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**

**«ОСНОВЫ СОБЫТИЙНО-ОРИЕНТИРОВАННОГО ПРОГРМАММИРОВАНИЯ»**

**ПО МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ**

Выполнил: студент учебной группы

ИСПк-101-51-00

Сунцов Александр Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы:** получить представление об организации хранения и использования информации посредством типизированных файлов, закрепить навыки создания пользовательского интерфейса.

**Скриншот задания**

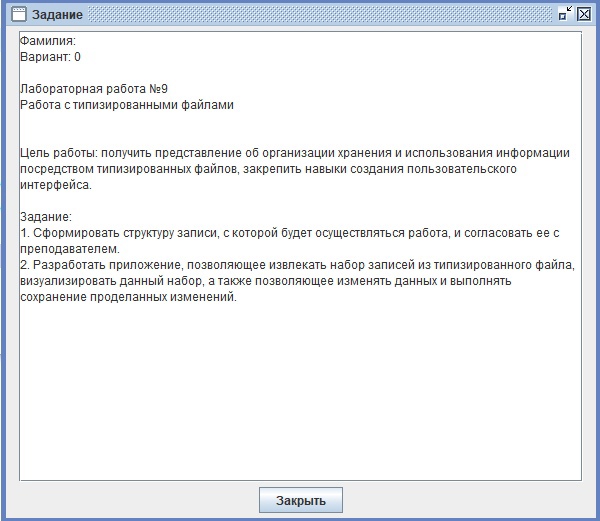


Рисунок 1 – Задание.

**Техническое обоснование**

При создании программы был выбран язык программирования Python, так как IDE Lazarus не подключается к выбранной мною базе данных PostgreSQL, поэтому было принято решение написать программу на Python c использованием библиотек tkinter и psycopg2. Библиотека tkinter предоставляет необходимые средства для создания простейшего пользовательского интерфейса. Библиотека psycopg2, предназначенная для обозначенной ранее базе данных PostgreSQL, была выбрана потому, что она обеспечивает удобные методы для выполнения SQL-запросов.

**Код программы**

import tkinter as tik

import psycopg2

from tkinter import messagebox, Tk, Label, ttk

from psycopg2 import connect

# Параметры базы данных

host = 'localhost'

port = '5432'

database = 'Movies'

user = 'postgres'

password = 'Zasada0902\_2000'

# Настройка первичной формы

class Movies(tik.Toplevel):

def \_\_init\_\_(self, parent):

super().\_\_init\_\_()

self.title("Ввод данных о фильмах")

self.parent = parent

self.geometry("300x190")

self.film\_label = tik.Label(self, text="Название:")

self.film\_label.grid(row=0, column=0)

self.film\_entry = tik.Entry(self)

self.film\_entry.grid(row=0, column=1)

self.status\_list = ["Просмотрено", "Брошено"]

self.status\_label = tik.Label(self, text="Статус:")

self.status\_label.grid(row=1, column=0)

self.status\_combobox = ttk.Combobox(self, values=self.status\_list, state="readonly")

self.status\_combobox.grid(row=1, column=1)

self.rate\_label = tik.Label(self, text="Оценка от 1 до 10:")

self.rate\_label.grid(row=2, column=0)

self.rate\_combobox = ttk.Combobox(self, values=list(range(1, 11)), state="readonly")

self.rate\_combobox.grid(row=2, column=1)

self.genre\_list = ["Приключение", "Хоррор", "Мелодрама", "Исторический", "Военный", "Боевик", "Драма",

"Мистика", "Фантастика", "Детектив"]

self.genre\_label = tik.Label(self, text="Жанр:")

self.genre\_label.grid(row=3, column=0)

self.genre\_combobox = ttk.Combobox(self, values=self.genre\_list, state="readonly")

self.genre\_combobox.grid(row=3, column=1)

self.type\_list = ["Фильм", "Сериал", "Мультфильм", "Аниме-фильм", "Аниме-сериал"]

self.type\_label = tik.Label(self, text="Тип:")

self.type\_label.grid(row=4, column=0)

self.type\_combobox = ttk.Combobox(self, values=self.type\_list, state="readonly")

self.type\_combobox.grid(row=4, column=1)

self.insert\_btn = tik.Button(self, text="Внести в базу данных", command=self.insert\_data)

self.insert\_btn.grid(row=5, column=1, columnspan=2)

self.back\_btn = tik.Button(self, text="Назад", command=self.get\_back)

self.back\_btn.grid(row=6, column=1, columnspan=2)

def insert\_data(self):

conn = psycopg2.connect(host=host, port=port, database=database, user=user, password=password)

cur = conn.cursor()

rating = int(self.rate\_combobox.get())

data = (

self.film\_entry.get(), self.status\_combobox.get(), rating, self.genre\_combobox.get(),

self.type\_combobox.get()

)

cur.execute("INSERT INTO Film (title, status, rating, genre, type) VALUES (%s, %s, %s, %s, %s)", data)

conn.commit()

messagebox.showinfo("Успех", "Данные успешно добавлены в таблицу")

cur.close()

conn.close()

self.destroy()

self.parent.deiconify()

def get\_back(self):

self.destroy()

self.parent.deiconify()

class ViewData(tik.Toplevel):

def \_\_init\_\_(self, parent):

super().\_\_init\_\_()

self.title("Просмотр данных")

self.parent = parent

self.back\_btn = tik.Button(self, text="Назад", command=self.get\_back, width=10, height=2)

self.back\_btn.pack()

self.data\_frame = tik.Frame(self)

self.data\_frame.pack()

self.column\_labels = ['Название фильма', 'Статус', 'Оценка фильма', 'Жанр', 'Тип']

self.read\_data()

def get\_back(self):

self.destroy()

self.parent.deiconify()

# Фуникция чтения

def read\_data(self):

self.title("База данных просмотренных шедевров и не очень")

self.geometry("700x500")

conn = psycopg2.connect(host=host, port=port, database=database, user=user, password=password)

cur = conn.cursor()

cur.execute("SELECT \* FROM Film")

results = cur.fetchall()

# Отчистка предыдущей попытки ввода данных

for widget in self.data\_frame.grid\_slaves():

widget.destroy()

# Описание колонок

for col, label\_text in enumerate(self.column\_labels):

label = tik.Label(self.data\_frame, text=label\_text, font='bold')

label.grid(row=0, column=col)

# Отображение наших данных

for row, data in enumerate(results):

for col, value in enumerate(data):

if col == 2: # Оценка просмотренных фильмов, сериалов и тд.

value = str(value) # Изменение типа для отображение в базе

label = tik.Label(self.data\_frame, text=value)

label.grid(row=row + 1, column=col)

delete\_btn = tik.Button(self.data\_frame, text="Удалить",

command=lambda row=row: self.delete\_data(row))

delete\_btn.grid(row=row + 1, column=7)

cur.close()

conn.close()

# Функция удаления

def delete\_data(self, row):

conn = psycopg2.connect(host=host, port=port, database=database, user=user, password=password)

cur = conn.cursor()

cur.execute("SELECT \* FROM Film")

results = cur.fetchall()

# Удаление выбранной строки из базы

delete\_title = results[row][0]

cur.execute("DELETE FROM Film WHERE title = %s", (delete\_title,))

conn.commit()

cur.close()

conn.close()

print("Данные успешно удалены из базы!)")

# Повторный просмотр базы для графического интерфейсика

self.read\_data()

class Edit\_data(tik.Toplevel):

def \_\_init\_\_(self, parent):

super().\_\_init\_\_()

self.title("Изменение записи")

self.geometry("300x180")

self.parent = parent

self.film\_label = tik.Label(self, text="Название:")

self.film\_label.grid(row=0, column=0)

self.film\_entry = tik.Entry(self)

self.film\_entry.grid(row=0, column=1)

self.status\_list = ["Просмотрено", "Брошено"]

self.status\_label = tik.Label(self, text="Статус:")

self.status\_label.grid(row=1, column=0)

self.status\_combobox = ttk.Combobox(self, values=self.status\_list, state="readonly")

self.status\_combobox.grid(row=1, column=1)

self.rate\_label = tik.Label(self, text="Оценка фильма от 1 до 10:")

self.rate\_label.grid(row=2, column=0)

self.rate\_combobox = ttk.Combobox(self, values=list(range(1, 11)), state="readonly")

self.rate\_combobox.grid(row=2, column=1)

self.genre\_list = ["Приключение", "Хоррор", "Мелодрама", "Исторический", "Военный", "Боевик",

"Драма", "Мистика", "Фантастика", "Детектив"]

self.genre\_label = tik.Label(self, text="Жанр:")

self.genre\_label.grid(row=3, column=0)

self.genre\_combobox = ttk.Combobox(self, values=self.genre\_list, state="readonly")

self.genre\_combobox.grid(row=3, column=1)

self.type\_list = ["Фильм", "Сериал", "Мультфильм", "Аниме-фильм", "Аниме-сериал"]

self.type\_label = tik.Label(self, text="Тип:")

self.type\_label.grid(row=4, column=0)

self.type\_combobox = ttk.Combobox(self, values=self.type\_list, state="readonly")

self.type\_combobox.grid(row=4, column=1)

self.search\_button = tik.Button(self, text="Поиск по названию", command=self.search\_by\_title)

self.search\_button.grid(row=5, column=0, columnspan=2)

self.update\_button = tik.Button(self, text="Обновить данные", command=self.update\_data)

self.update\_button.grid(row=6, column=0, columnspan=2)

self.back\_btn = tik.Button(self, text="Назад", command=self.get\_back)

self.back\_btn.grid(row=7, column=0, columnspan=2)

self.conn = psycopg2.connect(host=host, port=port, database=database, user=user, password=password)

self.cursor = self.conn.cursor()

def fetch\_data(self):

self.cursor.execute("SELECT title, status, rating, genre, "

"type FROM Film")

result = self.cursor.fetchone()

if result:

self.film\_entry.insert(0, result[0])

self.status\_combobox.set(result[1])

self.rate\_combobox.set(result[2])

self.genre\_combobox.insert(0, result[3])

self.type\_combobox.insert(0, result[4])

def get\_back(self):

self.destroy()

self.parent.deiconify()

def search\_by\_title(self):

title = self.film\_entry.get()

self.cursor.execute("SELECT title, status, rating, genre, type"

" FROM Film WHERE title=%s", (title,))

result = self.cursor.fetchone()

if result:

self.status\_combobox.set(result[1])

self.rate\_combobox.set(result[2])

self.genre\_combobox.set(result[3])

self.type\_combobox.set(result[4])

messagebox.showinfo("Результат поиска", "Запись найдена!")

else:

messagebox.showinfo("Результат поиска", "Запись не найдена.")

def update\_data(self):

title = self.film\_entry.get()

status = self.status\_combobox.get()

rating = self.rate\_combobox.get()

genre = self.genre\_combobox.get()

type = self.type\_combobox.get()

self.cursor.execute("UPDATE Film SET title=%s, status=%s, rating=%s, genre=%s, type=%s WHERE title=%s",

(title, status, rating, genre, type, title))

self.conn.commit()

messagebox.showinfo("Обновлено", "Данные успешно обновлены!")

# Создание главной формы для всей программы

class MainForm(tik.Tk):

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

super().\_\_init\_\_()

self.title("Главный экран")

self.geometry("300x100")

self.input\_btn = tik.Button(self, text="Внести данные о фильме", command=self.open\_input\_form)

self.input\_btn.pack()

self.view\_btn = tik.Button(self, text="Просмотр базы данных", command=self.open\_view\_form)

self.view\_btn.pack()

self.edit\_btn = tik.Button(self, text="Изменить запись", command=self.open\_editer\_form)

self.edit\_btn.pack()

def open\_view\_form(self):

self.withdraw() # Скрытие главной формы

view\_form = ViewData(self)

view\_form.protocol("WM\_DELETE\_WINDOW", self.on\_close\_view\_form)

def open\_input\_form(self):

self.withdraw() # Скрытие той же главной формы

input\_form = Movies(self)

input\_form.protocol("WM\_DELETE\_WINDOW", self.on\_close\_input\_form)

def open\_editer\_form(self):

self.withdraw()

edit\_form = Edit\_data(self)

edit\_form.protocol("WM\_DELETE\_WINDOW", self.on\_close\_edit\_form)

def on\_close\_input\_form(self):

self.withdraw() # Скрытие формы

self.deiconify() # Открытие главной формы после добавки Аниме в базу

def on\_close\_view\_form(self):

self.withdraw() # Скрытие той же главной формы

self.deiconify() # Открытие главной формы после просмотра базы

def on\_close\_edit\_form(self):

self.withdraw() # Скрытие той же главной формы

self.deiconify() # Открытие главной формы после просмотра базы

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

main\_form = MainForm()

main\_form.mainloop()

**Результат выполнение программы**

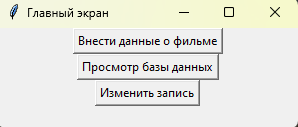
****

Рисунок 1 – Главный экран

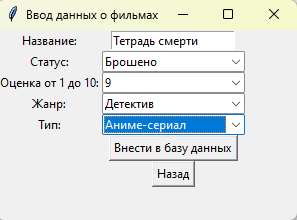


Рисунок 2 – Экран с внесением данных

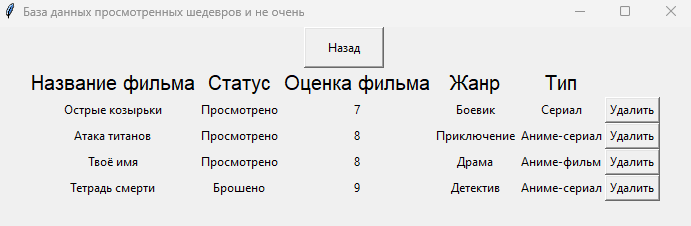


Рисунок 3 – Экран с таблицей данных

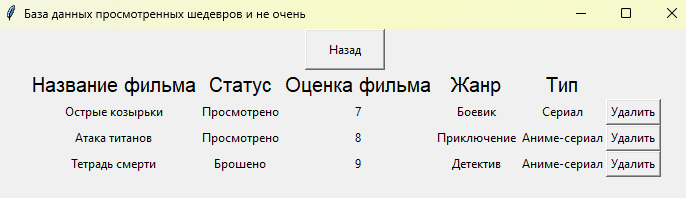


Рисунок 4 – Экран с таблицей данных после удаления одной строки

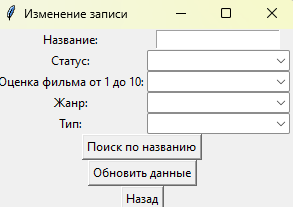


Рисунок 5 – Экран для изменения данных

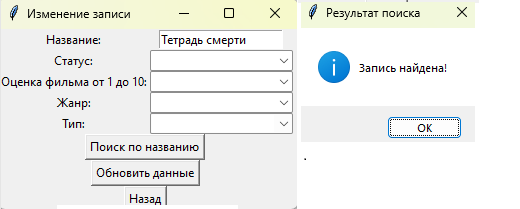


Рисунок 6 – Поиск по названию в экране изменения данных

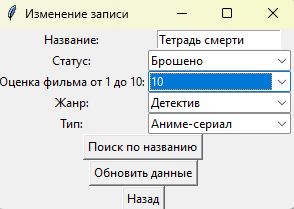


Рисунок 7 – Внесение изменений данных

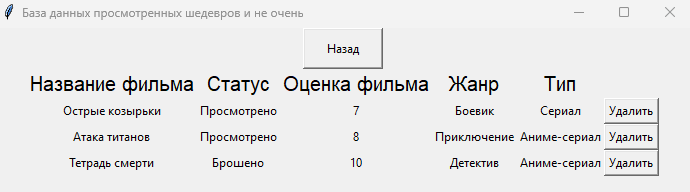


Рисунок 8 – Измененные данные на экране с таблицей данных

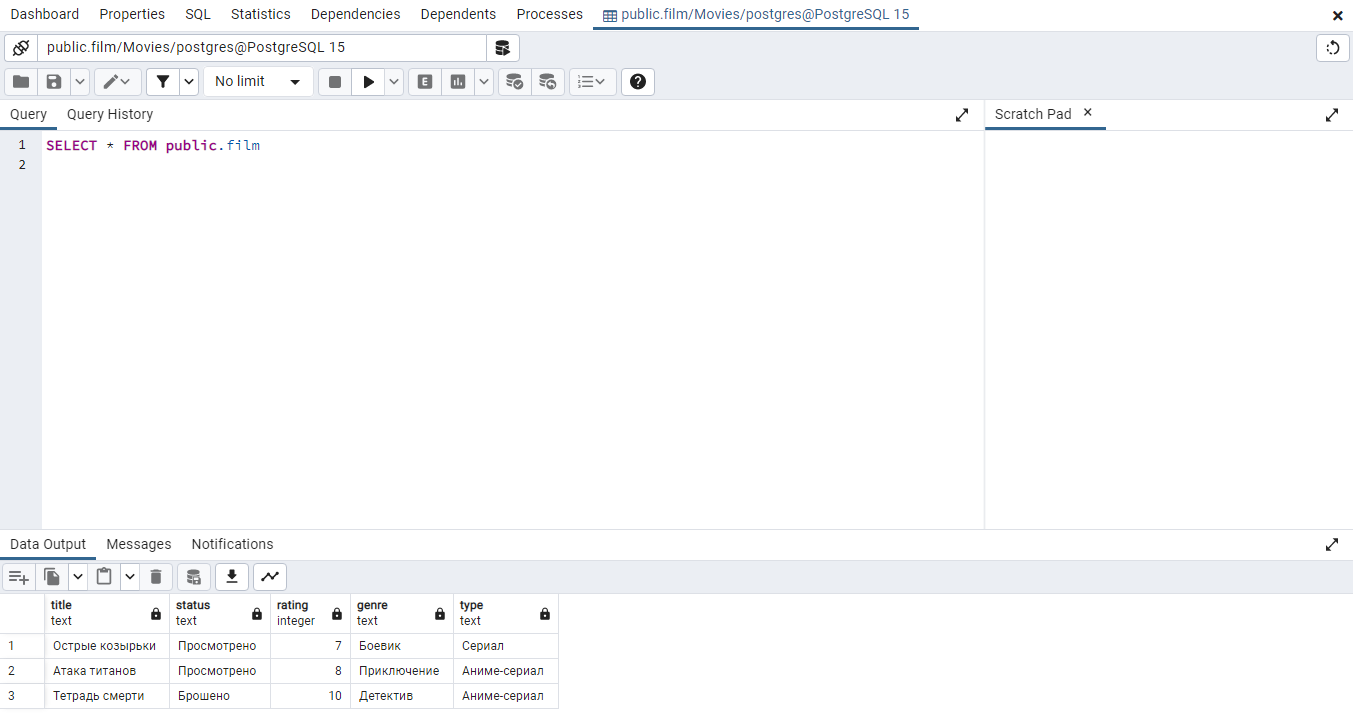


Рисунок 9 – Таблица в pgAdmin

**Вывод**

В результате выполнения Домашней контрольной работы №9 была создана программа, работающая с сервером PostgreSQL. При выполнении данной работы были получены дополнительные навыки работы с Python, библиотекой tkinter и получены навыки работы с библиотекой psycopg2. В итоге было получено представление об организации хранения и использования информации посредством типизированных файлов, а также закреплены навыки создания пользовательского интерфейса.