Листинг кода приложения Список контактов.

Файл app.py

Импортируем библиотеку sqlite3 для работы с базой данных.  
import sqlite3

Импортируем класс графической оболочки **MainWindow** и функции для работы с боазой данных

from main\_window import MainWindow

from database import data\_fetch, database\_setup

Запускаем программу

if \_\_name\_\_ == '\_\_main\_\_':

Соединяемся с базой данных. Используем менеджер контекста, чтобы исключить возможность не закрытия базы соединения с базой данных при получении ошибки во время работы

with sqlite3.connect("contacts.db") as conn:

Создаём базу данных, если её не существует.

database\_setup(conn)

Создаём объект класса графической оболочки

app = MainWindow(conn=conn)

Читаем базу данных и сохраняем ранее сохранённые контакты

data\_fetch(conn=conn, tree=app.tree)

Запускаем графическую оболочку

app.root.mainloop()

Главный запускаемый скрипт.

Файл main\_window.py

import tkinter.ttk as ttk

import tkinter.messagebox as tk\_message\_box

from sqlite3 import Connection

from tkinter import Tk, StringVar, Frame, SOLID, TOP, LEFT, RIGHT, Label, X, Button, Scrollbar, HORIZONTAL, VERTICAL, Y, \

BOTTOM, W, NO, Event, Toplevel, Radiobutton, Entry

from typing import Optional

from database import data\_insert, data\_fetch, database\_update, data\_delete

Класс MainWindow

class MainWindow:

Конструктор класса

def \_\_init\_\_(self, conn: Connection):

Сохраняем переменные объекта класса

Корневой объект графического интерфейса

self.root = Tk()

Соединение с базой данных

self.conn = conn

# variables

Временные переменные для хранения данных контакта

self.firstname = StringVar()

self.surname = StringVar()

self.gender = StringVar()

self.age = StringVar()

self.address = StringVar()

self.contact = StringVar()

self.mem\_id = ""

# frames

Рамки графического интерфейса для рисования в главном окне

self.top = Frame(self.root, width=500, bd=0, relief=SOLID, bg="#094391")

self.top.pack(side=TOP)

self.mid = Frame(self.root, width=500, bg="#094391")

self.mid.pack(side=TOP)

self.mid\_left = Frame(self.mid, width=0)

self.mid\_left.pack(side=LEFT, pady=5)

self.mid\_left\_padding = Frame(self.mid, width=420, bg="#094391")

self.mid\_left\_padding.pack(side=LEFT)

self.mid\_right = Frame(self.mid, width=100)

self.mid\_right.pack(side=RIGHT, pady=5)

self.table\_margin = Frame(self.root, width=500)

self.table\_margin.pack(side=TOP)

# Кнопки

self.btn\_add = Button(self.mid\_left, text="Добавить контакт", bg="#BECBFE", command=self.on\_add\_new\_contact)

self.btn\_add.pack()

self.btn\_delete = Button(self.mid\_right, text="Удалить контакт", bg="#BECBFE", command=self.on\_delete)

self.btn\_delete.pack(side=RIGHT)

# Таблицы

self.scroll\_bar\_x = Scrollbar(self.table\_margin, orient=HORIZONTAL)

self.scroll\_bar\_y = Scrollbar(self.table\_margin, orient=VERTICAL)

self.tree = ttk.Treeview(self.table\_margin,

columns=("MemberID", "Firstname", "Lastname", "Gender", "Age", "Address", "Contact"),

height=400, selectmode="extended", yscrollcommand=self.scroll\_bar\_y.set,

xscrollcommand=self.scroll\_bar\_x.set)

self.scroll\_bar\_y.config(command=self.tree.yview)

self.scroll\_bar\_y.pack(side=RIGHT, fill=Y)

self.scroll\_bar\_x.config(command=self.tree.xview)

self.scroll\_bar\_x.pack(side=BOTTOM, fill=X)

# Окна

self.new\_window: Optional[Toplevel] = None

# настройка главного окна

self.setup\_main\_window()

# настройка списка контактов

self.setup\_tree()

def setup\_tree(self):

Настройка заголовков окна с контактами

self.tree.heading('MemberID', text="MemberID", anchor=W)

self.tree.heading('Firstname', text="Имя", anchor=W)

self.tree.heading('Lastname', text="Фамилия", anchor=W)

self.tree.heading('Gender', text="Пол", anchor=W)

self.tree.heading('Age', text="Возраст", anchor=W)

self.tree.heading('Address', text="Адрес", anchor=W)

self.tree.heading('Contact', text="Контакт", anchor=W)

настройка колонок окна с контактами

self.tree.column('#0', stretch=NO, minwidth=0, width=0)

self.tree.column('#1', stretch=NO, minwidth=0, width=0)

self.tree.column('#2', stretch=NO, minwidth=0, width=80)

self.tree.column('#3', stretch=NO, minwidth=0, width=120)

self.tree.column('#4', stretch=NO, minwidth=0, width=90)

self.tree.column('#5', stretch=NO, minwidth=0, width=80)

self.tree.column('#6', stretch=NO, minwidth=0, width=120)

self.tree.column('#7', stretch=NO, minwidth=0, width=120)

self.tree.pack()

Привязка к двойному нажатию кнопки мыши обработчик self.on\_select\_row

self.tree.bind('<Double-Button-1>', self.on\_select\_row)

def setup\_main\_window(self):

Настройка главного окна

self.root.title("Список контактов")

width = 700

height = 400

screen\_width = self.root.winfo\_screenwidth()

screen\_height = self.root.winfo\_screenheight()

x = (screen\_width / 2) - (width / 2)

y = (screen\_height / 2) - (height / 2)

self.root.geometry("%dx%d+%d+%d" % (width, height, x, y))

self.root.resizable(False, False)

self.root.config(bg="#094391")

def on\_add\_new\_contact(self):

Функция обработчик события нажатия на кнопку Добавить

Создаётся новое окно

self.firstname.set("")

self.surname.set("")

self.gender.set("М")

self.age.set("")

self.address.set("")

self.contact.set("")

self.new\_window = Toplevel()

self.new\_window.title("Добавить контакт | Список контактов")

# self.new\_window.bind("Return", self.on\_submit())

width = 336

height = 260

screen\_width = self.root.winfo\_screenwidth()

screen\_height = self.root.winfo\_screenheight()

x = ((screen\_width / 2) - 455) - (width / 2)

y = ((screen\_height / 2) + 20) - (height / 2)

self.new\_window.resizable(False, False)

self.new\_window.geometry("%dx%d+%d+%d" % (width, height, x, y))

if 'UpdateWindow' in globals():

UpdateWindow.destroy()

form\_title = Frame(self.new\_window)

form\_title.pack(side=TOP)

contact\_form = Frame(self.new\_window)

contact\_form.pack(side=TOP, pady=10)

radio\_group = Frame(contact\_form)

Radiobutton(radio\_group, text="М", variable=self.gender, value="М", font=('arial', 14)).pack(side=LEFT)

Radiobutton(radio\_group, text="Ж", variable=self.gender, value="Ж", font=('arial', 14)).pack(

side=LEFT)

lbl\_firstname = Label(contact\_form, text="Имя", font=('arial', 14), bd=5)

lbl\_firstname.grid(row=0, sticky=W)

lbl\_lastname = Label(contact\_form, text="Фамилия", font=('arial', 14), bd=5)

lbl\_lastname.grid(row=1, sticky=W)

lbl\_gender = Label(contact\_form, text="Пол", font=('arial', 14), bd=5)

lbl\_gender.grid(row=2, sticky=W)

lbl\_age = Label(contact\_form, text="Возраст", font=('arial', 14), bd=5)

lbl\_age.grid(row=3, sticky=W)

lbl\_address = Label(contact\_form, text="Адрес", font=('arial', 14), bd=5)

lbl\_address.grid(row=4, sticky=W)

lbl\_contact = Label(contact\_form, text="Контакт", font=('arial', 14), bd=5)

lbl\_contact.grid(row=5, sticky=W)

firstname = Entry(contact\_form, textvariable=self.firstname, font=('arial', 14))

firstname.grid(row=0, column=1)

lastname = Entry(contact\_form, textvariable=self.surname, font=('arial', 14))

lastname.grid(row=1, column=1)

radio\_group.grid(row=2, column=1)

age = Entry(contact\_form, textvariable=self.age, font=('arial', 14))

age.grid(row=3, column=1)

address = Entry(contact\_form, textvariable=self.address, font=('arial', 14))

address.grid(row=4, column=1)

contact = Entry(contact\_form, textvariable=self.contact, font=('arial', 14))

contact.grid(row=5, column=1)

К кнопке Сохранить привязывается функция self.on\_submit()

btn\_add\_con = Button(contact\_form, text="Сохранить", width=44, command=self.on\_submit)

btn\_add\_con.grid(row=6, columnspan=2, pady=10)

Функция обработчик нажатия кнопки Сохранить в окне Добавить контакт

def on\_submit(self, event: Event = None):

Проверка все ли данные введены

if (self.firstname.get() == "" or self.surname.get() == "" or self.gender.get() == "" or

self.age.get() == "" or self.address.get() == "" or self.contact.get() == ""):

tk\_message\_box.showwarning('', 'Пожалуйста заполните требуемое поле', icon="warning")

else:

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

Вставка данных в базу данных

data\_insert(self.conn, [str(self.firstname.get()), str(self.surname.get()), str(self.gender.get()),

int(self.age.get()), str(self.address.get()), str(self.contact.get())])

Обновление списка контактов из базы данных

data\_fetch(self.conn, self.tree)

self.set\_all\_values\_empty()

Функция для обнуления временных переменных

def set\_all\_values\_empty(self):

self.firstname.set("")

self.surname.set("")

self.gender.set("М")

self.age.set("")

self.address.set("")

self.contact.set("")

Функция обработчик при нажатии кнопки Обновить в окне обновления контакта

def on\_update\_data(self, event: Event = None):

Проверка все ли данные введены

if self.gender.get() == "":

tk\_message\_box.showwarning('', 'Пожалуйста заполните требуемое поле', icon="warning")

else:

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

Добавление данных в базу данных

database\_update(conn=self.conn,

values=[str(self.firstname.get()), str(self.surname.get()), str(self.gender.get()),

str(self.age.get()), str(self.address.get()), str(self.contact.get()),

int(self.mem\_id)])

Обновление списка контактов из базы данных

data\_fetch(conn=self.conn, tree=self.tree)

Функция обработчик события двойного нажатия на контакт для обновления

def on\_select\_row(self, event: Event = None):

global UpdateWindow

cur\_item = self.tree.focus()

contents = (self.tree.item(cur\_item))

selected\_item = contents['values']

Заполнение временных переменных данными

self.mem\_id = selected\_item[0]

self.firstname.set(selected\_item[1])

self.surname.set(selected\_item[2])

self.gender.set(selected\_item[3])

self.age.set(selected\_item[4])

self.address.set(selected\_item[5])

self.contact.set(selected\_item[6])

Построение нового окна

UpdateWindow = Toplevel()

UpdateWindow.title("Обновить контакт | Список контактов")

width = 336

height = 260

screen\_width = self.root.winfo\_screenwidth()

screen\_height = self.root.winfo\_screenheight()

x = ((screen\_width / 2) + 450) - (width / 2)

y = ((screen\_height / 2) + 20) - (height / 2)

UpdateWindow.resizable(False, False)

UpdateWindow.geometry("%dx%d+%d+%d" % (width, height, x, y))

if 'NewWindow' in globals():

self.new\_window.destroy()

# frames

form\_title = Frame(UpdateWindow)

form\_title.pack(side=TOP)

contact\_form = Frame(UpdateWindow)

contact\_form.pack(side=TOP, pady=10)

radio\_group = Frame(contact\_form)

Radiobutton(radio\_group, text="М", variable=self.gender, value="М", font=('arial', 14)).pack(side=LEFT)

Radiobutton(radio\_group, text="Ж", variable=self.gender, value="Ж", font=('arial', 14)).pack(

side=LEFT)

# labels

lbl\_firstname = Label(contact\_form, text="Имя", font=('arial', 14), bd=5)

lbl\_firstname.grid(row=0, sticky=W)

lbl\_lastname = Label(contact\_form, text="Фамилия", font=('arial', 14), bd=5)

lbl\_lastname.grid(row=1, sticky=W)

lbl\_gender = Label(contact\_form, text="Пол", font=('arial', 14), bd=5)

lbl\_gender.grid(row=2, sticky=W)

lbl\_age = Label(contact\_form, text="Возраст", font=('arial', 14), bd=5)

lbl\_age.grid(row=3, sticky=W)

lbl\_address = Label(contact\_form, text="Адрес", font=('arial', 14), bd=5)

lbl\_address.grid(row=4, sticky=W)

lbl\_contact = Label(contact\_form, text="Контакт", font=('arial', 14), bd=5)

lbl\_contact.grid(row=5, sticky=W)

# entry

firstname = Entry(contact\_form, textvariable=self.firstname, font=('arial', 14))

firstname.grid(row=0, column=1)

lastname = Entry(contact\_form, textvariable=self.surname, font=('arial', 14))

lastname.grid(row=1, column=1)

radio\_group.grid(row=2, column=1)

age = Entry(contact\_form, textvariable=self.age, font=('arial', 14))

age.grid(row=3, column=1)

address = Entry(contact\_form, textvariable=self.address, font=('arial', 14))

address.grid(row=4, column=1)

contact = Entry(contact\_form, textvariable=self.contact, font=('arial', 14))

contact.grid(row=5, column=1)

# buttons

Привязка в нажатию на кнопку Обновить функции обработчика self.on\_update()

btn\_update\_con = Button(contact\_form, text="Обновить", width=44, command=self.on\_update\_data)

btn\_update\_con.grid(row=6, columnspan=2, pady=10)

Функция обработчик события нажатия кнопки Удалить

def on\_delete(self, event: Event = None):

Проверка выбран ли хоть один контакт

if not self.tree.selection():

tk\_message\_box.showwarning('', 'Сначала выберите что-нибудь!', icon="warning")

else:

result = tk\_message\_box.askquestion('', 'Вы уверены, что хотите удалить эту запись?', icon="warning")

if result == 'yes':

cur\_item = self.tree.focus()

contents = (self.tree.item(cur\_item))

selected\_item = contents['values']

self.tree.delete(cur\_item)

Удаляем элемент из базы данных

data\_delete(conn=self.conn, value=selected\_item[0])

Обновляем список контактов из базы данных

data\_fetch(conn=self.conn, tree=self.tree)

Файл database.py

Здесь собраны все функции для работы с базой данных

Импортируем название типа Connection из библиотеки sqlite3

from sqlite3 import Connection

Функция для создания базы данных

def database\_setup(conn: Connection):

"""

Create database if it does not exist

:return: None

"""

cursor = conn.cursor()

cursor.execute(

"CREATE TABLE IF NOT EXISTS `member` (mem\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, firstname TEXT, "

"lastname TEXT, gender TEXT, age TEXT, address TEXT, contact TEXT)")

cursor.close()

Функция для обновления списка контактов из базы данных

def data\_fetch(conn: Connection, tree):

"""

fetch all data from database and save it to ttk.TreeView

:param conn: sqlite3 connection

:param tree: ttk.TreeView object

:return: None

"""

cursor = conn.cursor()

cursor.execute("SELECT \* FROM `member` ORDER BY `lastname` ASC")

fetch = cursor.fetchall()

for data in fetch:

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

Функция для добавления данных в базу

def data\_insert(conn: Connection, values: list[str | int]) -> None:

cursor = conn.cursor()

cursor.execute(

"INSERT INTO `member` (firstname, lastname, gender, age, address, contact) VALUES(?, ?, ?, ?, ?, ?)", values)

conn.commit()

Функция для обновления конаткта в базе данных

def database\_update(conn: Connection, values: list[str | int]) -> None:

cursor = conn.cursor()

cursor.execute(

"UPDATE `member` SET `firstname` = ?, `lastname` = ?, `gender` =?, `age` = ?, `address` = ?, "

"`contact` = ? WHERE `mem\_id` = ?", values)

conn.commit()

Функция для удаления контакта из базы данных

def data\_delete(conn: Connection, value: str | int) -> None:

cursor = conn.cursor()

cursor.execute("DELETE FROM `member` WHERE `mem\_id` = %d" % value)

conn.commit()