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-21  
Литвинюк Т. В.

Проверил:  
Савонюк В. А.

Брест 2023

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

**Задание**

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

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

**Код программы:**

import tkinter as tk

from tkinter import ttk

import pyodbc

from tkinter import messagebox

from prettytable import PrettyTable

DRIVER = "{SQL Server}"

SERVER = "VICTUS\SQLEXPRESS"

DATABASE = 'lab1'

USERNAME = 'Victus\Victus'

PASSWORD = ''

connectionString = f'DRIVER={DRIVER};SERVER={SERVER};DATABASE={DATABASE};UID={USERNAME};PWD={PASSWORD}'

class DatabaseApp:

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

self.root = root

self.root.title("Database Viewer")

# Создаем меню

self.menu\_bar = tk.Menu(root)

root.config(menu=self.menu\_bar)

# Меню "Файл"

self.file\_menu = tk.Menu(self.menu\_bar, tearoff=0)

self.menu\_bar.add\_cascade(label="Файл", menu=self.file\_menu)

self.file\_menu.add\_command(label="Выход", command=root.destroy)

# Фрейм для отображения таблицы

self.table\_frame = ttk.Frame(root)

self.table\_frame.pack(pady=10)

# Выпадающий список для выбора таблицы

self.table\_var = tk.StringVar()

self.table\_var.set("Выберите таблицу")

self.table\_dropdown = ttk.Combobox(self.table\_frame, textvariable=self.table\_var, state="readonly")

self.table\_dropdown.grid(row=0, column=0, padx=10, pady=10)

# Кнопка для обновления данных

self.refresh\_button = ttk.Button(self.table\_frame, text="Показать", command=self.display\_table)

self.refresh\_button.grid(row=0, column=1, padx=10, pady=10)

# Подключение к базе данных

try:

# self.connection = pyodbc.connect(connectionString)

self.connection = pyodbc.connect('Driver={SQL Server};'

'Server=VICTUS\SQLEXPRESS;'

'Database=lab1;'

'Trusted\_Connection=yes;')

self.cursor = self.connection.cursor()

self.populate\_table\_dropdown()

except pyodbc.Error as e:

messagebox.showerror("Ошибка подключения", f"Ошибка: {str(e)}")

root.destroy()

# Кнопка для добавления данных

self.add\_button = ttk.Button(root, text="Добавить", command=self.add\_data)

self.add\_button.pack()

self.del\_button = ttk.Button(root, text="Удалить", command=self.del\_data)

self.del\_button.pack()

self.upd\_button = ttk.Button(root, text="Изменить", command=self.upd\_data)

self.upd\_button.pack()

def add\_data(self):

def add\_data\_func():

values = ", ".join([f"'{var.get()}'" for var in entries])

query = f"INSERT INTO {selected\_table} VALUES ({values})"

print(query)

try:

self.cursor.execute(query)

self.connection.commit()

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

except pyodbc.Error as e:

messagebox.showerror("Ошибка запроса", f"Ошибка: {str(e)}")

win.destroy()

# Добавление данных в выбранную таблицу

selected\_table = self.table\_var.get()

if selected\_table != "Выберите таблицу":

columns = self.get\_columns(selected\_table)

win = tk.Tk()

win.title("Добавление данных")

# Создание текстовых полей для ввода данных

entries = []

for i, label in enumerate(columns):

ttk.Label(win, text=label).grid(row=i, column=0, padx=10, pady=5)

entries.append(ttk.Entry(win))

entries[-1].grid(row=i, column=1, padx=10, pady=5)

tk.Button(win, text="Добавить", command=add\_data\_func).grid(row=5, column=0)

def del\_data(self):

def del\_data\_func(number):

query = f"DELETE FROM {selected\_table} WHERE [{columns[0]}] = {number}"

print(query)

try:

self.cursor.execute(query)

self.connection.commit()

messagebox.showinfo("Успех", "Данные успешно удалены.")

except pyodbc.Error as e:

messagebox.showerror("Ошибка запроса", f"Ошибка: {str(e)}")

win.destroy()

selected\_table = self.table\_var.get()

if selected\_table != "Выберите таблицу":

columns = self.get\_columns(selected\_table)

win = tk.Tk()

win.title("Удалить строку")

tk.Label(win, text="Введите номер строки:").pack()

entry = tk.Entry(win)

entry.pack()

tk.Button(win, text="Удалить", command=lambda: del\_data\_func(entry.get())).pack()

def upd\_data(self):

def upd\_data\_func(number):

def upd\_data\_func2():

query = f"UPDATE {selected\_table} SET [{columns[1]}] = \'{entries[1].get()}\', [{columns[2]}] = \'{entries[2].get()}\', [{columns[3]}] = \'{entries[3].get()}\' where [{columns[0]}] = {number}"

print(query)

try:

self.cursor.execute(query)

self.connection.commit()

messagebox.showinfo("Успех", "Данные успешно изменены.")

except pyodbc.Error as e:

messagebox.showerror("Ошибка запроса", f"Ошибка: {str(e)}")

query = f"SELECT \* FROM {selected\_table} where [{columns[0]}] = {number}"

self.cursor.execute(query)

values = [row for row in self.cursor.fetchone()]

win2 = tk.Tk()

win2.title("Изменение данных")

# Создание текстовых полей для ввода данных

entries = []

for i, label in enumerate(columns):

ttk.Label(win2, text=label).grid(row=i, column=0, padx=10, pady=5)

entries.append(ttk.Entry(win2))

entries[i].insert(0, str(values[i]).strip())

entries[-1].grid(row=i, column=1, padx=10, pady=5)

tk.Button(win2, text="Изменить", command=upd\_data\_func2).grid(row=5, column=0)

selected\_table = self.table\_var.get()

if selected\_table != "Выберите таблицу":

columns = self.get\_columns(selected\_table)

win = tk.Tk()

win.title("Изменить строку")

tk.Label(win, text="Введите номер строки:").pack()

entry = tk.Entry(win)

entry.pack()

tk.Button(win, text="Изменить", command=lambda: upd\_data\_func(entry.get())).pack()

def get\_columns(self, table\_name):

columns\_query = f"SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = '{table\_name}'"

self.cursor.execute(columns\_query)

return [row.COLUMN\_NAME for row in self.cursor.fetchall()]

def populate\_table\_dropdown(self):

# Получаем список таблиц из базы данных

tables = [table.table\_name for table in self.cursor.tables(tableType='TABLE')]

self.table\_dropdown["values"] = tables

def display\_table(self):

selected\_table = self.table\_var.get()

if selected\_table == "Выберите таблицу":

return

table = PrettyTable(self.get\_columns(selected\_table))

query = f"SELECT \* FROM {selected\_table}"

self.cursor.execute(query)

for item in self.cursor.fetchall():

table.add\_row(item)

win = tk.Tk()

text = tk.Label(win, text=str(table), font=("Consolas", 15))

text.pack()

win.mainloop()

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

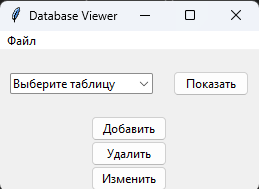
root = tk.Tk()

app = DatabaseApp(root)

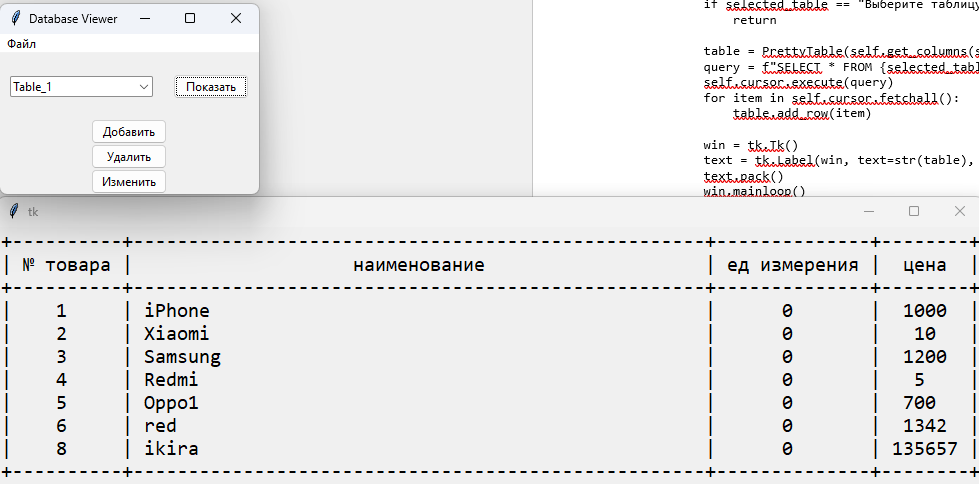
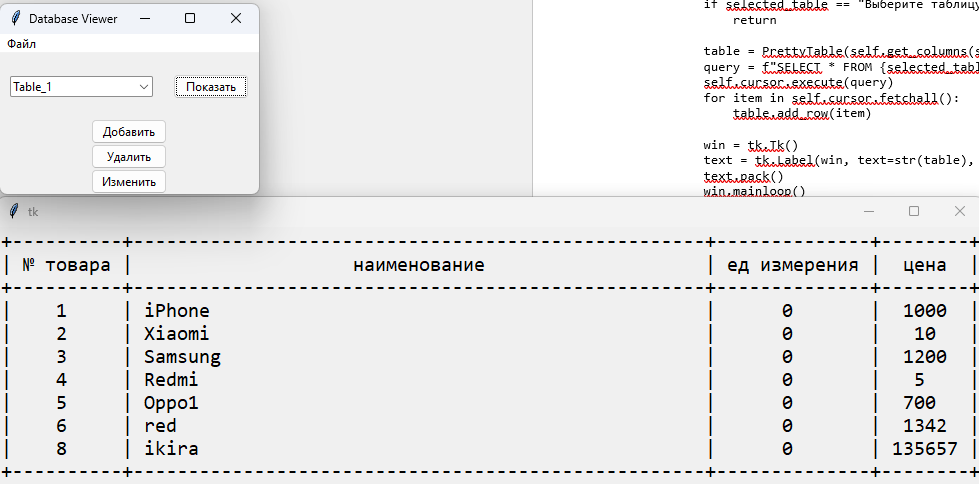
root.mainloop()

**Описание работы программы:**

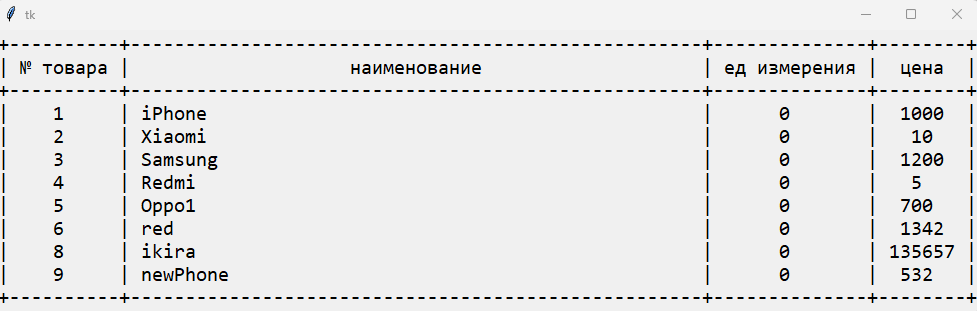
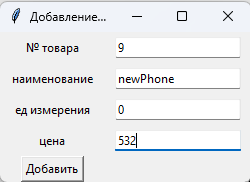
Главное окно:



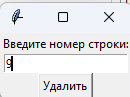
Просмотр данных таблицы:

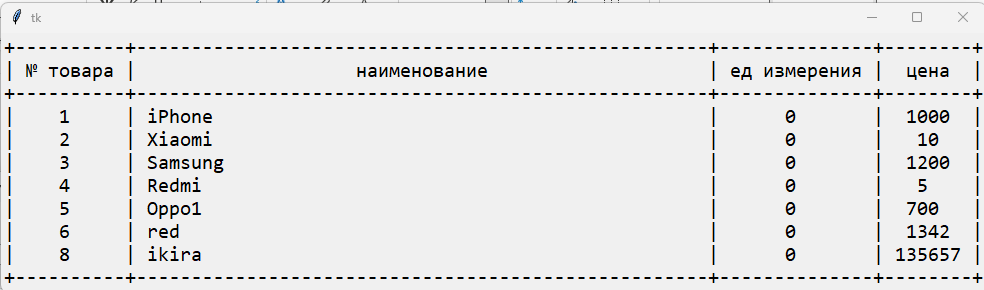


Добавление записи в таблицу:

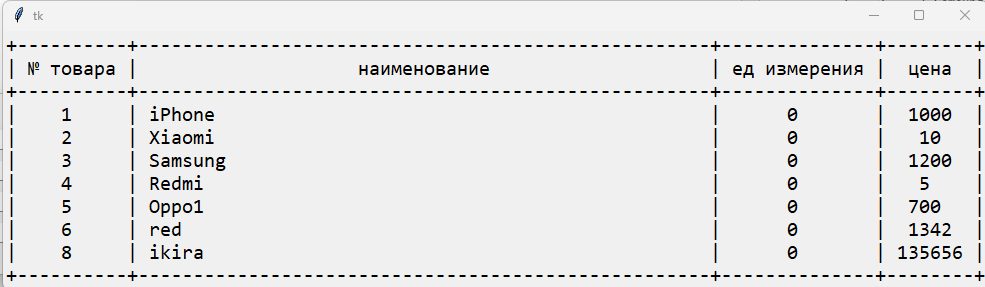
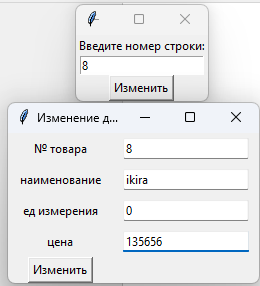


Удаление записи:



****

Изменение записи:

****

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