Отчёт по практической работа №6

МДК 01.01 Зорина Михаил Дмитриевича

Предметная область городской транспорт

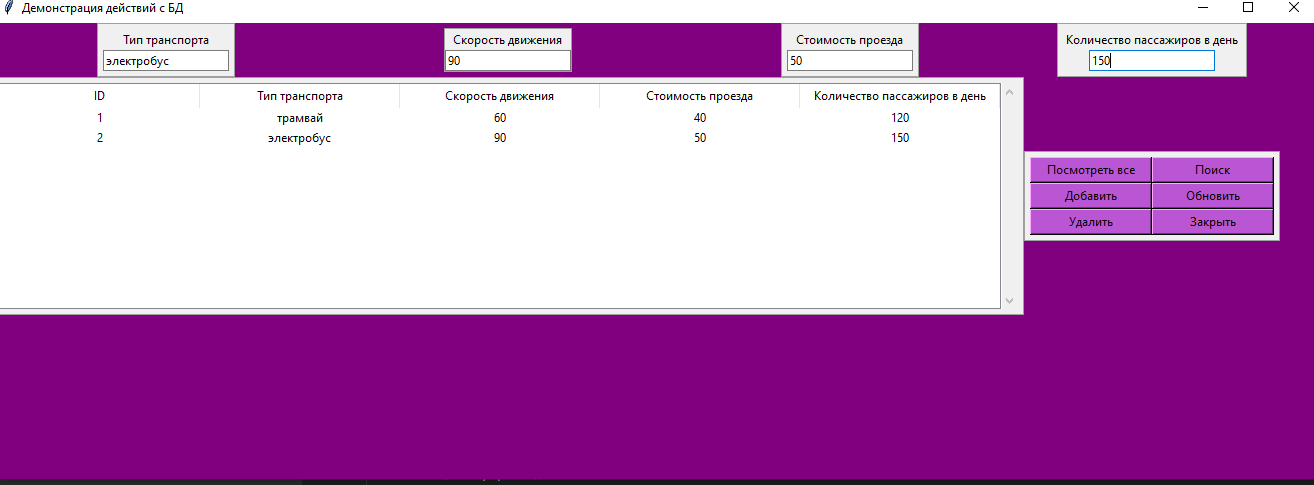
1. Интерфейс

from tkinter import \*  
from tkinter import ttk, messagebox  
from Cls import DB  
  
  
  
def get\_selected\_row(event):  
 global person  
 selection = table.selection()  
 item = table.item(selection[0])  
 person = item["values"]  
 lastname\_entry.delete(0, END)  
 lastname\_entry.insert(END, person[1])  
 firstname\_entry.delete(0, END)  
 firstname\_entry.insert(END, person[2])  
 scholarship\_entry.delete(0, END)  
 scholarship\_entry.insert(END, person[3])  
 passengers\_entry.delete(0, END)  
 passengers\_entry.insert(END, person[4]) # Added line  
  
def view\_command():  
 table.delete(\*table.get\_children())  
 for row in database\_students.view():  
 table.insert('', END, values=row)  
  
def search\_command():  
 table.delete(\*table.get\_children())  
 for row in database\_students.search(lastname\_text.get()):  
 table.insert('', END, values=row) # change here  
  
def add\_command():  
 database\_students.insert(lastname\_text.get(),  
 firstname\_text.get(),  
 scholarship\_text.get(),  
 passengers\_text.get()) # Added argument  
 view\_command()  
  
def delete\_command():  
 database\_students.delete(person[0])  
 view\_command()  
  
def update\_command():  
 database\_students.update(person[0],  
 lastname\_text.get(),  
 firstname\_text.get(),  
 scholarship\_text.get(),  
 passengers\_text.get()) # Added argument  
  
 view\_command()  
  
  
window = Tk()  
window.title("Демонстрация действий с БД")  
window.configure(bg="#800080")  
  
def on\_closing():  
 if messagebox.askokcancel("", "Закрыть программу?"):  
 window.destroy()  
  
  
window.protocol("WM\_DELETE\_WINDOW", on\_closing)  
frame = ttk.Frame(borderwidth=1, relief=SOLID, padding=5)  
l1 = Label(frame, text="Тип транспорта")  
l1.pack()  
lastname\_text = StringVar()  
lastname\_entry = ttk.Entry(frame, textvariable=lastname\_text)  
lastname\_entry.pack()  
frame.grid(row=0, column=0)  
frame = ttk.Frame(borderwidth=1, relief=SOLID, )  
l2 = Label(frame, text="Скорость движения", )  
l2.pack()  
firstname\_text = StringVar()  
firstname\_entry = ttk.Entry(frame, textvariable=firstname\_text)  
firstname\_entry.pack()  
frame.grid(row=0, column=1)  
frame = ttk.Frame(borderwidth=1, relief=SOLID, padding=5)  
l3 = Label(frame, text="Стоимость проезда", )  
l3.pack()  
scholarship\_text = StringVar()  
scholarship\_entry = ttk.Entry(frame, textvariable=scholarship\_text)  
scholarship\_entry.pack()  
frame.grid(row=0, column=2)  
frame = ttk.Frame(borderwidth=1, relief=SOLID, padding=5)  
l4 = Label(frame, text="Количество пассажиров в день", ) # Added line  
l4.pack() # Added line  
passengers\_text = StringVar() # Added line  
passengers\_entry = ttk.Entry(frame, textvariable=passengers\_text) # Added line  
passengers\_entry.pack() # Added line  
frame.grid(row=0, column=3) # Added line  
  
  
frame = ttk.Frame(borderwidth=1, relief=SOLID, padding=5)  
table = ttk.Treeview(frame, show='headings')  
heads = ['ID', 'Тип транспорта', 'Скорость движения', 'Стоимость проезда', 'Количество пассажиров в день'] # Added column  
table['columns'] = heads  
table['displaycolumns'] = ['ID', 'Тип транспорта', 'Скорость движения', 'Стоимость проезда', 'Количество пассажиров в день'] # Added column  
for head in heads:  
 table.heading(head, text=head, anchor='center')  
 table.column(head, anchor='center')  
table.pack(side=LEFT, fill=BOTH, expand=1)  
sb1 = Scrollbar(frame)  
sb1.pack(side=RIGHT, fill=Y)  
table.configure(yscrollcommand=sb1.set)  
sb1.configure(command=table.yview)  
frame.grid(row=1, column=0, columnspan=3) # Changed columnspan  
table.bind('<<TreeviewSelect>>', get\_selected\_row)  
  
frame = ttk.Frame(borderwidth=1, relief=SOLID, padding=5)  
b1 = Button(frame, text="Посмотреть все", width=16, command=view\_command, bg="#BA55D3")  
b1.grid(row=0, column=0)  
b2 = Button(frame, text="Поиск", width=16, command=search\_command, bg="#BA55D3")  
b2.grid(row=0, column=1)  
b3 = Button(frame, text="Добавить", width=16, command=add\_command, bg="#BA55D3")  
b3.grid(row=1, column=0)  
b4 = Button(frame, text="Обновить", width=16, command=update\_command, bg="#BA55D3")  
b4.grid(row=1, column=1)  
b5 = Button(frame, text="Удалить", width=16, command=delete\_command, bg="#BA55D3")  
b5.grid(row=2, column=0)  
b6 = Button(frame, text="Закрыть", width=16, command=on\_closing, bg="#BA55D3")  
b6.grid(row=2, column=1)  
frame.grid(row=1, column=3) # Changed grid  
database\_students = DB()  
window.mainloop()  
frame = ttk.Frame(borderwidth=1, relief=SOLID, padding=5)

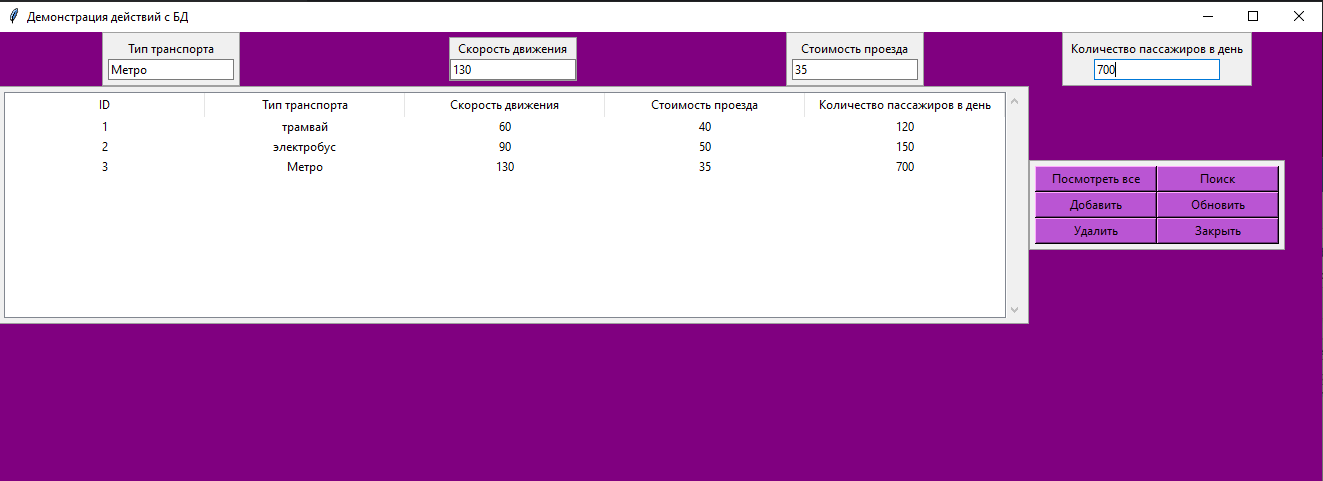
2.Классы

# подключаем библиотеку для работы с базой данных  
import sqlite3  
  
  
# создаём класс для работы с базой данных  
class DB:  
 def \_\_init\_\_(self):  
 self.conn = sqlite3.connect("transport.db")  
 self.cur = self.conn.cursor()  
 self.cur.execute(  
 "CREATE TABLE IF NOT EXISTS transport (id INTEGER PRIMARY KEY, "  
 "transport\_type TEXT, speed TEXT, ticket\_price TEXT, daily\_passenger\_count TEXT)")  
 self.conn.commit()  
  
 def \_\_del\_\_(self):  
 self.conn.close()  
  
 def view(self):  
 self.cur.execute("SELECT \* FROM transport")  
 rows = self.cur.fetchall()  
 return rows  
  
 def insert(self, transport\_type, speed, ticket\_price, daily\_passenger\_count):  
 self.cur.execute("INSERT INTO transport VALUES (NULL,?,?,?,?)",  
 (transport\_type, speed, ticket\_price, daily\_passenger\_count))  
 self.conn.commit()  
  
 def update(self, id, transport\_type, speed, ticket\_price, daily\_passenger\_count):  
 self.cur.execute("UPDATE transport SET transport\_type=?, speed=?, ticket\_price=?, daily\_passenger\_count=? WHERE id=?",  
 (transport\_type, speed, ticket\_price, daily\_passenger\_count, id))  
 self.conn.commit()  
  
 def delete(self, id):  
 self.cur.execute("DELETE FROM transport WHERE id=?", (id,))  
 self.conn.commit()  
  
 def search(self, transport\_type=""):  
 self.cur.execute("SELECT \* FROM transport WHERE transport\_type=?", (transport\_type,))  
 rows = self.cur.fetchall()  
 return rows  
  
  
# создаём экземпляр базы данных на основе класса  
db = DB()

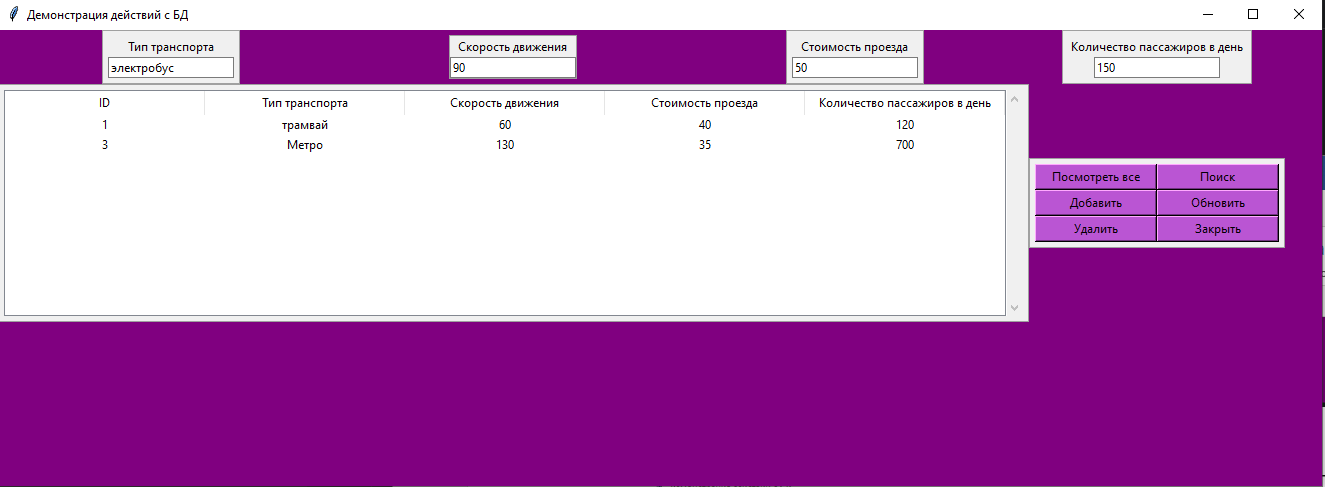
3.



4. Просмотр всех данных



5. Удаление



6.

