Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №9**

**«РЕАЛИЗАЦИЯ СПРАВОЧНИКА С ПОДКЛЮЧЕННОЙ СУБД»**

**ПО МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ**

Выполнил: студент учебной группы

ИСПк-203-52-00

Криницын Олег Сергеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

1. **Цель домашней контрольной работы:**

Овладеть базовыми навыками разработки приложения и подключения системы управления базами данных (СУБД).

1. **Формулировка задания**
2. Разработать аниме справочник на Python используя библиотеку psycopg2;
3. Создать БД в pgAdmin;
4. Реализовать добавление, изменение и удаление данных в БД.
5. **Код программы**

import tkinter as tk

import psycopg2

from tkinter import ttk

host = "127.0.0.1"

user = "postgres"

password = "qwerty"

db\_name = "anime\_db"

root = tk.Tk()

root.title("Аниме Справочник")

root.configure(background='light green')

root.resizable(False, False)

connection = psycopg2.connect(

host=host,

user=user,

password=password,

database=db\_name

)

connection.autocommit = True

cursor = connection.cursor()

def show\_details():

selected\_item = treeview.focus()

anime\_details = treeview.item(selected\_item)

details\_text.delete("1.0", tk.END)

if 'values' in anime\_details:

values = anime\_details['values']

if len(values) >= 1:

details\_text.insert(tk.END, f"Описание: {values[2]}\n")

def add\_window():

def add\_entry():

name = name1\_entry.get()

year = year1\_entry.get()

genre = genre1\_entry.get()

description = description1\_entry.get()

if name and year and genre and description:

insert\_query = "INSERT INTO anime\_table (anime\_name, age\_vyh, genre, soderjanie) VALUES (%s, %s, %s, %s)"

data = (name, year, genre, description)

cursor.execute(insert\_query, data)

connection.commit()

name1\_entry.delete(0, tk.END)

year1\_entry.delete(0, tk.END)

genre1\_entry.delete(0, tk.END)

description1\_entry.delete(0, tk.END)

output\_table()

window\_add.destroy()

window\_add = tk.Tk()

window\_add.title("Добавление записи")

window\_add.geometry("400x300")

window\_add.configure(background='light green')

window\_add.resizable(False, False)

redac\_label = ttk.Label(window\_add, text="Название:", background='light green')

redac\_label.pack()

name1\_entry = ttk.Entry(window\_add)

name1\_entry.pack()

year1\_label = ttk.Label(window\_add, text="Год:", background='light green')

year1\_label.pack()

year1\_entry = ttk.Entry(window\_add)

year1\_entry.pack()

genre1\_label = ttk.Label(window\_add, text="Жанр:", background='light green')

genre1\_label.pack()

genre1\_entry = ttk.Entry(window\_add)

genre1\_entry.pack()

description1\_label = ttk.Label(window\_add, text="Описание:", background='light green')

description1\_label.pack()

description1\_entry = ttk.Entry(window\_add)

description1\_entry.pack()

add1\_button = tk.Button(window\_add, text="Добавить", command=add\_entry, bg='aquamarine')

add1\_button.pack(pady=5)

def delete\_entry():

selected\_item = treeview.focus()

if selected\_item:

anime\_name = treeview.item(selected\_item, "text")

delete\_query = "DELETE FROM anime\_table WHERE anime\_name = %s"

cursor.execute(delete\_query, (anime\_name,))

connection.commit()

treeview.delete(selected\_item)

output\_table()

def edit\_window():

def update\_entry():

selected\_item = treeview.focus()

if selected\_item:

name = name\_entry.get()

year = year\_entry.get()

genre = genre\_entry.get()

description = description\_entry.get()

if name and year and genre and description:

update\_query = "UPDATE anime\_table SET age\_vyh = %s, genre = %s, soderjanie = %s WHERE anime\_name = %s"

data = (year, genre, description, name)

cursor.execute(update\_query, data)

connection.commit()

treeview.item(selected\_item, text=name, values=(year, genre, description))

output\_table()

edit\_window.destroy()

selected\_item = treeview.focus()

if selected\_item:

edit\_window = tk.Toplevel(root)

edit\_window.title("Редактирование записи")

edit\_window.geometry("400x300")

edit\_window.configure(background='light green')

edit\_window.resizable(False, False)

anime\_details = treeview.item(selected\_item)

values = anime\_details['values']

name\_label = ttk.Label(edit\_window, text="Название:", background='light green')

name\_label.pack()

name\_entry = ttk.Entry(edit\_window)

name\_entry.pack()

name\_entry.insert(tk.END, anime\_details['text'])

year\_label = ttk.Label(edit\_window, text="Год:", background='light green')

year\_label.pack()

year\_entry = ttk.Entry(edit\_window)

year\_entry.pack()

year\_entry.insert(tk.END, values[0])

genre\_label = ttk.Label(edit\_window, text="Жанр:", background='light green')

genre\_label.pack()

genre\_entry = ttk.Entry(edit\_window)

genre\_entry.pack()

genre\_entry.insert(tk.END, values[1])

description\_label = ttk.Label(edit\_window, text="Описание:", background='light green')

description\_label.pack()

description\_entry = ttk.Entry(edit\_window)

description\_entry.pack()

description\_entry.insert(tk.END, values[2])

save\_button = tk.Button(edit\_window, text="Сохранить", command=update\_entry, bg='light goldenrod')

save\_button.pack(pady=5)

def output\_table():

treeview.delete(\*treeview.get\_children())

select\_query = "SELECT \* FROM anime\_table"

cursor.execute(select\_query)

records = cursor.fetchall()

for row in records:

name = row[0]

year = row[1]

genre = row[2]

description = row[3]

treeview.insert("", tk.END, text=name, values=(year, genre, description))

treeview = ttk.Treeview(root)

treeview["columns"] = ("year", "genre", "description")

treeview.heading("#0", text="Название")

treeview.heading("year", text="Год")

treeview.heading("genre", text="Жанр")

treeview.heading("description", text="Описание")

treeview.pack(pady=10)

details\_text = tk.Text(root, height=4, width=40)

details\_text.pack(pady=5)

# Create a frame to hold the buttons

button\_frame = tk.Frame(root, bg='light green')

button\_frame.pack(pady=5)

details\_button = tk.Button(button\_frame, text="Показать детали", command=show\_details, bg='light blue')

details\_button.grid(row=0, column=0, padx=5)

add\_button = tk.Button(button\_frame, text="Добавить", command=add\_window, bg='aquamarine')

add\_button.grid(row=0, column=1, padx=5)

edit\_button = tk.Button(button\_frame, text="Редактировать", command=edit\_window, bg='light goldenrod')

edit\_button.grid(row=0, column=2, padx=5)

delete\_button = tk.Button(button\_frame, text="Удалить", command=delete\_entry, bg='coral1')

delete\_button.grid(row=0, column=3, padx=5)

output\_button = tk.Button(button\_frame, text="Выйти?", command=root.destroy, bg='tomato3')

output\_button.grid(row=0, column=4, padx=5)

def on\_closing():

cursor.close()

connection.close()

root.destroy()

root.protocol("WM\_DELETE\_WINDOW", on\_closing)

output\_table()

root.mainloop()

1. **Результат выполнения программы**

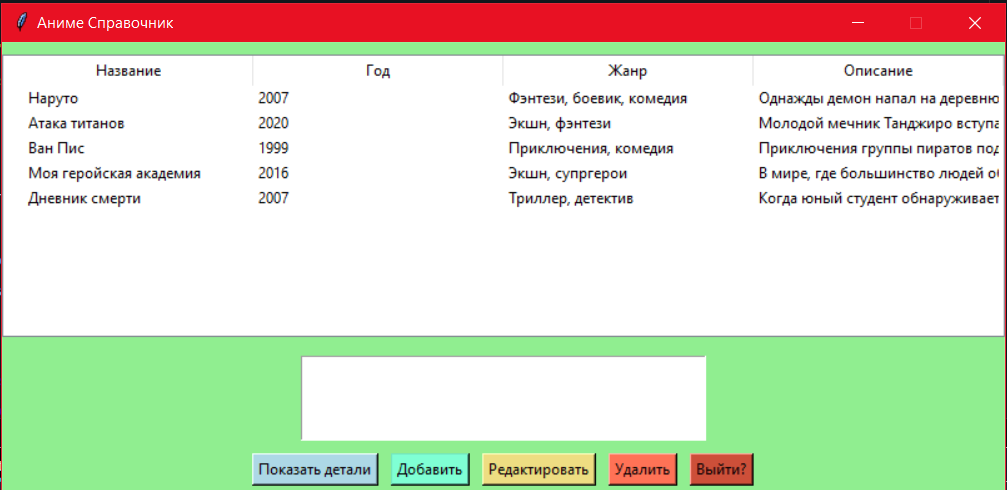


Рисунок 1 – Результат выполнения программы

Алгоритм кода:

1. Импортирование необходимых модулей и библиотек, таких как tkinter для создания GUI и psycopg2 для работы с PostgreSQL базой данных.
2. Определение параметров подключения к базе данных, таких как хост, имя пользователя, пароль и имя базы данных.
3. Создание главного окна с настройками, заголовком и фоновым цветом.
4. Установление соединения с базой данных PostgreSQL, используя параметры подключения.
5. Создание курсора для выполнения SQL-запросов.
6. Определение функции show\_details, которая отображает описание выбранного аниме.
7. Определение функции add\_window, которая создает дочернее окно для добавления новой записи об аниме.
8. Определение функции delete\_entry, которая удаляет выбранную запись об аниме.
9. Определение функции edit\_window, которая создает дочернее окно для редактирования выбранной записи об аниме.
10. Определение функции output\_table, которая выводит данные из базы данных в виде таблицы в главном окне.
11. Создание виджета treeview для отображения данных в виде таблицы.
12. Создание виджета details\_text для отображения описания выбранного аниме.
13. Создание фрейма для размещения кнопок.
14. Создание кнопок, которые выполняют соответствующие функции при нажатии.
15. Определение функции on\_closing, которая закрывает курсор и соединение с базой данных при закрытии приложения.
16. Вызов функции output\_table для начальной загрузки данных из базы данных.
17. Запуск главного цикла обработки событий (root.mainloop()), чтобы приложение оставалось открытым и отзывчивым для пользовательских действий.

Элементы, используемые в коде:

1. tkinter: Библиотека для создания графического интерфейса. Используется для создания окон, кнопок, текстовых полей и других виджетов GUI.
2. psycopg2: Библиотека для работы с PostgreSQL базой данных. Используется для установления соединения с базой данных, выполнения SQL-запросов и получения результатов.
3. Tk(): Класс, представляющий главное окно приложения.
4. Toplevel(): Класс, представляющий дочернее окно приложения.
5. Frame(): Виджет, представляющий контейнер для группировки других виджетов.
6. Treeview(): Виджет, представляющий дерево или таблицу для отображения данных.
7. Text(): Виджет, представляющий текстовое поле для отображения текста.
8. Button(): Виджет, представляющий кнопку.
9. Label(): Виджет, представляющий текстовую метку.
10. Entry(): Виджет, представляющий поле для ввода текста.
11. **Вывод**

Нами была поставлена цель работы: Овладеть базовыми навыками разработки приложения и подключения системы управления базами данных (СУБД). Работа была разделена на несколько этапов: цель, формулировка задания, написание кода программы, результат выполнения программы, вывод. Код программы был написан на Python с использованием библиотек tkinter и psycopg2. Итак, делая вывод, хочу сказать, что в ходе работы все цели были достигнуты.