МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский государственный гуманитарный университет»

(ФГБОУ ВО «РГГУ»)

Институт информационных наук и технологий безопасности

Факультет информационных систем и безопасности

Кафедра информационных технологий и систем

Бобров Алексей Владимирович

Лабораторная работа

по дисциплинам «Методы анализа предметных областей в гуманитарной сфере» и «Программная инженерия»

студента 3 курса по направлению подготовки:

09.03.03 «Прикладная информатика»,

профиль: «Прикладная информатика в гуманитарной сфере»

|  |  |
| --- | --- |
|  | Проверила:  Охапкина Е.П.  Отметка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва 2022

**Анализ предметной области**

**Цель работы:**

Выбор предметной области и её подробное описание. Для лабораторной работы была выбрана предметная область «Бар». Далее идёт ознакомление непосредственно с работой бара, в ходе которой происходит реализация напитков и продуктов питания.

**Ход работы**

Бар — это предприятие общественного питания, оборудованное барной стойкой и реализующее напитки, закуски и блюда в ограниченном ассортименте. Такие предприятия могут выступать в качестве либо полностью обособленного и самостоятельного заведения, либо как составляющее ресторана или коллективных средств размещения (гостиниц, отелей и др.).

Как и другие предприятия общественного питания, бар заказывает товары. Из доставленных и принятых товаров, которые поступают на хранение и реализацию, состоит инвентарь бара. Список товаров, с которыми работает бар, в наши дни чаще всего хранится в электронном виде в базе данных, как и информация об инвентаре заведения.

Бармен оформляет заказ на товар из таблицы товаров. Далее товар получается и происходит учёт товара. Его приблизительный сценарий:

1) Добавление позиции в приложение учёта

Используется специализированное ПО. Работник бара вносит характеристики (название, типа, розничная и оптовая цены, объём, место хранения, срок годности).

2) Актуализации информации об объёме и других характеристиках уже имеющейся продукции.

Не вся продукция является новой. Тем не менее в течение дня используется уже имеющиеся в баре, и необходимо обновлять её характеристики.

3) Автоматический подсчёт объёма использованного товара, выручки и др. данных.

Важная функция специализированного ПО – составление отчётов. Автоматически происходит взаимодействие с данными из базы данных и впоследствии формирование отчёта в Exсel.

В ходе работы бара несколько раз в неделю проводится инвентаризация. Инвентаризация — это проверка наличия имущества организации и состояния её финансовых обязательств на определённую дату путём сличения фактических данных с данными бухгалтерского учёта. Это основной способ фактического контроля за сохранностью имущественных ценностей и средств, для которого используется таблица инвентаря бара. Во время инвентаризации учитывается каким товаром располагает бар, в каком количестве и сколько составляет его остаток. Например, бар располагает 5 бутылками водки, объёмом по 0,5 л и у двух из них остаток составляет 0,35 л. Данные сведения заносятся в базу данных с помощью приложения для снятия остатков.

**Вывод**

Была выбрана предметная область «Бар» и проанализирована информация о процессе его работы, благодаря чему в следующей лабораторной будет осуществляться создание приложения для снятия остатков.

**Создание приложения**

**Цель работы:**

Целью данной работы является создание приложения для снятия остатков и для работы с ассортиментом бара (осуществляется только работником бара, которые имеют специальные данные для входа в приложение).

**Ход работы**

У работников бара есть множество обязанностей, обозначим лишь некоторые из них, которые будут реализованы в ходе этой и последующих лабораторных работ. В первую очередь бармен является продавцом, добавим Use Case «Учёт продаж». Распишем документацию для данного Use Case и добавим её в модель.

Учёт продаж проводится с каждым клиентом в индивидуальном порядке. Он проводится в несколько этапов:

1) Выполнение расчёта с клиентом

Принятие заказа у клиента, пробитие (составление счёта) всех отпускаемых товаров и выбор клиентом способа оплаты.

Кассовый чек является документом, который подтверждает факт приобретения какого-либо товара за наличный расчет или безналичный расчет и выдаётся клиенту. Когда посетители рассчитываются картой, нужно выдать ещё один документ: слип-чек. Слип — это чек, подтверждающий лишь то, что операция прошла по банковской карте через POS-терминал. Печатается в двух экземплярах, один остаётся у продавца.

2) Автоматическое обновление базы данных

ККТ(контрольно-кассовая техника) автоматически обновляет данные об отпускаемой продукции и совершаемых транзакциях в базе данных.

3) Пробитие чеков

Печать и выдача соответствующих чеков.

Кроме того, работник бара проводит все операции продажи через ККТ(контрольно-кассовая техника). Добавим Use Case «Работа с ККТ». Распишем документацию для данного Use Case и добавим её в модель.

В начале смены проводится проверка работы ККТ.

1) Открытие кассовой смены

Производится вручную. Происходит автоматическое формирование документа на текущий торговый день

2) Учёт продаж

См. документацию "Учёт продаж"

3) Автоматическая передача данных в ОФД

Касса сама формирует и передаёт отчётности в контролирующие органы

4) Закрытие кассовой смены

Производится вручную, не позже, чем через 24 часа от открытия смены.

Также работник бара должен контролировать весь отпускаемый товар. Добавим Use Case «Учёт товара». Распишем документацию для данного Use Case и добавим её в модель.

Учёт товара проводится ежедневно при закрытии смены.

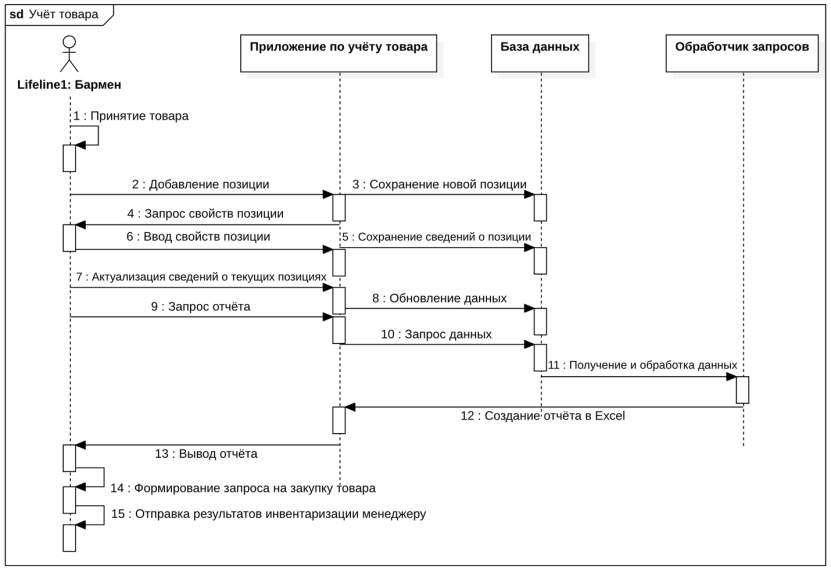
Сценарий описан в предыдущей лабораторной работе.

Помимо актёра «Бармен» необходимо добавить актёра «Клиент». Он не будет взаимодействовать с оборудованием бара, но без присутствия данного актёра в модели невозможно представить действия работника бара.

У актёра «Клиент» есть 3 Use Case: «Сделать заказ», «Получить чек» и «Получить заказ». Эти действия не требуют документации, однако их необходимо добавить для дальнейшего их учёта в Sequence Diagram. На рисунке ниже (рис. 1) представлена готовая Use Case диаграмма.

Рисунок 1. Use Case Diagram.

Также необходимо создать Sequence Diagram. Добавляем в неё актёра «Бармен». Кроме того, добавляем Lifeline’ы «Приложение по учёту товара», «База данных» и «Обработчик запросов». Добавляем между ними Message’ы. На рисунке ниже (рис. 2) представлена готовая Sequence Diagram Учёт товара.

Рисунок 2. Sequence Diagram Учёт товара.

Чтобы создать приложение для снятия остатков использовался язык программирования Python и реляционная система управления базами данных MySQL от Oracle.

Для создания графического интерфейса использовалась стандартная графическая библиотека Tkinter, а для подключения базы данных к программе — коннектор MySQL для Python.

Были импортированы следующие модели python:

Сначала был написан класс app, который создаёт окно приложения (далее, контейнер) с фиксированным параметрами (интерфейс не будет адаптивным).

Элементы интерфейса создаются с помощью классов, создающих виджеты модуля Tkinter, и помещаются в контейнер. Далее приведён список классов и их описание (табл. 1); в таблице упоминаются рисунки 3-8, которые представлены на страницах 8-10:

Таблица 1. Классы программы.

|  |  |
| --- | --- |
| Класс | Его описание |
| WindowInfo | Создаёт в контейнере дочернее окно для просмотра, изменения и добавления информации о товаре (рис. 7). |
| WindowMessage | Создаёт в контейнере сверху дочернее всплывающее окно, уведомление (рис. 4). |
| FrameIcon | Создаёт в контейнере виджет иконки приложения (рис. 3). |
| FrameMenu | Создаёт в контейнере виджет главного меню приложения (рис. 5). |
| FrameInventory | Создаёт в контейнере виджет работы с инвентарём бара (рис. 8). |
| FrameItems | Создаёт в контейнере виджет работы с ассортиментом товаров бара (рис. 6). |
| FrameCopyright | Создаёт в контейнере виджет копирайта (рис. 3). |
| FrameLogin | Создаёт в контейнере виджет авторизации (рис. 3). |
| App | Создаёт контейнер |

В условии if \_\_name\_\_ == '\_\_main\_\_' создаётся объект класса App, то есть приложение для снятия остатков. В него помещают объект класса FrameLogin. Код представлен на страницах 11-21.

**Вывод**

Было реализовано приложение для снятия остатков для барменов, которое позволяет обновлять ассортимент товаров доступных для заказа в бар, ассортимент товаров, которыми бар располагает в данный момент с информацией об их остатках и выводить отчёты.

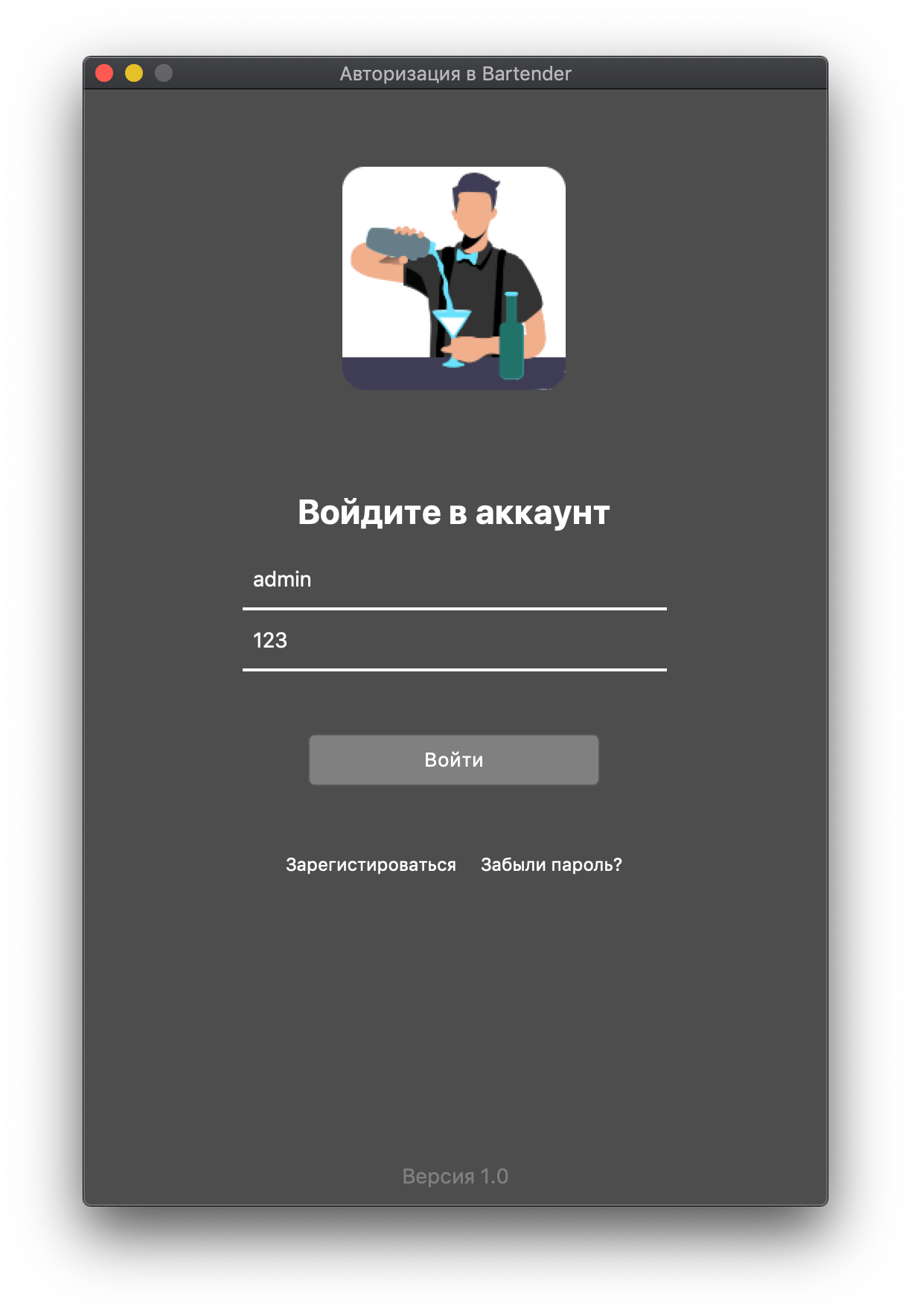
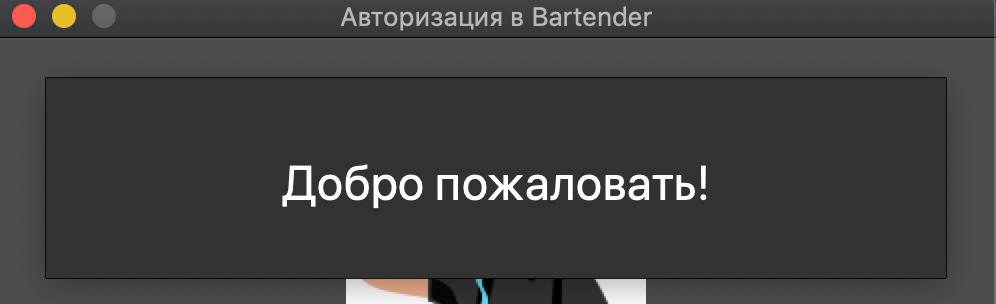
Рисунок 3. Снимок экрана окна авторизации

Рисунок 4. Снимок экрана уведомления

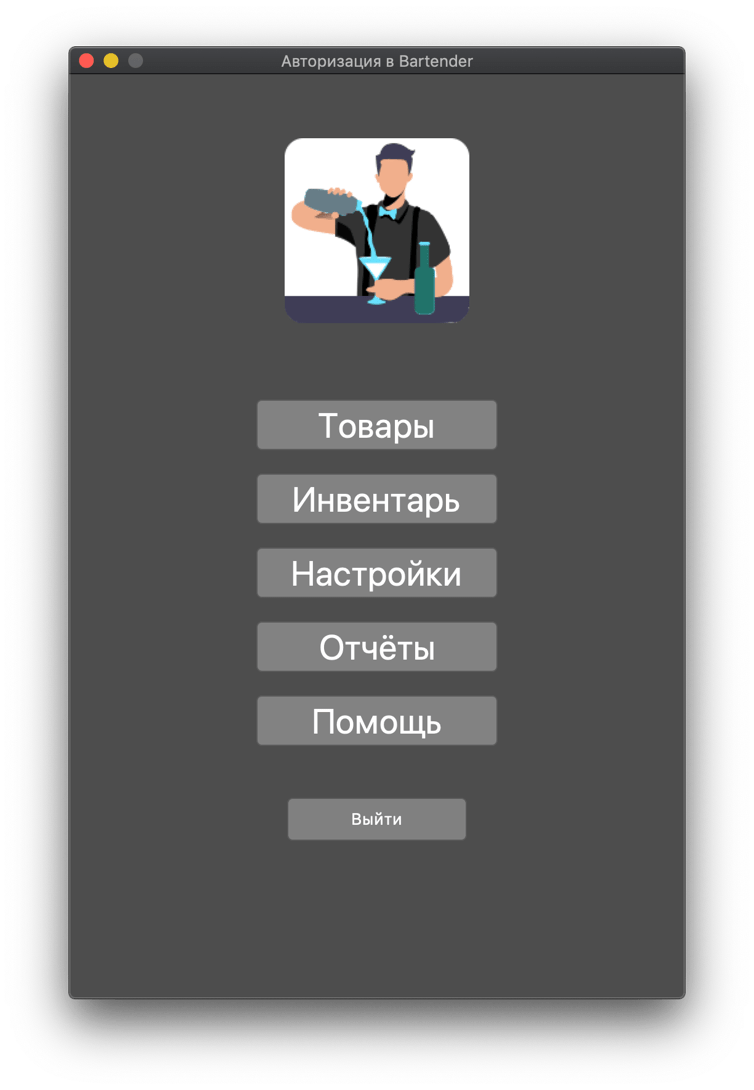
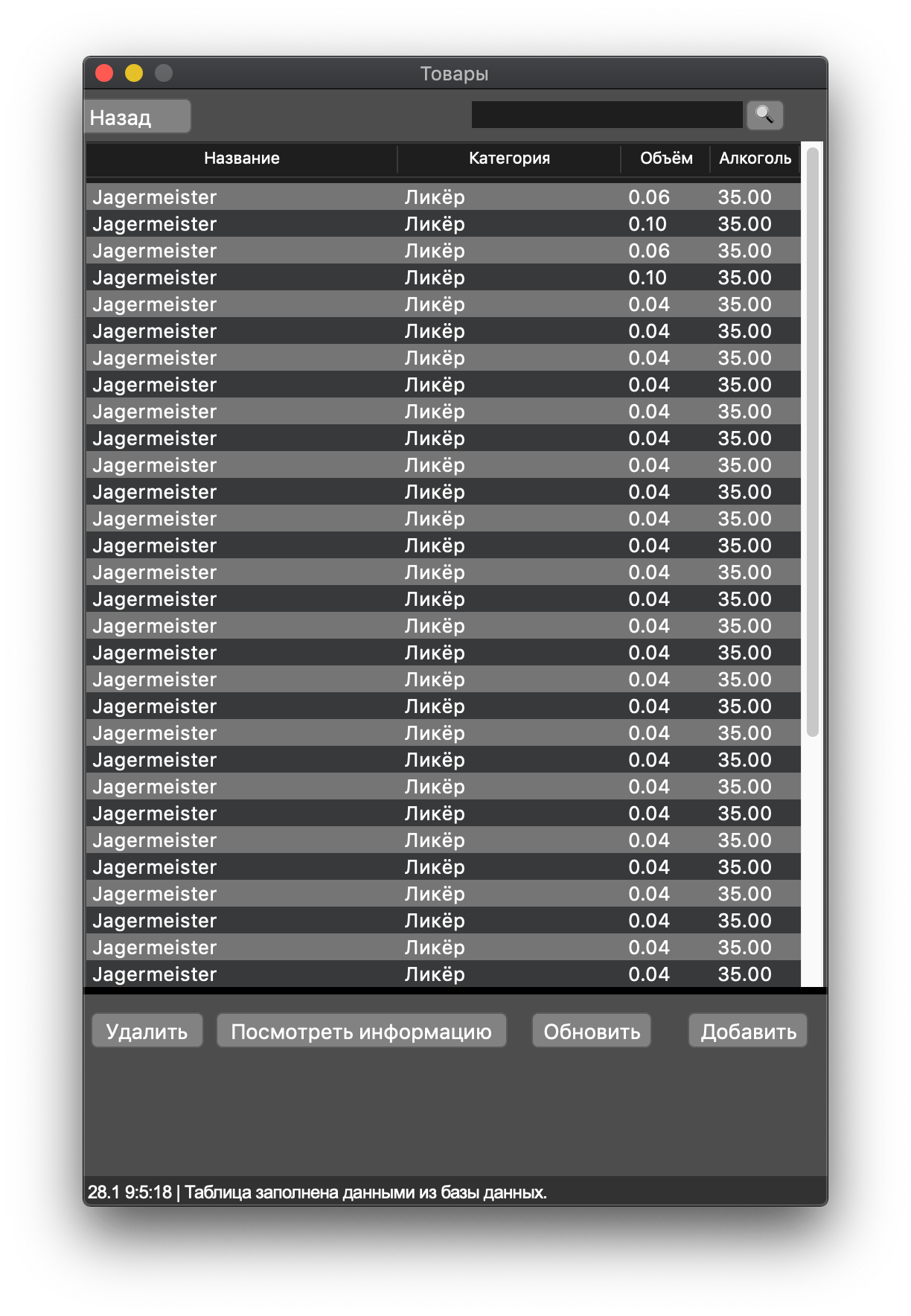
Рисунок 5. Снимок экрана окна главного меню

Рисунок 6. Снимок экрана окна работы с товаром

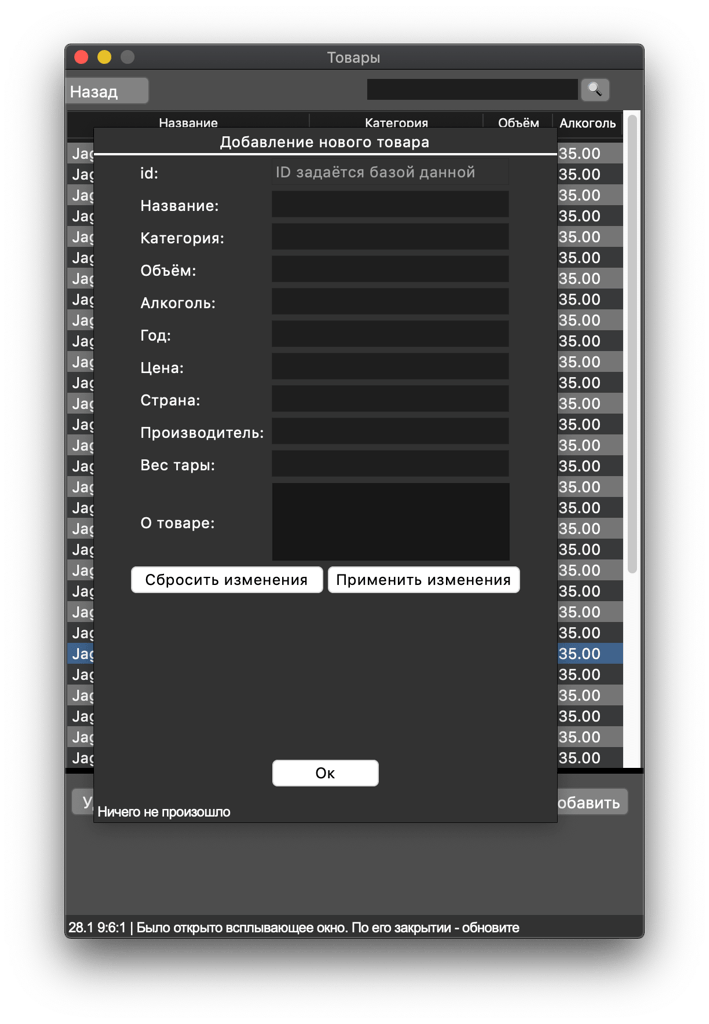
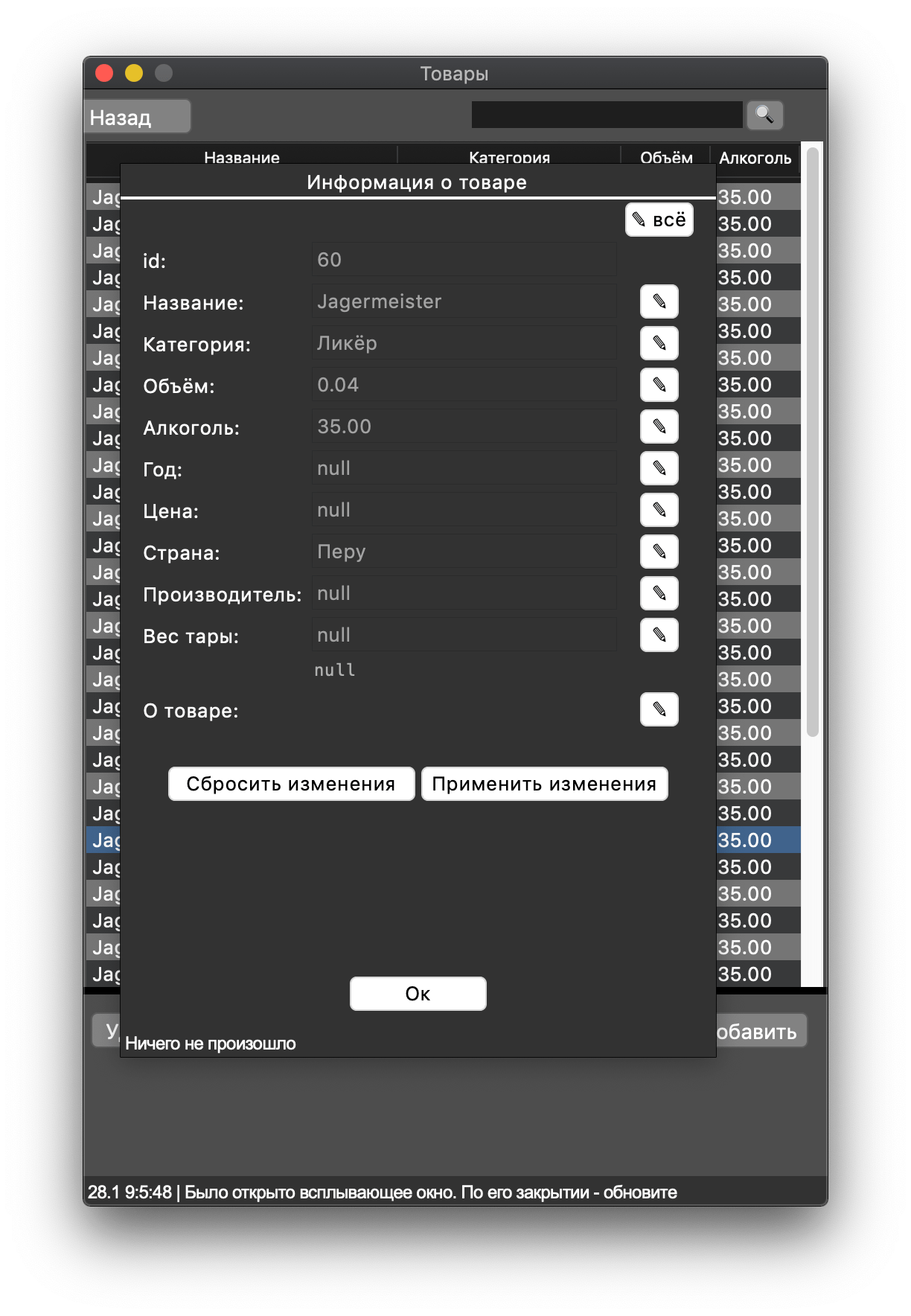
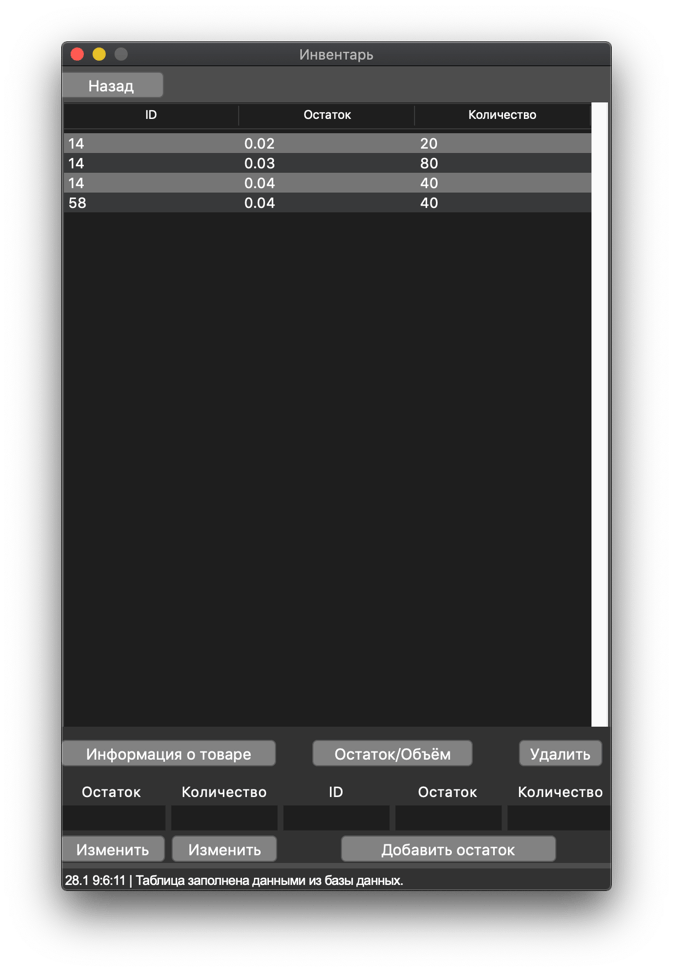
Рисунок 7. Снимок экрана окна просмотра/изменения информации о товаре и добавления нового товара

Рисунок 8. Снимок экрана окна работы с инвентарём

**#импорт модулей**

import mysql.connector as sql

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from tkmacosx import Button

from PIL import ImageTk, Image

from sqlalchemy import create\_engine

from datetime import datetime

**#создание шаблонов цветов, подлкючение локальной БД, функции получения текущего времени**

bgc="#4D4D4D"

btc='#828282'

conn=sql.connect(host='localhost',username='root',password='AjrG@#0kM',database='bartender')

conn.autocommit = True

cursor=conn.cursor()

def noow():

now = datetime.now()

a=str(now.day)+'.'+str(now.month)+' '+str(now.hour)+':'+str(now.minute)+':'+str(now.second)+' | '

return a

**#класс дочернего окна для просмотра, изменения и добавления информации о товаре**

class WindowInfo(tk.Toplevel):

def \_\_init\_\_(self, container, selected, new):

super().\_\_init\_\_(container)

x=container.winfo\_rootx()+50

y=container.winfo\_rooty()+50

self.geometry('400x600+%s+%s'%(x,y))

self.grab\_set()

self.overrideredirect(True)

buttons=[]

entrys=[]

#заменает NoneType на null

for i in range(len(selected)):

if selected[i]==None: selected[i]='null'

**#заменяет символы для корректной работы sql-запросов**

def rplc(a):

a=a.replace(',','.').replace('%','').replace('% ','')

return a

**#преобразует виджет в виджет добавления нового товара**

def new\_item():

for i in range(10):

entrys[i].config(state='normal')

buttons[i].destroy()

entrys[0].delete(0,END)

entrys[0].insert(0,'ID задаётся базой данной')

entrys[0].config(state='disabled')

about\_text.config(state='normal',bg='#171717',fg='white')

buttons[10].destroy()

buttons[11].destroy()

name.config(text='Добавление нового товара')

commit.config(command=insert\_item)

rst.config(text='Очистить поля',command=clear)

**#выход из дочернего окна**

def exit():

self.destroy()

**#работа кнопок редактирования**

def edit(a):

if entrys[a]['state'] == 'disabled':

entrys[a].config(state='normal')

buttons[a].config(bg=bgc)

else:

entrys[a].config(state='disable')

buttons[a].config(bg='white')

def editabout():

if about\_text['state']=='disabled':

about\_text.config(state='normal',bg='#171717',fg='white')

buttons[10].config(bg=bgc)

else:

about\_text.config(state='disable',bg='#323232',fg='#B1B2B1')

buttons[10].config(bg='white')

def editall():

if b['bg']=='white':

for i in range (1,10):

entrys[i].config(state='normal')

buttons[i].config(bg=bgc)

about\_text.config(state='normal',bg='#171717')

buttons[10].config(bg=bgc)

buttons[11].config(bg=bgc)

else:

for i in range (1,10):

entrys[i].config(state='disable')

buttons[i].config(bg='white')

about\_text.config(state='disable',bg='#323232')

buttons[10].config(bg='white')

buttons[11].config(bg='white')

**#возвращает tulpe с данными из полей**

def take\_data():

new\_data=[]

for i in range(1,10):

new\_data.append(entrys[i].get())

new\_data.append(about\_text.get(1.0,END))

arr=[new\_data[2],new\_data[3],new\_data[4],new\_data[5],new\_data[8]]

if entrys[0].get()!='ID задаётся базой данной':

new\_data.append(entrys[0].get())

elif '' in arr:

new\_data[2]='0'

new\_data[3]='0'

new\_data[4]='0'

new\_data[5]='0'

new\_data[8]='0'

new\_data[2]=rplc(new\_data[2])

new\_data[3]=rplc(new\_data[3])

new\_data[8]=rplc(new\_data[8])

new\_data[9]=new\_data[9].replace('\n',' ')

for i in range(len(new\_data)):

if 'null' in new\_data[i]: new\_data[i]=None

return tuple(new\_data)

**#обновление/вставка товара**

def update\_item():

sql='update items set name=%s,type=%s,volume=%s,alcohol=%s,year=%s,price=%s,country=%s,producer=%s,empty\_weight=%s,about=%s where item\_id=%s'#,str(entry[0])

cursor.execute(sql,take\_data())

status.config(text=noow()+'Обновлены сведения товара с id='+entrys[0].get())

def insert\_item():

sql='insert into items (name,type,volume,alcohol,year,price,country,producer,empty\_weight,about) values '+str(take\_data())

cursor.execute(sql)

status.config(text=noow()+'Добавлен новый товар')

**#очищает поля ввода**

def clear():

for i in range(10):

entrys[i].delete(0,END)

entrys[i].insert(0,selected[i])

about\_text.delete(1.0, END)

about\_text.insert(1.0,selected[10])

**#возвращает данные, которые были изначально**

def reset():

clear()

update\_item()

name=Label(self, text='Информация о товаре')

name.pack()

status=Label(self,text='',font=("Arial", 12),anchor=W) #до 55 символов

status.config(text='Ничего не произошло')

status.pack(padx=0,pady=0,fill=X,side=BOTTOM)

Frame(self, width=400, height=2, bg='white').pack()

frame=Frame(self)

frame.pack()

for i in range(11):

lt=['id',"Название","Категория","Объём","Алкоголь",'Год','Цена','Страна','Производитель','Вес тары','О товаре']

Label(frame,text=lt[i]+':').grid(row=i+1,column=0,sticky=W)

e=Entry(frame,width=22)

e.grid(row=i+1,column=1,sticky=W)

e.insert(0,selected[i])

e.config(state='disable')

entrys.append(e)

b=Button(frame,width=30,text='✎',command=lambda a=i:edit(a))

b.grid(row=i+1,column=2)

buttons.append(b)

b=Button(frame,width=50,text='✎ всё',bg='white')

b.grid(row=0,column=2)

b.config(command=editall)

buttons.append(b)

entrys[10].destroy()

buttons[0].destroy()

about\_text=Text(frame,width=29,height=5,bg='#323232')

about\_text.insert(1.0,selected[10])

about\_text.config(state='disable',fg='#B1B2B1')

about\_text.grid(row=11,column=1,sticky=W)

buttons[10].config(command=editabout)

frame=Frame(self,bg='red')

frame.pack()

rst=Button(frame,width=170,text='Сбросить изменения',command=reset)

rst.grid(row=0,column=0)

commit=Button(frame,width=170,text='Применить изменения',command=update\_item)

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

if new==1:

new\_item()

Button(self,text='Ок',command=exit).pack(pady=10,side=BOTTOM)

self.wm\_transient(container)

self.resizable(False, False)

**#класс дочернего окна (уведомление)**

class WindowMessage(tk.Toplevel):

def \_\_init\_\_(self, container, case):

super().\_\_init\_\_(container)

x=container.winfo\_rootx()+25

y=container.winfo\_rooty()+20

self.geometry('450x100+%s+%s'%(x,y))

self.grab\_set()

self.overrideredirect(True)

self.wm\_transient(container)

self.resizable(False, False)

self.after(2000,lambda:self.destroy())

label=Label(self, text='',font=('Microsoft YaHei UI Light', 24))

label.place(relx=0.5,rely=0.5,anchor=CENTER)

if case==1:

label.config(text='Добро пожаловать!')

elif case==2:

label.config(text='Неправильный пароль.')

elif case==3:

label.config(text='Неправильный логин или пароль.')

**#класс виджета с иконкой приложения**

class FrameIcon(ttk.Frame):

def \_\_init\_\_(self, container):

super().\_\_init\_\_(container)

self.img=PhotoImage(file='images/bartender\_icon.png')

Label(self, image=self.img, bg=bgc).pack()

self.pack(pady=50,side=TOP)

**#класс виджета с главным меню приложения**

class FrameMenu(ttk.Frame):

def \_\_init\_\_(self, container):

super().\_\_init\_\_(container)

def logout():

container.destroy()

FrameLogin(App()).pack()

def ranframeitems():

self.destroy()

frame=FrameItems(container)

container.title("Товары")

frame.place(relx=0.5, y=0, anchor=N)

def ranframeinventory():

self.destroy()

frame=FrameInventory(container)

container.title("Инвентарь")

frame.place(relx=0.5, y=0, anchor=N)

def passs(): pass

frame=Frame(self,width=460, height=400, bg=bgc)

frame.pack(padx=0,pady=0)

for i in range(5):

c=[ranframeitems,ranframeinventory,passs,passs,passs]

n=['Товары','Инвентарь','Настройки','Отчёты','Помощь']

a=[30,90,150,210,270]

Button(frame, width=200, text=n[i], bg=btc, border=0, font=('Microsoft YaHei UI Light', 28), fg='white', borderless=1, cursor="hand2",command=c[i]).place(relx=0.5, y=a[i], anchor=CENTER)

logout=Button(frame, width=150, pady=7, text='Выйти', fg='white', bg='grey', borderless=1, cursor="hand2", command=logout)

logout.place(relx=0.5, y=350, anchor=CENTER)

self.pack(padx=0,pady=0)

**#класс виджета для работы с инвентарём бара**

class FrameInventory(tk.Frame):

def \_\_init\_\_(self, container):

super().\_\_init\_\_(container)

self.config(bg=bgc)

#выход в главное меню

def doexit():

self.destroy()

status.destroy()

FrameMenu(container)

**#заполнение таблицы**

def fill\_table():

for row in tree.get\_children():

tree.delete(row)

counter\_rows=0

cursor.execute('select item\_id,weight\_left,amount from inventory')

for i in cursor:

if counter\_rows%2==0:

tree.insert('',END,counter\_rows,text="",values=(i[0],i[1],i[2]),tags=("lighter"))

else:

tree.insert('',END,counter\_rows,text="",values=(i[0],i[1],i[2]),tags=("darker"))

counter\_rows+=1

status.config(text=noow()+'Таблица заполнена данными из базы данных.')

**#возвращает массив с информацией о выбранном товаре**

def info\_of\_selected(a):

if a==None:

a=tree.item(tree.selection())['values'][0]

cursor.execute('select \* from items where item\_id='+str(a))

arr=[]

for i in cursor:

for j in range(11):

arr.append(i[j])

return arr

**#запускает дочернее окно с информацией о выбранном твоаре**

def info():

if len(tree.selection())!=1:

status.config(text=noow()+'Ошибка: Должна быть выбрана ОДНА строка для её удаления или справки')

pass

arr=info\_of\_selected(None)

WindowInfo(self,arr,0)

status.config(text=noow()+'Было открыто всплывающее окно. По его закрытии - обновите')

**#выводит в статус-бар соотношение остатка и объёма**

def ost\_vol():

if len(tree.selection())!=1:

status.config(text=noow()+'Ошибка: Должна быть выбрана ОДНА строка для её удаления или справки')

pass

text=tree.item(tree.selection())['values'][1]

vol=info\_of\_selected(None)[3]

status.config(text=noow()+'Остаток составляет %s от %s'%(text,vol))

**#добавить новый товар в инвентарь**

def insert\_ost():

text\_entrys=[]

for i in range(len(entrys)):

text\_entrys.append(entrys[i].get())

if info\_of\_selected(text\_entrys[0])!=[]:

if float(text\_entrys[1])<=float(info\_of\_selected(text\_entrys[0])[3]):

try:

cursor.execute('insert into inventory (item\_id,weight\_left,amount) values '+str(tuple(text\_entrys)))

fill\_table()

status.config(text=noow()+'Добавлен новый товар в инвентарь')

except sql.Error as err:

status.config(text=noow()+str(err))

if '1062' in status['text']:

cursor.execute('SET SQL\_SAFE\_UPDATES = 0')

cursor.execute('update inventory set amount=amount+%s where item\_id=%s and weight\_left=%s',(text\_entrys[2],text\_entrys[0],text\_entrys[1]))

cursor.execute('SET SQL\_SAFE\_UPDATES = 1')

fill\_table()

status.config(text=noow()+'Остаток совпал с другой строкой, увеличено количество')

pass

else:

status.config(text=noow()+'Остаток не может быть больше объёма/текущего остатка! (Объём=%s)'%info\_of\_selected(text\_entrys[0])[3])

else:

status.config(text=noow()+'Нет товара с id='+text\_entrys[0])

**#изменить количество выбранного товара**

def change\_amnt():

if len(tree.selection())!=1:

status.config(text=noow()+'Ошибка: Должна быть выбрана ОДНА строка для её удаления или справки')

pass

new\_amnt=amnt.get()

curamount=tree.item(tree.selection())['values'][2]

itemid=tree.item(tree.selection())['values'][0]

curost=tree.item(tree.selection())['values'][1]

if new\_amnt!=0:

cursor.execute('update inventory set amount=%s where item\_id=%s and weight\_left=%s',(new\_amnt,itemid,curost))

fill\_table()

else: delete\_it()

**#удалить выбранный товар из инвентаря**

def delete\_it():

if len(tree.selection())!=1:

status.config(text=noow()+'Ошибка: Должна быть выбрана ОДНА строка для её удаления или справки')

pass

itemid=tree.item(tree.selection())['values'][0]

curost=tree.item(tree.selection())['values'][1]

cursor.execute('SET SQL\_SAFE\_UPDATES = 0')

cursor.execute('delete from inventory where item\_id=%s and weight\_left=%s',(itemid,curost))

cursor.execute('SET SQL\_SAFE\_UPDATES = 1')

fill\_table()

**#изменить остаток выбранного товара**

def change\_ost():

if len(tree.selection())!=1:

status.config(text=noow()+'Ошибка: Должна быть выбрана ОДНА строка для её удаления или справки')

pass

itemid=tree.item(tree.selection())['values'][0]

curost=tree.item(tree.selection())['values'][1]

curamount=tree.item(tree.selection())['values'][2]

vol=info\_of\_selected(None)[3]

text=float(ost.get())

if text<=vol and text<float(curost):

try:

cursor.execute('update inventory set weight\_left=%s where item\_id=%s and weight\_left=%s',(ost.get(),itemid,curost))

fill\_table()

status.config(text=noow()+'Остаток обновлён для товара с id='+str(itemid))

except sql.Error as err:

status.config(text=noow()+str(err))

if '1062' in status['text']:

cursor.execute('SET SQL\_SAFE\_UPDATES = 0')

cursor.execute('delete from inventory where item\_id=%s and weight\_left=%s',(itemid,curost))

cursor.execute('update inventory set amount=amount+%s where item\_id=%s and weight\_left=%s',(curamount,itemid,ost.get()))

cursor.execute('SET SQL\_SAFE\_UPDATES = 1')

fill\_table()

status.config(text=noow()+'Остаток совпал с другой строкой, увеличено количество')

pass

else:

status.config(text=noow()+'Остаток не может быть больше объёма/текущего остатка! (Объём=%s)'%info\_of\_selected()[3])

exit=Button(self, width=100, text='Назад', bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=doexit)

exit.pack(padx=0,pady=3, anchor=NW)

status=Label(container,text='',font=("Arial", 12),anchor=W) #до 55 символов

status.config(text='Ничего не произошло')

status.pack(padx=0,pady=0,fill=X,side=BOTTOM)

frame\_tree\_scroll=Frame(self,bg=bgc)

scrollbar=Scrollbar(frame\_tree\_scroll,orient=VERTICAL)

tree=ttk.Treeview(frame\_tree\_scroll,yscrollcommand=scrollbar.set, selectmode=EXTENDED, show='headings', height=30)

scrollbar.config(command=tree.yview)

scrollbar.pack(side=RIGHT,fill=Y)

heads=["ID","Остаток","Количество"]

tree["columns"] = heads

for i in heads:

tree.heading(i,text=i)

for i in heads:

tree.column(i,width=160)

tree.tag\_configure('lighter',background='#757575')

tree.tag\_configure('darker',background='#38393A')

fill\_table()

tree.pack(padx=0,pady=0)

frame\_tree\_scroll.pack()

f=Frame(self)

f.pack(padx=0,pady=0,fill=X)

full\_info=Button(f, width=200, text='Информация о товаре', bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=info)

full\_info.grid(row=0,column=0,columnspan=2,pady=10)

ostvol=Button(f, width=150, text='Остаток/Объём', bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=ost\_vol)

ostvol.grid(row=0,column=2,columnspan=2,pady=10)

dlt=Button(f, width=80, text='Удалить', bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=delete\_it)

dlt.grid(row=0,column=4,pady=10)

ost\_l=Label(f,text='Остаток',fg='white')

ost\_l.grid(row=1,column=0)

ost=Entry(f,width=10)

ost.grid(row=2,column=0)

ost\_b=Button(f, width=100, text='Изменить', bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=change\_ost)

ost\_b.grid(row=3,column=0)

amnt\_l=Label(f,text='Количество',fg='white')

amnt\_l.grid(row=1,column=1)

amnt=Entry(f,width=10)

amnt.grid(row=2,column=1)

amnt\_b=Button(f, width=100, text='Изменить', bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=change\_amnt)

amnt\_b.grid(row=3,column=1)

entrys=[]

for i in range(3):

l=Label(f,text=heads[i])

l.grid(row=1,column=i+2)

e=Entry(f,width=10)

e.grid(row=2,column=i+2)

entrys.append(e)

insrtost=Button(f, width=200, text='Добавить остаток', bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=insert\_ost)

insrtost.grid(row=3,column=2,columnspan=3,padx=50)

**#класс виджета для работы с ассортиментом товаров бара**

class FrameItems(tk.Frame):

def \_\_init\_\_(self, container):

super().\_\_init\_\_(container)

self.config(bg=bgc)

**#выход в главное меню**

def doexit():

self.destroy()

status.destroy()

FrameMenu(container)

**#удаление выбранного объекта**

def delete\_item():

slct\_id=tree.item(tree.selection())['values'][4]

cursor.execute('SET SQL\_SAFE\_UPDATES = 0')

cursor.execute('delete from items where item\_id='+str(slct\_id))

cursor.execute('SET SQL\_SAFE\_UPDATES = 1')

fill\_table('select name,type,volume,alcohol,item\_id from items',None)

status.config(text=noow()+'Удалён товар с id='+str(slct\_id)+'. Таблица заполнена данными из базы данных.')

**#заполнение таблицы**

def fill\_table(sql,tlp):

for row in tree.get\_children():

tree.delete(row)

counter\_rows=0

cursor.execute(sql,tlp)

a=None

for i in cursor:

if counter\_rows%2==0:

tree.insert('',END,counter\_rows,text="",values=(i[0],i[1],str(i[2]),str(i[3]),str(i[4])),tags=("lighter"))

else:

tree.insert('',END,counter\_rows,text="",values=(i[0],i[1],str(i[2]),str(i[3]),str(i[4])),tags=("darker"))

counter\_rows+=1

a=str(i[4])

status.config(text=noow()+'Таблица заполнена данными из базы данных.')

return a

**#возвращает массив информации о выбранном товаре**

def info\_of\_selected():

a=tree.item(tree.selection())['values'][4]

cursor.execute('select \* from items where item\_id='+str(a))

arr=[]

for i in cursor:

for j in range(11):

arr.append(i[j])

status.config(text=noow()+'Показаны сведения товара с id='+str(a))

return arr

**#запускает дочернее окно с информацией о выбранном товаре**

def info():

arr=info\_of\_selected()

WindowInfo(self,arr,0)

status.config(text=noow()+'Было открыто всплывающее окно. По его закрытии - обновите')

**#запускает дочернее окно для добавления нового товара**

def new\_item\_info():

arr=['']\*11

WindowInfo(self,arr,1)

status.config(text=noow()+'Было открыто всплывающее окно. По его закрытии - обновите')

**#осуществляет поиск по таблице**

def search():

tlp=('%'+search\_e.get()+'%',)\*10

sql='select name,type,volume,alcohol,item\_id from items where name like %s or type like %s or volume like %s or alcohol like %s or year like %s or price like %s or country like %s or producer like %s or empty\_weight like %s or about like %s'

fill\_table(sql,tlp)

status.config(text=noow()+'Результаты поиска по запросу "%s"'%search\_e.get())

**#проверяет выбрана ли одна и только одна строка**

def check\_selection(function):

if len(tree.selection())!=1: status.config(text=noow()+'Ошибка: Должна быть выбрана ОДНА строка для её удаления или справки')

else:

function()

**#устанавливает команду на кнопку через проверку строки**

def set\_command(this\_button,com):

this\_button.config(command=lambda a=com:check\_selection(a))

**#кнопка выхода в главное меню**

frame\_top=Frame(self,bg=bgc)

frame\_top.pack(padx=0,pady=4,fill=X,side=TOP)

exit=Button(frame\_top, width=100, text='Назад', bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=doexit)

exit.grid(row=0,column=0,sticky=W)

Label(frame\_top,width=20,bg=bgc).grid(row=0,column=1)

search\_e=Entry(frame\_top,border=0, highlightthickness=0)

search\_e.grid(row=0,column=2,sticky=E)

search\_b=Button(frame\_top,width=30,text='🔍',bg=btc,border=0,borderless=1,font=('Arial',10), command=search)

search\_b.grid(row=0,column=3,sticky=W)

status=Label(container,text='',font=("Arial", 12),anchor=W) #до 55 символов

status.config(text='Ничего не произошло')

status.pack(padx=0,pady=0,fill=X,side=BOTTOM)

**#фрейм таблицы и её скроллбара**

frame\_tree\_scroll=Frame(self,bg=bgc)

scrollbar=Scrollbar(frame\_tree\_scroll,orient=VERTICAL)

tree=ttk.Treeview(frame\_tree\_scroll,yscrollcommand=scrollbar.set, selectmode=EXTENDED, show='headings', height=30)

scrollbar.config(command=tree.yview)

scrollbar.pack(side=RIGHT,fill=Y)

heads=["Название","Категория","Объём","Алкоголь","ID"]

tree["columns"] = heads

for i in heads:

tree.heading(i,text=i)

for i in heads:

tree.column(i,width=60)

tree.column(heads[0],width=210)

tree.column(heads[1],width=150)

tree.column(heads[4],width=0,stretch=0)

tree.tag\_configure('lighter',background='#757575')

tree.tag\_configure('darker',background='#38393A')

tree.pack(padx=0,pady=0)

frame\_buttons=Frame(self,bg=bgc)

buttons=[]

for i in range(4):

w=[80,200,85,85]

n=['Удалить','Посмотреть информацию','Обновить','Добавить']

c=[delete\_item,info,lambda a='select name,type,volume,alcohol,item\_id from items':fill\_table(a,None),new\_item\_info]

p=[2,2,10,10]

b=Button(frame\_buttons, width=w[i], text=n[i], bg=btc, border=0, font=('Microsoft YaHei UI Light', 14), fg='white', borderless=1, cursor="hand2",command=c[i])

b.grid(row=0,column=i,padx=p[i])

if i==0 or i==1:

set\_command(b,c[i])

buttons.append(b)

fill\_table('select name,type,volume,alcohol,item\_id from items',None)

frame\_tree\_scroll.pack(padx=0,pady=0)

Frame(self, width=550, height=5, bg='Black').pack()

frame\_buttons.pack(padx=0,pady=10)

self.place(x=0,y=0)

**#класс виджета с копирайтом**

class FrameCopyright(ttk.Frame):

def \_\_init\_\_(self, container):

super().\_\_init\_\_(container)

copyright=Label(self, text="Версия 1.0", fg="#808080", bg=bgc, font=('Microsoft YaHei UI Light', 14)) #bg=bgc

copyright.pack(padx=0,pady=0)

self.pack(pady=10,side=BOTTOM)

**#класс виджета авторизации**

class FrameLogin(tk.Frame):

def \_\_init\_\_(self, container):

super().\_\_init\_\_(container,bg=bgc)

**#обработка логина и пароля**

def login():

query\_get\_login\_pswrd='SELECT login,password FROM users'

cursor.execute(query\_get\_login\_pswrd)

counter=0

users\_num=0

passwords=[]

logins=[]

for i in cursor:

for j in i:

if counter%2 == 0:

logins.append(j)

else:

passwords.append(j)

counter += 1

username=user.get()

password=psword.get()

for i in range(len(logins)):

if logins[i]==username and passwords[i]==password:

cprt.destroy()

self.destroy()

FrameIcon(container)

FrameMenu(container)

WindowMessage(container,1)

elif username in logins:

psword.delete(0, END)

psword.insert(0, 'Пароль')

WindowMessage(container,2)

else:

user.delete(0, END)

user.insert(0, 'Логин')

psword.delete(0, END)

psword.insert(0, 'Пароль')

WindowMessage(container,3)

**#обработка событий полей ввода**

def in\_user(e):

if user.get()=="Логин":

user.delete(0, END)

def in\_pswrd(e):

if psword.get()=="Пароль":

psword.delete(0, END)

def out\_entry(e):

if psword.get()=="":

psword.insert (0, 'Пароль')

elif user.get()=="":

user.insert (0, 'Логин')

FrameIcon(self)

Label(self, text='Войдите в аккаунт', fg='white', bg=bgc, font=('Microsoft YaHei UI Light', 23, 'bold')).pack(pady=10)

user=Entry(self, width=30, fg='white', border=0, highlightthickness=0, bg=bgc, font=('Microsoft YaHei UI Light', 14))

user.pack(pady=10)

user.insert(0, 'admin')

Frame(self, width=285, height=2, bg='white').pack()

user.bind('<FocusIn>', in\_user)

user.bind('<FocusOut>', out\_entry)

psword=Entry(self, width=30, fg='white', border=0, highlightthickness=0, bg=bgc, font=('Microsoft YaHei UI Light', 14))

psword.pack(pady=10)

psword.insert(0, '123')

Frame(self, width=285, height=2, bg='white').pack()

psword.bind('<FocusIn>', in\_pswrd)

psword.bind('<FocusOut>', out\_entry)

go=Button(self, width=200, pady=7, text='Войти', fg='white', bg='grey', borderless=1, command=login, cursor="hand2")

go.pack(pady=40)

frame=Frame(self,bg=bgc)

frame.pack(pady=0)

signup=Label(frame, text="Зарегистироваться", fg='white', bg=bgc, font=('Microsoft YaHei UI Light' ,12))

signup.grid(row=0,column=0,padx=5)

forgotpsword=Label(frame, text="Забыли пароль?", fg='white', bg=bgc, font=('Microsoft YaHei UI Light' ,12))

forgotpsword.grid(row=0,column=1,padx=5)

self.pack(padx=0,pady=0)

cprt=FrameCopyright(container)

**#класс создания контейнера**

class App(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.title("Авторизация в Bartender")

self.geometry("500x750+1100+100")

self.config(bg=bgc)

self.resizable(False, False)

if \_\_name\_\_ == "\_\_main\_\_":

app=App()

FrameLogin(app).pack()

app.mainloop()