редагування даних:**

**def** taskUpdate**(**self**,** table**,** column**,** **id,** value**):**

cursor **=** self**.**connection**.**cursor**()**

columnID **=** f'{table}ID'

cursor**.**execute**(**f'UPDATE "{table}" SET "{column}" = \'{value}\' WHERE "{columnID}" = {**id**}'**)**

self**.**connection**.**commit**()**

**Функція вилучення даних:**

**def** taskDelete**(**self**,** table**,** **id):**

cursor **=** self**.**connection**.**cursor**()**

cursor**.**execute**(**f'DELETE FROM "{table}" WHERE "{table}ID" = {**id**}'**)**

self**.**connection**.**commit**()**

**Функція генерування даних:**

**def** generateRows**(**self**,** table**,** rowNumber**):**

cursor **=** self**.**connection**.**cursor**()**

cursor**.**execute**(**f"SELECT column\_name FROM information\_schema.columns WHERE table\_name = '{table}'"**)**

columnNames**=** cursor**.**fetchall**()**

cursor**.**execute**(**f"SELECT data\_type FROM information\_schema.columns WHERE table\_name = '{table}'"**)**

columnTypes **=** cursor**.**fetchall**()**

columnID **=** f'{table}ID'

**for** i **in** **range(**rowNumber**):**

insert **=** f'INSERT INTO "{table}" ('

selectSubquery **=** ""

**for** j **in** **range(**0**,** **len(**columnNames**)):**

**if** columnNames**[**j**][**0**]** **==** columnID**:**

cursor**.**execute**(**f'SELECT max("{columnID}") FROM "{table}"'**)**

maxID **=** cursor**.**fetchone**()[**0**]** **or** 0

selectSubquery **+=** f'{maxID **+** 1},'

**elif** columnNames**[**j**][**0**].**endswith**(**'ID'**):**

relatedTable **=** columnNames**[**j**][**0**][:-**2**]**

cursor**.**execute**(**f'SELECT "{relatedTable}ID" FROM "{relatedTable}" ORDER BY RANDOM() LIMIT 1'**)**

relatedID **=** cursor**.**fetchone**()[**0**]**

selectSubquery **+=** f'{relatedID},'

**elif** columnNames**[**j**][**0**]** **==** "Duration"**:**

selectSubquery **+=** "1,"

**elif** columnTypes**[**j**][**0**]** **==** 'integer'**:**

selectSubquery **+=** f'trunc(random()\*1000)::INT,'

**elif** columnTypes**[**j**][**0**]** **==** 'character varying'**:**

selectSubquery **+=** f'chr(trunc(65 + random()\*25)::INT) || chr(trunc(65 + random()\*25)::INT),'

**elif** columnTypes**[**j**][**0**]** **==** 'date'**:**

selectSubquery **+=** "'2020-02-15',"

**else:**

**continue**

insert **+=** f'"{columnNames**[**j**][**0**]**}",'

insert **=** insert**.**rstrip**(**','**)** **+** f') VALUES ({selectSubquery**[:-**1**]**})'

cursor**.**execute**(**insert**)**

self**.**connection**.**commit**()**

**Повний код програми**

**main.py:**

**from** controller **import** Controller

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

Controller**().**run**()**

**Model.py:**

**import** psycopg2

**import** random

**class** **Model:**

**def** \_\_init\_\_**(**self**):**

self**.**connection **=** psycopg2**.**connect**(**

dbname**=**"postgres"**,**

user**=**"postgres"**,**

password**=**"Qwerty12345"**,**

host**=**"localhost"**,**

port**=**5432

**)**

**def** getTables**(**self**):**

cursor **=** self**.**connection**.**cursor**()**

cursor**.**execute**(**"SELECT table\_name FROM information\_schema.tables WHERE table\_schema = 'public'"**)**

**return** cursor**.**fetchall**()**

**def** getColumns**(**self**,** table**):**

cursor **=** self**.**connection**.**cursor**()**

cursor**.**execute**(**f"SELECT column\_name FROM information\_schema.columns WHERE table\_name = '{table}'"**)**

**return** cursor**.**fetchall**()**

**def** taskAdd**(**self**,** table**,** columns**,** values**):**

cursor **=** self**.**connection**.**cursor**()**

columnsString **=** ", "**.**join**(**columns**)**

placeHolders **=** ", "**.**join**(**values**)**

cursor**.**execute**(**f'INSERT INTO "{table}" ({columnsString}) VALUES ' **+** f"({placeHolders})"**)**

self**.**connection**.**commit**()**

**def** taskUpdate**(**self**,** table**,** column**,** **id,** value**):**

cursor **=** self**.**connection**.**cursor**()**

columnID **=** f'{table}ID'

cursor**.**execute**(**f'UPDATE "{table}" SET "{column}" = \'{value}\' WHERE "{columnID}" = {**id**}'**)**

self**.**connection**.**commit**()**

**def** taskDelete**(**self**,** table**,** **id):**

cursor **=** self**.**connection**.**cursor**()**

cursor**.**execute**(**f'DELETE FROM "{table}" WHERE "{table}ID" = {**id**}'**)**

self**.**connection**.**commit**()**

**def** generateRows**(**self**,** table**,** rowNumber**):**

cursor **=** self**.**connection**.**cursor**()**

cursor**.**execute**(**f"SELECT column\_name FROM information\_schema.columns WHERE table\_name = '{table}'"**)**

columnNames**=** cursor**.**fetchall**()**

cursor**.**execute**(**f"SELECT data\_type FROM information\_schema.columns WHERE table\_name = '{table}'"**)**

columnTypes **=** cursor**.**fetchall**()**

columnID **=** f'{table}ID'

**for** i **in** **range(**rowNumber**):**

insert **=** f'INSERT INTO "{table}" ('

selectSubquery **=** ""

**for** j **in** **range(**0**,** **len(**columnNames**)):**

**if** columnNames**[**j**][**0**]** **==** columnID**:**

cursor**.**execute**(**f'SELECT max("{columnID}") FROM "{table}"'**)**

maxID **=** cursor**.**fetchone**()[**0**]** **or** 0

selectSubquery **+=** f'{maxID **+** 1},'

**elif** columnNames**[**j**][**0**].**endswith**(**'ID'**):**

relatedTable **=** columnNames**[**j**][**0**][:-**2**]**

cursor**.**execute**(**f'SELECT "{relatedTable}ID" FROM "{relatedTable}" ORDER BY RANDOM() LIMIT 1'**)**

relatedID **=** cursor**.**fetchone**()[**0**]**

selectSubquery **+=** f'{relatedID},'

**elif** columnNames**[**j**][**0**]** **==** "Duration"**:**

selectSubquery **+=** "1,"

**elif** columnTypes**[**j**][**0**]** **==** 'integer'**:**

selectSubquery **+=** f'trunc(random()\*1000)::INT,'

**elif** columnTypes**[**j**][**0**]** **==** 'character varying'**:**

selectSubquery **+=** f'chr(trunc(65 + random()\*25)::INT) || chr(trunc(65 + random()\*25)::INT),'

**elif** columnTypes**[**j**][**0**]** **==** 'date'**:**

selectSubquery **+=** "'2020-02-15',"

**else:**

**continue**

insert **+=** f'"{columnNames**[**j**][**0**]**}",'

insert **=** insert**.**rstrip**(**','**)** **+** f') VALUES ({selectSubquery**[:-**1**]**})'

cursor**.**execute**(**insert**)**

self**.**connection**.**commit**()**

**View.py:**

**class** **View:**

**def** \_\_init\_\_**(**self**):**

self**.**shown **=** **False**

**def** menuShow**(**self**):**

**if** **(not** self**.**shown**):**

**print(**"Enter a number to choose an option: \n \

1. Output tables' names \n \

2. Output columns' names \n \

3. Add data \n \

4. Update data \n \

5. Delete data \n \

6. Generate data \n \

7. Exit \n"**)**

self**.**shown **=** **True**

userInput **=** **input(**"Print a number: "**)**

**return** userInput

**def** printMessage**(**self**,** string**):**

**print(**string**)**

**def** askTableName**(**self**):**

table **=** **input(**"Enter table's name: "**)**

**return** table

**def** askColumnName**(**self**,** table**):**

column **=** **input(**"Enter columns in " **+** table **+** " table: "**).**split**()**

**return** column

**def** askValue**(**self**,** string**):**

value **=** **input(**"Enter data to be " **+** string**).**split**()**

**return** value

**def** askID**(**self**,** tableName**):**

**id** **=** **input(**"Enter " **+** tableName **+** "ID: "**)**

**return** **id**

**def** askRowNumber**(**self**):**

number **=** **input(**"How many rows to generate: "**)**

**return** number

**def** taksUpdate**(**self**):**

table **=** **input(**"Enter table's name: "**)**

column **=** **input(**"Enter column's name in " **+** table **+** " table: "**)**

**id** **=** **input(**"Enter " **+** table **+** "ID of the row: "**)**

value **=** **input(**"Enter new data: "**)**

**return** table**,** column**,** **id,** value

**def** printTables**(**self**,** tables**):**

**print(**"Tables' names:"**)**

**for** table **in** tables**:**

**print(**table**[**0**])**

**print(**"\n"**)**

**def** printColumns**(**self**,** columns**,** table**):**

**print(**"Columns in " **+** table **+** " table: "**)**

**for** column **in** columns**:**

**print(**column**[**0**])**

**print(**"\n"**)**

**Controller.py:**

**from** model **import** Model

**from** view **import** View

**class** **Controller:**

**def** \_\_init\_\_**(**self**):**

self**.**model **=** Model**()**

self**.**view **=** View**()**

**def** run**(**self**):**

**while** **True:**

choice **=** self**.**view**.**menuShow**()**

**if** **(**choice **==** "1"**):**

self**.**outputTables**()**

**elif** **(**choice **==** "2"**):**

self**.**outputColumns**()**

**elif** **(**choice **==** "3"**):**

self**.**taskAdd**()**

**elif** **(**choice **==** "4"**):**

self**.**taskUpdate**()**

**elif** **(**choice **==** "5"**):**

self**.**taskDelete**()**

**elif** **(**choice **==** "6"**):**

self**.**taskGenerate**()**

**elif** **(**choice **==** "7"**):**

**return**

**else:**

self**.**view**.**printMessage**(**"\n" **+** "Wrong number. Please, try again."**)**

**def** outputTables**(**self**):**

tables **=** self**.**model**.**getTables**()**

self**.**view**.**printTables**(**tables**)**

**def** outputColumns**(**self**):**

tableName **=** self**.**view**.**askTableName**()**

columns **=** self**.**model**.**getColumns**(**tableName**)**

self**.**view**.**printColumns**(**columns**,** tableName**)**

**def** checkForQuotes**(**self**,** vector**):**

index **=** 0

quoteBegin **=** 0

quoteEnd **=** 0

**for** i **in** vector**:**

**if** **(**i**[**0**]** **==** '"'**):**

quoteBegin **=** index

**if** **(**i**[-**1**]** **==** '"'**):**

quoteEnd **=** index

index **+=** 1

**for** i **in** **range(**quoteBegin **+** 1**,** quoteEnd **+** 1**):**

vector**[**quoteBegin**]** **+=** " " **+** vector**[**i**]**

**for** i **in** **range(**quoteBegin **+** 1**,** quoteEnd **+** 1**):**

**del** vector**[**quoteBegin **+** 1**]**

**if** **(**quoteEnd **!=** 0**):**

withOutQuotes **=** vector**[**quoteBegin**][**1**:-**1**]**

vector**[**quoteBegin**]** **=** withOutQuotes

**return** vector

**def** addToBeginAndEnd**(**self**,** vector**,** symbol**):**

index **=** 0

**for** i **in** vector**:**

vector**[**index**]** **=** symbol **+** i **+** symbol

index **+=** 1

**return** vector

**def** taskAdd**(**self**):**

table **=** self**.**view**.**askTableName**()**

column **=** self**.**view**.**askColumnName**(**table**)**

column **=** self**.**addToBeginAndEnd**(**column**,** '"'**)**

value **=** self**.**view**.**askValue**(**"added: "**)**

value **=** self**.**checkForQuotes**(**value**)**

value **=** self**.**addToBeginAndEnd**(**value**,** "'"**)**

**print(**value**)**

self**.**model**.**taskAdd**(**table**,** column**,** value**)**

self**.**view**.**printMessage**(**"Data added successfully. \n"**)**

**def** taskUpdate**(**self**):**

table**,** column**,** **id,** newValue **=** self**.**view**.**taksUpdate**()**

self**.**model**.**taskUpdate**(**table**,** column**,** **id,** newValue**)**

self**.**view**.**printMessage**(**"Data updated successfully. \n"**)**

**def** taskDelete**(**self**):**

table **=** self**.**view**.**askTableName**()**

**id** **=** self**.**view**.**askID**(**table**)**

self**.**model**.**taskDelete**(**table**,** **id)**

self**.**view**.**printMessage**(**"Data deleted successfully. \n"**)**

**def** taskGenerate**(**self**):**

table **=** self**.**view**.**askTableName**()**

rowNumber **=** **int(**self**.**view**.**askRowNumber**())**

self**.**model**.**generateRows**(**table**,** rowNumber**)**

self**.**view**.**printMessage**(**f"{rowNumber} rows in " **+** table **+** " table successfully generated. \n"**)**

**Результати роботи програми**

Отримання назв таблиць:

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Отримання назв стовпців таблиць:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

**Додавання рядків до таблиць:**

Таблиця User до додавання:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Додавання даних:

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Таблиця User після додавання рядка:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Таблиця Rental до додавання рядка:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Додавання рядка:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Таблиця Rental після додавання рядка:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Таблиця Equipment до додавання рядка:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Додавання рядка:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Таблиця Equipment після додавання рядка:

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

**Редагування даних:**

Таблиця Equipment до редагування рядка 5:

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

**Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание**

Таблиця Equipment після редагування рядка 5:

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

**Видалення даних:**

Таблиця Rental до видалення рядка 6:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Таблиця Rental після видалення рядка 6:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**Генерація даних:**

Таблиця User до генерації 5-ти рядків:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Таблиця User після генерації 5-ти рядків:

Изображение выглядит как текст, число, снимок экрана, Шрифт

Автоматически созданное описание