**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

**(национальный исследовательский университет)»**

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 311 «Прикладные программные средства и математические методы»

**ЛАБОРАТОРНАЯ РАБОТА №1**

**по дисциплине:**

**«Проектирование информационных систем»**

Выполнил:

Студент гр. М3О-216Б-21

Лошаков Иван Евгеньевич

Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель:

к.т.н. доцент

Смирнов Владимир Юрьевич

Подпись руководителя:\_\_\_\_\_\_\_\_\_\_

**Содержание**

[Цель лабораторной работы 3](#_Toc130644366)

[Создание sql базы данных 6](#_Toc130644367)

[Используемое ПО и схема БД 7](#_Toc130644368)

[Вывод 28](#_Toc130644369)

[Список литературы 29](#_Toc130644370)

# Цель лабораторной работы

Разработать согласно требованиям и реализовать интерфейс автоматизированного рабочего места оператора по продажам для выполнения следующих действий:

1. Открыть базу клиентов
2. Ввод имени клиента, при вводе имени осуществляется поиск по списку клиентов, и если такой находится, то вывести:

* общий счет покупок клиента (сумма всех покупок)
* текущий счет клиента (деньги на счету фирмы для покупок);
* потолок кредита (кредитный лимит, который не изменяется);
* текущий долг клиента (;
* остаток кредита (разница между потолком кредита и текущим долгом);
* комментарий (о причине долга клиента, его надежности и т.п.).

1. Открыть базу товаров
2. Создать новый заказ

*Нужна таблица склада товаров.*

1. Выбор вида продажи:

* наличный расчет (увеличивается общий счет покупок клиента, остальные счета не изменяются, количество товаров уменьшается);
* безналичный расчет (увеличивается общий счет покупок клиента, уменьшается текущий счет клиента, остальные счета не изменяются, количество товаров уменьшается);
* кредит (увеличивается общий счет покупок клиента, увеличивается текущий долг клиента, уменьшается остаток кредита, текущий счет клиента в этом случае нулевой или недостаточный для покупки (во втором случае он должен быть учтен при покупке и обнулится), количество товаров уменьшается; при приближении текущего долга клиента к потолку кредита (90% от потолка кредита) подсвечивается красным текущий долг и остаток кредита; количество товаров уменьшается)
* бартер, т.е. обмен товаров без использования денег (никакие счета не изменяются, на складе количество одних товаров должно увеличиться, количество других уменьшиться, стоимость товаров, количество которых увеличилось, должно быть равно стоимости товаров, количество которых уменьшилось);
* взаимозачет (уменьшается текущий долг клиента, увеличивается остаток кредита, остальные счета не изменяются, количество товаров увеличивается).

1. Открыть базу товаров.
2. В новом заказе в форме в разделе таблица выбор товара из перечня, а также ввод количество товаров, которые пользователь заказывает.
3. Ввод перечня товаров и продаваемого количества в табличной части, с автоматическим указанием:

* цены (из списка товаров); -?
* суммы за единицу; -? Сумма за единицу заказа, сумма всего заказа, состоящего из списка товаров, общая цена за список товаров.
* выводом текущего остатка.
* Если введенное количество товаров меньше текущего остатка или равно ему, то цвет шрифта остатка должен быть одним (например, зеленым).
* Если введенное количество товаров больше текущего остатка, то цвет шрифта остатка должен быть другим (например, красным), а также должно сработать звуковое предупреждение. (в случае наличия устройства воспроизведения звука)

1. Автоматический подсчет общей суммы купленных товаров для каждого клиента. Запрос на формирование таблицы, состоящей из полей таблиц, а также из дополнительного вычисляемого поля, где подсчитана общая сумма для каждого заказа клиента. Построение отчета на основе данной таблицы.
2. Формирование предупреждения, если общая сумма купленных товаров для клиента превышает потолок его кредита за минусом текущего долга клиента. При формировании нового заказа в случае, если выбрать способ покупки с помощью кредита.
3. Формирование предупреждения если есть строки с незаполненным товаром, нулевой ценой и нулевым количеством. Данное предупреждение реализовывается в разделе заполнения базы товаров. Открыть базу товаров.

# Создание sql базы данных

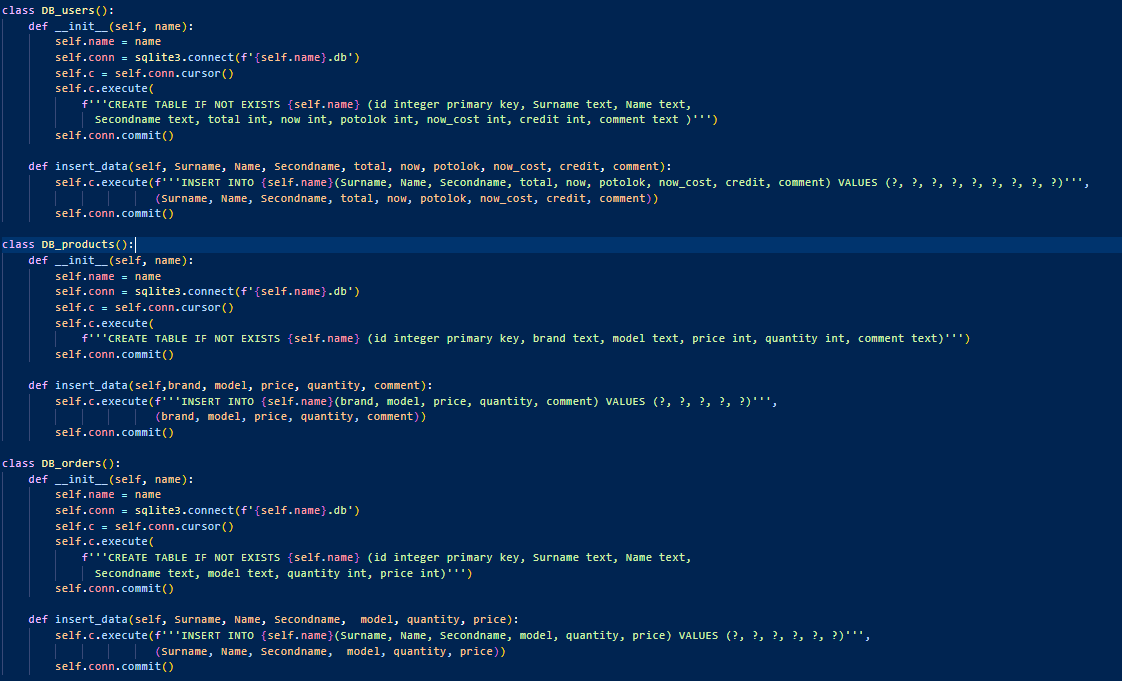


Рисунок 1.1 – Создание базы данных

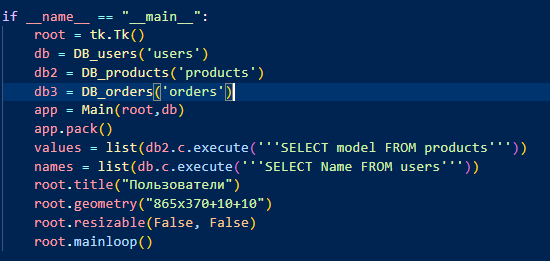


Рисунок 1.2 – Создание главного окна интерфейса

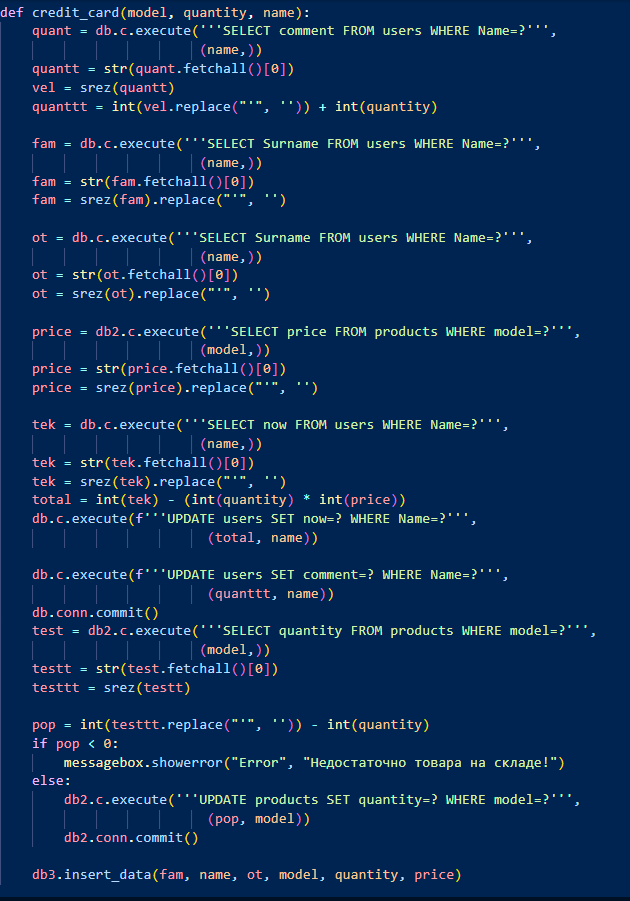
# Используемое ПО и схема БД

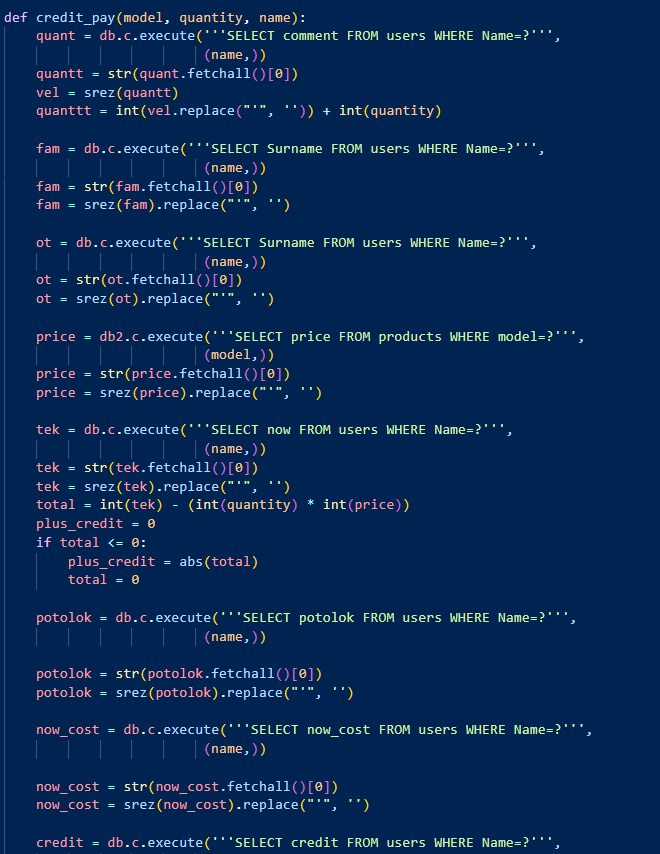
Интерфейс написан на python с использованием модуля tkinter.

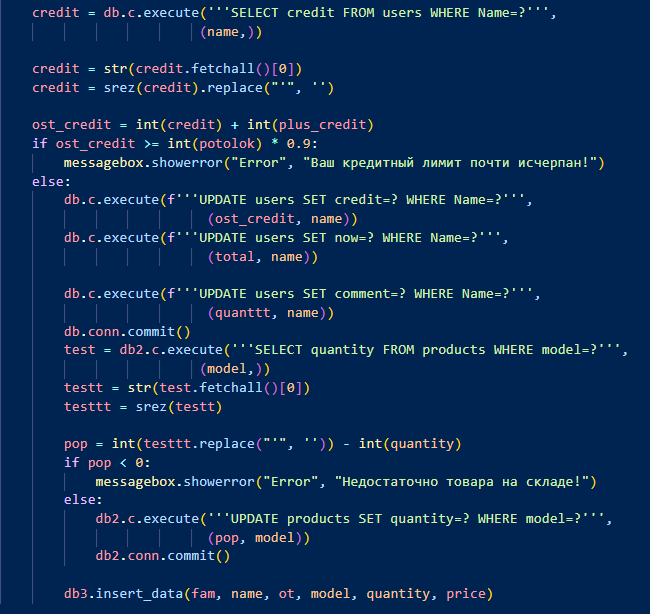
Используемая база данных – Встроенная в python библиотека sqlite3

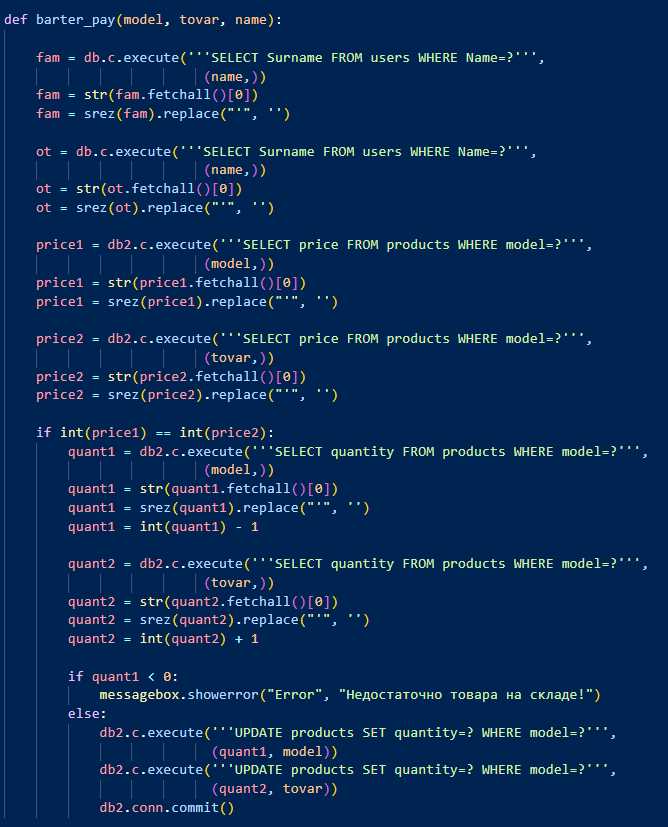
* Участок кода для создания интерфейса программы:
* class Main(tk.Frame):
* def \_\_init\_\_(self, root, db):
* super().\_\_init\_\_(root)
* self.init\_main()
* self.db = db
* self.view\_records()
* def init\_main(self):
* toolbar = tk.Frame(bg='#d7d8e0', bd=2)
* toolbar.pack(side=tk.TOP, fill=tk.X)
* btn\_open\_dialog = tk.Button(toolbar, text='Добавить позицию', command=self.open\_dialog, bg='#d7d8e0', bd=0,
* compound=tk.TOP)
* btn\_open\_dialog.pack(side=tk.LEFT)
* btn\_edit\_dialog = tk.Button(toolbar, text='Редактировать', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.open\_update\_dialog)
* btn\_edit\_dialog.pack(side=tk.LEFT)
* btn\_delete = tk.Button(toolbar, text='Удалить позицию', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.delete\_records)
* btn\_delete.pack(side=tk.LEFT)
* btn\_search = tk.Button(toolbar, text='Поиск', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.open\_search\_dialog)
* btn\_search.pack(side=tk.LEFT)
* btn\_refresh = tk.Button(toolbar, text='Обновить', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.view\_records)
* btn\_refresh.pack(side=tk.LEFT)
* btn\_create = tk.Button(toolbar, text='Открыть таблицу товаров', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.create\_second)
* btn\_create.pack(side=tk.LEFT)
* self.tree = ttk.Treeview(self, columns=('ID', 'Surname', 'Name', 'Secondname', 'total', 'now', 'potolok', 'now\_cost', 'credit', 'comment'), height=15, show='headings')
* self.tree.column('ID', width=30, anchor=tk.CENTER)
* self.tree.column('Surname', width=120, anchor=tk.CENTER)
* self.tree.column('Name', width=100, anchor=tk.CENTER)
* self.tree.column('Secondname', width=100, anchor=tk.CENTER)
* self.tree.column('total', width=80, anchor=tk.CENTER)
* self.tree.column('now', width=80, anchor=tk.CENTER)
* self.tree.column('potolok', width=80, anchor=tk.CENTER)
* self.tree.column('now\_cost', width=80, anchor=tk.CENTER)
* self.tree.column('credit', width=80, anchor=tk.CENTER)
* self.tree.column('comment', width=80, anchor=tk.CENTER)
* self.tree.heading('ID', text='ID')
* self.tree.heading('Surname', text='Фамилия')
* self.tree.heading('Name', text='Имя')
* self.tree.heading('Secondname', text='Отчество')
* self.tree.heading('total', text='Общий счет')
* self.tree.heading('now', text='Текущий счет')
* self.tree.heading('potolok', text='Потолок кредита')
* self.tree.heading('now\_cost', text='Текущий долг')
* self.tree.heading('credit', text='Остаток кредита')
* self.tree.heading('comment', text='Кол-во покупок')
* 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, Surname, Name, Secondname, total, now, potolok, now\_cost, credit, comment):
* self.db.insert\_data(Surname, Name, Secondname, total, now, potolok, now\_cost, credit, comment)
* self.view\_records()
* names.append(Name)
* def update\_record(self, Surname, Name, Secondname, total, now, potolok, now\_cost, credit, comment):
* self.db.c.execute(f'''UPDATE {self.db.name} SET Surname=?, Name=?, Secondname=?, total=?, now=?, potolok=?, now\_cost=?, credit=?, comment=? WHERE ID=?''',
* (Surname, Name, Secondname, total, now, potolok, now\_cost, credit, comment, self.tree.set(self.tree.selection()[0], '#1')))
* self.db.conn.commit()
* self.view\_records()
* def view\_records(self):
* self.db.c.execute(f'''SELECT \* FROM {self.db.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()]
* self.tree.get\_children()
* def delete\_records(self):
* for selection\_item in self.tree.selection():
* self.db.c.execute(f'''DELETE FROM {self.db.name} WHERE id=?''', (self.tree.set(selection\_item, '#1'),))
* self.db.conn.commit()
* self.view\_records()
* def search\_records(self, Name):
* Name = ('%' + "%" + Name + '%' + '%' + '%' + '%' + '%' + '%' + '%', )
* self.db.c.execute(f'''SELECT \* FROM {self.db.name} WHERE Name LIKE ?''',  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(self)
* def open\_update\_dialog(self):
* Update(self)
* def open\_search\_dialog(self):
* Search(self)
* def create\_second(self):
* window = Products(db2, self)
* window.title('Товары')
* window.geometry('665x370+870+10')
* window.resizable(False, False)
* class Child(tk.Toplevel):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_(master)
* self.init\_child()
* self.view = master
* def init\_child(self):
* self.title('Добавить пользователя')
* self.geometry('400x360+90+150')
* self.resizable(False, False)
* label\_description = tk.Label(self, text='Фамилия:')
* label\_description.place(x=50, y=50)
* label\_select = tk.Label(self, text='Имя:')
* label\_select.place(x=50, y=80)
* label\_sum = tk.Label(self, text='Отчество:')
* label\_sum.place(x=50, y=110)
* label\_description = tk.Label(self, text='Общий счет:')
* label\_description.place(x=50, y=140)
* label\_select = tk.Label(self, text='Текущий счет:')
* label\_select.place(x=50, y=170)
* label\_sum = tk.Label(self, text='Потолок кредита:')
* label\_sum.place(x=50, y=200)
* label\_description = tk.Label(self, text='Текущий долг:')
* label\_description.place(x=50, y=230)
* label\_select = tk.Label(self, text='Остаток кредита:')
* label\_select.place(x=50, y=260)
* label\_sum = tk.Label(self, text='Кол-во покупок:')
* label\_sum.place(x=50, y=290)
* self.entry\_fam = ttk.Entry(self)
* self.entry\_fam.place(x=200, y=50)
* self.entry\_name = ttk.Entry(self)
* self.entry\_name.place(x=200, y=80)
* self.entry\_ot = ttk.Entry(self)
* self.entry\_ot.place(x=200, y=110)
* self.entry\_score = ttk.Entry(self)
* self.entry\_score.place(x=200, y=140)
* self.entry\_now = ttk.Entry(self)
* self.entry\_now.place(x=200, y=170)
* self.entry\_pot = ttk.Entry(self)
* self.entry\_pot.place(x=200, y=200)
* self.entry\_tek = ttk.Entry(self)
* self.entry\_tek.place(x=200, y=230)
* self.entry\_ost = ttk.Entry(self)
* self.entry\_ost.place(x=200, y=260)
* self.entry\_com = ttk.Entry(self)
* self.entry\_com.place(x=200, y=290)
* btn\_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)
* btn\_cancel.place(x=300, y=320)
* self.btn\_ok = ttk.Button(self, text='Добавить')
* self.btn\_ok.place(x=220, y=320)
* self.btn\_ok.bind('<Button-1>', lambda event: self.view.records(self.entry\_fam.get(),
* self.entry\_name.get(),
* self.entry\_ot.get(),
* self.entry\_score.get(),
* self.entry\_now.get(),
* self.entry\_pot.get(),
* self.entry\_tek.get(),
* self.entry\_ost.get(),
* self.entry\_com.get()) if len(self.entry\_com.get()) != 0 and len(self.entry\_fam.get()) != 0 and
* len(self.entry\_name.get()) != 0 and len(self.entry\_tek.get()) != 0 and
* len(self.entry\_ost.get()) != 0 and len(self.entry\_score.get()) != 0 and
* len(self.entry\_ot.get()) != 0 and len(self.entry\_pot.get()) != 0 and
* len(self.entry\_ost.get()) != 0 and
* self.entry\_now.get().isdigit() and self.entry\_score.get().isdigit() and self.entry\_pot.get().isdigit() and self.entry\_tek.get().isdigit() and self.entry\_ost.get().isdigit()
* else messagebox.showerror("Error", "Данные введены неверно!"))
* self.grab\_set()
* self.focus\_set()
* class Update(Child):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_(master)
* self.init\_edit()
* self.view = master
* def init\_edit(self):
* self.title('Редактировать позицию')
* btn\_edit = ttk.Button(self, text='Редактировать')
* btn\_edit.place(x=205, y=320)
* btn\_edit.bind('<Button-1>', lambda event: self.view.update\_record(self.entry\_fam.get(),
* self.entry\_name.get(),
* self.entry\_ot.get(),
* self.entry\_score.get(),
* self.entry\_now.get(),
* self.entry\_pot.get(),
* self.entry\_tek.get(),
* self.entry\_ost.get(),
* self.entry\_com.get()) if len(self.entry\_com.get()) != 0 and len(self.entry\_fam.get()) != 0 and
* len(self.entry\_name.get()) != 0 and len(self.entry\_tek.get()) != 0 and
* len(self.entry\_ost.get()) != 0 and len(self.entry\_score.get()) != 0 and
* len(self.entry\_ot.get()) != 0 and len(self.entry\_pot.get()) != 0 and
* len(self.entry\_ost.get()) != 0 and
* self.entry\_now.get().isdigit() and self.entry\_score.get().isdigit() and self.entry\_pot.get().isdigit() and
* self.entry\_tek.get().isdigit() and self.entry\_ost.get().isdigit() and self.entry\_com.get().isdigit()
* else messagebox.showerror("Error", "Данные введены неверно!"))
* self.btn\_ok.destroy()
* class Search(tk.Toplevel):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_()
* self.init\_search()
* self.view = master
* def init\_search(self):
* self.title('Поиск')
* self.geometry('300x100+90+150')
* 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 Products(tk.Toplevel):
* def \_\_init\_\_(self,db, master):
* super().\_\_init\_\_(master)
* self.init\_main()
* self.db = db
* self.view\_records()
* def init\_main(self):
* toolbar1 = tk.Frame(bg='#d7d8e0', bd=2, master=self)
* toolbar1.pack(side=tk.TOP, fill=tk.X)
* btn\_open\_dialog = tk.Button(toolbar1, text='Добавить позицию', command=self.open\_dialog1, bg='#d7d8e0', bd=0,
* compound=tk.TOP)
* btn\_open\_dialog.pack(side=tk.LEFT)
* btn\_edit\_dialog = tk.Button(toolbar1, text='Редактировать', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.open\_update\_dialog)
* btn\_edit\_dialog.pack(side=tk.LEFT)
* btn\_delete = tk.Button(toolbar1, text='Удалить позицию', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.delete\_records)
* btn\_delete.pack(side=tk.LEFT)
* btn\_search = tk.Button(toolbar1, text='Поиск', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.open\_search\_dialog)
* btn\_search.pack(side=tk.LEFT)
* btn\_refresh = tk.Button(toolbar1, text='Обновить', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.view\_records)
* btn\_refresh.pack(side=tk.LEFT)
* btn\_refresh = tk.Button(toolbar1, text='Заказать', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.open\_order)
* btn\_refresh.pack(side=tk.LEFT)
* btn\_create = tk.Button(toolbar1, text='Открыть таблицу заказов', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.create\_second)
* btn\_create.pack(side=tk.LEFT)
* self.tree = ttk.Treeview(self, columns=('ID', 'brand', 'model', 'price', 'quantity', 'comment'), height=15, show='headings')
* self.tree.column('ID', width=30, anchor=tk.CENTER)
* self.tree.column('brand', width=110, anchor=tk.CENTER)
* self.tree.column('model', width=100, anchor=tk.CENTER)
* self.tree.column('price', width=100, anchor=tk.CENTER)
* self.tree.column('quantity', width=100, anchor=tk.CENTER)
* self.tree.column('comment', width=200, anchor=tk.CENTER)
* self.tree.heading('ID', text='ID')
* self.tree.heading('brand', text='Бренд')
* self.tree.heading('model', text='Модель')
* self.tree.heading('price', text='Цена за 1 шт')
* self.tree.heading('quantity', text='Количество на складе')
* self.tree.heading('comment', text='Комментарий')
* 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, brand, model, price, quantity, comment):
* self.db.insert\_data(brand, model, price,quantity, comment)
* values.append(model)
* self.view\_records()
* def update\_record(self, brand, model, price, quantity, comment):
* self.db.c.execute(f'''UPDATE {self.db.name} SET brand=?,model=?, price=?, quantity=?, comment=? WHERE ID=?''',
* (brand, model, price, quantity, comment, self.tree.set(self.tree.selection()[0], '#1')))
* self.db.conn.commit()
* self.view\_records()
* def view\_records(self):
* self.db.c.execute(f'''SELECT \* FROM {self.db.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 delete\_records(self):
* for selection\_item in self.tree.selection():
* self.db.c.execute(f'''DELETE FROM {self.db.name} WHERE id=?''', (self.tree.set(selection\_item, '#1'),))
* self.db.conn.commit()
* self.view\_records()
* def search\_records(self, model):
* model = ('%' + '%' + model + '%' + '%' + '%',)
* self.db.c.execute(f'''SELECT \* FROM {self.db.name} WHERE model LIKE ?''', model)
* [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 create\_second(self):
* window = Orders(db3, self)
* window.title('Заказы')
* window.geometry('665x370+50+400')
* window.resizable(False, False)
* window.grab\_set()
* window.focus\_set()
* def open\_dialog1(self):
* Child\_Sec(self)
* def open\_update\_dialog(self):
* Update\_Sec(self)
* def open\_search\_dialog(self):
* Search\_Sec(self)
* def open\_order(self):
* Order(self)
* class Child\_Sec(tk.Toplevel):
* def \_\_init\_\_(self,master):
* super().\_\_init\_\_(master)
* self.init\_child()
* def init\_child(self):
* self.title('Добавить товар')
* self.geometry('400x240+950+100')
* self.resizable(False, False)
* label\_brand = tk.Label(self, text='Фирма:')
* label\_brand.place(x=50, y=50)
* label\_model = tk.Label(self, text='Модель:')
* label\_model.place(x=50, y=80)
* label\_sum = tk.Label(self, text='Цена за 1 шт:')
* label\_sum.place(x=50, y=110)
* label\_quantity = tk.Label(self, text='Количесто на складе:')
* label\_quantity.place(x=50, y=140)
* label\_comm = tk.Label(self, text='Комментарий:')
* label\_comm.place(x=50, y=170)
* self.entry\_brand = ttk.Entry(self)
* self.entry\_brand.place(x=200, y=50)
* self.entry\_model = ttk.Entry(self)
* self.entry\_model.place(x=200, y=80)
* self.entry\_sum = ttk.Entry(self)
* self.entry\_sum.place(x=200, y=110)
* self.entry\_quantity = ttk.Entry(self)
* self.entry\_quantity.place(x=200, y=140)
* self.entry\_comm = ttk.Entry(self)
* self.entry\_comm.place(x=200, y=170)
* btn\_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)
* btn\_cancel.place(x=300, y=200)
* self.btn\_ok = ttk.Button(self, text='Добавить')
* self.btn\_ok.place(x=220, y=200)
* self.btn\_ok.bind('<Button-1>', lambda event: self.master.records(self.entry\_brand.get(),
* self.entry\_model.get(),
* self.entry\_sum.get(),
* self.entry\_quantity.get(),
* self.entry\_comm.get())  if len(self.entry\_brand.get()) != 0 and len(self.entry\_model.get()) != 0 and
* len(self.entry\_sum.get()) != 0 and len(self.entry\_quantity.get()) != 0 and
* len(self.entry\_comm.get()) != 0 and
* self.entry\_sum.get().isdigit() and self.entry\_quantity.get().isdigit()
* else messagebox.showerror("Error", "Данные введены неверно!"))
* self.grab\_set()
* self.focus\_set()
* class Update\_Sec(Child\_Sec):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_(master)
* self.init\_edit()
* self.view = master
* def init\_edit(self):
* self.title('Редактировать позицию')
* btn\_edit = ttk.Button(self, text='Редактировать')
* btn\_edit.place(x=205, y=200)
* btn\_edit.bind('<Button-1>', lambda event: self.view.update\_record(self.entry\_brand.get(),
* self.entry\_model.get(),
* self.entry\_sum.get(),
* self.entry\_quantity.get(),
* self.entry\_comm.get()) if len(self.entry\_brand.get()) != 0 and len(self.entry\_model.get()) != 0 and
* len(self.entry\_sum.get()) != 0 and len(self.entry\_quantity.get()) != 0 and
* len(self.entry\_comm.get()) != 0 and
* self.entry\_sum.get().isdigit() and self.entry\_quantity.get().isdigit()
* else messagebox.showerror("Error", "Данные введены неверно!"))
* self.btn\_ok.destroy()
* class Search\_Sec(tk.Toplevel):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_(master)
* self.init\_search()
* self.view = master
* def init\_search(self):
* self.title('Поиск')
* self.geometry('300x100+950+100')
* 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 Order(tk.Toplevel):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_(master)
* self.init\_order()
* self.view = master
* def init\_order(self):
* self.title('Заказ')
* self.geometry('200x120+900+450')
* self.resizable(False, False)
* choise = ['money', 'credit card', 'credit', 'barter', 'netting']
* self.combobox = ttk.Combobox(self, values=choise)
* self.combobox.current(0)
* self.combobox.place(x = 30, y = 40)
* btn\_order = ttk.Button(master = self, text='Далее')
* btn\_order.place(x=30, y=80)
* btn\_order.bind('<Button-1>', lambda event: self.checkcmbo())
* def checkcmbo(self):
* if self.combobox.get() == 'money':
* money = Calculation\_money(self)
* money.btn\_ok.bind('<Button-1>', lambda event: calc\_money(money.combobox1.get(), money.entry\_quant.get(),  money.combobox2.get()))
* elif self.combobox.get() == "credit card":
* card = Calculation\_money(self)
* card.btn\_ok.bind('<Button-1>', lambda event: credit\_card(card.combobox1.get(), card.entry\_quant.get(),  card.combobox2.get()))
* elif self.combobox.get() == "credit":
* credit = Calculation\_money(self)
* credit.btn\_ok.bind('<Button-1>', lambda event: credit\_pay(credit.combobox1.get(), credit.entry\_quant.get(),  credit.combobox2.get()))
* elif self.combobox.get() == "barter":
* barter = Calculation\_barter(self)
* barter.btn\_ok.bind('<Button-1>', lambda event: barter\_pay(barter.combobox1.get(), barter.combobox.get(),  barter.combobox2.get()))
* elif self.combobox.get() == "netting":
* netting = Calculation\_money(self)
* netting.btn\_ok.bind('<Button-1>', lambda event: netting\_pay(netting.combobox1.get(), netting.entry\_quant.get(),  netting.combobox2.get()))
* class Calculation(tk.Toplevel):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_(master)
* self.init\_calc()
* def init\_calc(self):
* self.title('Добавить товар')
* self.geometry('400x220+1100+450')
* self.resizable(False, False)
* label\_model = tk.Label(self, text='Товар:')
* label\_model.place(x=100, y=50)
* label\_quant = tk.Label(self, text='Пользователь:')
* label\_quant.place(x=100, y=110)
* self.combobox1 = ttk.Combobox(self, values=values)
* self.combobox1.current(0)
* self.combobox1.place(x = 200, y = 50)
* self.combobox2 = ttk.Combobox(self, values=names)
* self.combobox2.current(0)
* self.combobox2.place(x = 200, y = 110)
* btn\_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)
* btn\_cancel.place(x=300, y=180)
* self.btn\_ok = ttk.Button(self, text='Добавить')
* self.btn\_ok.place(x=220, y=180)
* class Calculation\_money(Calculation):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_(master)
* self.init\_money()
* self.view = master
* def init\_money(self):
* label\_quant = tk.Label(self, text='Количество:')
* label\_quant.place(x=100, y=80)
* self.entry\_quant = ttk.Entry(self)
* self.entry\_quant.place(x=200, y=80)
* class Calculation\_barter(Calculation):
* def \_\_init\_\_(self, master):
* super().\_\_init\_\_(master)
* self.init\_money()
* self.view = master
* def init\_money(self):
* label\_quant = tk.Label(self, text='Ваш товар:')
* label\_quant.place(x=100, y=80)
* self.combobox = ttk.Combobox(self, values=values)
* self.combobox.current(0)
* self.combobox.place(x = 200, y = 80)
* class Orders(tk.Toplevel):
* def \_\_init\_\_(self,db, master):
* super().\_\_init\_\_(master)
* self.init\_main()
* self.db = db
* self.view\_records()
* def init\_main(self):
* toolbar1 = tk.Frame(bg='#d7d8e0', bd=2, master=self)
* toolbar1.pack(side=tk.TOP, fill=tk.X)
* btn\_search = tk.Button(toolbar1, text='Поиск', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.open\_search\_dialog)
* btn\_search.pack(side=tk.LEFT)
* btn\_refresh = tk.Button(toolbar1, text='Обновить', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.view\_records)
* btn\_refresh.pack(side=tk.LEFT)
* btn\_delete = tk.Button(toolbar1, text='Удалить позицию', bg='#d7d8e0', bd=0,
* compound=tk.TOP, command=self.delete\_records)
* btn\_delete.pack(side=tk.LEFT)
* self.tree = ttk.Treeview(self, columns=('ID', 'Surname', 'Name', 'Secondname',  'model', 'quantity', 'price'), height=15, show='headings')
* self.tree.column('ID', width=30, anchor=tk.CENTER)
* self.tree.column('Surname', width=130, anchor=tk.CENTER)
* self.tree.column('Name', width=100, anchor=tk.CENTER)
* self.tree.column('Secondname', width=130, anchor=tk.CENTER)
* self.tree.column('model', width=90, anchor=tk.CENTER)
* self.tree.column('quantity', width=80, anchor=tk.CENTER)
* self.tree.column('price', width=80, anchor=tk.CENTER)
* self.tree.heading('ID', text='ID')
* self.tree.heading('Surname', text='Фамилия')
* self.tree.heading('Name', text='Имя')
* self.tree.heading('Secondname', text='Отчество')
* self.tree.heading('model', text='Модель')
* self.tree.heading('quantity', text='Количество')
* self.tree.heading('price', text='Цена')
* 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 view\_records(self):
* self.db.c.execute(f'''SELECT \* FROM {self.db.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 search\_records(self, Name):
* Name = ('%' + '%' + Name + '%' + '%' + '%',)
* self.db.c.execute(f'''SELECT \* FROM {self.db.name} WHERE Name LIKE ?''', 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 delete\_records(self):
* for selection\_item in self.tree.selection():
* self.db.c.execute(f'''DELETE FROM {self.db.name} WHERE id=?''', (self.tree.set(selection\_item, '#1'),))
* self.db.conn.commit()
* self.view\_records()
* def open\_search\_dialog(self):
* Search\_Orders(self)
* Функции для реализации заказов:

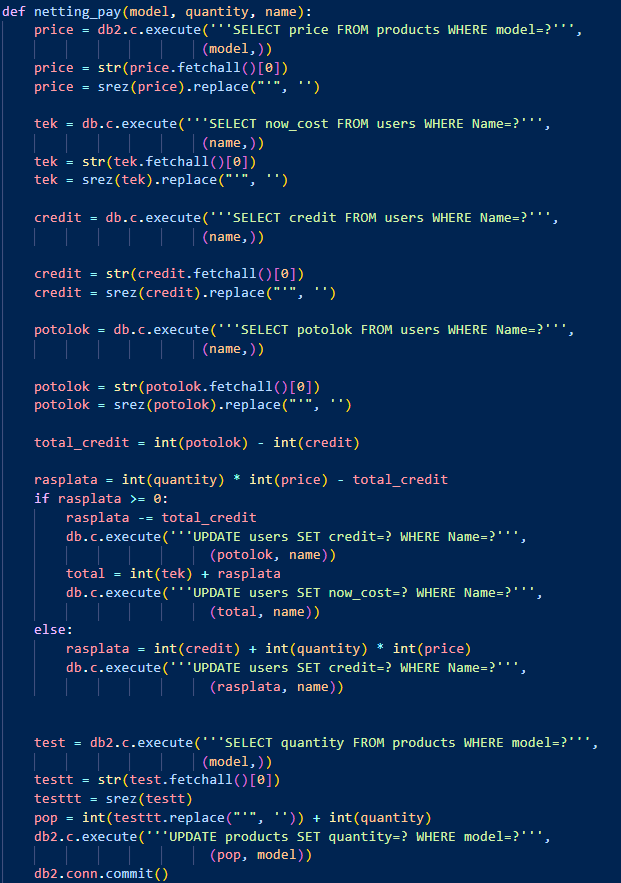




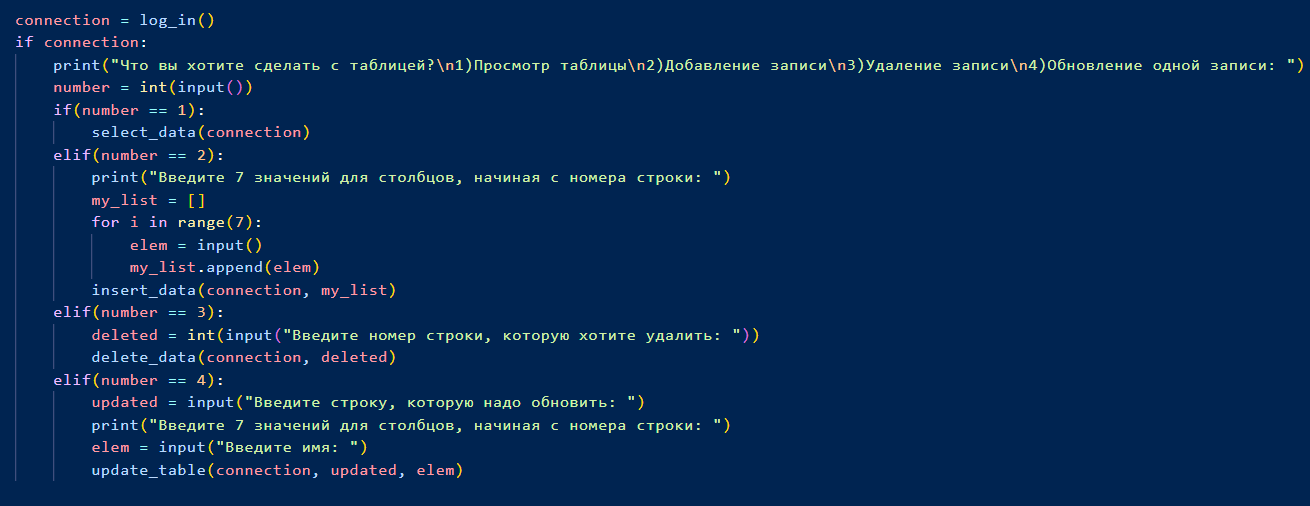








* Реализация интерфейса терминала:



# Вывод

Мы научились реализовывать интерфейс автоматизированного рабочего места оператора по продажам.

# Список литературы

* [Как создавать классы в приложении Tkinter ~ PythonRu](https://pythonru.com/uroki/oop-v-prilozhenii-tkinter-tkinter)- Создание приложения на tkinter
* [tkinter — Интерфейс Python к Tcl/Tk — Документация по Python 3.11.2](https://docs.python.org/3/library/tkinter.html)– Документация к ООП в tkinter;