Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ОТЧЕТ

по лабораторной работе № 6

Создание приложения для базы данных.

на тему

«Сеть клубов карате»

Студент: Р.С. Кочеров

Преподаватель: Д.В. Куприянова

МИНСК 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc195089237)

[1. ИНТЕРФЕЙС ПРОГРАММЫ 4](#_Toc195089238)

[1.1 Главное окно приложения 4](#_Toc195089239)

[1.2 Окно создания нового запроса 5](#_Toc195089240)

[1.3 Окно удаления запроса 6](#_Toc195089241)

[1.4 Окно для выполнения запросов 7](#_Toc195089242)

[1.5 Окно создания новой таблицы 7](#_Toc195089243)

[1.6 Окно удаления таблицы 8](#_Toc195089244)

[1.7 Окно добавления данных в таблицу 8](#_Toc195089245)

[1.8 Окно просмотра данных из таблицы 9](#_Toc195089246)

[1.9 Окно удаления данных из таблицы 10](#_Toc195089247)

[1.10 Кнопка экспорта в Excel 10](#_Toc195089248)

[1.11 Кнопка создания резервной копии 11](#_Toc195089249)

[ЗАКЛЮЧЕНИЕ 12](#_Toc195089250)

[ПРИЛОЖЕНИЕ А 13](#_Toc195089251)

# **ВВЕДЕНИЕ**

В современном мире управления данными важность эффективного взаимодействия с базами данных невозможно переоценить. Программа «Работа с базой данных Сеть клубов карате» разработана для упрощения процессов управления данными в контексте клуба карате. Она предоставляет пользователям возможность создавать, редактировать и удалять таблицы, а также выполнять SQL-запросы для получения нужной информации. Используя графический интерфейс на базе Tkinter, программа делает взаимодействие с базой данных интуитивно понятным и доступным даже для пользователей без глубоких технических знаний.

Программа включает в себя функции для экспорта данных в формат Excel, создания резервных копий базы данных и управления пользовательскими запросами. Это позволяет не только сохранить важную информацию, но и упростить её анализ и обработку.

Для запуска программы необходима операционная система Windows 10 или выше (другие ОС, такие как Linux и macOS, также поддерживаются при наличии установленного Python и необходимых библиотек). Убедитесь, что у вас установлен Python версии 3.7 или выше и PostgreSQL версии 9.6 или выше. Программа требует библиотеки psycopg2, pandas, sqlalchemy, openpyxl, а также библиотеку tkinter, которая входит в стандартный пакет Python. Минимальные аппаратные требования включают 2 ГБ оперативной памяти и 100 МБ свободного места на диске.

Чтобы запустить программу, сначала установите Python с официального сайта и добавьте его в системный PATH. Затем установите PostgreSQL и настройте базу данных kerete\_club, создав пользователя postgres с паролем 1111. Далее откройте командную строку и установите необходимые библиотеки, выполнив команды: pip install psycopg2 pandas sqlalchemy openpyxl. Сохраните код в файл, например, database\_app.py, откройте командную строку, перейдите в директорию с файлом и запустите программу командой python database\_app.py. Программа должна запуститься, и вы сможете взаимодействовать с базой данных через графический интерфейс.

# **1. ИНТЕРФЕЙС ПРОГРАММЫ**

## **1.1 Главное окно приложения**

Главное окно приложения «Работа с базой данных сеть клубов карате» предоставляет пользователю удобный интерфейс для управления базой данных. Оно разделено на три панели: левая панель для работы с таблицами, центральная панель для управления SQL-запросами и правая панель для создания и удаления таблиц, а также создания резервных копий. Это позволяет пользователю быстро и легко выполнять различные задачи.

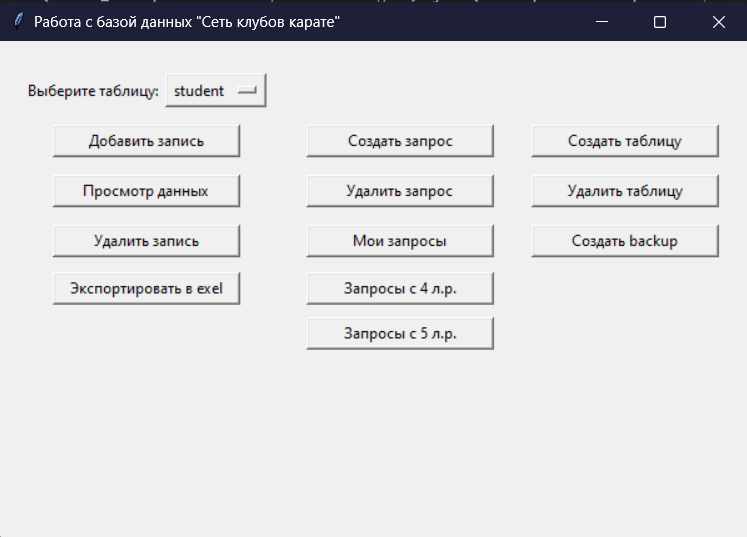


Рисунок 1.1 – Главное окно приложения

Кнопка «Выберите таблицу» представляет собой выпадающий список, который позволяет пользователю выбрать конкретную таблицу для работы с данными, связанными с сетью клубов карате. В этом контексте пользователю доступны различные таблицы, такие как клубы, тренеры, учащиеся, тренировки, соревнования и другие. После выбора таблицы система активирует все соответствующие функции и возможности, что позволяет эффективно управлять информацией, хранящейся в этих таблицах.

Кнопка «Добавить запись» открывает форму для ввода новой записи в выбранную таблицу. Пользователь может ввести информацию о клубе, тренере, учащемся или тренировке. Например, для таблицы «Клубы» это могут быть название клуба, его адрес, контактные данные и количество участников. Для таблицы «Тренеры» можно указать имя, квалификацию и опыт работы. После заполнения формы запись будет добавлена в базу данных, что позволяет поддерживать актуальность и полноту информации о клубах и их деятельности.

Кнопка «Создать запрос» предоставляет возможность формировать SQL-запросы для извлечения данных из таблиц. При нажатии на эту кнопку открывается интерфейс, позволяющий писать и редактировать SQL-запросы. Пользователь может задавать условия для получения конкретной информации, например, список тренеров, работающих в определенном клубе, или расписание тренировок для группы учащихся. Это значительно упрощает процесс анализа данных и позволяет быстро находить нужную информацию.

Кнопка «Удалить запрос» предназначена для удаления ранее созданного запроса из списка, что помогает пользователю управлять своими запросами и поддерживать порядок в интерфейсе. Это особенно полезно, если пользователь работает с большим количеством запросов и хочет сосредоточиться только на актуальных.

Кнопка «Удалить таблицу» позволяет удалить выбранную таблицу из базы данных, если она больше не актуальна или содержит устаревшие данные. Удаление требует подтверждения, что добавляет уровень безопасности и предотвращает случайные действия.

Кнопка «Создать backup» служит для создания резервной копии базы данных. Эта функция крайне важна для защиты данных от потерь, вызванных ошибками, повреждениями или другими непредвиденными обстоятельствами. Создание резервной копии позволяет восстановить информацию в случае необходимости, обеспечивая безопасность и надежность данных.

Кнопка «Экспортировать в Excel» позволяет экспортировать данные из выбранной таблицы или результаты запросов в файл Excel. Это делает данные более доступными для анализа и отчетности, поскольку Excel предоставляет инструменты для работы с данными, включая создание графиков и диаграмм. Экспорт в Excel также упрощает дальнейшую работу с данными в других приложениях, что делает эту функцию особенно полезной для тренеров и администраторов клубов.

При нажатии на кнопки «Запросы с 4 лабораторной работы» и «Запросы с 5 лабораторной работы.» выполняются заранее определенные запросы.

## **1.2 Окно создания нового запроса**

Окно создания нового запроса предназначено для ввода и сохранения SQL-запросов. Пользователь может ввести описание запроса и сам SQL-код, после чего сохранить его для дальнейшего использования. Это полезно для организации часто используемых запросов. Перед сохранением пользователь может посмотреть на отображение результата запроса.

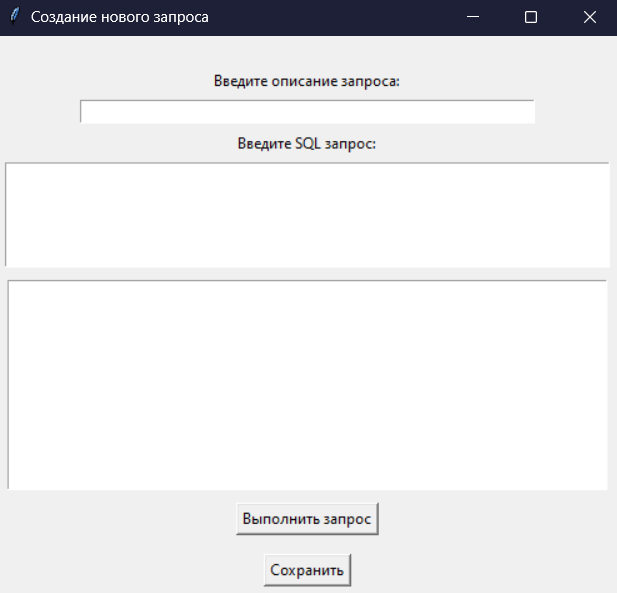


Рисунок 1.2 – Таблица Окно создания нового запроса

## **1.3 Окно удаления запроса**

Окно удаления запроса позволяет пользователю удалить ранее сохраненные SQL-запросы. В нем представлен список доступных запросов, из которого можно выбрать и удалить нужный. Это помогает поддерживать порядок и актуальность запросов.

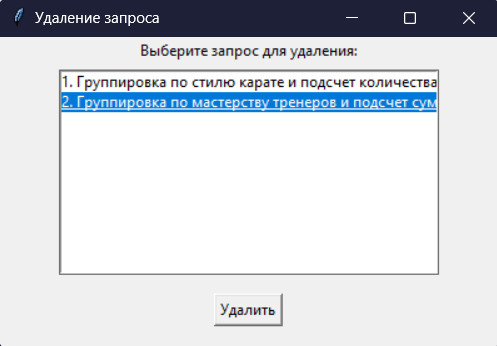


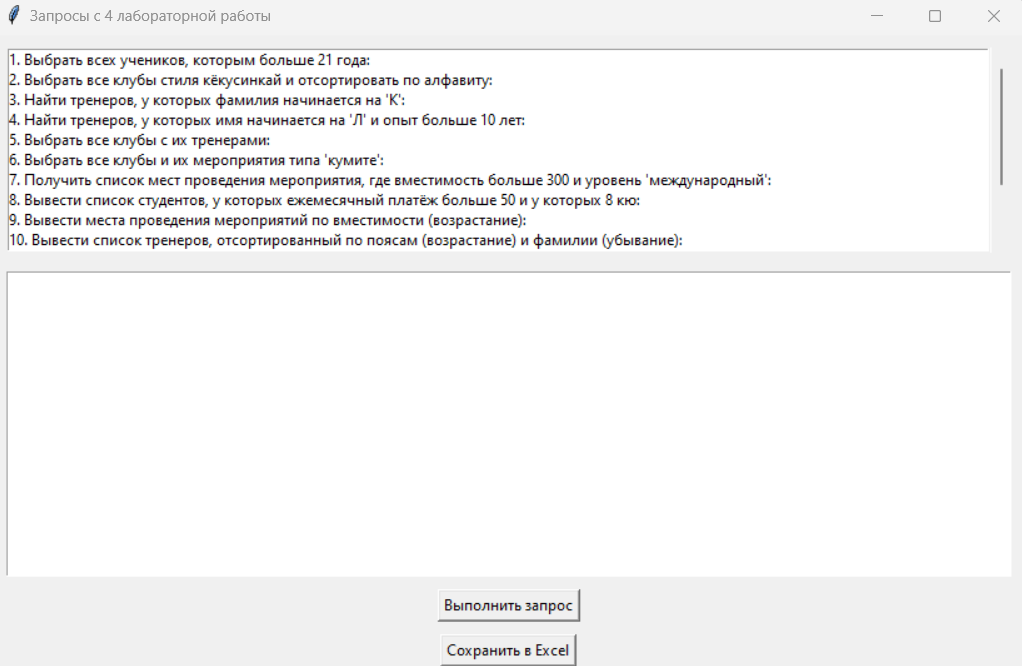
Рисунок 1.3 – Окно удаления запроса

## **1.4 Окно для выполнения запросов**

Окно для выполнения запросов предоставляет пользователю возможность запускать SQL-запросы и получать результаты их выполнения. В этом окне пользователи могут выбрать один из сохраненных запросов из списка, после чего нажать кнопку для выполнения.

Результаты запроса отображаются в текстовом поле под списком, что позволяет пользователю быстро просмотреть полученные данные. Это окно особенно полезно для анализа данных, так как оно позволяет выполнять как простые, так и более сложные запросы к базе данных, обеспечивая удобный способ получения информации и принятия решений на основе результатов. В случае возникновения ошибок при выполнении запроса, пользователю будет показано сообщение с описанием ошибки, что позволяет ему быстро понять, что пошло не так и внести необходимые корректировки. Данные можно экспортировать в exel.

Так же можно выполнить запросы из 4 и 5 лабораторных работ.



1.4 – Окно для выполнения запросов

## **1.5 Окно создания новой таблицы**

Окно создания новой таблицы используется для добавления новых таблиц в базу данных. Пользователь вводит название таблицы и поля, определяя структуру данных, которые будут храниться. После ввода данных таблица создается в базе.

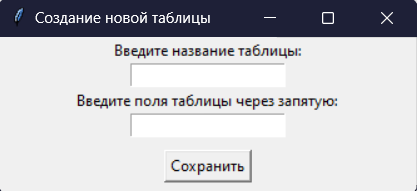
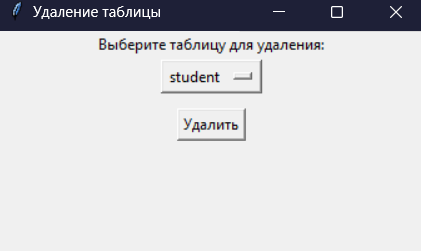


Рисунок 1.5 – Окно создания новой таблицы

## **1.6 Окно удаления таблицы**

Окно удаления таблицы позволяет пользователю выбрать и удалить существующую таблицу. В нем представлен выпадающий список всех доступных таблиц, что упрощает процесс удаления ненужных данных.



1.6 – Окно удаления таблицы

## **1.7 Окно добавления данных в таблицу**

Окно добавления данных в таблицу предназначено для ввода новых записей. Пользователь может выбрать таблицу и заполнить соответствующие поля данными, которые будут сохранены в базе данных, что позволяет легко обновлять информацию.

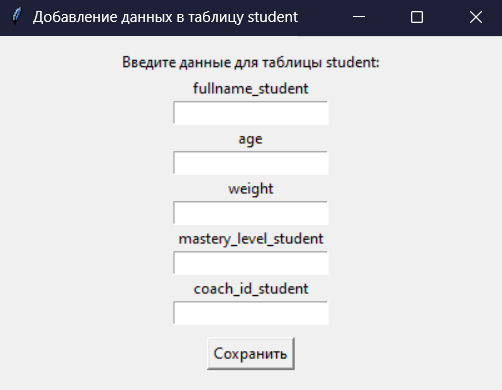


Рисунок 1.7 – Окно добавления данных в таблицу

## **1.8 Окно просмотра данных из таблицы**

Окно просмотра данных из таблицы позволяет пользователю видеть содержимое выбранной таблицы. Данные отображаются в текстовом формате, что удобно для быстрого анализа информации.

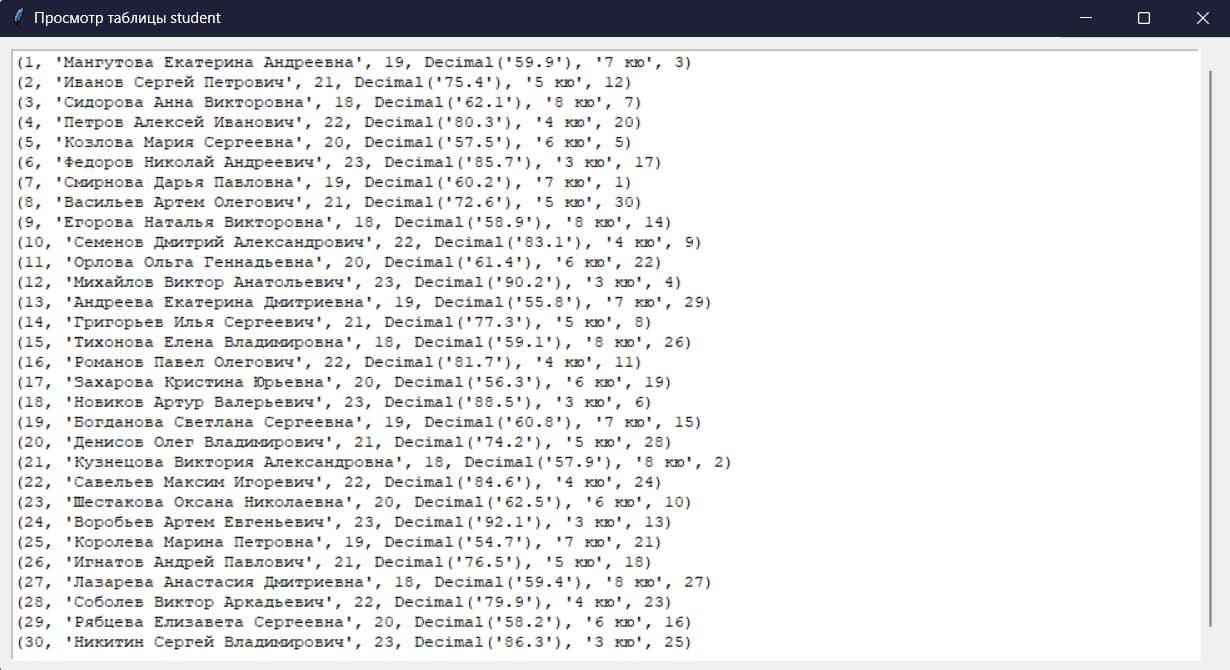
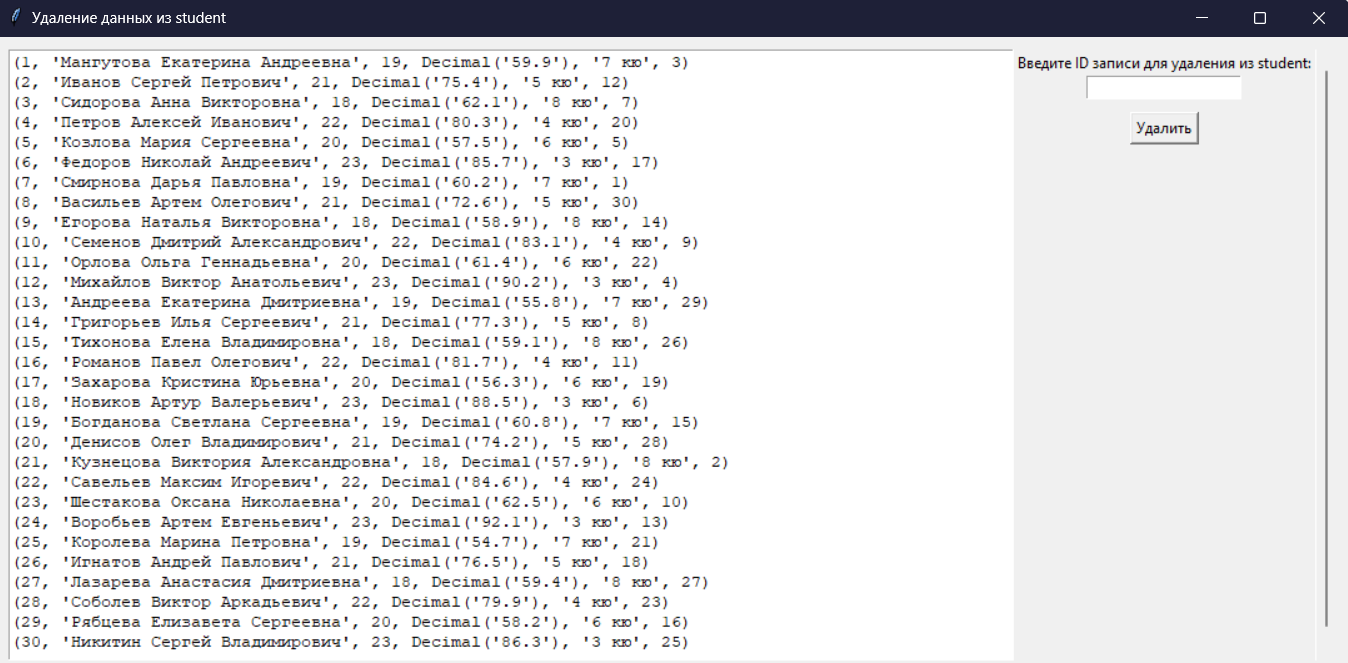


Рисунок 1.8 – Окно просмотра данных из таблицы

## **1.9 Окно удаления данных из таблицы**

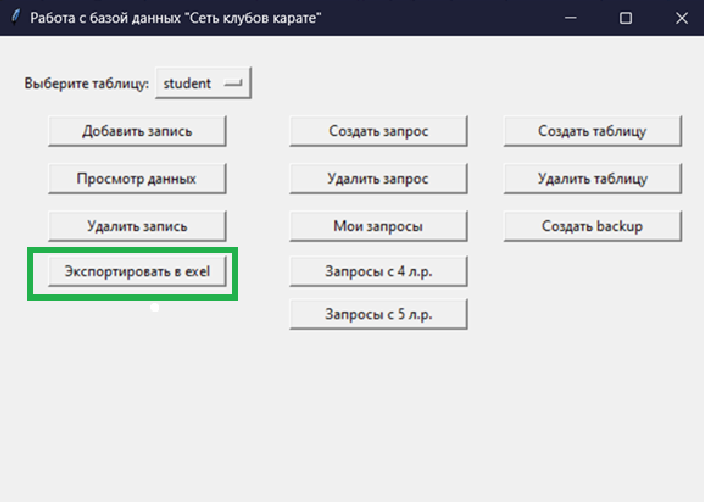
Окно удаления данных из таблицы дает возможность удалить конкретную запись по ее ID. Пользователь может просмотреть данные таблицы и ввести ID записи, которую он хочет удалить, что упрощает управление записями.



1.9 – Окно удаления данных из таблицы

## **1.10 Кнопка экспорта в Excel**

Кнопка экспорта в Excel позволяет пользователю экспортировать данные выбранной таблицы в файл формата Excel. Это полезно для дальнейшего анализа и работы с данными вне приложения.



1.10 – Кнопка экспорта в Excel

## **1.11 Кнопка создания резервной копии**

Кнопка создания резервной копии дает пользователю возможность создать бэкап базы данных. Пользователь может нажать кнопку для создания резервной копии, что важно для защиты данных и их сохранности на случай непредвиденных обстоятельств.

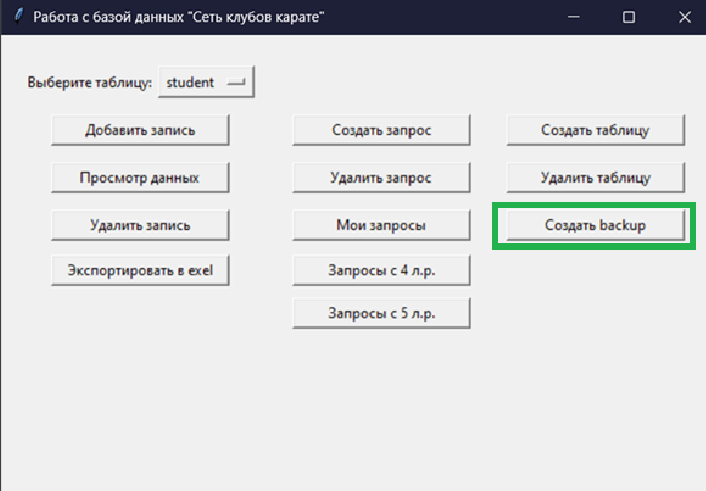


Рисунок 1.11 – Кнопка создания резервной копии

# **ЗАКЛЮЧЕНИЕ**

В ходе разработки программы было достигнуто несколько ключевых целей:

1. Упрощение управления данными: Пользователи могут легко добавлять, редактировать и удалять записи в базе данных, что значительно упрощает процесс работы с информацией.
2. Гибкость в работе с запросами: Возможность создания и удаления пользовательских SQL-запросов позволяет адаптировать программу под конкретные нужды клуба, обеспечивая необходимую гибкость.
3. Экспорт и резервное копирование: Реализованные функции экспорта данных в Excel и создания резервных копий базы данных способствуют надежному хранению и обработке информации, что является критически важным для любого бизнеса.
4. Доступность для пользователей: Графический интерфейс делает программу доступной для пользователей с различными уровнями знаний, что позволяет расширить её аудиторию.

Таким образом, программа «Работа с базой данных сеть клубов карате» представляет собой мощный инструмент для управления данными и упрощает множество процессов, связанных с хранением и анализом информации. В будущем возможна дальнейшая оптимизация и расширение функционала приложения, что позволит улучшить его пользовательский опыт и эффективность.

# **ПРИЛОЖЕНИЕ А** (обязательное)

Код программы

Файл main.py:

import psycopg2

from tkinter import Tk, Button, Label, Toplevel, Listbox, Scrollbar, Text, Frame, Entry, StringVar, OptionMenu, \

messagebox

from tkinter import END

import pandas as pd

import os

import sqlalchemy

import subprocess

import datetime

from datetime import datetime

TABLE\_FIELDS = {}

entry\_fields = {}

queries\_dict\_4\_lab = {}

queries\_dict\_5\_lab = {}

my\_queries = {}

# Функция удаления запроса

def delete\_quires():

top = Toplevel(root)

top.title("Удаление запроса")

top.geometry("400x250")

Label(top, text="Выберите запрос для удаления:").pack()

listbox = Listbox(top, width=50)

listbox.pack(pady=5)

for query in my\_queries.keys():

listbox.insert(END, query)

def delete\_selected():

selected\_index = listbox.curselection()

if not selected\_index:

messagebox.showwarning("Ошибка", "Выберите запрос для удаления!")

return

selected\_query = listbox.get(selected\_index)

# Удаляем из хэш-таблицы

del my\_queries[selected\_query]

# Перезаписываем файл без удаленного запроса

with open("my\_queries.txt", "w", encoding="utf-8") as file:

for name, sql in my\_queries.items():

file.write(f"{name}:\n{sql}\n\n")

messagebox.showinfo("Успех", "Запрос удален!")

top.destroy()

Button(top, text="Удалить", command=delete\_selected).pack(pady=10)

def load\_queries\_from\_file(filename):

queries\_dict = {}

with open(filename, "r", encoding="utf-8") as file:

lines = file.readlines()

query\_title = None

query\_text = []

for line in lines:

line = line.strip()

if line.isdigit() or line.endswith(":"): # Заголовок запроса (номер + описание)

if query\_title and query\_text:

queries\_dict[query\_title] = " ".join(query\_text) # Добавляем предыдущий запрос

query\_title = line

query\_text = []

else:

query\_text.append(line)

if query\_title and query\_text: # Добавляем последний запрос

queries\_dict[query\_title] = " ".join(query\_text)

return queries\_dict

def create\_quires():

create\_query\_window = Toplevel()

create\_query\_window.title("Создание нового запроса")

create\_query\_window.geometry("500x450") # Увеличенное окно

frame = Frame(create\_query\_window)

frame.pack(padx=10, pady=10, fill="both", expand=True)

Label(create\_query\_window, text="Введите описание запроса:").pack()

quires\_name\_entry = Entry(create\_query\_window, width=60)

quires\_name\_entry.pack(pady=5)

Label(create\_query\_window, text="Введите SQL запрос:").pack()

quires\_entry = Text(create\_query\_window, height=5, width=60)

quires\_entry.pack(pady=5)

output = Text(create\_query\_window, wrap="word", height=10, width=60)

output.pack(padx=10, pady=5, fill="both", expand=True)

def execute\_query\_from\_input():

sql\_query = quires\_entry.get("1.0", END).strip()

if sql\_query:

execute\_selected\_query(sql\_query=sql\_query, output=output)

def save\_quires():

global my\_queries

quires\_name = quires\_name\_entry.get().strip()

quires = quires\_entry.get("1.0", END).strip()

if quires\_name and quires:

with open("my\_queries.txt", "a", encoding="utf-8") as file:

file.write(f"{quires\_name}:\n{quires}\n\n")

my\_queries = load\_queries\_from\_file("my\_queries.txt")

create\_query\_window.destroy() # Закрытие окна после сохранения

Button(create\_query\_window, text="Выполнить запрос", command=execute\_query\_from\_input).pack(pady=5)

Button(create\_query\_window, text="Сохранить", command=save\_quires).pack(pady=10)

def create\_table():

top = Toplevel(root)

top.title("Создание новой таблицы")

Label(top, text="Введите название таблицы:").pack()

table\_name\_entry = Entry(top)

table\_name\_entry.pack()

Label(top, text="Введите поля таблицы через запятую:").pack()

fields\_entry = Entry(top)

fields\_entry.pack()

def save\_table():

table\_name = table\_name\_entry.get().strip()

fields = fields\_entry.get().strip()

if table\_name and fields:

field\_list = [f.strip() for f in fields.split(",")]

TABLE\_FIELDS[table\_name] = field\_list

field\_definitions = ", ".join([f"{field} TEXT" for field in field\_list])

create\_query = f"CREATE TABLE IF NOT EXISTS {table\_name} (id SERIAL PRIMARY KEY, {field\_definitions});"

try:

conn = connect\_to\_db()

cur = conn.cursor()

cur.execute(create\_query)

conn.commit()

cur.close()

conn.close()

with open("table\_fields.txt", "a", encoding="utf-8") as file:

file.write(f"{table\_name}: {fields}\n")

messagebox.showinfo("Успех", f"Таблица {table\_name} создана.")

load\_table\_fields()

update\_table\_list()

top.destroy()

except Exception as e:

print("всё ок")

messagebox.showerror("Ошибка", f"Ошибка создания таблицы: {e}")

else:

messagebox.showerror("Ошибка", "Название таблицы и поля не могут быть пустыми!")

Button(top, text="Сохранить", command=save\_table).pack(pady=10)

def delete\_table():

top = Toplevel(root)

top.title("Удаление таблицы")

Label(top, text="Выберите таблицу для удаления:").pack()

selected\_table = StringVar(top)

tables = list(TABLE\_FIELDS.keys())

if tables:

selected\_table.set(tables[0])

option\_menu = OptionMenu(top, selected\_table, \*tables)

option\_menu.pack()

else:

Label(top, text="Нет доступных таблиц").pack()

return

def confirm\_delete():

table = selected\_table.get()

if table:

try:

conn = connect\_to\_db()

cur = conn.cursor()

cur.execute(f"DROP TABLE IF EXISTS {table};")

conn.commit()

cur.close()

conn.close()

del TABLE\_FIELDS[table]

with open("table\_fields.txt", "r", encoding="utf-8") as file:

lines = file.readlines()

with open("table\_fields.txt", "w", encoding="utf-8") as file:

for line in lines:

if not line.startswith(f"{table}:"):

file.write(line)

messagebox.showinfo("Успех", f"Таблица {table} удалена.")

load\_table\_fields()

update\_table\_list()

top.destroy()

except Exception as e:

messagebox.showerror("Ошибка", f"Ошибка удаления таблицы: {e}")

Button(top, text="Удалить", command=confirm\_delete).pack(pady=10)

def update\_table\_list():

tables = list(TABLE\_FIELDS.keys())

selected\_table.set(tables[0] if tables else "")

menu = frame\_left.winfo\_children()[1] # Получаем `OptionMenu`, который находится на втором месте

menu["menu"].delete(0, "end")

for table in tables:

menu["menu"].add\_command(label=table, command=lambda value=table: selected\_table.set(value))

# Функция для загрузки данных из .txt файла

def load\_table\_fields(filename="table\_fields.txt"):

global TABLE\_FIELDS

TABLE\_FIELDS.clear() # Очищаем словарь перед загрузкой новых данных

try:

with open(filename, "r", encoding="utf-8") as file:

for line in file:

line = line.strip()

if line: # Пропускаем пустые строки

table, fields = line.split(":")

TABLE\_FIELDS[table.strip()] = [field.strip() for field in fields.split(",")]

print("Данные успешно загружены")

except FileNotFoundError:

print(f"Файл {filename} не найден!")

except Exception as e:

print(f"Ошибка при чтении файла: {e}")

# Функция подключения к БД

def connect\_to\_db():

try:

conn = psycopg2.connect(

dbname="kerete\_club",

user="postgres",

password="1111",

host="localhost",

port="5432"

)

return conn

except Exception as e:

print("Ошибка подключения:", e)

return None

# Функция для выполнения запроса к БД

def execute\_query(query, output):

try:

conn = connect\_to\_db()

cur = conn.cursor()

cur.execute(query)

results = cur.fetchall()

output.delete("1.0", "end") # Очистка предыдущего результата

for row in results:

output.insert("end", f"{row}\n")

cur.close()

conn.close()

except Exception as e:

output.delete("1.0", "end")

output.insert("end", f"Ошибка выполнения запроса: {e}\n")

# Функция для выполнения запроса к БД для выбранного элемента листа через хэш-таблицу

def execute\_selected\_query(queries\_list=None, queries\_dict=None, output=None, sql\_query=None):

if sql\_query:

# Если передан конкретный SQL-запрос, выполняем его

execute\_query(sql\_query, output)

elif queries\_list and queries\_dict:

# Если передан список и словарь, выполняем выбранный запрос из списка

selected\_index = queries\_list.curselection()

if selected\_index:

selected\_query = queries\_list.get(selected\_index[0])

sql\_query = queries\_dict.get(selected\_query, None)

if sql\_query:

execute\_query(sql\_query, output)

# Экран для запросов

def show\_queries\_window(window\_title, queries\_dict):

query\_window = Toplevel()

query\_window.title(window\_title)

frame = Frame(query\_window)

frame.pack(padx=10, pady=10, fill="both", expand=True)

queries\_list = Listbox(frame, height=10, width=100)

scrollbar = Scrollbar(frame)

queries\_list.config(yscrollcommand=scrollbar.set)

scrollbar.config(command=queries\_list.yview)

for query\_desc in queries\_dict.keys():

queries\_list.insert("end", query\_desc)

queries\_list.pack(side="left", fill="both", expand=True)

scrollbar.pack(side="right", fill="y")

output = Text(query\_window, wrap="word", height=15, width=100)

output.pack(padx=10, pady=5, fill="both", expand=True)

Button(query\_window, text="Выполнить запрос",

command=lambda: execute\_selected\_query(queries\_list, queries\_dict, output)).pack(pady=5)

Button(query\_window, text="Сохранить в Excel",

command=lambda: save\_to\_excel(output.get("1.0", END))).pack(pady=5)

def save\_to\_excel(data):

# Создаем папку excel, если ее нет

folder\_path = 'excel'

if not os.path.exists(folder\_path):

os.makedirs(folder\_path)

# Генерируем осмысленное имя файла

timestamp = datetime.now().strftime("%Y%m%d\_%H%M%S")

file\_name = f"results\_queries\_{timestamp}.xlsx"

file\_path = os.path.join(folder\_path, file\_name)

# Преобразуем текст в список значений, убирая скобки и кавычки

rows = []

for line in data.strip().split('\n'):

if line:

# Убираем скобки и одинарные кавычки, а затем разбиваем по запятой

cleaned\_line = line.replace("(", "").replace(")", "").replace("'", "").strip()

rows.append(cleaned\_line.split(','))

# Создаем DataFrame и записываем в Excel

df = pd.DataFrame(rows)

df.to\_excel(file\_path, index=False, header=False)

print(f"Данные сохранены в {file\_path}")

def create\_entry\_fields(root, selected\_table):

global entry\_fields

entry\_fields.clear()

if selected\_table not in TABLE\_FIELDS:

print("Неизвестная таблица")

return

top = Toplevel(root)

top.title(f"Добавление данных в таблицу {selected\_table}")

frame = Frame(top)

frame.pack(padx=10, pady=10)

Label(frame, text=f"Введите данные для таблицы {selected\_table}:").pack()

for field in TABLE\_FIELDS[selected\_table]:

Label(frame, text=field).pack()

entry = Entry(frame)

entry.pack()

entry\_fields[field] = entry

Button(frame, text="Сохранить", command=lambda: insert\_data(selected\_table)).pack(pady=10)

# Функция вставки данных в таблицу

def insert\_data(table):

conn = connect\_to\_db()

if not conn:

return

try:

cur = conn.cursor()

fields = TABLE\_FIELDS[table]

values = [entry\_fields[field].get() for field in fields]

placeholders = ', '.join(['%s'] \* len(values))

query = f"INSERT INTO {table} ({', '.join(fields)}) VALUES ({placeholders})"

cur.execute(query, values)

conn.commit()

print(f"Данные успешно добавлены в {table}")

cur.close()

conn.close()

except Exception as e:

print("Ошибка вставки данных:", e)

conn.rollback()

conn.close()

def view\_table\_data():

table = selected\_table.get()

if table not in TABLE\_FIELDS:

print("Неизвестная таблица")

return

top = Toplevel(root)

top.title(f"Просмотр таблицы {table}")

frame = Frame(top)

frame.pack(padx=10, pady=10, fill="both", expand=True)

output = Text(frame, wrap="word", height=15, width=100)

scrollbar = Scrollbar(frame, command=output.yview)

output.config(yscrollcommand=scrollbar.set)

scrollbar.pack(side="right", fill="y")

output.pack(side="left", fill="both", expand=True)

query = f"SELECT \* FROM {table};"

execute\_query(query, output)

def confirm\_delete(table, entry\_id):

record\_id = entry\_id.get()

if not record\_id.isdigit():

print("Ошибка: ID должен быть числом")

return

conn = connect\_to\_db()

if not conn:

return

try:

cur = conn.cursor()

query = f"DELETE FROM {table} WHERE id = %s"

cur.execute(query, (record\_id,))

conn.commit()

print(f"Запись с ID {record\_id} успешно удалена из {table}")

cur.close()

conn.close()

except Exception as e:

print("Ошибка удаления:", e)

conn.rollback()

conn.close()

def delete\_data():

table = selected\_table.get()

if table not in TABLE\_FIELDS:

print("Неизвестная таблица")

return

delete\_window = Toplevel(root)

delete\_window.title(f"Удаление данных из {table}")

frame = Frame(delete\_window)

frame.pack(padx=10, pady=10)

frame.pack(padx=10, pady=10, fill="both", expand=True)

output = Text(frame, wrap="word", height=15, width=100)

scrollbar = Scrollbar(frame, command=output.yview)

output.config(yscrollcommand=scrollbar.set)

scrollbar.pack(side="right", fill="y")

output.pack(side="left", fill="both", expand=True)

query = f"SELECT \* FROM {table};"

execute\_query(query, output)

Label(frame, text=f"Введите ID записи для удаления из {table}:").pack()

entry\_id = Entry(frame)

entry\_id.pack()

Button(frame, text="Удалить", command=lambda: confirm\_delete(table, entry\_id)).pack(pady=10)

def connect\_to\_db\_exel():

# Подключение к базе данных PostgreSQL через SQLAlchemy

db\_url = "postgresql://postgres:1111@localhost:5432/kerete\_club"

return sqlalchemy.create\_engine(db\_url)

def export\_to\_excel():

table\_name = selected\_table.get()

if table\_name not in TABLE\_FIELDS:

print("Неизвестная таблица")

return

try:

engine = connect\_to\_db\_exel() # SQLAlchemy engine вместо обычного подключения

# Проверяем, существует ли таблица

with engine.connect() as connection:

result = connection.execute(sqlalchemy.text("SELECT to\_regclass(:table\_name)"), {"table\_name": table\_name})

if result.fetchone()[0] is None:

messagebox.showerror("Ошибка", f"Таблица '{table\_name}' не найдена в базе данных.")

return

# Запрос данных

query = f"SELECT \* FROM {table\_name}"

df = pd.read\_sql(query, engine) # Используем SQLAlchemy engine

# Создание папки 'excel', если её нет

folder\_path = os.path.join(os.getcwd(), "excel")

os.makedirs(folder\_path, exist\_ok=True)

# Путь к файлу

file\_path = os.path.join(folder\_path, f"{table\_name}.xlsx")

# Экспорт в Excel

df.to\_excel(file\_path, index=False, engine='openpyxl')

messagebox.showinfo("Успех", f"Данные успешно экспортированы в {file\_path}!")

except Exception as e:

messagebox.showerror("Ошибка", f"Ошибка при экспорте: {e}")

def backup\_database():

# Установите пароль для PostgreSQL

os.environ["PGPASSWORD"] = "1111"

# Генерация имени файла для бэкапа с текущей датой и временем

timestamp = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")

backup\_filename = f"backup\_{timestamp}.sql"

# Папка для бэкапов

backup\_dir = "backups"

# Создаем папку, если её нет

if not os.path.exists(backup\_dir):

os.makedirs(backup\_dir)

# Полный путь к файлу бэкапа

backup\_path = os.path.join(backup\_dir, backup\_filename)

# Команда для выполнения

command = [

"C:/Program Files/PostgreSQL/17/bin/pg\_dump.exe",

"-U", "postgres",

"-d", "kerete\_club",

"-h", "127.0.0.1",

"-p", "5432",

"-F", "p",

"-f", backup\_path

]

try:

# Выполнить команду

subprocess.run(command, check=True)

print(f"Бекап выполнен успешно. Файл сохранён как {backup\_path}.")

except subprocess.CalledProcessError as e:

print(f"Ошибка при выполнении бекапа: {e}")

# Загружаем данные

load\_table\_fields()

queries\_dict\_4\_lab = load\_queries\_from\_file("queries\_4lab.txt")

queries\_dict\_5\_lab = load\_queries\_from\_file("queries\_5lab.txt")

my\_queries = load\_queries\_from\_file("my\_queries.txt")

# Основное окно приложения

root = Tk()

root.title("Работа с базой данных \"Сеть клубов карате\"")

root.geometry("600x400")

# Основной фрейм

frame\_main = Frame(root)

frame\_main.pack(padx=10, pady=10, fill="both", expand=True)

# Создаём три колонки

frame\_left = Frame(frame\_main)

frame\_left.grid(row=0, column=0, padx=10, pady=10, sticky="n")

frame\_center = Frame(frame\_main)

frame\_center.grid(row=0, column=1, padx=20, pady=10, sticky="n")

frame\_right = Frame(frame\_main)

frame\_right.grid(row=0, column=2, padx=10, pady=10, sticky="n")

# Выравниваем строки

for i in range(4):

frame\_left.grid\_rowconfigure(i, minsize=40)

frame\_center.grid\_rowconfigure(i, minsize=40)

frame\_right.grid\_rowconfigure(i, minsize=40)

# Левая панель (Работа с таблицами)

Label(frame\_left, text="Выберите таблицу:").grid(row=0, column=0, sticky="w")

selected\_table = StringVar(root)

tables = list(TABLE\_FIELDS.keys())

selected\_table.set(tables[0])

OptionMenu(frame\_left, selected\_table, \*tables).grid(row=0, column=1, sticky="w")

Button(frame\_left, text="Добавить запись", command=lambda: create\_entry\_fields(root, selected\_table.get()), width=20).grid(row=1, column=0, columnspan=2, pady=5)

Button(frame\_left, text="Просмотр данных", command=view\_table\_data, width=20).grid(row=2, column=0, columnspan=2, pady=5)

Button(frame\_left, text="Удалить запись", command=delete\_data, width=20).grid(row=3, column=0, columnspan=2, pady=5)

Button(frame\_left, text="Экcпортировать в exel", command=export\_to\_excel, width=20).grid(row=4, column=0, columnspan=2, pady=5)

# Центральная панель (Запросы)

Button(frame\_center, text="Создать запрос", command=create\_quires, width=20).grid(row=1, column=0, pady=5)

Button(frame\_center, text="Удалить запрос", command=delete\_quires, width=20).grid(row=2, column=0, pady=5)

Button(frame\_center, text="Мои запросы", command=lambda: show\_queries\_window("Мои запросы", my\_queries), width=20).grid(row=3, column=0, pady=5)

Button(frame\_center, text="Запросы с 4 л.р.", command=lambda: show\_queries\_window("Запросы с 4 лабораторной работы", queries\_dict\_4\_lab), width=20).grid(row=4, column=0, pady=5)

Button(frame\_center, text="Запросы с 5 л.р.", command=lambda: show\_queries\_window("Запросы с 5 лабораторной работы", queries\_dict\_5\_lab), width=20).grid(row=5, column=0, pady=5)

# Правая панель (Создание/удаление таблиц)

Button(frame\_right, text="Создать таблицу", command=create\_table, width=20).grid(row=1, column=0, pady=5)

Button(frame\_right, text="Удалить таблицу", command=delete\_table, width=20).grid(row=2, column=0, pady=5)

Button(frame\_right, text="Создать backup", command=backup\_database, width=20).grid(row=3, column=0, pady=5)

root.mainloop()