

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
1. ИНТЕРФЕЙС ПРОГРАММЫ.....	4
1.1 Главное окно приложения.....	4
1.2 Окно создания нового запроса.....	5
1.3 Окно удаления запроса .....	6
1.4 Окно для выполнения запросов.....	7
1.5 Окно создания новой таблицы.....	7
1.6 Окно удаления таблицы .....	8
1.7 Окно добавления данных в таблицу .....	8
1.8 Окно просмотра данных из таблицы.....	9
1.9 Окно удаления данных из таблицы.....	10
1.10 Кнопка экспорта в Excel.....	10
1.11 Кнопка создания резервной копии .....	11
ЗАКЛЮЧЕНИЕ .....	12
ПРИЛОЖЕНИЕ А .....	13

## 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.

# 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.

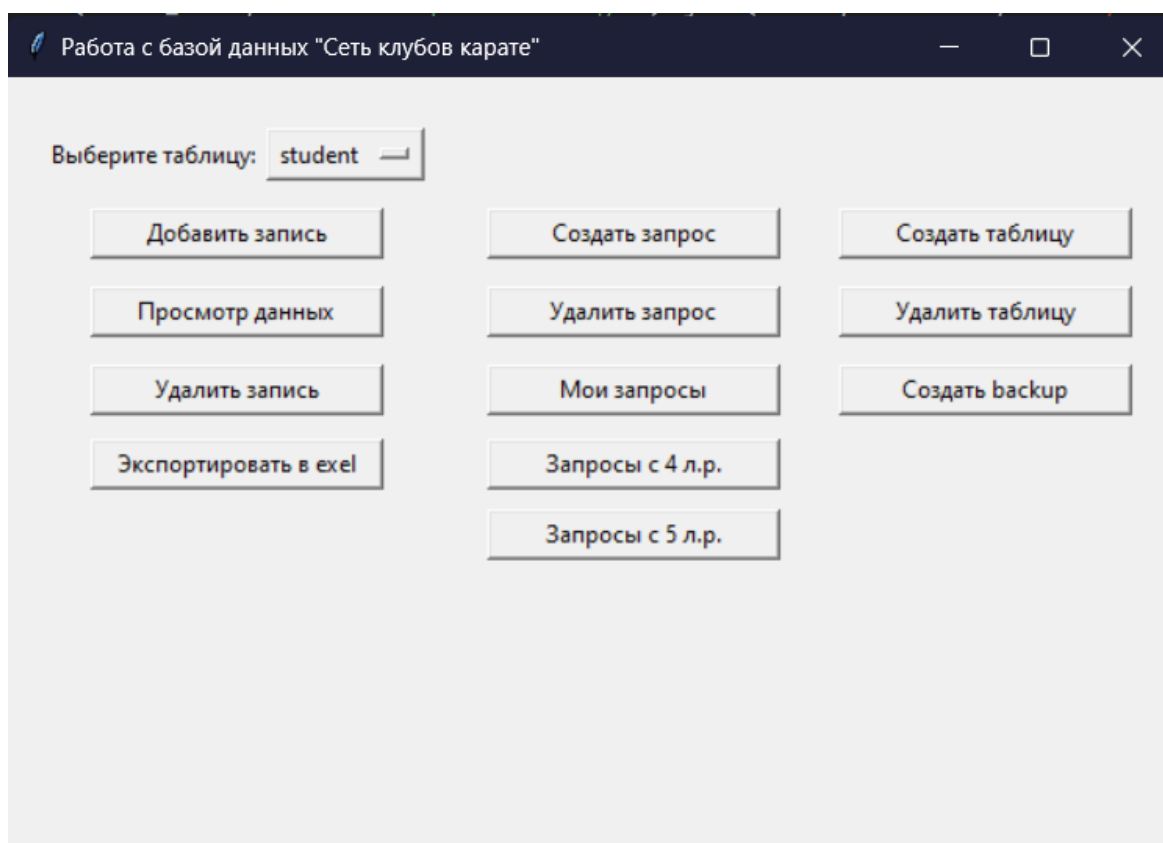


Рисунок 1.1 – Главное окно приложения

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 л.р." и "Запросы с 5 л.р." 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.

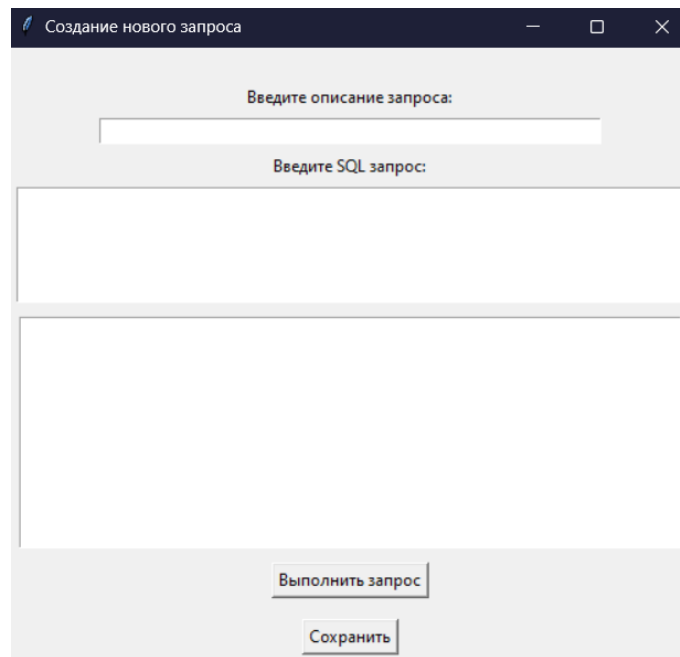


Рисунок 1.2 – Таблица Окно создания нового запроса

### 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.

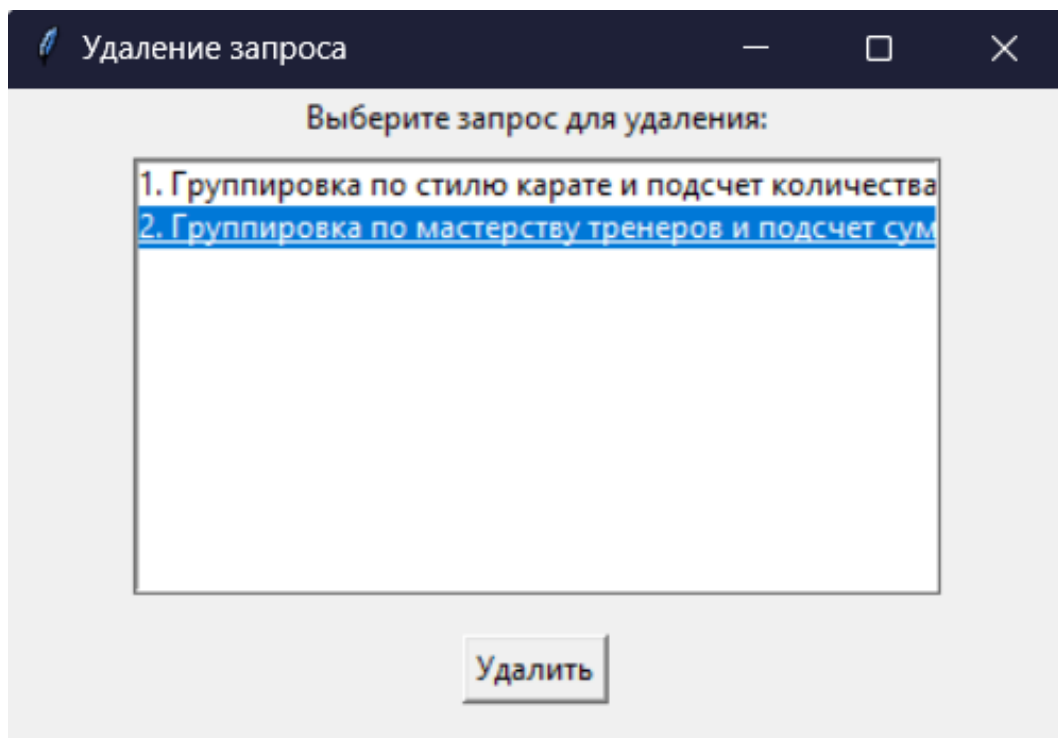


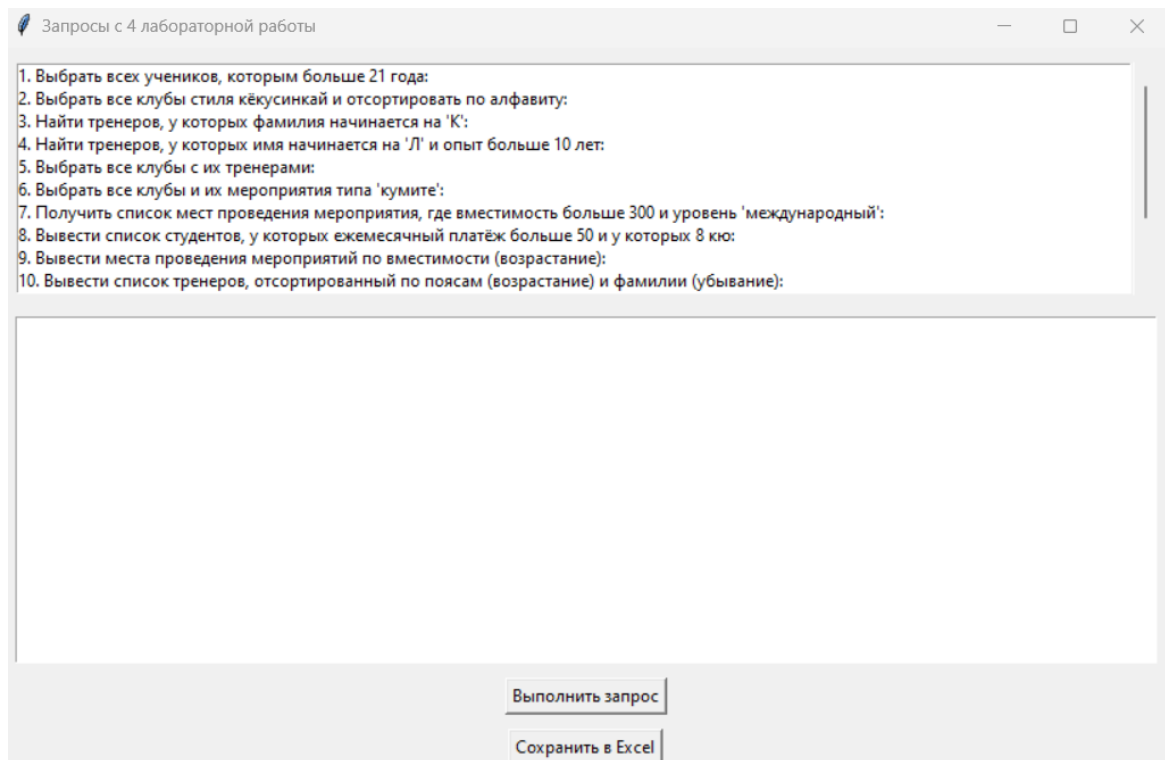
Рисунок 1.3 – Окно удаления запроса

## 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 excel.

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.

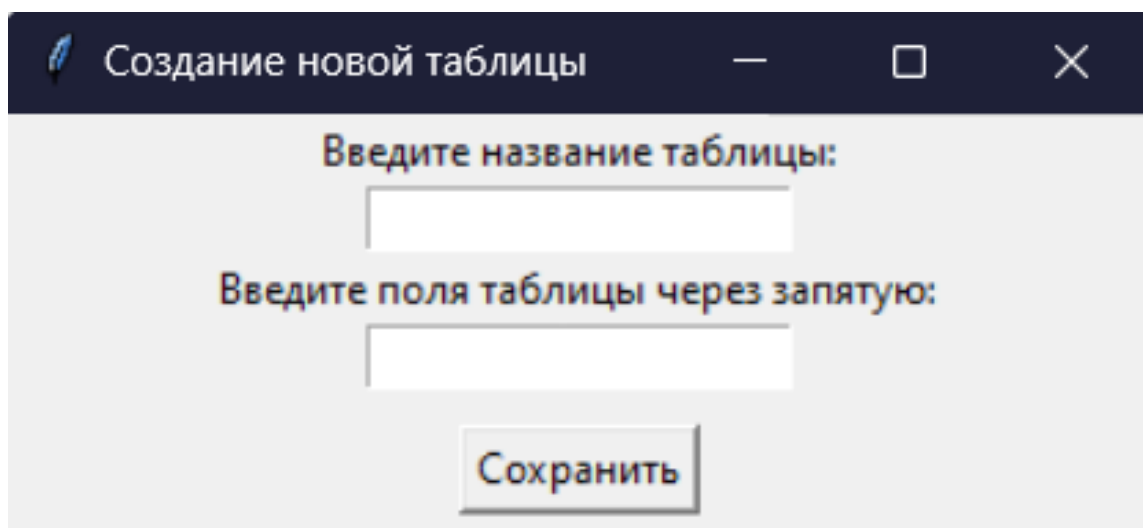
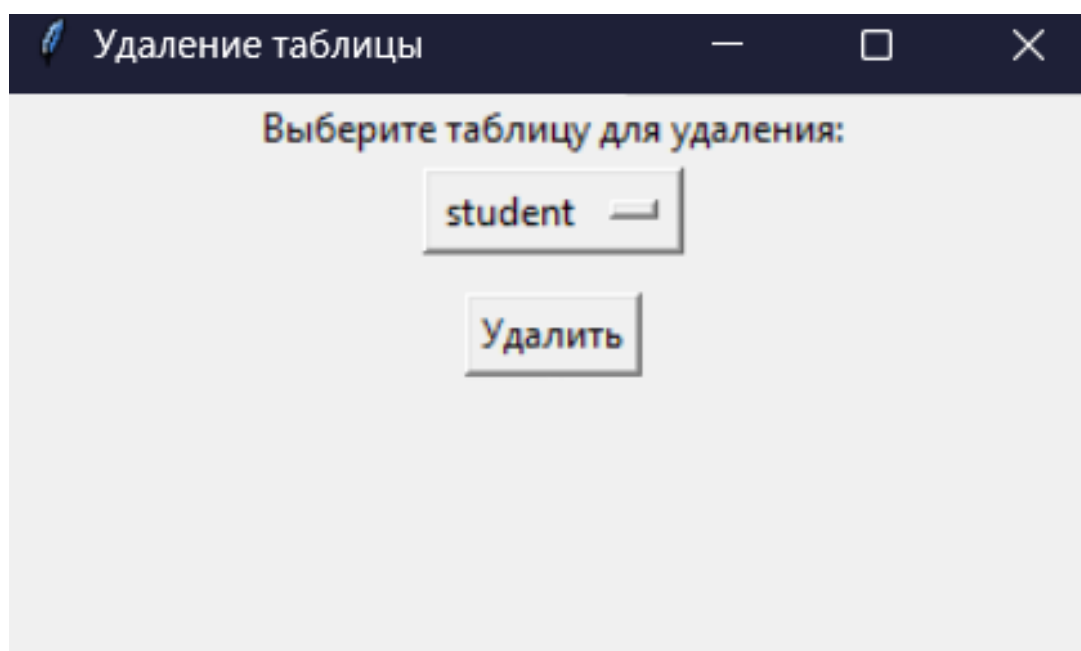


Рисунок 1.5 – Окно создания новой таблицы

### 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.



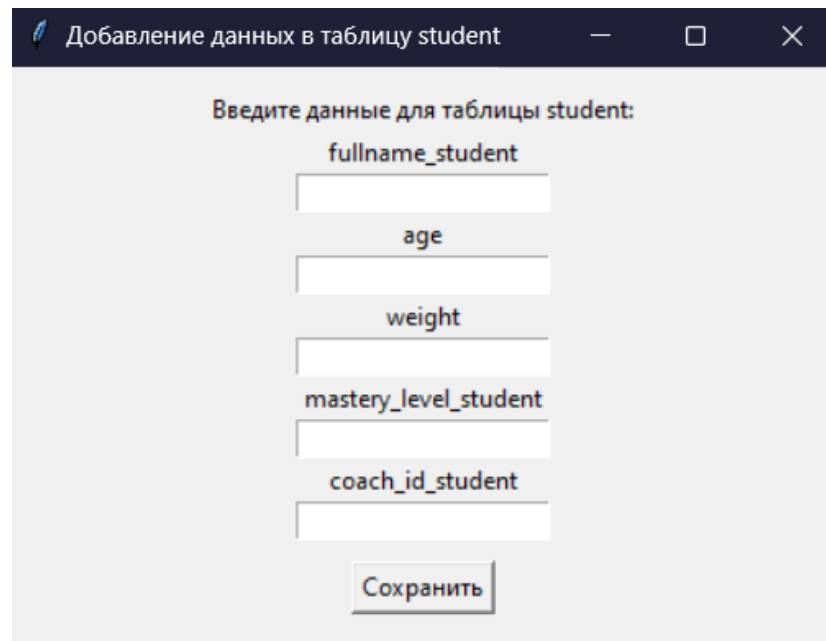
1.6 – Окно удаления таблицы

### 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.



Добавление данных в таблицу student

Введите данные для таблицы student:

fullname\_student

age

weight

mastery\_level\_student

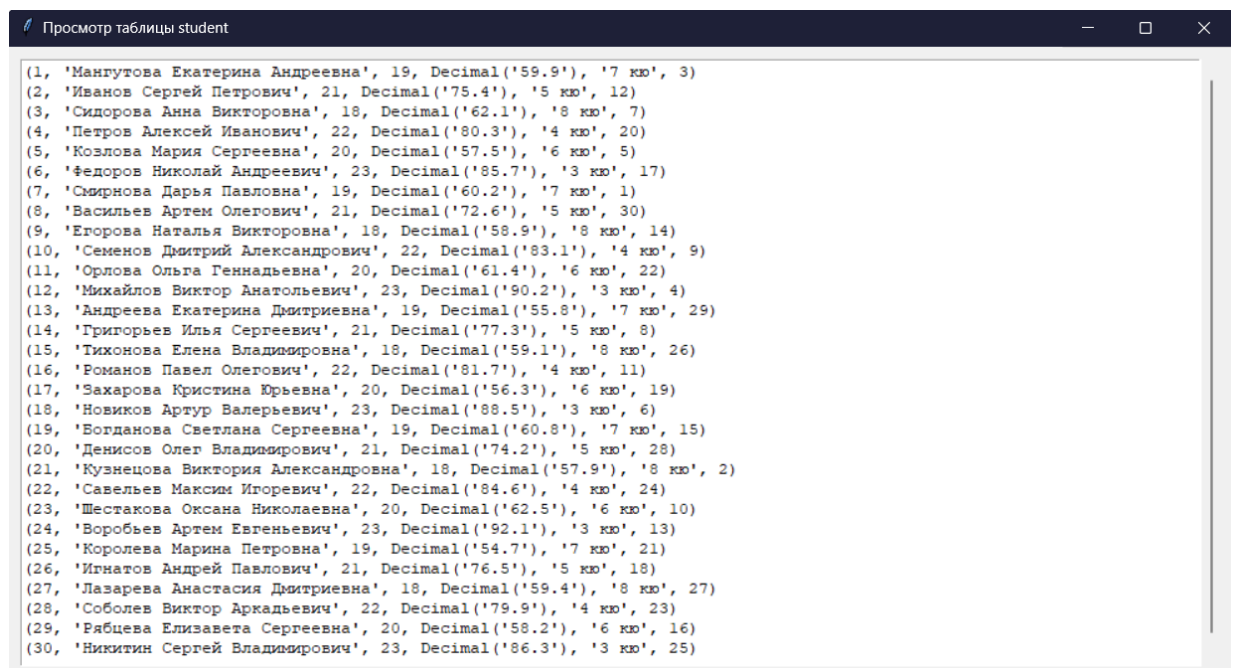
coach\_id\_student

Сохранить

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.



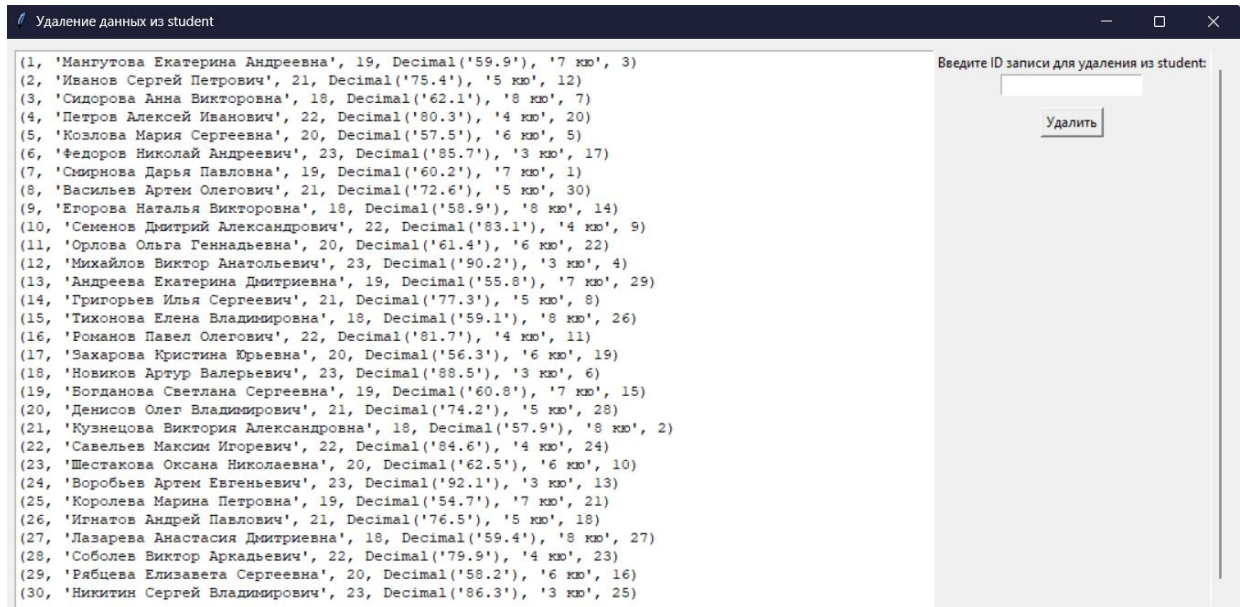
Просмотр таблицы student

(1, 'Мангутова Екатерина Андреевна', 19, Decimal('59.9'), '7 кю', 3)
(2, 'Иванов Сергей Петрович', 21, Decimal('75.4'), '5 кю', 12)
(3, 'Сидорова Анна Викторовна', 18, Decimal('62.1'), '8 кю', 7)
(4, 'Петров Алексей Иванович', 22, Decimal('80.3'), '4 кю', 20)
(5, 'Козлова Мария Сергеевна', 20, Decimal('57.5'), '6 кю', 5)
(6, 'Федоров Николай Андреевич', 23, Decimal('85.7'), '3 кю', 17)
(7, 'Смирнова Дарья Павловна', 19, Decimal('60.2'), '7 кю', 1)
(8, 'Васильев Артем Олегович', 21, Decimal('72.6'), '5 кю', 30)
(9, 'Егорова Наталья Викторовна', 18, Decimal('58.9'), '8 кю', 14)
(10, 'Семенов Дмитрий Александрович', 22, Decimal('83.1'), '4 кю', 9)
(11, 'Орлова Ольга Геннадьевна', 20, Decimal('61.4'), '6 кю', 22)
(12, 'Михайлов Виктор Анатольевич', 23, Decimal('90.2'), '3 кю', 4)
(13, 'Андреева Екатерина Дмитриевна', 19, Decimal('55.8'), '7 кю', 29)
(14, 'Григорьев Илья Сергеевич', 21, Decimal('77.3'), '5 кю', 8)
(15, 'Тихонова Елена Владимировна', 18, Decimal('59.1'), '8 кю', 26)
(16, 'Романов Павел Олегович', 22, Decimal('81.7'), '4 кю', 11)
(17, 'Захарова Кристина Юрьевна', 20, Decimal('56.3'), '6 кю', 19)
(18, 'Новиков Артур Валерьевич', 23, Decimal('88.5'), '3 кю', 6)
(19, 'Богданова Светлана Сергеевна', 19, Decimal('60.8'), '7 кю', 15)
(20, 'Денисов Олег Владимирович', 21, Decimal('74.2'), '5 кю', 28)
(21, 'Кузнецова Виктория Александровна', 18, Decimal('57.9'), '8 кю', 2)
(22, 'Савельев Максим Игоревич', 22, Decimal('84.6'), '4 кю', 24)
(23, 'Шестакова Оксана Николаевна', 20, Decimal('62.5'), '6 кю', 10)
(24, 'Воробьев Артем Евгеньевич', 23, Decimal('92.1'), '3 кю', 13)
(25, 'Королева Марина Петровна', 19, Decimal('54.7'), '7 кю', 21)
(26, 'Игнатов Андрей Павлович', 21, Decimal('76.5'), '5 кю', 18)
(27, 'Лазарева Анастасия Дмитриевна', 18, Decimal('59.4'), '8 кю', 27)
(28, 'Соболев Виктор Аркадьевич', 22, Decimal('79.9'), '4 кю', 23)
(29, 'Рябцева Елизавета Сергеевна', 20, Decimal('58.2'), '6 кю', 16)
(30, 'Никитин Сергей Владимирович', 23, Decimal('86.3'), '3 кю', 25)

Рисунок 1.8 – Окно просмотра данных из таблицы

## 1.9 Окно удаления данных из таблицы

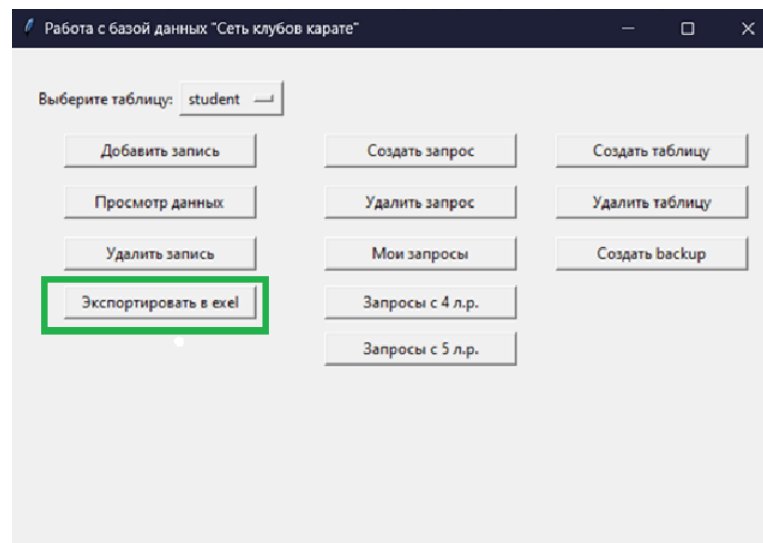
Окно удаления данных из таблицы дает возможность удалить конкретную запись по ее ID. Пользователь может просмотреть данные таблицы и ввести ID записи, которую он хочет удалить, что упрощает управление записями.



## 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.

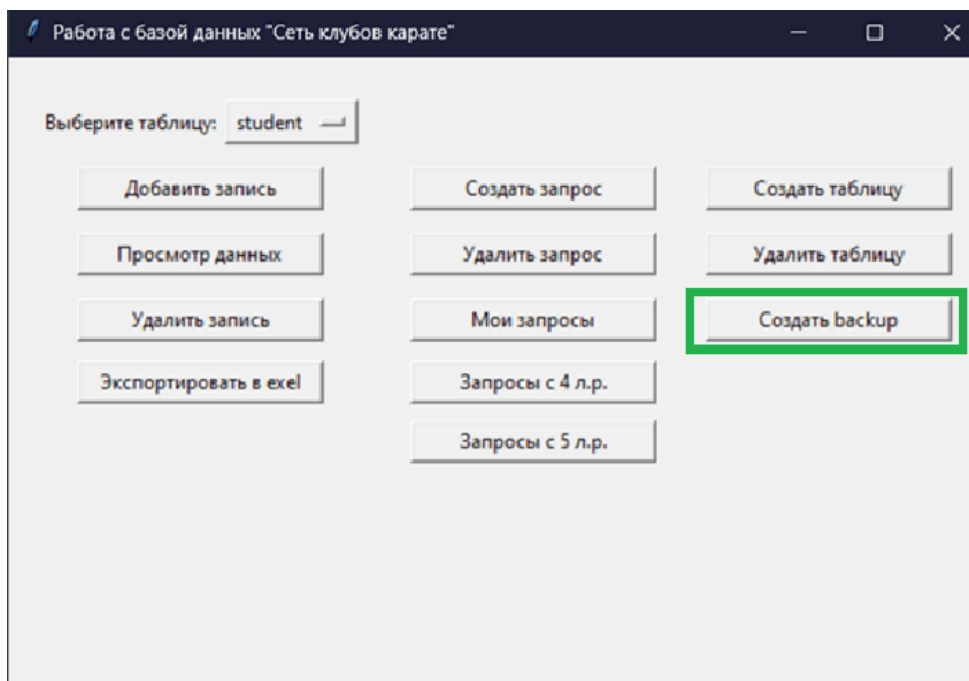


Рисунок 1.11 – Кнопка создания резервной копии

## 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

Файл main.py:

```
import psycpg2
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() # Заккрытие 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.wininfo_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="Экспортировать 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()
```