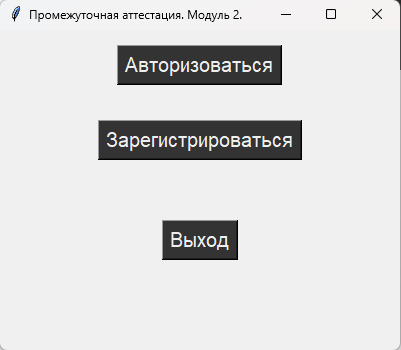
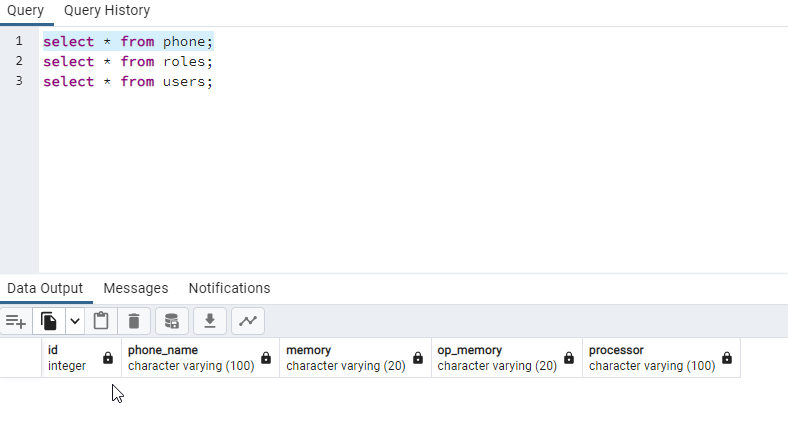
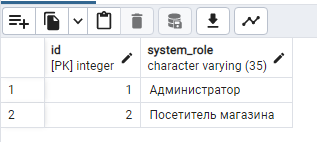
from tkinter import \*  
from tkinter import messagebox  
import psycopg2  
  
######## PSQL  
conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
cursor = conn.cursor()  
# создание таблицы телефонов  
createPhone = """  
CREATE TABLE IF NOT EXISTS phone(  
 id serial NOT NULL,  
 phone\_name VARCHAR(100) NOT NULL,  
 memory VARCHAR(20) NOT NULL,  
 op\_memory VARCHAR(20) NOT NULL,  
 processor VARCHAR(100) NOT NULL  
);"""  
# создание таблицы ролей  
createRoles = """  
CREATE TABLE IF NOT EXISTS roles (  
 id serial PRIMARY KEY NOT NULL,  
 system\_role VARCHAR(35) NOT NULL  
 );"""  
# создание таблицы пользователей  
createUsers = """  
CREATE TABLE IF NOT EXISTS users (  
 id serial NOT NULL,  
 surname VARCHAR(50) NOT NULL,  
 name VARCHAR(50) NOT NULL,  
 patronymic VARCHAR(50) NOT NULL,  
 login VARCHAR(50) NOT NULL,  
 password VARCHAR(50) NOT NULL,  
 roles\_id int DEFAULT 2,  
 FOREIGN KEY (roles\_id) REFERENCES roles (id),  
 is\_deleted boolean DEFAULT FALSE  
 );"""  
# добавление ролей  
insertRole = """  
INSERT INTO roles (system\_role) VALUES  
('Администратор'),  
('Посетитель магазина');  
"""  
  
cursor.execute(createPhone)  
cursor.execute(createRoles)  
cursor.execute(createUsers)  
cursor.execute(insertRole)  
conn.commit()  
conn.close()  
  
  
def login\_window():  
 login\_win = Toplevel(root)  
 login\_win.title('Окно авторизации')  
 login\_win.geometry('250x160+200+200')  
  
 login\_frame = Frame(login\_win)  
 login\_frame.pack(pady=10)  
  
 login\_label = Label(login\_frame, text="Введите логин и пароль")  
 login\_label.grid(row=1, columnspan=2, pady=10)  
  
 login\_entry = Entry(login\_frame)  
 login\_entry.grid(row=2, column=0, padx=5, pady=5)  
  
 password\_entry = Entry(login\_frame, show="\*")  
 password\_entry.grid(row=3, column=0, padx=5, pady=5)  
  
 btn\_login = Button(login\_frame, text="Войти", command=lambda: login(login\_entry.get(), password\_entry.get()))  
 btn\_login.grid(row=2, column=1, padx=5, pady=5)  
  
 btn\_cancel = Button(login\_frame, text="Отмена", command=login\_win.destroy)  
 btn\_cancel.grid(row=3, column=1, padx=5, pady=5)  
  
 def login(login, password):  
 if not (login and password):  
 messagebox.showerror(title='Ошибка', message='Заполните все поля')  
 else:  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
 cur = conn.cursor()  
 try:  
 cur.execute('SELECT \* FROM users WHERE login=%s AND password=%s AND is\_deleted = false', (login, password))  
 user = cur.fetchone()  
 if user:  
 if user is not None:  
 if user[6] == 1:  
 login\_win.destroy()  
 admin\_window(user)  
 elif user[6] == 2:  
 login\_win.destroy()  
 visitor\_window(user)  
 else:  
 messagebox.showerror(title='Ошибка', message='Ошибка в логине или пароле')  
  
  
 cur.close()  
 conn.close()  
 except Exception:  
 print('Ошибка авторизации')  
  
def delete\_user(login):  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
 cur = conn.cursor()  
 try:  
 cur.execute(f"UPDATE users SET is\_deleted = true WHERE login='{login}';")  
 conn.commit()  
 cur.close()  
 conn.close()  
 messagebox.showinfo(title='Информация', message='Аккаунт удален.\nПрограмма будет закрыта.')  
 root.destroy()  
 except Exception:  
 print('Ошибка удаления аккаунта')  
  
def admin\_window(user):  
 admin\_win = Toplevel(root)  
 admin\_win.grab\_set()  
 admin\_win.title("Окно администратора")  
  
 admin\_frame = LabelFrame(admin\_win, text='Информация о пользователе', pady=5, padx=5)  
 admin\_frame.pack(side=TOP, fill=X)  
 # логин активного пользователя  
 login\_label = Label(admin\_frame, text=f"Вы вошли как - {user[4]}", font=('Calibri', 12))  
 login\_label.pack(side=LEFT)  
 # кнопка удалить аккаунт  
 btn\_delete = Button(admin\_frame, text="Удалить аккаунт", font=('Calibri', 11), command=lambda: delete\_user(user[4]))  
 btn\_delete.pack(side=RIGHT)  
  
 # фрейм добавления телефона  
 add\_phone\_frame = LabelFrame(admin\_win, text='Добавление телефона')  
 add\_phone\_frame.pack(side=TOP, fill=X, pady=10)  
 # название телефона  
 l\_ph\_name = Label(add\_phone\_frame, text='Модель')  
 l\_ph\_name.grid(row=1, column=0, padx=5, pady=10)  
 e\_ph\_name = Entry(add\_phone\_frame)  
 e\_ph\_name.grid(row=1, column=1, padx=5, pady=10)  
 # память телефона  
 l\_memory = Label(add\_phone\_frame, text='Размер памяти')  
 l\_memory.grid(row=1, column=2, padx=5, pady=10)  
 e\_memory = Entry(add\_phone\_frame)  
 e\_memory.grid(row=1, column=3, padx=5, pady=10)  
 # оперативная память  
 l\_op\_memory = Label(add\_phone\_frame, text='Оперативная память')  
 l\_op\_memory.grid(row=1, column=4, padx=5, pady=10)  
 e\_op\_memory = Entry(add\_phone\_frame)  
 e\_op\_memory.grid(row=1, column=5, padx=5, pady=10)  
 # процессор  
 l\_processor = Label(add\_phone\_frame, text='Процессор')  
 l\_processor.grid(row=1, column=6, padx=5, pady=10)  
 e\_processor = Entry(add\_phone\_frame)  
 e\_processor.grid(row=1, column=7, padx=5, pady=10)  
 # кнопка добавить телефон  
 b\_add\_phone = Button(add\_phone\_frame, text='Добавить', command=lambda: insertPhone())  
 b\_add\_phone.grid(row=2, column=0, pady=10, padx=2)  
  
  
 # изменить роль пользователя  
 change\_role\_frame = LabelFrame(admin\_win, text='Редактирование роли пользователя (1 - Admin, 2 - User)')  
 change\_role\_frame.pack(side=BOTTOM, fill=X, pady=10)  
 l\_user\_id = Label(change\_role\_frame, text='Введите ID пользователя')  
 l\_user\_id.pack(side=LEFT, padx=7)  
 e\_user\_id = Entry(change\_role\_frame)  
 e\_user\_id.pack(side=LEFT, padx=7)  
 l\_new\_role = Label(change\_role\_frame, text='и ID новой роли')  
 l\_new\_role.pack(side=LEFT, padx=7)  
 e\_new\_role = Entry(change\_role\_frame)  
 e\_new\_role.pack(side=LEFT, padx=7)  
 b\_change\_role = Button(change\_role\_frame, text='Изменить роль', command=lambda: changeRole())  
 b\_change\_role.pack(side=LEFT, pady=7, padx=20)  
  
 # удаление телефона  
 delete\_phone\_frame = LabelFrame(admin\_win, text='Удаление телефона')  
 delete\_phone\_frame.pack(side=BOTTOM, fill=X, pady=35)  
 l\_phone\_id = Label(delete\_phone\_frame, text='Удалить телефон с ID')  
 l\_phone\_id.pack(side=LEFT, padx=7)  
 e\_phone\_id = Entry(delete\_phone\_frame)  
 e\_phone\_id.pack(side=LEFT)  
 b\_delete\_phone = Button(delete\_phone\_frame, text='Удалить телефон', command=lambda: deletePhone())  
 b\_delete\_phone.pack(side=LEFT, pady=7, padx=20)  
  
  
 # список телефонов  
 phone\_list = LabelFrame(admin\_win, text='Список телефонов')  
 phone\_list.pack(side=LEFT, fill=X, padx=2)  
 list\_1 = Label(phone\_list, height=20, width=64)  
 list\_1.grid(row=1, column=0)  
 b\_show\_phone = Button(phone\_list, text='Показать\Обновить список', command=lambda: showPhone())  
 b\_show\_phone.grid(row=8, column=0)  
  
 # список пользователей  
 user\_list = LabelFrame(admin\_win, text='Список пользователей')  
 user\_list.pack(side=LEFT, fill=X, padx=1)  
 list\_2 = Label(user\_list, height=20, width=64)  
 list\_2.grid(row=1, column=0)  
 b\_show\_user = Button(user\_list, text='Показать\Обновить список', command=lambda: showUsers())  
 b\_show\_user.grid(row=8, column=0)  
  
  
 # функция изменения роли  
 def changeRole():  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
 cursor = conn.cursor()  
 param1 = e\_user\_id.get()  
 param2 = e\_new\_role.get()  
 try:  
 cursor.execute('UPDATE users SET roles\_id = %s WHERE id = %s', (param2, param1))  
 conn.commit()  
 messagebox.showinfo(title='Информация', message='Роль изменена')  
 except Exception:  
 print('Ошибка изменения роли')  
 conn.rollback()  
 conn.close()  
  
 # функция удаления телефона по номеру ИД  
 def deletePhone():  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
 cursor = conn.cursor()  
 try:  
 cursor.execute('DELETE FROM phone WHERE id = ' + e\_phone\_id.get())  
 conn.commit()  
 messagebox.showinfo(title='Информация', message='Телефон удален')  
 except Exception:  
 print('Ошибка удаления телефона')  
 conn.rollback()  
 conn.close()  
  
 # добавить телефон  
 def insertPhone():  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
 cursor = conn.cursor()  
 try:  
 cursor.execute("""INSERT INTO phone (phone\_name, memory, op\_memory, processor) VALUES (%s, %s, %s, %s)""",  
 (e\_ph\_name.get(), e\_memory.get(), e\_op\_memory.get(), e\_processor.get()))  
 conn.commit()  
 messagebox.showinfo(title='Информация', message='Телефон добавлен')  
 except Exception:  
 print('Ошибка добавления телефона')  
 conn.rollback()  
 conn.close()  
  
 # показать список телефонов  
 def showPhone():  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
 cursor = conn.cursor()  
 try:  
 cursor.execute('SELECT \* FROM phone')  
 records = cursor.fetchall()  
 output = ''  
  
 for record in records:  
 list\_1.config(text=f'{output} \n {record[0]} - {record[1]}, {record[2]}, {record[3]}, {record[4]}')  
 output = list\_1['text']  
 except Exception:  
 print('Ошибка формирования списка')  
 conn.rollback()  
 conn.close()  
  
 # показать список пользователей  
 def showUsers():  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
 cursor = conn.cursor()  
 try:  
 cursor.execute('SELECT \* FROM users')  
 records = cursor.fetchall()  
 output = ''  
  
 for record in records:  
 list\_2.config(text=f'{output} \n ID: {record[0]} - ФИО: {record[1]} {record[2]} {record[3]}.\n '  
 f'Логин: {record[4]}. Пароль: {record[5]}.\n Роль в системе: {record[6]}.'  
 f'Логическое удаление: {record[7]}')  
 output = list\_2['text']  
 except Exception:  
 print('Ошибка формирования списка')  
 conn.rollback()  
 conn.close()  
  
def visitor\_window(user):  
 visitor\_win = Toplevel(root)  
 visitor\_win.title("Окно пользователя")  
 visitor\_win.geometry('600x400+200+200')  
 # инфо о пользователе  
 visitor\_frame = LabelFrame(visitor\_win, text='Информация о пользователе', pady=5, padx=5)  
 visitor\_frame.pack(side=TOP, fill=X)  
 # логин активного пользователя  
 login\_label = Label(visitor\_frame, text=f"Вы вошли как - {user[4]}", font=('Calibri', 12))  
 login\_label.pack(side=LEFT)  
 # кнопка удалить аккаунт  
 btn\_delete = Button(visitor\_frame, text="Удалить аккаунт", font=('Calibri', 11), command=lambda: delete\_user(user[4]))  
 btn\_delete.pack(side=RIGHT)  
  
 # список телефонов  
 phone\_list = LabelFrame(visitor\_win, text='Список телефонов')  
 phone\_list.pack(side=TOP, fill=X)  
 list\_1 = Label(phone\_list, height=18, width=63)  
 list\_1.pack()  
 b\_show\_phone = Button(phone\_list, text='Показать\Обновить список', command=lambda: showPhone())  
 b\_show\_phone.pack(side=BOTTOM)  
  
 def showPhone():  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost', port='5432')  
 cursor = conn.cursor()  
 try:  
 cursor.execute('SELECT \* FROM phone')  
 records = cursor.fetchall()  
 output = ''  
  
 for record in records:  
 list\_1.config(text=f'{output} \n {record[0]} - {record[1]}, {record[2]}, {record[3]}, {record[4]}')  
 output = list\_1['text']  
 except Exception:  
 print('Ошибка формирования списка')  
 conn.rollback()  
 conn.close()  
  
def reg\_window():  
 reg\_win = Toplevel(root)  
 reg\_win.title("Регистрация")  
 reg\_win.geometry('400x320+200+200')  
 reg\_win.grab\_set()  
  
 reg\_frame = Frame(reg\_win)  
 reg\_frame.pack(pady=10)  
  
 label = Label(reg\_frame, text="Заполните все поля для регистрации")  
 label.pack()  
  
 l\_surname = Label(reg\_frame, text='Фамилия:').pack()  
 e\_surname = Entry(reg\_frame)  
 e\_surname.pack()  
 l\_name = Label(reg\_frame, text='Имя:').pack()  
 e\_name = Entry(reg\_frame)  
 e\_name.pack()  
 l\_patronymic = Label(reg\_frame, text='Отчество:').pack()  
 e\_patronymic = Entry(reg\_frame)  
 e\_patronymic.pack()  
 l\_log = Label(reg\_frame, text='Логин:').pack()  
 e\_log = Entry(reg\_frame)  
 e\_log.pack()  
 l\_regpass = Label(reg\_frame, text='Пароль:').pack()  
 e\_regpass = Entry(reg\_frame)  
 e\_regpass.pack()  
  
 btn\_register = Button(reg\_frame, text="Зарегистрироваться",  
 command=lambda: registration(e\_surname.get(), e\_name.get(), e\_patronymic.get(),  
 e\_log.get(), e\_regpass.get()))  
 btn\_register.pack(pady=15)  
  
 btn\_cancel = Button(reg\_frame, text="Отмена", command=reg\_win.destroy)  
 btn\_cancel.pack()  
  
 def registration(surname, name, patronymic, login, password):  
 surname = e\_surname.get()  
 name = e\_name.get()  
 patronymic = e\_patronymic.get()  
 login = e\_log.get()  
 password = e\_regpass.get()  
 if not (surname and name and patronymic and login and password):  
 messagebox.showerror(title='Ошибка', message='Заполните все поля')  
 return False  
 else:  
 conn = psycopg2.connect(database='postgres', user='postgres', password='0712', host='localhost',  
 port='5432')  
 cursor = conn.cursor()  
 try:  
 cursor.execute("""INSERT INTO users (surname, name, patronymic, login, password)   
 VALUES (%s, %s, %s, %s, %s)""", (surname, name, patronymic, login, password))  
 conn.commit()  
 cursor.close()  
 conn.close()  
 messagebox.showinfo(title='Информация', message='Успешная регистрация')  
 reg\_win.destroy()  
 return True  
 except Exception:  
 print('Ошибка регистрации')  
 return False  
  
  
root = Tk()  
root.title('Промежуточная аттестация. Модуль 2.')  
root.geometry('400x320+200+200')  
  
button\_1 = Button(text='Авторизоваться', bg='#333', fg='#eee', font='18', activebackground='#070',  
 activeforeground='#006', state='normal', command=login\_window).pack(pady=15)  
  
button\_2 = Button(text='Зарегистрироваться', bg='#333', fg='#eee', font='18', activebackground='#070',  
 activeforeground='#006', state='normal', command=reg\_window).pack(pady=20)  
  
quitButton = Button(text='Выход', bg='#333', fg='#eee', font='18', command=root.destroy).pack(pady=40)  
  
  
root.mainloop()

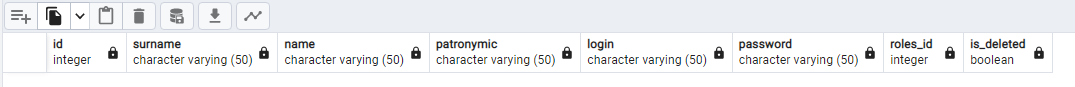
Запуск приложения (Главное окно)



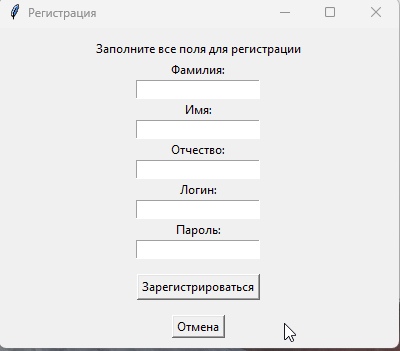
Созданы таблицы (users, phones, roles)



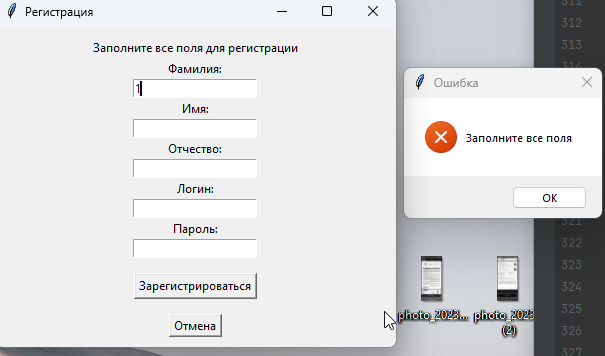




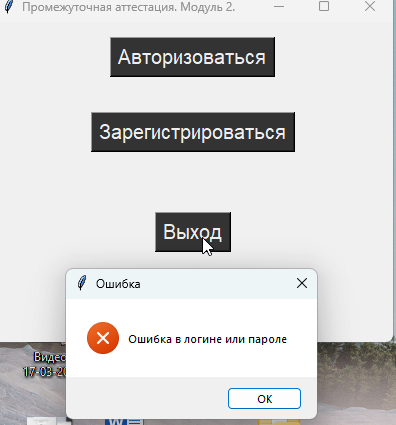
Окно регистрации



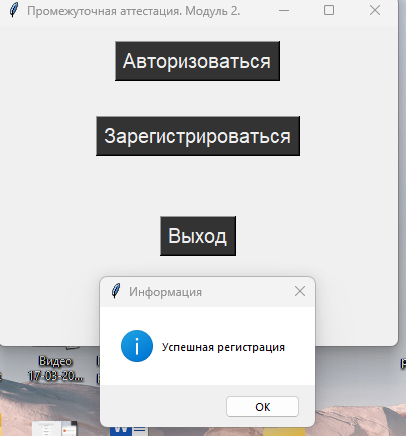
Проверка на заполнение полей

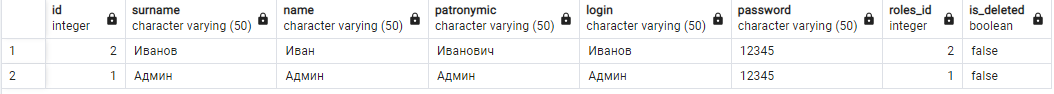


Если пользователя нет в БД, ошибка при авторизации

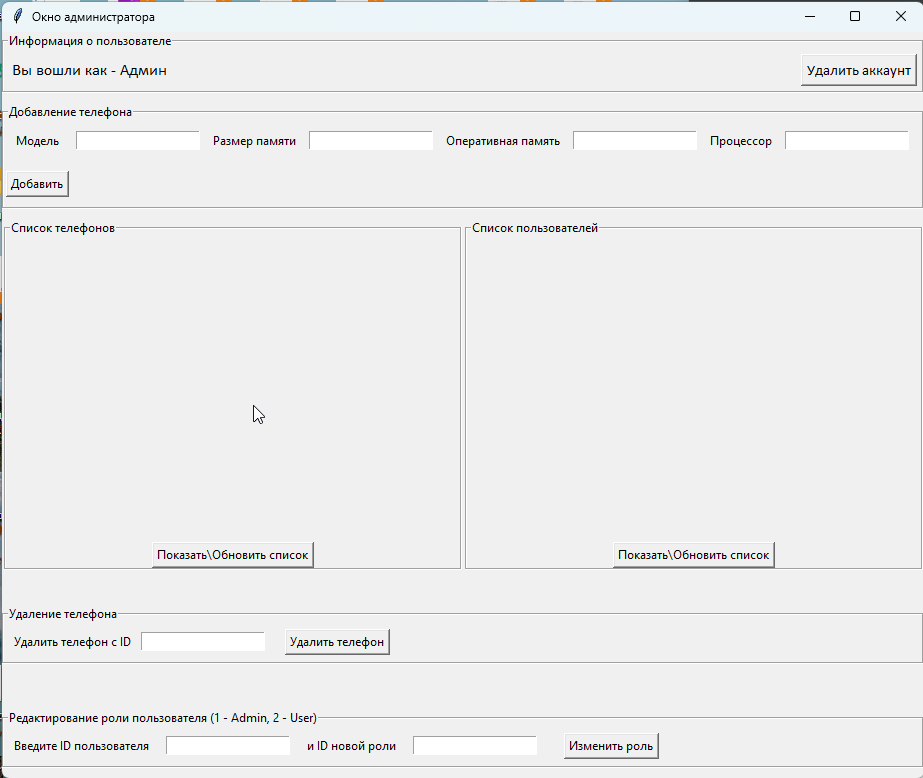


После успешной регистрации, форма регистрации закрывается и появляется окно, что регистрация успешна. Пользователь добавляется в БД

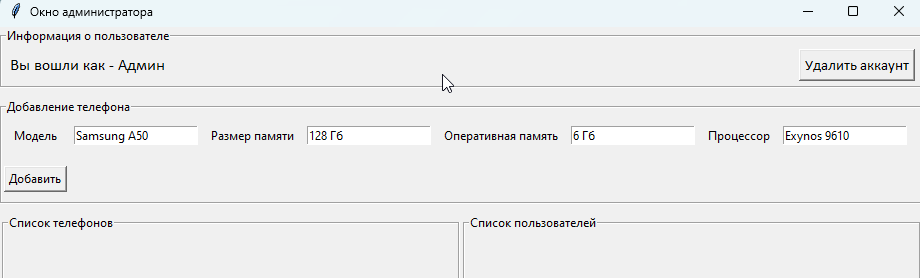


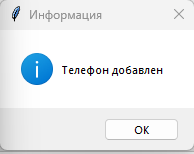


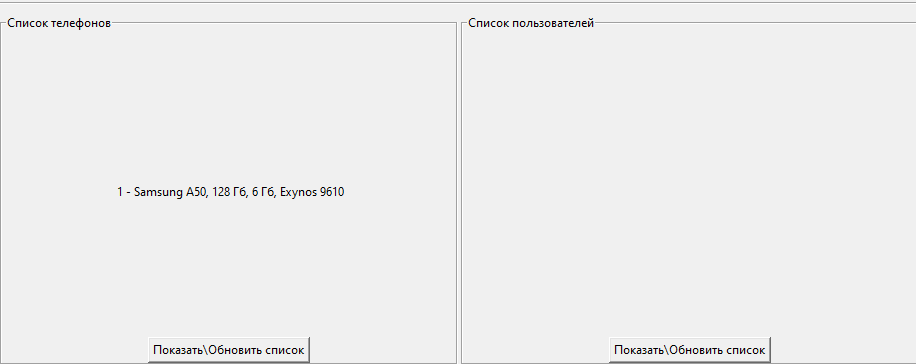
Если пользователь вошел под админом, появляется окно администратора

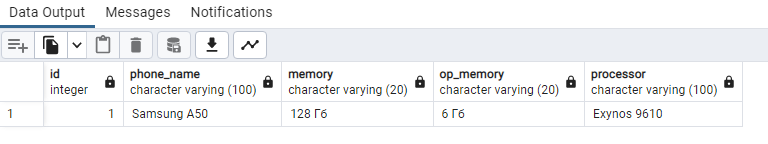


Добавление телефона

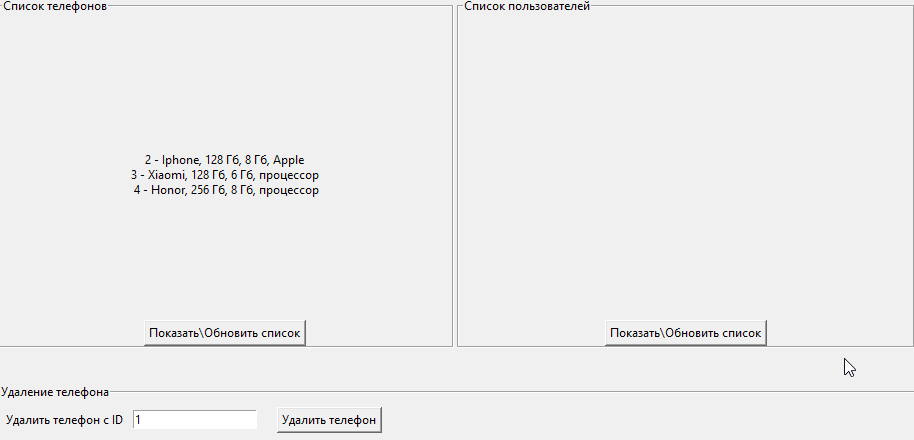


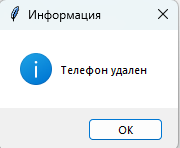


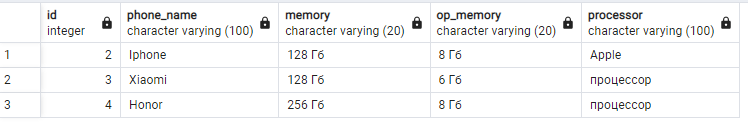




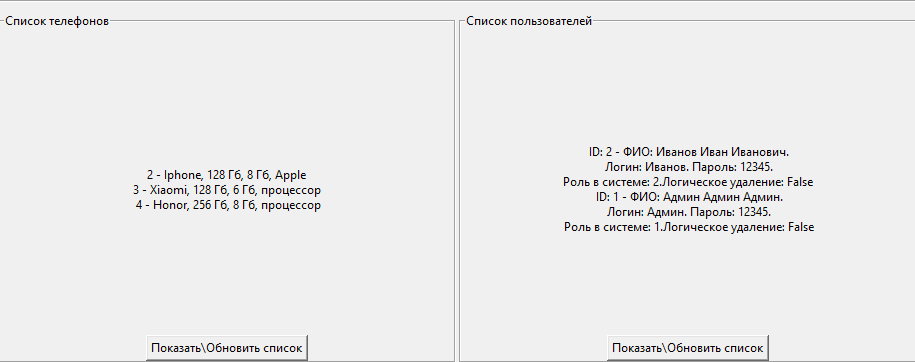
Удаление телефона



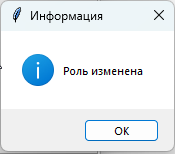


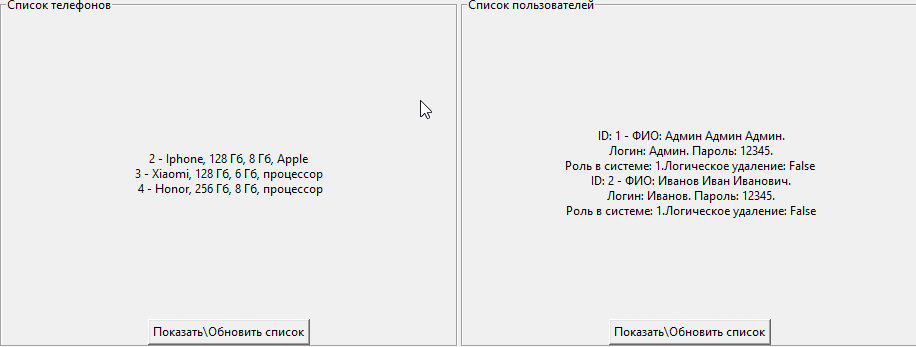


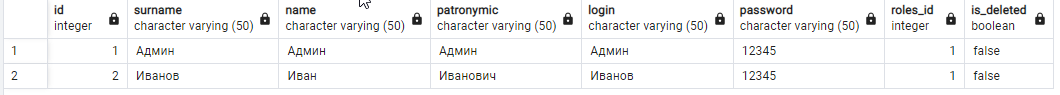
Список пользователей



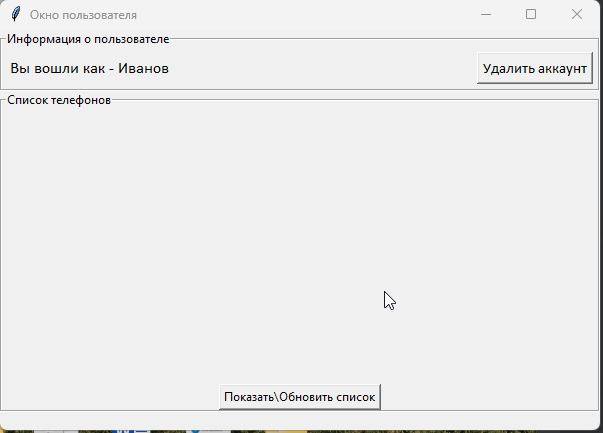
Изменение роли



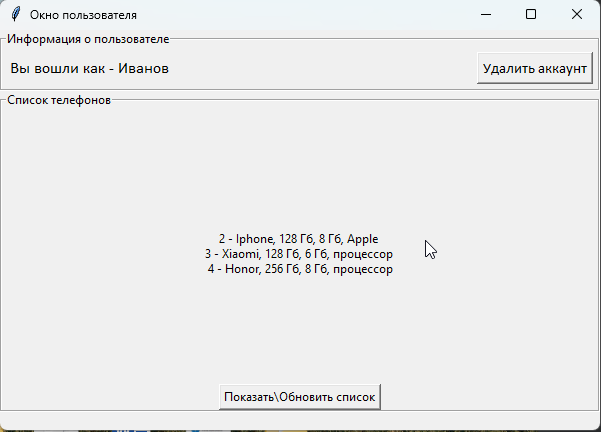




Окно пользователя



Список телефонов



Удаление аккаунта

Сделал без подтверждения. Удаление одним кликом

