Министерство образования Республики Беларусь  
Учреждение образования  
«Брестский государственный технический университет»  
Кафедра ИИТ

Лабораторная работа №5  
По дисциплине: «Создание приложения для работы с БД и организация пользовательского интерфейса.»

Выполнил:  
студент 3 курса  
группы ИИ-23  
Романюк А. П.

Проверил:  
Савонюк В. А.

Брест 2024

**Цель работы**: получить навыки разработки приложений БД и организации пользовательского интерфейса.

**Задание**

Изучить материал, приведенный в “К лабораторной работе 5.doc”.

На основании логической модели (в соответствии с вариантом Лабораторной работы №1) создать приложение и организовать пользовательский интерфейс, используя соответствующие элементы и методы.

import tkinter as tk

from tkinter import ttk

import sqlite3

from tkinter import messagebox

class DatabaseApp:

def \_\_init\_\_(self, master, connection\_params):

self.master = master

self.connection\_params = connection\_params

self.master.title("СМОТРИ БД")

self.notebook = ttk.Notebook(master)

self.notebook.pack(expand=True, fill='both')

self.conn = sqlite3.connect(\*\*connection\_params)

self.cursor = self.conn.cursor()

self.table\_names = self.get\_table\_names()

for table\_name in self.table\_names:

frame = tk.Frame(self.notebook)

self.notebook.add(frame, text=table\_name)

self.create\_table\_view(frame, table\_name)

def get\_table\_names(self):

self.cursor.execute("SELECT name FROM sqlite\_master WHERE type='table';")

table\_names = [row[0] for row in self.cursor.fetchall()]

return table\_names

def create\_table\_view(self, frame, table\_name):

self.cursor.execute(f"PRAGMA table\_info({table\_name});")

columns = [row[1] for row in self.cursor.fetchall()]

tree = ttk.Treeview(frame, columns=columns, show='headings', selectmode='browse')

tree.pack(expand=True, fill='both')

for col in columns:

tree.heading(col, text=col)

tree.column(col, width=100, anchor='center')

self.populate\_treeview(tree, table\_name)

add\_button = tk.Button(frame, text="Добавить", command=lambda: self.add\_row(tree, table\_name))

add\_button.pack(side=tk.LEFT, padx=10)

delete\_button = tk.Button(frame, text="Удалить", command=lambda: self.delete\_row(tree, table\_name))

delete\_button.pack(side=tk.LEFT, padx=10)

edit\_button = tk.Button(frame, text="Изменить", command=lambda: self.edit\_row(tree, table\_name))

edit\_button.pack(side=tk.LEFT, padx=10)

refresh\_button = tk.Button(frame, text="Обновить", command=lambda: self.populate\_treeview(tree, table\_name))

refresh\_button.pack(side=tk.LEFT, padx=10)

def populate\_treeview(self, tree, table\_name):

self.cursor.execute(f"SELECT \* FROM {table\_name};")

data = self.cursor.fetchall()

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

for row in data:

tree.insert('', 'end', values=row)

def add\_row(self, tree, table\_name):

self.cursor.execute(f"PRAGMA table\_info({table\_name});")

columns = [row[1] for row in self.cursor.fetchall()]

add\_dialog = tk.Toplevel(self.master)

add\_dialog.title("Добавить строку")

entry\_widgets = []

for col in columns:

label = tk.Label(add\_dialog, text=col)

label.grid(row=columns.index(col), column=0, padx=10, pady=5, sticky='e')

entry = tk.Entry(add\_dialog)

entry.grid(row=columns.index(col), column=1, padx=10, pady=5, sticky='w')

entry\_widgets.append(entry)

def insert\_row():

values = [entry.get() for entry in entry\_widgets]

placeholders = ', '.join(['?' for \_ in values])

query = f"INSERT INTO {table\_name} VALUES ({placeholders});"

self.cursor.execute(query, values)

self.conn.commit()

self.populate\_treeview(tree, table\_name)

add\_dialog.destroy()

submit\_button = tk.Button(add\_dialog, text="Подтвердить", command=insert\_row)

submit\_button.grid(row=len(columns), columnspan=2, pady=10)

def delete\_row(self, tree, table\_name):

selected\_item = tree.selection()

if not selected\_item:

messagebox.showwarning("Предупреждение", "Пожалуйста, выберите строку для удаления.")

return

confirm = messagebox.askyesno("Подтверждение", "Вы уверены, что хотите удалить эту строку?")

if not confirm:

return

values = tree.item(selected\_item)['values']

where\_clause = ' AND '.join([f"{column} = ?" for column in tree['columns']])

query = f"DELETE FROM {table\_name} WHERE {where\_clause};"

self.cursor.execute(query, values)

self.conn.commit()

self.populate\_treeview(tree, table\_name)

def edit\_row(self, tree, table\_name):

selected\_item = tree.selection()

if not selected\_item:

messagebox.showwarning("Предупреждение", "Пожалуйста, выберите строку для изменения.")

return

values = tree.item(selected\_item)['values']

self.cursor.execute(f"PRAGMA table\_info({table\_name});")

columns = [row[1] for row in self.cursor.fetchall()]

edit\_dialog = tk.Toplevel(self.master)

edit\_dialog.title("Изменить строку")

entry\_widgets = []

for col, value in zip(columns, values):

label = tk.Label(edit\_dialog, text=col)

label.grid(row=columns.index(col), column=0, padx=10, pady=5, sticky='e')

entry = tk.Entry(edit\_dialog)

entry.insert(0, value)

entry.grid(row=columns.index(col), column=1, padx=10, pady=5, sticky='w')

entry\_widgets.append(entry)

def update\_row():

new\_values = [entry.get() for entry in entry\_widgets]

set\_clause = ', '.join([f"{column} = ?" for column in columns])

where\_clause = ' AND '.join([f"{column} = ?" for column in columns])

query = f"UPDATE {table\_name} SET {set\_clause} WHERE {where\_clause};"

self.cursor.execute(query, new\_values + values)

self.conn.commit()

self.populate\_treeview(tree, table\_name)

edit\_dialog.destroy()

submit\_button = tk.Button(edit\_dialog, text="Подтвердить", command=update\_row)

submit\_button.grid(row=len(columns), columnspan=2, pady=10)

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

connection\_params = {"database": "mydb.sqlite3"}

try:

root = tk.Tk()

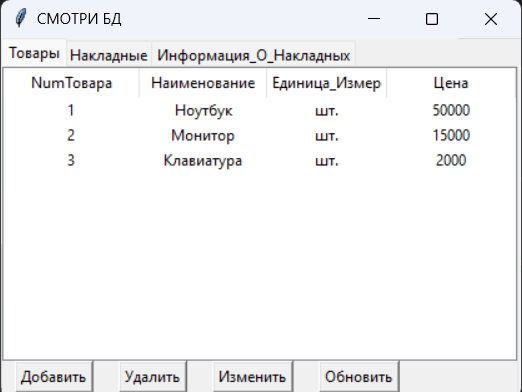
app = DatabaseApp(root, connection\_params)

root.mainloop()

except sqlite3.Error as err:

print(f"Error: {err}")

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**Вывод**: в ходе лабораторной работы я создал приложение с графическим интерфейсом для управления базой данных.