

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №2**

з дисципліни

**«Бази даних і засоби управління»**

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

Виконав: студент ІII курсу

ФПМ групи КВ-82

Чернухін Владислав

Перевірив: Павловський В. І.

Київ – 2020

*Мета роботи:* здобуття практичних навичок проектування та побудови реляційних баз даних та створення прикладних програм з базами даних

*Завдання* роботи полягає у наступному:

1. Реалізувати функціональні вимоги, наведені нижче.

*Функціональні вимоги:*

1. Реалізувати внесення, редагування та вилучення даних у базі засобами консольного інтерфейсу;
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі;
3. Забезпечити реалізацію пошуку за двома-трьома атрибутами з двох сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як перелічення, для логічного типу – значення True/False, для дат – у рамках діапазону дат;

*Додаткові вимоги:*

1. Передбачити перехоплення помилок. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки;
2. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

*Вимоги до інтерфейсу користувача:*

1. Використовувати консольний інтерфейс користувача.

**Логічна модель (Структура) БД “** Ігрова платформа **”** (засобами SqlDMB)

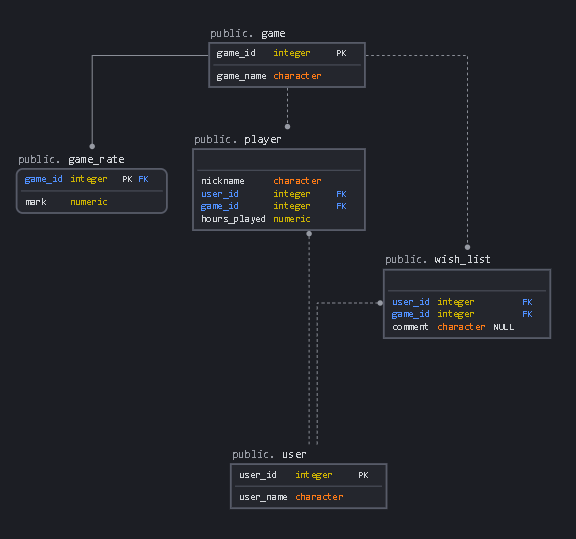


Рисунок 1 - Логічна модель (Структура) БД “ Ігрова платформа ” (засобами SqlDMB)

# Опис програми

Програма створена за допомогою мови програмування Python в середовищі розробки PyCharm Community Edition 2020.2.3

Програма створена за патерном MVC (Model-View-Controller). Складається відповідно з класів Model, View та Controller. Програма створена для управління базою даних за допомогою базових операцій СУБД PostgreSQL та реалізовує функціональні вимоги, що наведені у завданні. Вона складається з 4 модулів:

1. main.py –викликає функцію меню із Сontroller.py;
2. Model.py – модуль Model, який містить методи для управління даними програми та БД;
3. View.py – модуль View, який містить методи для відображення результатів роботи Model на екран;
4. Controller.py – модуль Controller, який містить методи для взаємодії з користувачем та виклику методів Model.

# Опис структури меню програми

На рисунку 2 зображена концептуальна модель програми

# 

Рисунок 2 – Концептуальна модель програми

**Нотація:** Модель побудована засобами програми draw.io

**Меню для навігації**

1. [main.py](#main)
2. [Controller.py](#contr)
3. [View.py](#view)
4. [Model.py](#model)
   1. Завдання 1
      1. [Метод, який додає інформацію до БД](#insert)
      2. [Метод, який видаляє інформацію з БД](#delete)
      3. [Метод, який редагує дані у БД](#update)
      4. [Результат роботи методу додавання](#res_ins)
      5. [Результат роботи методу видалення](#res_del)
      6. [Результат роботи методу редагування](#res_upd)
   2. Завдання 2
      1. [Метод, який додає випадкові дані до таблиці «user»](#rand)
      2. [Результат роботи методу додавання випадкових даних](#res_rand)
   3. Завдання 3
      1. [Метод пошуку](#select)
      2. [Результат роботи методу пошуку](#res_select)
5. Додаткові вимоги
   1. [Передбачення перехоплення помилок](#Controller)
   2. [Приклад перехоплення помилки](#res_try)

# Лістинг модуля Model.py

import psycopg2  
import random  
from View import View  
  
  
class Model:  
 @staticmethod  
 def Select():  
 connection = psycopg2.connect(host="localhost", port="5432",  
 database="steam",  
 user="postgres", password="")  
 cursor = connection.cursor()  
 border1 = "'" + input("first border: ") + "'"  
 border2 = "'" + input("second border: ") + "'"  
 select = 'SELECT \* FROM public."user" AS r1 ' \  
 'INNER JOIN (SELECT \* FROM public."player") AS c1 ON r1.user\_id = c1.user\_id ' \  
 f'WHERE user\_id BETWEEN {border1} AND {border2}'  
 cursor.execute(select)  
 records = cursor.fetchall()  
 cursor.close()  
 connection.close()  
 for row in records:  
 print("\nuser\_id =", row[0])  
 print("user\_name =", row[1])  
 print("nickname =", row[2])  
 print("game\_id =", row[5])  
 print("hours\_played =", row[6])  
  
 @staticmethod  
 def Insert():  
 flag = 0  
 connection = psycopg2.connect(host="localhost", port="5432",  
 database="steam",  
 user="postgres", password="")  
 cursor = connection.cursor()  
 while flag == 0:  
 table = View.list()  
 if table < 1 or table > 5:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif table == 1:  
 name = "'" + input("user\_id = ") + "'"  
 id = input("user\_name = ")  
 cursor.execute(f'INSERT INTO public."user" (user\_id, user\_name) VALUES ({id}, {name});')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag = 1  
 elif table == 2:  
 nickname = "'" + input("nickname = ") + "'"  
 user\_id = input("user\_id = ")  
 game\_id = input("game\_id = ")  
 hours\_played = input("hours\_played = ")  
 cursor.execute(f'INSERT INTO public."player" (nickname, user\_id, game\_id, hours\_played) '  
 f'VALUES ({nickname}, {user\_id}, {game\_id}, {hours\_played});')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag = 1  
 elif table == 3:  
 game\_id = input("game\_id = ")  
 game\_name = "'" + input("game\_name = ") + "'"  
  
 cursor.execute('INSERT INTO public."game" (game\_id ' \  
 f'game\_name) VALUES ({game\_id}, {game\_name});')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag = 1  
 elif table == 4:  
 comment = "'" + input("comment = ") + "'"  
 user\_id = input("user\_id = ")  
 game\_id = input("game\_id = ")  
 cursor.execute('INSERT INTO public."comments" (user\_id, game\_id, comment) ' \  
 f'VALUES ({user\_id}, {game\_id}, {comment});')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag = 1  
 elif table == 5:  
 game\_id = input("game\_id = ")  
 mark = input("mark = ")  
 cursor.execute('INSERT INTO public."rate" (game\_id, ' \  
 f'mark) VALUES ({game\_id}, {mark});')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag = 1  
  
 @staticmethod  
 def Delete():  
 flag1 = 0  
 flag2 = 0  
 connection = psycopg2.connect(host="localhost", port="5432",  
 database="steam",  
 user="postgres", password="")  
 cursor = connection.cursor()  
 while flag1 == 0:  
 table = View.list()  
 if table < 1 or table > 5:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif table == 1:  
 while flag2 == 0:  
 attribute = View.attribute\_list(1)  
 if attribute < 1 or attribute > 2:  
 print("\n...Incorrect input, try again...")  
 elif attribute == 1:  
 value = "'" + input('user\_id to delete = ') + "'"  
 where = f'"user\_id" = {value}'  
 flag2 = 1  
 elif attribute == 2:  
 value = input('user\_name to delete = ')  
 where = f'"user\_name" = {value}'  
 flag2 = 1  
 cursor.execute(f'DELETE FROM public."user" WHERE {where}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
 elif table == 2:  
 while flag2 == 0:  
 attribute = View.attribute\_list(2)  
 if attribute < 1 or attribute > 4:  
 print("\n...Incorrect input, try again...")  
 elif attribute == 1:  
 value = "'" + input('nickname to delete = ') + "'"  
 where = f'"nickname" = {value}'  
 flag2 = 1  
 elif attribute == 2:  
 value = "'" + input('user\_id to delete = ') + "'"  
 where = f'"user\_id" = {value}'  
 flag2 = 1  
 elif attribute == 3:  
 value = "'" + input('game\_id to delete = ') + "'"  
 where = f'"game\_id" = {value}'  
 flag2 = 1  
 elif attribute == 4:  
 value = "'" + input('hours\_played to delete = ') + "'"  
 where = f'"hours\_played" = {value}'  
 flag2 = 1  
 cursor.execute(f'DELETE FROM public."player" WHERE {where}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
 elif table == 3:  
 while flag2 == 0:  
 attribute = View.attribute\_list(3)  
 if attribute < 1 or attribute > 2:  
 print("\n...Incorrect input, try again...")  
 elif attribute == 1:  
 value = "'" + input('game\_id to delete = ') + "'"  
 where = f'"game\_id" = {value}'  
 flag2 = 1  
 elif attribute == 2:  
 value = "'" + input('game\_name to delete = ') + "'"  
 where = f'"game\_name" = {value}'  
 flag2 = 1  
 cursor.execute(f'DELETE FROM public."game" WHERE {where}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
 elif table == 4:  
 while flag2 == 0:  
 attribute = View.attribute\_list(4)  
 if attribute < 1 or attribute > 3:  
 print("\n...Incorrect input, try again...")  
 elif attribute == 1:  
 value = "'" + input('user\_id to delete = ') + "'"  
 where = f'"user\_id" = {value}'  
 flag2 = 1  
 elif attribute == 2:  
 value = "'" + input('game\_id to delete = ') + "'"  
 where = f'"game\_id" = {value}'  
 flag2 = 1  
 elif attribute == 3:  
 value = "'" + input('comment to delete = ') + "'"  
 where = f'"comment" = {value}'  
 flag2 = 1  
 cursor.execute(f'DELETE FROM public."wish\_list" WHERE {where}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
 elif table == 5:  
 while flag2 == 0:  
 attribute = View.attribute\_list(5)  
 if attribute < 1 or attribute > 2:  
 print("\n...Incorrect input, try again...")  
 elif attribute == 1:  
 value = "'" + input('game\_id to delete = ') + "'"  
 where = f'"game\_id" = {value}'  
 flag2 = 1  
 elif attribute == 2:  
 value = "'" + input('mark to delete = ') + "'"  
 where = f'"mark" = {value}'  
 flag2 = 1  
 cursor.execute(f'DELETE FROM public."game\_rate" WHERE {where}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
  
  
 @staticmethod  
 def show\_table(number):  
 flag = 0  
 table = 0  
 while flag == 0:  
 if number == 1:  
 table = View.list()  
 flag = 1  
 elif number == 2:  
 table += 1  
 if table == 7:  
 flag = 1  
 connection = psycopg2.connect(host="localhost", port="5432",  
 database="steam",  
 user="postgres", password="")  
 cursor = connection.cursor()  
 if table == 1:  
 print("table: User")  
 cursor.execute('SELECT \* FROM public."user"')  
 rows = cursor.fetchall()  
 View(table, rows).show()  
 print("\n")  
 elif table == 2:  
 print("table: Player")  
 cursor.execute('SELECT \* FROM public."player"')  
 rows = cursor.fetchall()  
 View(table, rows).show()  
 print("\n")  
 elif table == 3:  
 print("table: Game")  
 cursor.execute('SELECT \* FROM public."game"')  
 rows = cursor.fetchall()  
 View(table, rows).show()  
 print("\n")  
 elif table == 4:  
 print("table: Wish List")  
 cursor.execute('SELECT \* FROM public."wish\_list"')  
 rows = cursor.fetchall()  
 View(table, rows).show()  
 print("\n")  
 elif table == 5:  
 print("table: Game Rate")  
 cursor.execute('SELECT \* FROM public."game\_rate"')  
 rows = cursor.fetchall()  
 View(table, rows).show()  
 print("\n")  
  
 @staticmethod  
 def Random():  
 flag = 0  
 connection = psycopg2.connect(host="localhost", port="5432",  
 database="steam",  
 user="postgres", password="")  
 cursor = connection.cursor()  
 while flag == 0:  
 counter = int(input('How much data to generate? Input: '))  
 names = create\_user(counter)  
 if counter > 1:  
 flag = 1  
 else:  
 print('\n...Incorrect input, try again...\n')  
 for i in range(counter):  
 name = names[i]  
 id = random.randint(20, 1000)  
 cursor.execute('INSERT INTO public."user" ' \  
 f'(user\_id, user\_name) VALUES ({id}, {name});')  
 connection.commit()  
 cursor.close()  
 connection.close()  
  
 @staticmethod  
 def Update():  
 flag1 = 0  
 flag2 = 0  
 flag3 = 0  
 connection = psycopg2.connect(host="localhost", port="5432",  
 database="steam",  
 user="postgres", password="")  
 cursor = connection.cursor()  
 while flag1 == 0:  
 table = View.list()  
 if table < 1 or table > 5:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif table == 1:  
 while flag2 == 0:  
 where = View.attribute\_list(1)  
 if where < 1 or where > 2:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif where == 1:  
 user\_id = input('Attribute to update where user\_id = ')  
 flag2 = 1  
 elif where == 2:  
 user\_name = "'" + input('Attribute to update where user\_name = ') + "'"  
 flag2 = 1  
 while flag3 == 0:  
 attribute = View.attribute\_list(1)  
 if attribute < 1 or attribute > 2:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif attribute == 1:  
 new\_value = input('New value of attribute = ')  
 set = f'"user\_id" = {new\_value}'  
 flag3 = 1  
 elif attribute == 2:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"user\_name" = {new\_value}'  
 flag3 = 1  
 if where == 1:  
 cursor.execute(  
 f'UPDATE public."user" SET {set} WHERE "user\_id" = {user\_id}')  
 elif where == 2:  
 cursor.execute(  
 f'UPDATE public."user" SET {set} WHERE "user\_name" = {user\_name}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
 elif table == 2:  
 while flag2 == 0:  
 where = View.attribute\_list(2)  
 if where < 1 or where > 4:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif where == 1:  
 nickname = "'" + input('Attribute to update where nickname = ') + "'"  
 flag2 = 1  
 elif where == 2:  
 user\_id =input('Attribute to update where user\_id = ')  
 flag2 = 1  
 elif where == 3:  
 game\_id = input('Attribute to update where game\_id = ')  
 flag2 = 1  
 elif where == 4:  
 hours\_played = input(  
 'Attribute to update where hours\_played = ')  
 flag2 = 1  
 while flag3 == 0:  
 attribute = View.attribute\_list(2)  
 if attribute < 1 or attribute > 4:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif attribute == 1:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"nickname" = {new\_value}'  
 flag3 = 1  
 elif attribute == 2:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"user\_id" = {new\_value}'  
 flag3 = 1  
 elif attribute == 3:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"game\_id" = {new\_value}'  
 flag3 = 1  
 elif attribute == 4:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"hours\_played" = {new\_value}'  
 flag3 = 1  
 if where == 1:  
 cursor.execute(  
 f'UPDATE public."player" SET {set} WHERE "blog\_name" = {nickname}')  
 elif where == 2:  
 cursor.execute(  
 f'UPDATE public."player" SET {set} WHERE "author\_name" = {user\_id}')  
 elif where == 3:  
 cursor.execute(  
 f'UPDATE public."player" SET {set} WHERE "author\_id" = {game\_id}')  
 elif where == 4:  
 cursor.execute(  
 f'UPDATE public."player" SET {set} WHERE "text" = {hours\_played}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
 elif table == 3:  
 while flag2 == 0:  
 where = View.attribute\_list(3)  
 if where < 1 or where > 2:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif where == 1:  
 game\_id = input('Attribute to update where game\_id = ')  
 flag2 = 1  
 elif where == 2:  
 game\_name = "'" + input(  
 'Attribute to update where game\_name = ') + "'"  
 flag2 = 1  
 while flag3 == 0:  
 attribute = View.attribute\_list(3)  
 if attribute < 1 or attribute > 2:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif attribute == 1:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"game\_id" = {new\_value}'  
 flag3 = 1  
 elif attribute == 2:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"game\_name" = {new\_value}'  
 flag3 = 1  
 if where == 1:  
 cursor.execute(  
 f'UPDATE public."game" SET {set} WHERE "game\_id" = {game\_id}')  
 elif where == 2:  
 cursor.execute(  
 f'UPDATE public."game" SET {set} WHERE "game\_name" = {game\_name}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
 elif table == 4:  
 while flag2 == 0:  
 where = View.attribute\_list(4)  
 if where < 1 or where > 3:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif where == 1:  
 user\_id = input('Attribute to update where user\_id = ')  
 flag2 = 1  
 elif where == 2:  
 game\_id = input('Attribute to update where game\_id = ')  
 flag2 = 1  
 elif where == 3:  
 comment = "'" + input('Attribute to update where comment = ') + "'"  
 flag2 = 1  
 while flag3 == 0:  
 attribute = View.attribute\_list(4)  
 if attribute < 1 or attribute > 3:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif attribute == 1:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"user\_id" = {new\_value}'  
 flag3 = 1  
 elif attribute == 2:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"game\_id" = {new\_value}'  
 flag3 = 1  
 elif attribute == 3:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"comment" = {new\_value}'  
 flag3 = 1  
 if where == 1:  
 cursor.execute(  
 f'UPDATE public."wish\_list" SET {set} WHERE "user\_id" = {user\_id}')  
 elif where == 2:  
 cursor.execute(  
 f'UPDATE public."wish\_list" SET {set} WHERE "game\_id" = {game\_id}')  
 elif where == 3:  
 cursor.execute(  
 f'UPDATE public."wish\_list" SET {set} WHERE "comment" = {comment}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
 elif table == 5:  
 while flag2 == 0:  
 where = View.attribute\_list(5)  
 if where < 1 or where > 2:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif where == 1:  
 game\_id = "'" + input('Attribute to update where game\_id = ') + "'"  
 flag2 = 1  
 elif where == 2:  
 mark = "'" + input('Attribute to update where mark = ') + "'"  
 flag2 = 1  
 while flag3 == 0:  
 attribute = View.attribute\_list(5)  
 if attribute < 1 or attribute > 2:  
 print("\n...Incorrect input, try again...")  
 continue  
 elif attribute == 1:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"game\_id" = {new\_value}'  
 flag3 = 1  
 elif attribute == 2:  
 new\_value = "'" + input('New value of attribute = ') + "'"  
 set = f'"mark" = {new\_value}'  
 flag3 = 1  
 if where == 1:  
 cursor.execute(  
 f'UPDATE public."game\_rate" SET {set} WHERE "game\_id" = {game\_id}')  
 elif where == 2:  
 cursor.execute(  
 f'UPDATE public."game\_rate" SET {set} WHERE "mark" = {mark}')  
 connection.commit()  
 cursor.close()  
 connection.close()  
 flag1 = 1  
  
  
  
  
first\_names = ['Liam', 'Noah', 'Oliver', 'William', 'Elijah', 'James', 'Benjamin', 'Lucas',  
 'Mason', 'Ethan', 'Alexander', 'Henry', 'Jacob', 'Michael', 'Daniel', 'Logan',  
 'Jackson', 'Sebastian', 'Jack', 'Aiden']  
last\_names = ['Smith', 'Johnson', 'Williams', 'Brown', 'Jones', 'Garcia', 'MillerDavis',  
 'Rodriguez', 'Martinez', 'Hernandez', 'Lopez', 'Gonzalez', 'Wilson', 'Anderson',  
 'Thomas', 'Taylor', 'Moore', 'Jackson', 'Martin']  
  
def create\_user(count):  
 students\_count = count  
 res = []  
 for i in range(students\_count):  
 buf\_first\_name = "'" + random.choice(first\_names) + " "  
 buf\_last\_name = random.choice(last\_names) + "'"  
 res.append(buf\_first\_name + buf\_last\_name)  
 return res

# Лістинг модуля Controller.py

from Model import Model  
  
  
class Controller:  
 @staticmethod  
 def menu():  
 flag = 0  
 while flag == 0:  
 print(" menu ")  
 print("1. Choose one table to show")  
 print("2. Show all tables")  
 print("3. Insert data")  
 print("4. Delete data")  
 print("5. Update data")  
 print("6. Random users")  
 print("7. Select data")  
 print("8. Exit")  
 number = int(input('\nMake your choise: '))  
 if number == 1 or number == 2:  
 Model.show\_table(number)  
 elif number == 3:  
 try:  
 Model.Insert()  
 except Exception:  
 print("\nError,try again ...\n")  
 elif number == 4:  
 try:  
 Model.Delete()  
 except Exception:  
 print("\nError,try again ...\n")  
 elif number == 5:  
 try:  
 Model.Update()  
 except Exception:  
 print("\nError,try again ...\n")  
 elif number == 6:  
 try:  
 Model.Random()  
 except Exception:  
 print("\nError,try again ...\n")  
 elif number == 7:  
 try:  
 Model.Select()  
 except Exception:  
 print("\nError,try again ...\n")  
 elif number == 8:  
 flag = 1

# Лістинг модуля View.py

class View:  
 def \_\_init\_\_(self, table, records):  
 self.table = table  
 self.records = records  
  
 @staticmethod  
 def list():  
 print(" 1. User\n", "2. Player\n", "3. Game\n",  
 "4. wish list\n", "5. game rate\n")  
 number = input("\nMake your number: ")  
 return int(number)  
  
 def show(self):  
 if self.table == 1:  
 for row in self.records:  
 print("\nuser\_id =", row[0])  
 print("user\_name =", row[1])  
 elif self.table == 2:  
 for row in self.records:  
 print("\nnickname =", row[0])  
 print("user\_id =", row[1])  
 print("game\_id =", row[2])  
 print("hours\_played =", row[3])  
 elif self.table == 3:  
 for row in self.records:  
 print("\ngame\_id =", row[0])  
 print("game\_name =", row[1])  
 elif self.table == 4:  
 for row in self.records:  
 print("\nuser\_id =", row[0])  
 print("game\_id =", row[1])  
 print("comment =", row[2])  
 elif self.table == 5:  
 for row in self.records:  
 print("\ngame\_id =", row[0])  
 print("mark =", row[1])  
  
 @staticmethod  
 def attribute\_list(table):  
 if table == 1:  
 print(" 1. user\_id\n", "2. user\_name\n", )  
 if table == 2:  
 print(" 1. nickname\n", "2. user\_id\n", "3. game\_id\n", "4. hours\_played\n")  
 if table == 3:  
 print(" 1. game\_id\n", "2. game\_name\n")  
 if table == 4:  
 print(" 1. user\_id\n", "2. game\_id\n",  
 "3. comment\n")  
 if table == 5:  
 print(" 1. game\_id\n", "2. mark\n")  
 number = input('Number of attribute: ')  
 return int(number)

# Лістинг модуля main.py

from Controller import Controller  
  
Controller.menu()

# Результати роботи програми

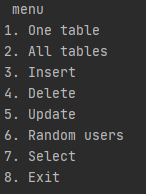


Рисунок 3 – Головне меню

# 

Рисунок 4 – Додавання даних до БД

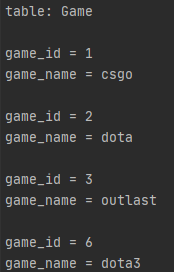


Рисунок 5 – Вміст таблиці «game» після додавання нових даних

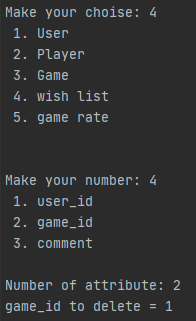


Рисунок 6 – Видалення даних з таблиці

**Нотація:** В даному прикладі ми видаляємо дані за назвою гри: З списку бажаних ігор усіх користувачів буде видалена гра з назвою dota . Програмою передбачено видалення за будь-яким атрибутом.

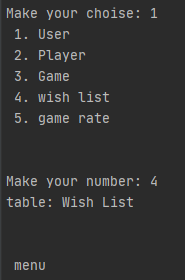


Рисунок 7 – Вміст таблиці «wish\_list» після видалення даних

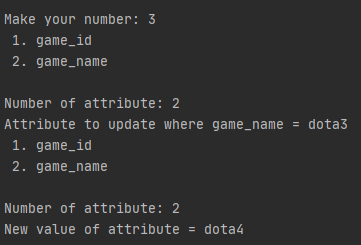


Рисунок 8 – Редагування даних у БД

**Нотація:** В даному прикладі користувач змінює назву гри dota3 на dota4 в таблиці game.

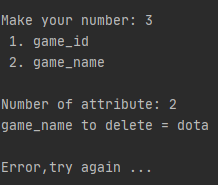


Рисунок 9 – Спроба видалити дані, які пов’язані з даними іншої таблиці зовнішнім ключем



Рисунок 10 – Генерування випадкових даних у таблиці «user»

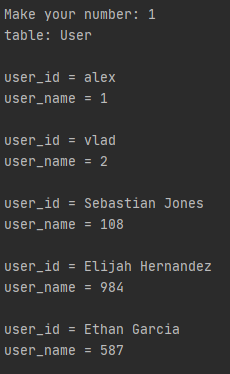


Рисунок 11 – Вміст таблиці «user» після додавання рандомізованих даних

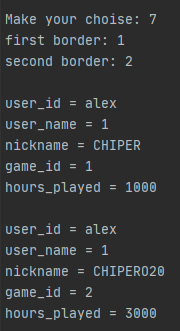


Рисунок 12 – Пошук з декількох сутностей одночасно

**Нотація:** В даному прикладі користувач з клавіатури вводить діапазон значень унікального id користувача, а програма виводить дані про усі профілі користувача в іграх, тобто користувач alex має профіль з ніком CHIPER в грі з id = 1 та профіль з ніком CHIPER0203 в грі з id = 2