Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №9**

**«РАБОТА С ТИПИЗИРОВАННЫМИ ФАЙЛАМИ»**

**ПО ДИСЦИПЛИНЕ «РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Куликов Иван Алексеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров, 2023

**Цель работы:** получить представление об организации и использовании информации посредством типизированных файлов, закрепить навыки создания пользовательского интерфейса.

**Задание (Вариант 12):**

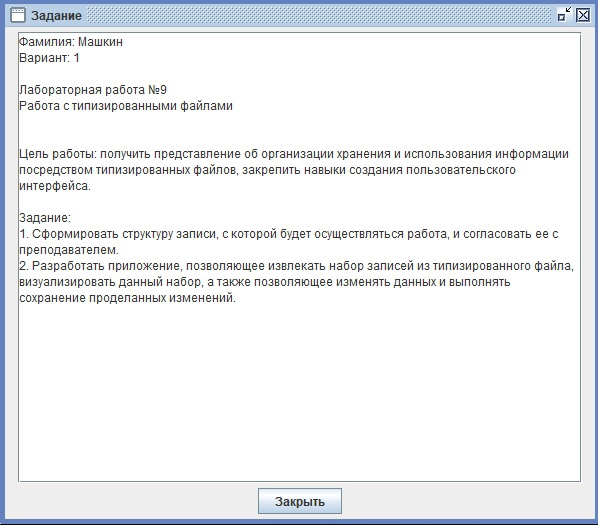
На рисунке 1 представлено индивидуальное задание по домашней контрольной работе №9.

Рисунок 1 - Индивидуальное задание

**Реализация алгоритмов функционирования**

При выполнении домашней контрольной работы была использована СУБД PostgreSQL 15. Работа с pgAdmin 4 представлена на рисунке 2.

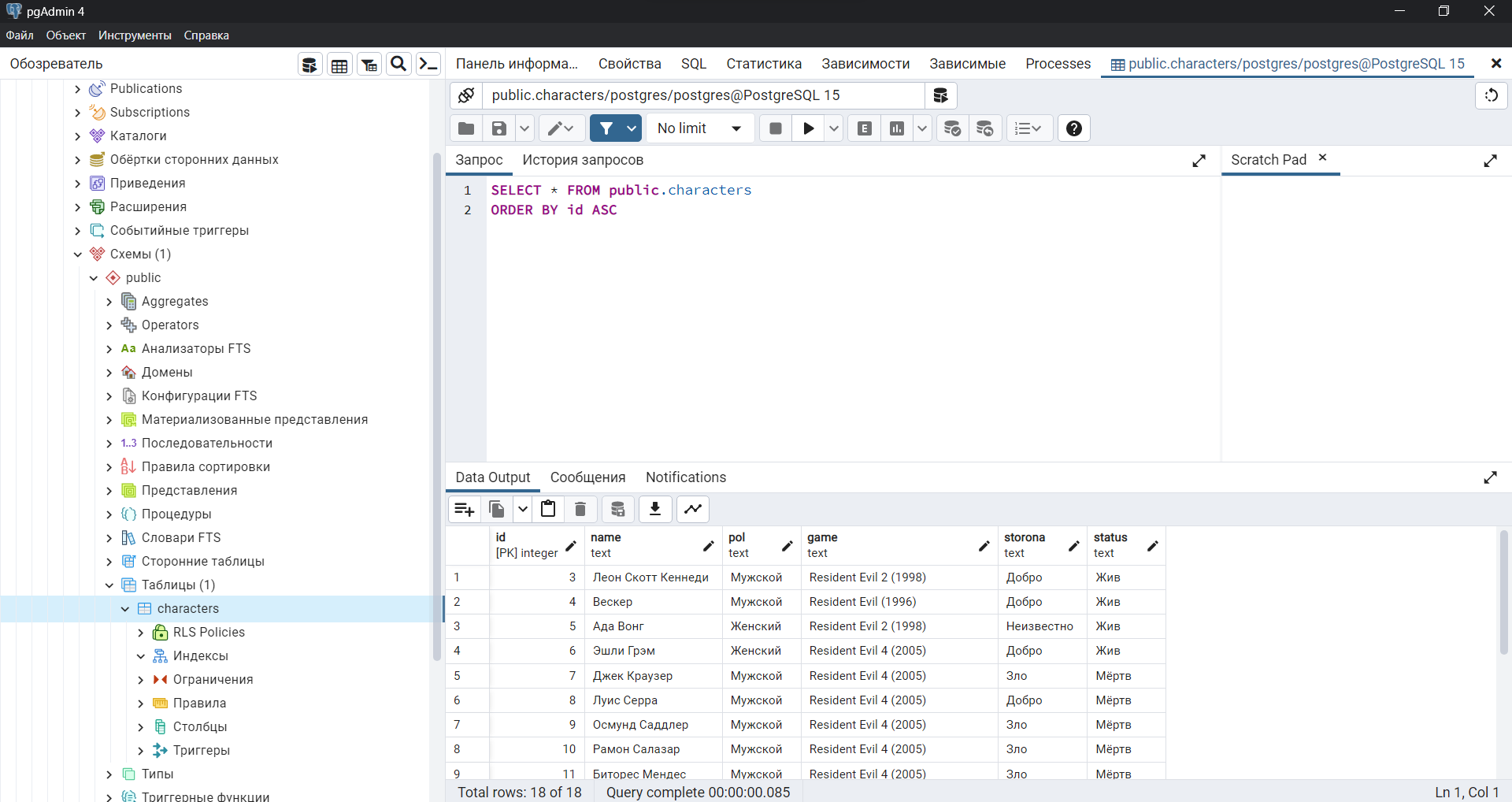


Рисунок 2 - pgAdmin 4

**Код программы:**

**import** tkinter **as** tk  
**import** tkinter.messagebox  
**from** tkinter **import** ttk  
**import** psycopg2  
**import** psycopg2.extras  
  
**class** Main(tk.Frame):  
 **def** \_\_init\_\_(self, root):  
 super().\_\_init\_\_(root)  
 self.init\_main()  
 self.db = db  
 self.view\_records()  
  
 **def** init\_main(self):  
 toolbar = tk.Frame(bg=**'#d7d8e0'**, bd=2)  
 toolbar.pack(side=tk.TOP, fill=tk.X)  
  
 self.add\_img = tk.PhotoImage(file=**'add.png'**)  
 btn\_open\_dialog = tk.Button(toolbar, text=**'Добавить персонажа'**, command=self.open\_dialog, bg=**'#d7d8e0'**, bd=0,  
 compound=tk.TOP, image=self.add\_img)  
 btn\_open\_dialog.pack(side=tk.LEFT)  
  
 self.update\_img = tk.PhotoImage(file=**'update.png'**)  
 btn\_edit\_dialog = tk.Button(toolbar, text=**'Редактировать'**, bg=**'#d7d8e0'**, bd=0, image=self.update\_img,  
 compound=tk.TOP, command=self.open\_update\_dialog)  
 btn\_edit\_dialog.pack(side=tk.LEFT)  
  
 self.delete\_img = tk.PhotoImage(file=**'delete.png'**)  
 btn\_delete = tk.Button(toolbar, text=**'Удалить персонажа'**, bg=**'#d7d8e0'**, bd=0, image=self.delete\_img,  
 compound=tk.TOP, command=self.delete\_records)  
 btn\_delete.pack(side=tk.RIGHT)  
  
 self.search\_img = tk.PhotoImage(file=**'search.png'**)  
 btn\_search = tk.Button(toolbar, text=**'Поиск'**, bg=**'#d7d8e0'**, bd=0, image=self.search\_img,  
 compound=tk.TOP, command=self.open\_search\_dialog)  
 btn\_search.pack(side=tk.RIGHT)  
  
 self.refresh\_img = tk.PhotoImage(file=**'refresh.png'**)  
 btn\_refresh = tk.Button(toolbar, text=**'Обновить'**, bg=**'#d7d8e0'**, bd=0, image=self.refresh\_img,  
 compound=tk.TOP, command=self.view\_records)  
 btn\_refresh.pack(side=tk.RIGHT)  
  
 label = tk.Label(root, text=**"Таблица 1 - Персонажи серии игр Resident Evil"**)  
 label.pack()  
  
 label = tk.Label(toolbar, text =**'Resident Evil'**, font=(**"Times New Roman"**, 52, **"bold"**))  
 label.pack(anchor=**'center'**)  
  
 self.tree = ttk.Treeview(self, columns=(**'ID'**, **'name'**, **'pol'**, **'game'**, **'storona'**, **'status'**), height=15, show=**'headings'**)  
  
 self.tree.column(**'ID'**, width=30, anchor=tk.CENTER)  
 self.tree.column(**'name'**, width=300, anchor=tk.CENTER)  
 self.tree.column(**'pol'**, width=75, anchor=tk.CENTER)  
 self.tree.column(**'game'**, width=270, anchor=tk.CENTER)  
 self.tree.column(**'storona'**, width=75, anchor=tk.CENTER)  
 self.tree.column(**'status'**, width=100, anchor=tk.CENTER)  
  
 self.tree.heading(**'ID'**, text=**'ID'**)  
 self.tree.heading(**'name'**, text=**'Полное имя'**)  
 self.tree.heading(**'pol'**, text=**'Пол'**)  
 self.tree.heading(**'game'**, text=**'Первое появление'**)  
 self.tree.heading(**'storona'**, text=**'Статус'**)  
 self.tree.heading(**'status'**, text=**'Сторона'**)  
  
 self.tree.pack(side=tk.LEFT)  
  
 scroll = tk.Scrollbar(self, command=self.tree.yview)  
 scroll.pack(side=tk.LEFT, fill=tk.Y)  
 self.tree.configure(yscrollcommand=scroll.set)  
  
 **def** records(self, name, pol, game, storona, status):  
 self.db.insert\_data(name, pol, game, storona, status)  
 self.view\_records()  
  
 **def** update\_record(self, name, pol, game, storona, status):  
 **try**:  
 self.db.c.execute(  
 **'''UPDATE characters SET name=%s, pol=%s, game=%s, storona=%s, status=%s WHERE ID=%s'''**,  
 (name, pol, game, storona, status, self.tree.set(self.tree.selection()[0], **"#1"**))  
 )  
 self.db.conn.commit()  
 self.view\_records()  
 **except** IndexError:  
 tkinter.messagebox.showinfo(title=**"Примечание"**, message=**"Для редактирования выберите нужную запись"**)  
  
 **def** view\_records(self):  
 self.db.c.execute(**"SELECT \* FROM characters"**)  
 [self.tree.delete(i) **for** i **in** self.tree.get\_children()]  
 [self.tree.insert(**""**, **"end"**, values=row) **for** row **in** self.db.c.fetchall()]  
  
 **def** delete\_records(self):  
 **for** selection\_item **in** self.tree.selection():  
 self.db.c.execute(**"DELETE FROM characters WHERE id=%s"**, (self.tree.set(selection\_item, **"#1"**),))  
 self.db.conn.commit()  
 self.view\_records()  
  
 **def** search\_records(self, name):  
 name = **"%"** + name + **"%"** self.db.c.execute(**"SELECT \* FROM characters WHERE name LIKE %s"**, (name,))  
 [self.tree.delete(i) **for** i **in** self.tree.get\_children()]  
 [self.tree.insert(**""**, **"end"**, values=row) **for** row **in** self.db.c.fetchall()]  
  
 **def** open\_dialog(self):  
 Child()  
  
 **def** open\_update\_dialog(self):  
 Update()  
  
 **def** open\_search\_dialog(self):  
 Search()  
  
**class** Child(tk.Toplevel):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_(root)  
 self.init\_child()  
 self.view = app  
  
 **def** init\_child(self):  
 self.title(**'Добавить персонажа'**)  
 self.geometry(**'400x220+400+300'**)  
 self.resizable(**False**, **False**)  
  
 label\_name = tk.Label(self, text=**'Полное имя:'**)  
 label\_name.place(x=50, y=20)  
 label\_select = tk.Label(self, text=**'Первое появление:'**)  
 label\_select.place(x=50, y=50)  
 label\_pol = tk.Label(self, text=**'Пол:'**)  
 label\_pol.place(x=50, y=80)  
 label\_pol = tk.Label(self, text=**'Сторона:'**)  
 label\_pol.place(x=50, y=110)  
 label\_status = tk.Label(self, text=**'Статус:'**)  
 label\_status.place(x=50, y=140)  
  
 self.entry\_name = ttk.Entry(self)  
 self.entry\_name.place(x=200, y=20)  
  
 self.combobox = ttk.Combobox(self,state=**"readonly"**, width=27,  
 values=[**u'Resident Evil (1996)'**,  
 **u'Resident Evil 2 (1998)'**,  
 **u'Resident Evil 3: Nemesis (1999)'**,  
 **u'Resident Evil Code: Veronica (2000)'**,  
 **u'Resident Evil Zero (2002)'**,  
 **u'Resident Evil 4 (2005)'**,  
 **u'Resident Evil 5 (2009)'**,  
 **u'Resident Evil 6 (2012)'**,  
 **u'Resident Evil 7: Biohazard (2017)'**,  
 **u'Resident Evil Village (2021)'**])  
 self.combobox.current(0)  
 self.combobox.place(x=200, y=50)  
  
 self.combobox\_pol = ttk.Combobox(self, state=**"readonly"**, width=25,  
 values=[**u'Мужской'**,  
 **u'Не определено'**,  
 **u'Женский'**])  
 self.combobox\_pol.current(0)  
 self.combobox\_pol.place(x=200, y=80)  
  
 self.combobox\_storona = ttk.Combobox(self, state=**"readonly"**, width=25,  
 values=[**u'Добро'**,  
 **u'Неизвестно'**,  
 **u'Зло'**])  
 self.combobox\_storona.current(0)  
 self.combobox\_storona.place(x=200, y=110)  
  
 self.combobox\_status = ttk.Combobox(self,state=**"readonly"**, width=25,  
 values=[**u'Жив'**,  
 **u'Неизвестно'**,  
 **u'Мёртв'**])  
 self.combobox\_status.current(0)  
 self.combobox\_status.place(x=200, y=140)  
  
 btn\_cancel = ttk.Button(self, text=**'Закрыть'**, command=self.destroy)  
 btn\_cancel.place(x=300, y=170)  
  
 self.btn\_ok = ttk.Button(self, text=**'Добавить'**)  
 self.btn\_ok.place(x=200, y=170)  
 self.btn\_ok.bind(**'<Button-1>'**, self.on\_button\_click)  
  
 **def** on\_button\_click(self, event):  
 *# проверка на пустые поля* **if** len(self.entry\_name.get()) == 0:  
 tkinter.messagebox.showerror(**"Ошибка"**, **"Заполните все поля!"**)  
 **return  
 else**:  
 *# выполнение действий при нажатии на кнопку* self.view.records(self.entry\_name.get(), self.combobox\_pol.get(), self.combobox.get(), self.combobox\_storona.get(), self.combobox\_status.get())  
 self.grab\_set()  
 self.focus\_set()  
 self.entry\_name.delete(0, len(self.entry\_name.get()))  
 tkinter.messagebox.showinfo(**"Добавление"**, **'Запись успешно добавлена'**)  
  
**class** Update(Child):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.init\_edit()  
 self.view = app  
 self.db = db  
 self.default\_data()  
  
 **def** init\_edit(self):  
 self.title(**'Редактировать позицию'**)  
 btn\_edit = ttk.Button(self, text=**'Редактировать'**)  
 btn\_edit.place(x=205, y=170)  
 btn\_edit.bind(**'<Button-1>'**, **lambda** event: self.view.update\_record(self.entry\_name.get(),  
 self.combobox\_pol.get(),  
 self.combobox.get(),  
 self.combobox\_storona.get(),  
 self.combobox\_status.get()))  
  
 self.btn\_ok.destroy()  
  
 **def** default\_data(self):  
 **try**:  
 self.db.c.execute(**"SELECT \* FROM characters WHERE id=%s"**,  
 (self.view.tree.set(self.view.tree.selection()[0], **"#1"**),))  
 row = self.db.c.fetchone()  
 self.entry\_name.insert(0, row[1])  
 self.combobox\_status.insert(0, row[3])  
 **except** IndexError:  
 tkinter.messagebox.showinfo(title=**"Примечание"**, message=**"Для редактирования выберите нужную запись"**)  
  
  
**class** Search(tk.Toplevel):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.init\_search()  
 self.view = app  
  
 **def** init\_search(self):  
 self.title(**'Поиск'**)  
 self.geometry(**'300x100+400+300'**)  
 self.resizable(**False**, **False**)  
  
 label\_search = tk.Label(self, text=**'Поиск'**)  
 label\_search.place(x=50, y=20)  
  
 self.entry\_search = ttk.Entry(self)  
 self.entry\_search.place(x=105, y=20, width=150)  
  
 btn\_cancel = ttk.Button(self, text=**'Закрыть'**, command=self.destroy)  
 btn\_cancel.place(x=185, y=50)  
  
 btn\_search = ttk.Button(self, text=**'Поиск'**)  
 btn\_search.place(x=105, y=50)  
 btn\_search.bind(**'<Button-1>'**, **lambda** event: self.view.search\_records(self.entry\_search.get()))  
 btn\_search.bind(**'<Button-1>'**, **lambda** event: self.destroy(), add=**'+'**)  
  
**class** DB:  
 **def** \_\_init\_\_(self):  
 self.conn = psycopg2.connect(dbname=**"postgres"**, user=**"postgres"**, password=**"TOTnot0671M"**, host=**"localhost"**)  
 self.c = self.conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)  
 self.c.execute(  
 **'''CREATE TABLE IF NOT EXISTS characters  
 (id serial primary key, name text, pol text, game text, storona text, status text);'''** )  
 self.conn.commit()  
  
 **def** insert\_data(self, name, pol, game, storona, status):  
 self.c.execute(  
 **'''INSERT INTO characters(name, pol, game, storona, status) VALUES (%s, %s, %s, %s, %s)'''**,  
 (name, pol, game, storona, status)  
 )  
 self.conn.commit()  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 root = tk.Tk()  
 db = DB()  
 app = Main(root)  
 app.pack()  
 root.title(**"Персонажи серии игр Resident Evil"**)  
 root.iconbitmap(**'Icon.ico'**)  
 root.geometry(**"905x450+430+200"**)  
 root.minsize(905, 450)  
 root.resizable(**True**, **True**)  
 root.mainloop()

**Результат выполнения программы**

Работы реализована на Python с использованием графической библиотеки Tkinter.

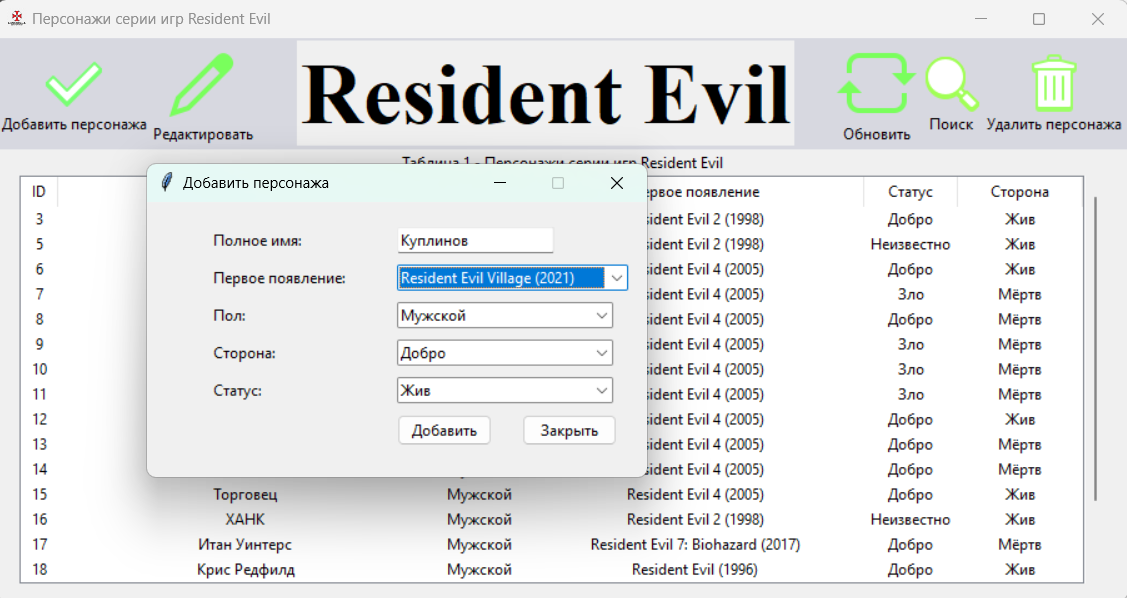
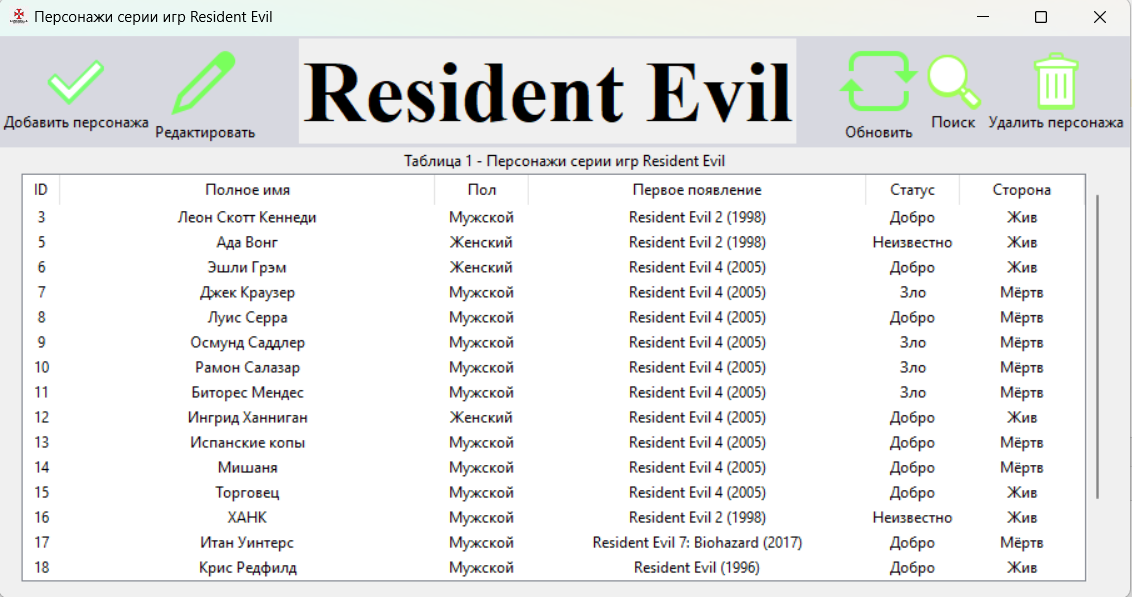
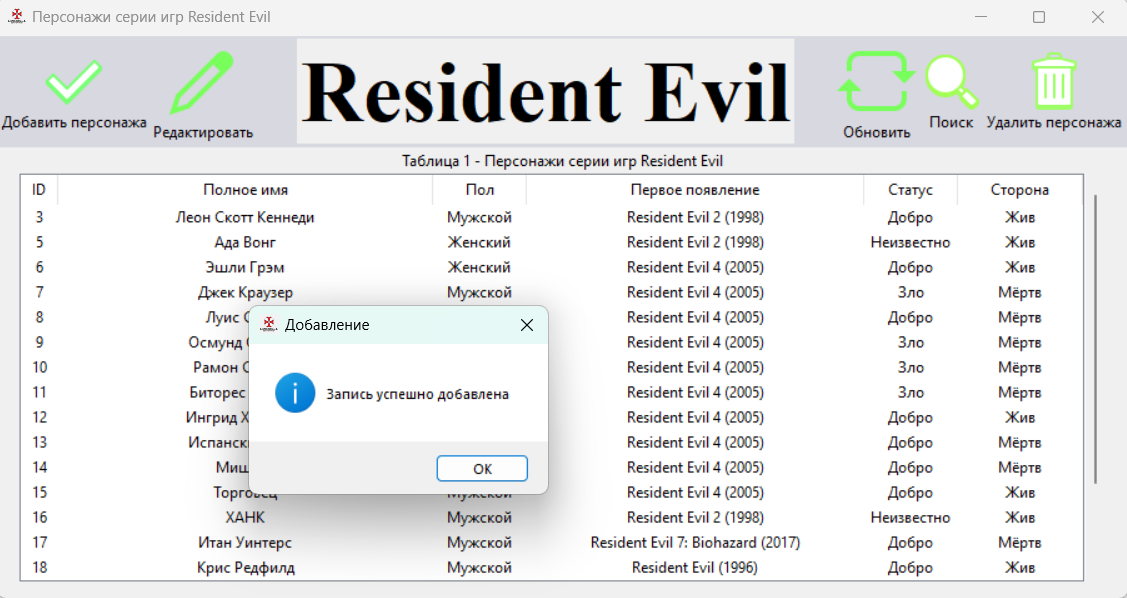


Рисунок 3 – Пользовательский интерфейс

Рисунок 4 – Добавление персонажа в таблицу

****

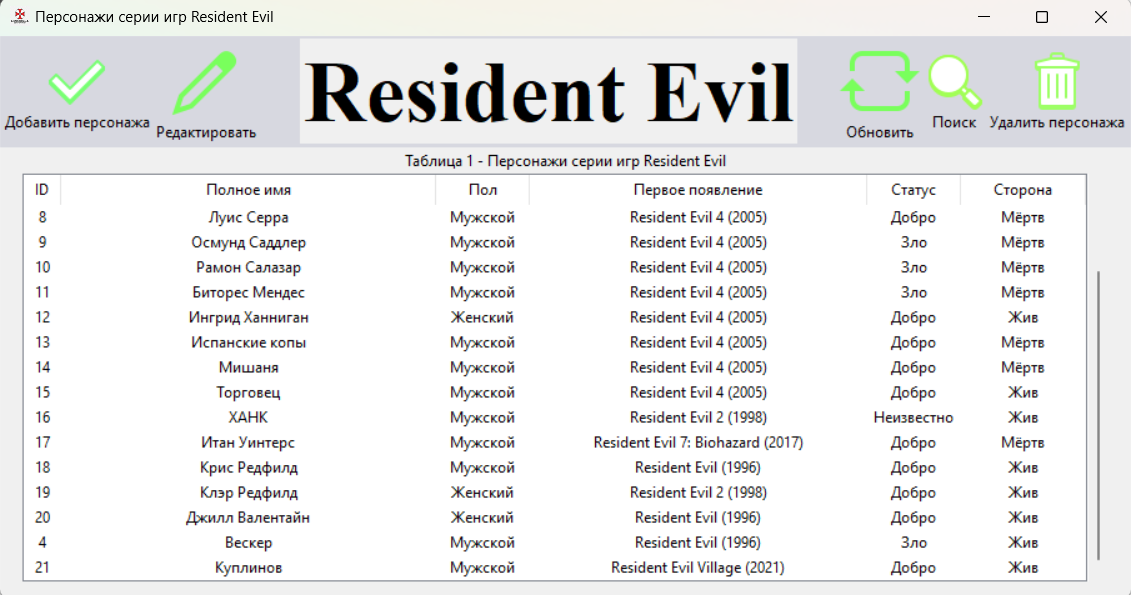
Рисунок 5 - Добавление персонажа в таблицу

Рисунок 6 - Персонаж добавлен

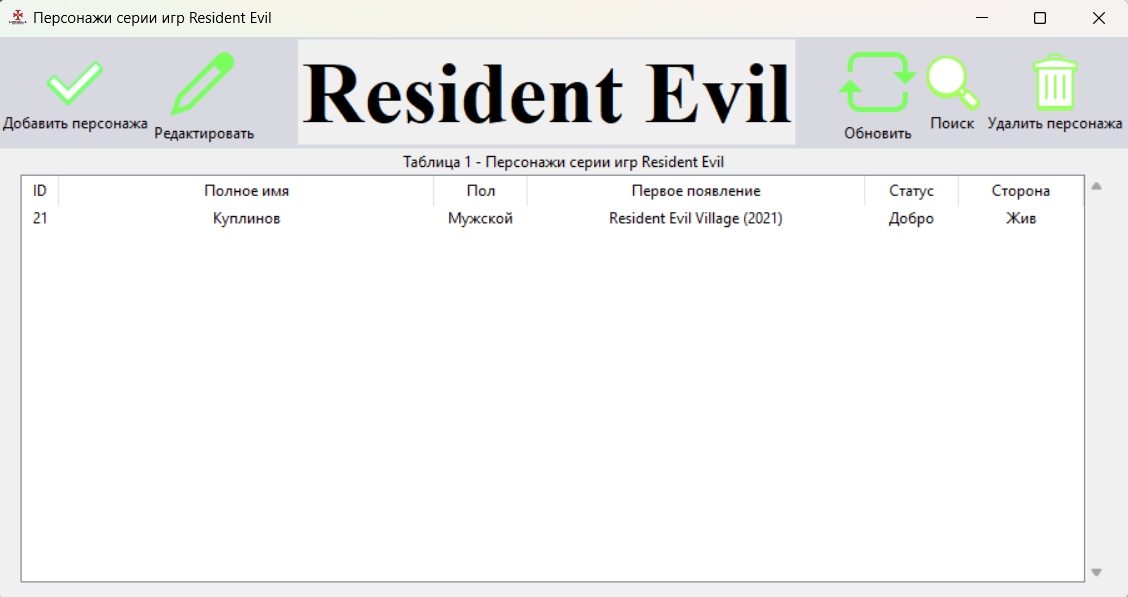
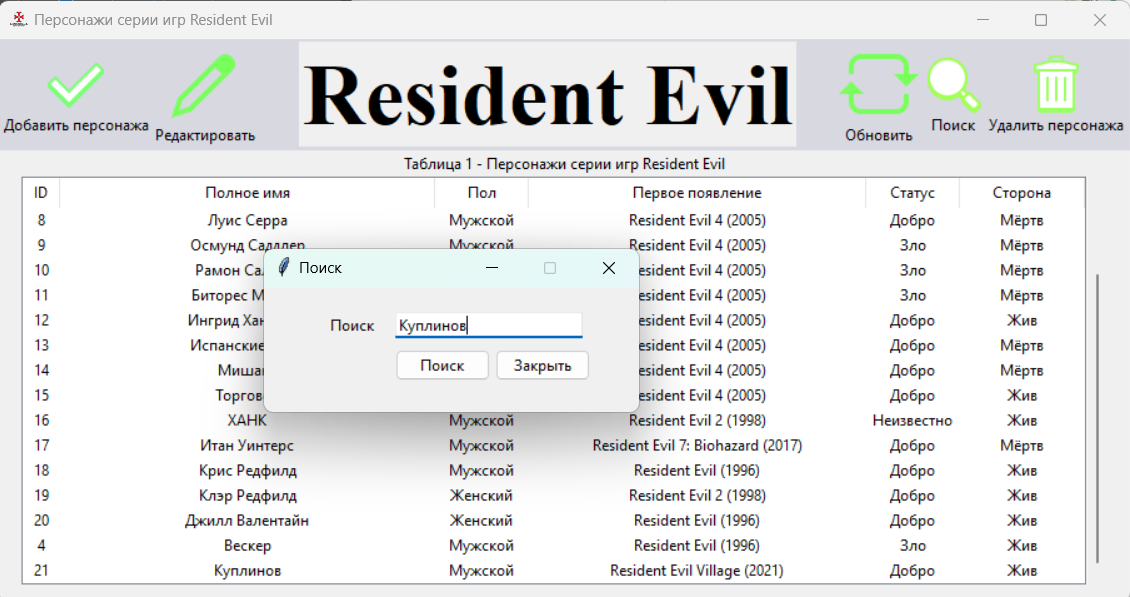
****

Рисунок 7 - Поиск добавленного персонажа

Рисунок 8 - Поиск произведён успешно

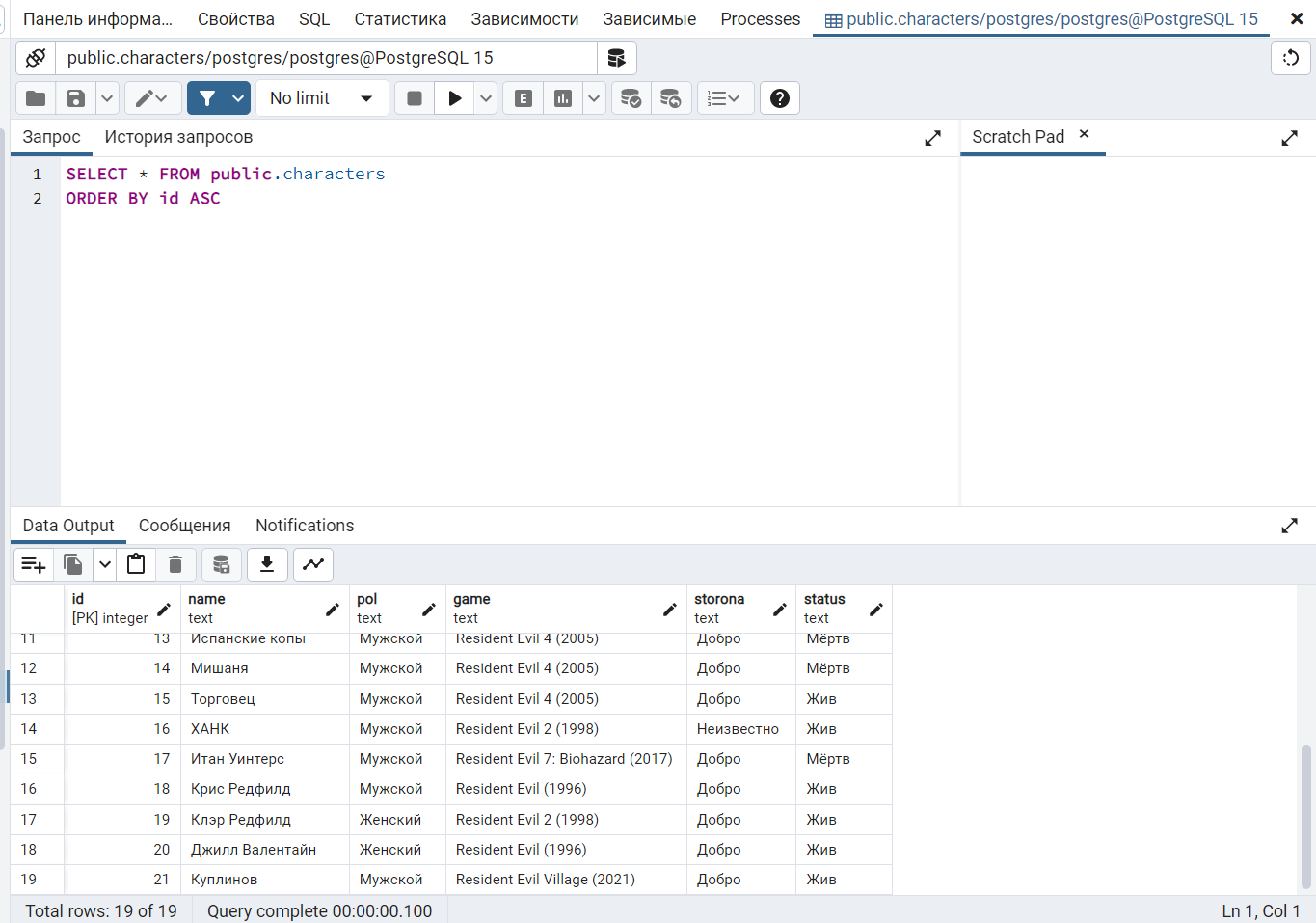
****

Рисунок 9 - Отображение добавленной записи в postgresql

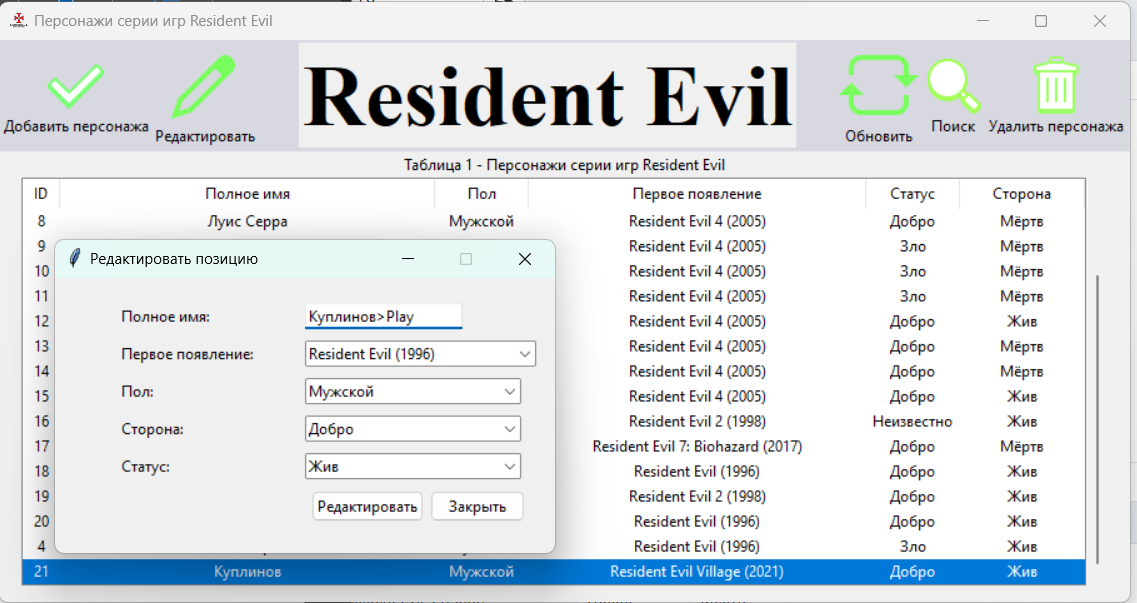
****

Рисунок 10 - Редактирование записи

**Вывод:**

Исходя из общего положения и пользуясь в качестве довода и логического обоснования совокупность ранее упомянутых эмпирических знаний можно сделать закономерный вывод, что в ходе работы мы использовали знания, полученные нами на столь замечательной и бесподобной дисциплине, как Разработка кода информационных систем. После домашней контрольной работы №9 мы можем смело сказать, что полученные на лекционных занятиях знания и умения в полной мере пригодились для грамотного и быстрого выполнения данной работы. Исходя из этого можно смело утверждать, что благодаря ДКР №9 возможно стать настоящим Python-разработчиком.

В ходе выполнения домашней контрольной работы №9 проблем не возникло.

Подводя итоги, можно смело утверждать, что было получено представление об организации и использовании информации посредством типизированных файлов, закреплены навыки создания пользовательского интерфейса.