МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ федеральное государственное автономное образовательное учреждение высшего образования «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» КАФЕДРА № 43

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ

ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доц., к.т.н |  |  |  | А.В. Туманова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ  Разработка программы |
| «Онлайн магазин» |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

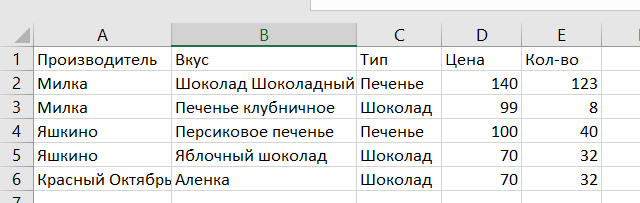
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

## **Описание задачи**

Задачей курсового проекта является разработка программы для предметной области, в моем случае это «Интернет магазин»

Работа представляет из себя пример реализации двух библиотек в виде онлайн магазина в оболочке терминала на языке программирования Python.

**Описание структур данных**

## **Описание функций программы**

## **Первая библиотека catalog\_hub.**

Библиотека представляет из себя инструмент управления базой данных excel, в которой организатор ведет учет товара

## **Вторая библиотека client\_hub**

Библиотека дает возможность полного управления информацией о пользователях магазина. Хранит информацию: login, password и корзину в json формате

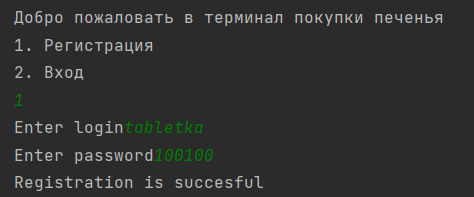
Вместе связка этих двух наборов функций дает возможность симулировать настоящий магазин. Для примера я завернул их реализацию в рамках терминала. Возможна интеграция и в другие api, которые поддерживают Python

## **Описание функций**

Исходный код функций указан выше, в листинге соответствует модуля

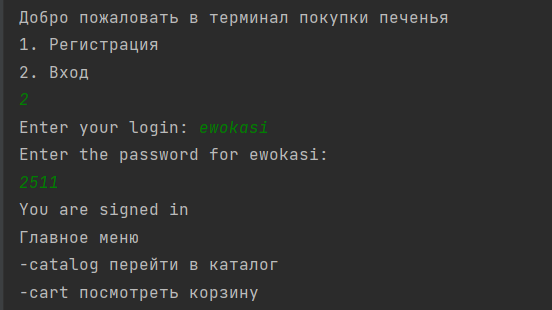
## **Client\_hub.registrate(login, psw);**

Осуществляет запись нового пользователя в json файл для дальнейшей авторизации



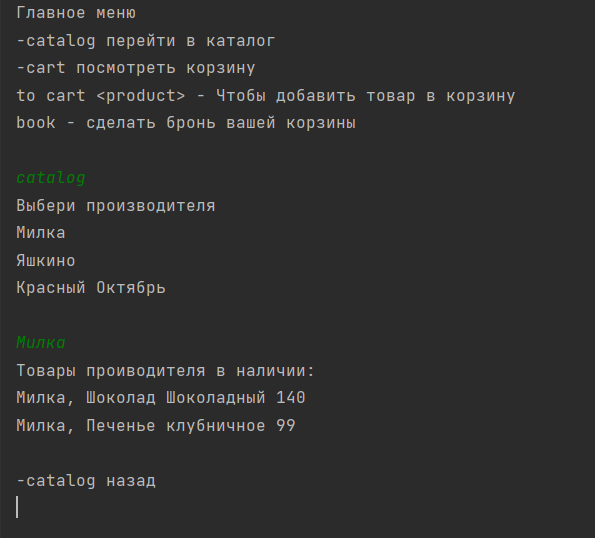
## **Client\_hub.sign\_in()**

Дает возможность зарегистрированному пользователю авторизоваться в системе



## **Branch\_hub.branch\_render(callback, user)**

Осуществляет весь графический интерфейс, соотнося его с профилем пользователя. На вход требует строку, которая обозначает выбор подветки интерфейса и данные о пользователе для хранения товара и его бронирования.



## **Hash(password)**

Все пароли хешируются, поэтому если база данных с паролями пользователя утечет, злоумышленники все равно не смогут подставить значение в программу

def hash(password):  
 res = 1  
 password = str(password)  
 for i in range(0, len(password)):  
 res = res \* int(password[i]) \* int(password[i])  
 return res

**Управление структурой, в которой хранится информация о товаре**

## Catalog\_hub.Catalog\_add(maker, taste, form, price, count)

Добавляет в конец таблицы товар с введенными значениями. Также доступен вариант редактирования напрямую через таблицу excel

## Catalog\_hub.Catalog\_del(maker, taste)

Реализует удаление товара по его ключам

**Хранение информации о пользователе**

Информация о пользователе хранится в json формате

{  
 "clients": [  
 {  
 "login": "ewokasi",  
 "password": "100",  
 "cart": [  
 {  
 "maker": "\u041a\u0440\u0430\u0441\u043d\u044b\u0439 \u041e\u043a\u0442\u044f\u0431\u0440\u044c",  
 "taste": "\u0410\u043b\u0435\u043d\u043a\u0430"  
 }  
 ]  
 },  
 {  
 "login": "tabletka",  
 "password": "0",  
 "cart": []  
 }  
 ]  
}

**В программе реализовано автоматическое управление базой данных клиентов**

## **Добавление товара в корзину**

def add\_to\_cart(login, call):  
  
 with open("databases/clients.json", 'r') as file:  
 db = json.load(file)  
 clients = db['clients']  
  
 maker = call.split(sep=", ")[0]  
 taste = call.split(sep=", ")[1]  
  
 offer = {  
 "login": login,  
 "cart": [  
 {  
 "maker": maker,  
 "taste": taste  
 }  
 ]  
  
 }  
  
 # Если заказ первый, и если второй  
 is\_found = 0  
 for client in clients:  
 if offer["login"] == client["login"]:  
 client["cart"].append(offer["cart"][0])  
 db["clients"] = clients  
 with open("databases/clients.json", 'w') as file:  
 json.dump(db, file, indent=2)  
  
 is\_found = 1  
 break  
  
 if is\_found == 0:  
 clients.append(offer)  
 db["clients"] = clients  
 with open("databases/clients.json", 'w') as file:  
 json.dump(db, file, indent=2)  
  
 return offer

## **Добавление брони товара для пользователя**

def add\_book(login):  
 # сделать бронь товара - перенести карт юзера из клинт в букс  
  
 with open("databases/books.json", 'r') as file:  
 books\_db = json.load(file)  
  
 with open("databases/clients.json", 'r') as file:  
 clients\_db = json.load(file)  
 clients = clients\_db["clients"]  
  
 for client in clients:  
 if login == client["login"]:  
 cart = client["cart"]  
 break  
  
 client = {  
 "login": login,  
  
 "key": str(login[0].upper()) + "-" + str(randint(100, 999)),  
 "cart": cart  
 }  
  
 with open("databases/books.json", "w") as file:  
 books\_db["books"].append(client)  
 json.dump(books\_db, file, indent=2)  
 clear\_cart(login)  
 return client

## **Описание пользовательского интерфейса**

Навигация по программе выполняется при помощи команд, которые предлагает ввести программа.

Catalog – для перехода в секцию товаров

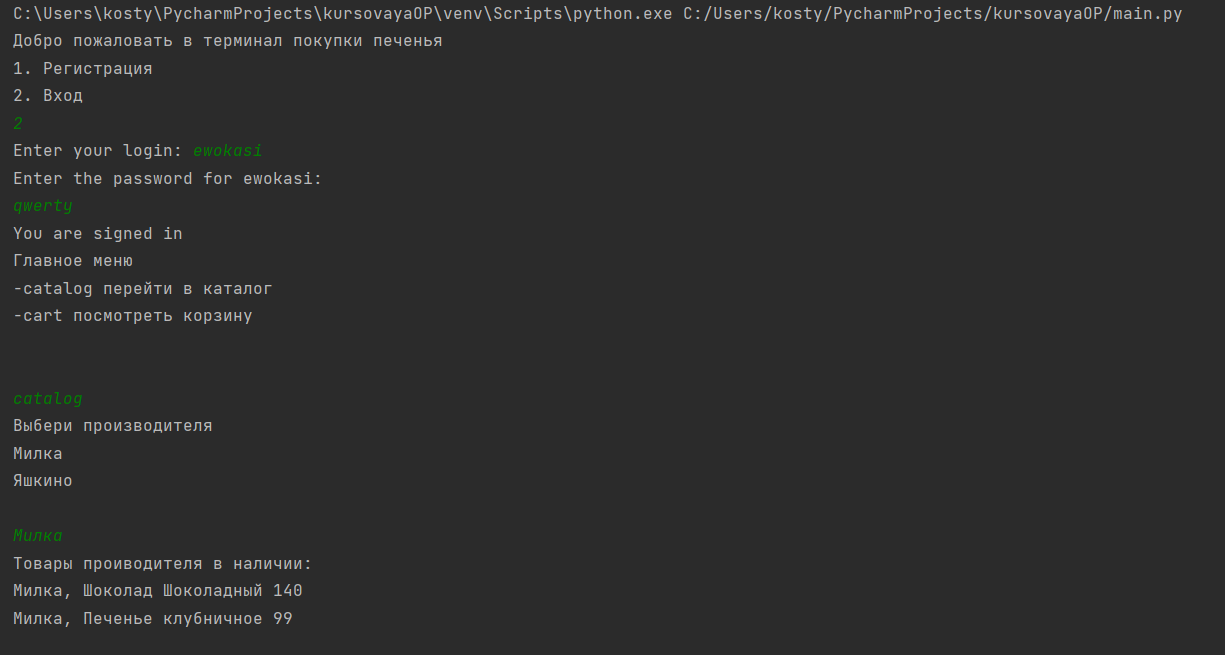
(название производителя) – для переъода к товару производителя

Cart – для перехода в корзину покупателя

Book – забронировать товар

Cancel book -отменить бронь товара

## **Результат работы программы**



## **Выводы**

Мной была реализована программа для ведения онлайн магазина, которая осуществляет менеджмент на складе и автоматизирует процесс заказа и бронирования товара.

Из достоинств можно ответить наличие интерфейса и наглядность работы

Недостатков в ходе проверки выявлено не было

## **Список литературы**

1. Ключарев А.А., Матьяш В.А., Щекин С.В. Структуры и алгоритмы обработки данных: Учебное пособие / СПбГУАП. СПб., 2004.

2. Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие / Колдаев В.Д; Под ред. проф.Л.Г. Гагариной - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2016. - 416 с. <http://znanium.com/catalog.php?bookinfo=537513>

6. Кнут, Д. Искусство программирования [Текст] = The art of computer programming : [в 3 т.]. Т. 1. Основные алгоритмы / Д. Кнут ; ред. Ю. В. Козаченко. - 3-е изд. - М. : Вильямс, 2014. - 720 с.

8. Вирт, Н Алгоритмы и структуры данных. Новая версия для Оберона + CD [Текст] / Н. Вирт ; пер. Д. Б. Подшивалов. - 2-е изд., испр. - М. : ДМК Пресс, 2012. - 272 с.

## **ПРИЛОЖЕНИЕ А. Код проекта**

import openpyxl  
  
  
def separator(string, prefix='/', slicer=", "):  
 *"""  
 Function for making a dictionary that would be comfortable to use with catalog\_controller* ***:param*** *string: the input line that would be separated on name, price and count* ***:param*** *prefix: bot prefix that deletes command characters from line (default '/')* ***:param*** *slicer: char set that would be sliced on parts (default ", ")  
 """* if prefix != None:  
 string = string.replace(prefix + ' ', '', 1)  
 string = string.split(slicer)  
 data = {  
 'maker': string[0].upper(),  
 'taste': string[1],  
 'form': None,  
 'price': 0,  
 'count': 1,  
  
 }  
  
 try:  
 if data['price'] < int(string[3]): data['price'] = int(string[3])  
 except Exception:  
 print(f"price по умолчанию {data['price']}")  
  
 try:  
 if data['count'] < int(string[4]):  
 data['count'] = int(string[4])  
 except Exception:  
 print(f"count по умолчанию {data['count']}")  
  
 print(data)  
 return data  
  
  
def catalog\_add(maker, taste, form, price=0, count=1, path="databases/catalog.xlsx"):  
  
 wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
  
 is\_found = 0  
 for i in range(1, sheet.max\_row + 1):  
 if maker in str(sheet.cell(i, 1).value) and taste in str(sheet.cell(i, 2).value):  
 sheet.cell(i, 5).value = int(sheet.cell(i, 5).value) + count  
 is\_found = 1  
 break  
  
 if is\_found == 0:  
 placeholder = sheet.max\_row + 1  
 sheet.cell(placeholder, 1).value = maker  
 sheet.cell(placeholder, 2).value = taste  
 sheet.cell(placeholder, 3).value = form  
 sheet.cell(placeholder, 4).value = price  
 sheet.cell(placeholder, 5).value = count  
  
 wb.save(path)  
 wb.close()  
  
  
def catalog\_del(maker, taste, path="databases/catalog.xlsx"):  
 *"""  
 Deletes row from catalog* ***:param*** *maker:* ***:param*** *taste:* ***:param*** *path:* ***:return****: maker and taste string  
 """* wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
 info = 0  
 for i in range(1, sheet.max\_row + 1):  
 if str(maker).lower() in str(sheet.cell(i, 1).value).lower() and str(taste).lower() in str(  
 sheet.cell(i, 2).value).lower():  
 workcell = i  
 info = str(sheet.cell(workcell, 1).value)  
 sheet.cell(workcell, 1).value = ''  
 sheet.cell(workcell, 2).value = ''  
 sheet.cell(workcell, 3).value = ''  
 sheet.cell(workcell, 4).value = ''  
 sheet.cell(workcell, 5).value = ''  
 break  
  
 wb.save(path)  
 wb.close()  
 return info  
  
  
def catalog\_sell(maker, taste, count=1, path="databases/catalog.xlsx"):  
 *"""  
 Decreases count of product, if last product - deletes row* ***:param*** *maker:* ***:param*** *taste:* ***:param*** *count:* ***:param*** *path:* ***:return****: maker and taste string  
 """* wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
 found = 0  
 print(taste)  
 # уменьшение количетва, если последний товар - удаление  
 for i in range(1, sheet.max\_row + 1):  
 if str(maker).lower() in str(sheet.cell(i, 1).value).lower() and str(taste).lower() in str(  
 sheet.cell(i, 2).value).lower():  
 workcell = i  
 found = f"{str(sheet.cell(workcell, 1).value)} {str(sheet.cell(workcell, 2).value)}"  
  
 sheet.cell(workcell, 5).value = int(sheet.cell(workcell, 5).value) - count  
 break  
  
 wb.save(path)  
 wb.close()  
 return found  
  
  
def remove\_last\_pos(path="databases/catalog.xlsx"):  
 *"""  
 Function removes last product row from the catalog* ***:param*** *path: path to excel table (default "catalog.excel")* ***:return****:  
 """* wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
 workcell = sheet.max\_row  
  
 sheet.cell(workcell, 1).value = ''  
 sheet.cell(workcell, 2).value = ''  
 sheet.cell(workcell, 3).value = ''  
 sheet.cell(workcell, 4).value = ''  
 sheet.cell(workcell, 5).value = ''  
  
 wb.save(path)  
 wb.close()  
  
  
def get\_makers(path="databases/catalog.xlsx", column=1):  
 *"""  
 Get all unic makers from xlxs (default - 1 column of each row)* ***:param*** *path:* ***:return****: list of makers  
 """* wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
 workcell = sheet.max\_row  
 makers = []  
 for i in range(2, workcell + 1):  
 maker = sheet.cell(i, column).value  
 if maker not in makers and maker is not None:  
 makers.append(str(sheet.cell(i, column).value))  
  
 wb.close()  
 return makers  
  
  
def get\_products(maker, path="databases/catalog.xlsx"):  
 wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
 workcell = sheet.max\_row  
 products = []  
  
 for i in range(2, workcell + 1):  
 if sheet.cell(i, 1).value == maker:  
 if int(sheet.cell(i, 5).value) > 0:  
 product = {  
 "maker": sheet.cell(i, 1).value,  
 "taste": sheet.cell(i, 2).value,  
 "form": sheet.cell(i, 3).value,  
 "price": sheet.cell(i, 4).value,  
 "count": sheet.cell(i, 5).value  
 }  
 if product not in products:  
 products.append(product)  
  
 return products  
  
  
def find\_product(maker, taste, path='databases/catalog.xlsx'):  
 wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
 info = {}  
 for i in range(1, sheet.max\_row + 1):  
 if str(maker).lower() in str(sheet.cell(i, 1).value).lower() and str(taste).lower() in str(  
 sheet.cell(i, 2).value).lower():  
 info['maker'] = sheet.cell(i,1).value  
 info['taste'] = sheet.cell(i, 2).value  
 info['form'] = sheet.cell(i, 3).value  
 info['price'] = sheet.cell(i, 4).value  
 info['count'] = sheet.cell(i, 5).value  
 wb.close()  
 return info  
 return 'Not Found'  
  
  
def catalog\_get(path='databases/catalog.xlsx'):  
 wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
 catalog = []  
 for i in range(2, sheet.max\_row + 1):  
  
 if str(type(sheet.cell(i,1).value)) != "<class 'NoneType'>":  
 product = {  
 'maker': str(sheet.cell(i, 1).value),  
 'taste': str(sheet.cell(i, 2).value),  
 'form': str(sheet.cell(i, 3).value),  
 'price': int(sheet.cell(i, 4).value),  
 'count': int(sheet.cell(i, 5).value)  
 }  
 catalog.append(product)  
 wb.close()  
 return catalog  
  
  
def catalog\_change\_count(maker, taste, num,path ='databases/catalog.xlsx'):  
 wb = openpyxl.load\_workbook(path)  
 sheet = wb.active  
 for i in range(1, sheet.max\_row + 1):  
 if str(maker).lower() in str(sheet.cell(i, 1).value).lower() and str(taste).lower() in str(  
 sheet.cell(i, 2).value).lower():  
 sheet.cell(i, 5).value = int(sheet.cell(i, 5).value) + num  
  
 product = {  
 'maker': str(sheet.cell(i, 1).value).upper(),  
 'taste': str(sheet.cell(i, 2).value),  
 'form': str(sheet.cell(i, 3).value),  
 'price': int(sheet.cell(i, 4).value),  
 'count': int(sheet.cell(i, 5).value)  
 }  
 wb.save(path)  
 wb.close()  
 return product

Листинг client\_hub

import json  
from random import randint  
  
def hash(password):  
 res = 1  
 password = str(password)  
 for i in range(0, len(password)):  
 res = res \* int(password[i])\*int(password[i])  
 return res  
  
def registrate(login, password):  
  
 with open("databases/clients.json", 'r') as file:  
 db = json.load(file)  
 clients = db['clients']  
  
 new\_user = {  
 "login": str(login),  
 "password": str(hash(password)),  
 "cart": []  
 }  
  
 is\_found = 0  
 for client in clients:  
 if new\_user["login"] == client["login"]:  
 is\_found = 1  
  
  
 if is\_found:  
 print("User is already registred")  
 return new\_user  
 else:  
 clients.append(new\_user)  
 db['clients']=clients  
 with open("databases/clients.json", 'w') as file:  
 json.dump(db, file, indent=2)  
  
 with open("databases/clients.json", 'r') as file:  
 db = json.load(file)  
 clients = db['clients']  
  
 registred = 0  
 for client in clients:  
 if new\_user["login"] == client["login"]:  
 registred = 1  
 break  
  
 if registred:  
 print("Registration is succesful")  
  
 return new\_user  
  
def sign\_in():  
 with open("databases/clients.json", 'r') as file:  
 db = json.load(file)  
 clients = db['clients']  
 login = input("Enter your login: ")  
 registred = 0  
 for client in clients:  
 if login == client["login"]:  
 registred = 1  
 break  
  
 if registred:  
 print(f"Enter the password for {login}: ")  
 password = input()  
 password = str(hash(password))  
 for client in clients:  
 if login == client["login"]:  
 if password == client["password"]:  
 print("You are signed in")  
 return {'login': login, 'password': password}  
 else:  
 print("password is incorrect")  
 else:  
 print("No such user. Try one more time")  
  
  
def get\_cart(login):  
 with open("databases/clients.json", 'r') as file:  
 db = json.load(file)  
 clients = db['clients']  
  
 for client in clients:  
 if login == client["login"]:  
 return client['cart']  
  
def clear\_cart(login):  
  
 with open("databases/clients.json", 'r') as file:  
 db = json.load(file)  
 clients = db['clients']  
  
 for client in clients:  
 if client["login"] == login:  
 client["cart"] = []  
 db["clients"] = clients  
 with open("databases/clients.json", 'w') as file:  
 json.dump(db, file, indent=2)  
 break  
  
def add\_to\_cart(login, call):  
  
 with open("databases/clients.json", 'r') as file:  
 db = json.load(file)  
 clients = db['clients']  
  
 maker = call.split(sep=", ")[0]  
 taste = call.split(sep=", ")[1]  
  
 offer = {  
 "login": login,  
 "cart": [  
 {  
 "maker": maker,  
 "taste": taste  
 }  
 ]  
  
 }  
  
 # Если заказ первый, и если второй  
 is\_found = 0  
 for client in clients:  
 if offer["login"] == client["login"]:  
 client["cart"].append(offer["cart"][0])  
 db["clients"] = clients  
 with open("databases/clients.json", 'w') as file:  
 json.dump(db, file, indent=2)  
  
 is\_found = 1  
 break  
  
 if is\_found == 0:  
 clients.append(offer)  
 db["clients"] = clients  
 with open("databases/clients.json", 'w') as file:  
 json.dump(db, file, indent=2)  
  
 return offer  
  
def add\_book(login):  
 # сделать бронь товара - перенести карт юзера из клинт в букс  
  
 with open("databases/books.json", 'r') as file:  
 books\_db = json.load(file)  
  
 with open("databases/clients.json", 'r') as file:  
 clients\_db = json.load(file)  
 clients = clients\_db["clients"]  
  
 for client in clients:  
 if login == client["login"]:  
 cart = client["cart"]  
 break  
  
 client = {  
 "login": login,  
  
 "key": str(login[0].upper()) + "-" + str(randint(100, 999)),  
 "cart": cart  
 }  
  
 with open("databases/books.json", "w") as file:  
 books\_db["books"].append(client)  
 json.dump(books\_db, file, indent=2)  
 clear\_cart(login)  
 return client  
  
def cancel\_book(key):  
 with open("databases/books.json", 'r') as file:  
 db = json.load(file)  
 books = db['books']  
  
 for book in books:  
 if key == book['key']:  
 deleted = book  
 books.remove(book)  
 db["books"] = books  
 with open("databases/books.json", 'w') as file:  
 json.dump(db, file, indent=2)  
  
 return deleted  
  
  
def get\_books(login):  
 with open("databases/books.json", 'r') as file:  
 db = json.load(file)  
 books = db['books']  
  
 output = []  
 for book in books:  
 key = str(book['key'])  
 username = str(book["login"])  
 cart = str(book['cart'])  
 data = username + "\n" + key + "\n" + cart + "\n"  
 if username == login:  
 output.append(data)  
  
 return output

## Branch\_hub

Для реализации проекта в терминале я создал дополнительную библиотеку branch\_hub, которая написана только для удобства проектирования и разгрузки main.py

В этом модуле написан GUI для каждого ключевого слова, с помощью которого реализуется управление магазином. Модуль содержит только одну функцию branch\_render(), которая выводит ответ на запрос пользователя

## Листинг branch\_hub

import catalog\_hub  
import client\_hub  
  
def hash(password):  
 res = 1  
 password = str(password)  
 for i in range(0, len(password)):  
 res = res \* int(password[i]) \* int(password[i])  
 return res  
  
def branch\_render(call, user):  
 call\_kw = {'mainmenu': "mainmenu",  
 'catalog': "catalog",  
 'cart': "cart"  
 }  
  
 makers\_kw = {}  
 tastes\_kw = {}  
 q=1  
 for item in catalog\_hub.catalog\_get():  
 tastes\_kw[f'taste{q}']=f"{item['maker']}, {item['taste']}"  
 q+=1  
  
 q = 1  
 for key in catalog\_hub.get\_makers():  
 makers\_kw[f'{str("maker") + str(q)}'] = key  
 call\_kw[f'{str("maker") + str(q)}'] = key  
 q = q + 1  
  
  
 answer = "Error, try again"  
 if call == call\_kw['mainmenu']:  
 answer = "Главное меню\n-catalog перейти в каталог\n-cart посмотреть корзину\nto cart <product> - Чтобы добавить товар в корзину \nbook - сделать бронь вашей корзины\n"  
  
 elif call == call\_kw['catalog']:  
 answer = f"Выбери производителя\n"  
 for maker in makers\_kw.values():  
 answer = answer + str(maker) + "\n"  
  
 elif call in makers\_kw.values():  
 answer = "Товары проиводителя в наличии:\n"  
 products = catalog\_hub.get\_products(call)  
 count = 1  
 for product in products:  
  
 answer = answer + product['maker'] + ", " + product["taste"] + " " + str(product['price']) + '\n'  
 count += 1  
 answer = answer + "\n-catalog назад"  
 elif call == call\_kw["cart"]:  
 cart = client\_hub.get\_cart(user["login"])  
 products = catalog\_hub.get\_products(call)  
 price = 0  
 if cart:  
 answer = ''  
 for item in cart:  
 answer = answer + item['maker']+" "+ item['taste'] + " "+str(catalog\_hub.find\_product(item['maker'], item['taste'])["price"])+"р\n"  
 pr = catalog\_hub.find\_product(item['maker'], item['taste'])  
 price = price + pr["price"]  
 answer = answer+f"\nОбщая стоимость {price}р"  
 else:  
 answer = "Ваша корзина пуста"  
  
 elif call in tastes\_kw.values():  
  
 call = call.split(sep=", ")  
 product\_call = {  
 "maker": call[0],  
 "taste": call[1]  
 }  
  
 product= catalog\_hub.find\_product(product\_call["maker"], product\_call["taste"])  
 if product!="Not Found":  
 answer = f"{product['taste']} \nЦена: {product['price']}\nКоличество: {product['count']}"  
  
 elif "to cart" in call:  
 call= call.replace("to cart ", '',1)  
 client\_hub.add\_to\_cart(user['login'], call)  
 answer = f"Добавлено в вашу корзину {call}"  
  
 elif call == "book":  
 info = client\_hub.add\_book(user['login'])  
 price = 0  
 products = catalog\_hub.get\_products(call)  
  
 for item in info['cart']:  
 maker = item['maker']  
 taste = item['taste']  
 catalog\_hub.catalog\_change\_count(maker, taste, -1)  
 price = price+ catalog\_hub.find\_product(maker, taste)["price"]  
 answer = f"Ваш номер заказа {info['key']}. Сумма заказа: {price}"  
  
 elif call =="clear cart":  
 if client\_hub.get\_cart():  
 client\_hub.clear\_cart(user['login'])  
 answer = "Ваша корзина очищена"  
 else:  
 answer = "Ваша корзина уже пуста"  
  
  
 elif "cancel book" in call :  
 call = call.replace("cancel book ", '', 1)  
 info= client\_hub.cancel\_book(call)  
  
 for item in info['cart']:  
 maker = item['maker']  
 taste = item['taste']  
 catalog\_hub.catalog\_change\_count(maker, taste, 1)  
  
 answer=f"Ваш заказ {call} отменен"  
  
 elif call == "show books":  
 info = client\_hub.get\_books(user['login'])  
 answer ="Заказы\n"  
 for i in info:  
 answer = answer+ i+"\n"  
  
 print(answer)

## Main

Главный модуль main, который позволяет зарегистрировать пользователя и реализует вызов brach\_render()

## Листинг main

import client\_hub  
import branch\_hub  
  
def hash(password):  
 res = 1  
 password = str(password)  
 for i in range(0, len(password)):  
 res = res \* int(password[i])\*int(password[i])  
 return res  
  
print("Добро пожаловать в терминал покупки печенья")  
print("1. Регистрация\n2. Вход")  
  
decide = int(input())  
if decide == 1:  
 login = input("Enter login")  
 psw = input("Enter password")  
  
 client\_hub.registrate(login, psw)  
 user = {  
 "login": login,  
 "password": psw  
 }  
  
elif decide == 2:  
 user = client\_hub.sign\_in()  
  
if user:  
  
 branch\_hub.branch\_render("mainmenu", user)  
 while 1:  
  
 branch = input()  
 branch\_hub.branch\_render(branch, user)

Оглавление

[«Онлайн магазин» 1](#_Toc122014765)

[по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ 1](#_Toc122014766)

[**Описание задачи** 2](#_Toc122014767)

[**Описание функций программы** 2](#_Toc122014768)

[**Первая библиотека catalog\_hub.** 2](#_Toc122014769)

[**Вторая библиотека client\_hub** 2](#_Toc122014770)

[**Описание функций** 2](#_Toc122014771)

[**Client\_hub.registrate(login, psw);** 3](#_Toc122014772)

[**Client\_hub.sign\_in()** 3](#_Toc122014773)

[**Branch\_hub.branch\_render(callback, user)** 3](#_Toc122014774)

[**Hash(password)** 4](#_Toc122014775)

[Catalog\_hub.Catalog\_add(maker, taste, form, price, count) 4](#_Toc122014776)

[Catalog\_hub.Catalog\_del(maker, taste) 5](#_Toc122014777)

[**Добавление товара в корзину** 5](#_Toc122014778)

[**Добавление брони товара для пользователя** 6](#_Toc122014779)

[**Описание пользовательского интерфейса** 7](#_Toc122014780)

[**Результат работы программы** 8](#_Toc122014781)

[**Выводы** 8](#_Toc122014782)

[**Список литературы** 8](#_Toc122014783)

[**ПРИЛОЖЕНИЕ А. Код проекта** 9](#_Toc122014784)

[Branch\_hub 20](#_Toc122014785)

[Листинг branch\_hub 20](#_Toc122014786)

[Main 24](#_Toc122014787)

[Листинг main 24](#_Toc122014788)