

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

должность, уч. степень, звание

подпись, дата

А.В. Туманова

инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ
Разработка программы
«Онлайн магазин»

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4134к

подпись, дата

Костяков Н.А.

инициалы, фамилия

Санкт-Петербург 2022

СОДЕРЖАНИЕ

Содержание проекта	2
Программные средства и инструменты реализации.....	2
1. Описание задачи	2
2. Описание структур данных	2
3. Описание функций программы.....	3
4. Описание пользовательского интерфейса.....	5
5. Результат тестирования программы	5
Выводы	7
Список литературы.....	7
ПРИЛОЖЕНИЕ А. Код проекта	7

Содержание проекта

Разработка инструментов для ведения онлайн магазина, которые автоматизирует процесс введения переписи склада и менеджмента пользовательской информации.

Программа реализована на языке программирования Python под операционную систему Windows и Linux.

Программные средства и инструменты реализации

Среда разработки PyCharm

Сторонние библиотеки: openpyxl, json, random

Язык программирования Python 3.8

1. 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.

Разработка ведется в среде программирования PyCharm.

2. Описание структур данных

Данные о товаре хранятся в таблице `xlsx`. Выбор такой СУБД обоснован тем, чтобы владельцу магазина было удобно вносить как через модули для Python, например с

телефона при подключении модулей к арі телеграм бота, так и напрямую через доступ к БД

	A	B	C	D	E
1	Производитель	Вкус	Тип	Цена	Кол-во
2	Милка	Шоколад Шоколадный	Печенье	140	123
3	Милка	Печенье клубничное	Шоколад	99	8
4	Яшкино	Персиковое печенье	Печенье	100	40
5	Яшкино	Яблочный шоколад	Шоколад	70	32
6	Красный Октябрь	Аленка	Шоколад	70	32

Выбор БД для хранения информации о пользователях основан на возможности JSON формата удобно добавлять нового пользователя и прочих атрибутов без необходимости изменения всех структуры БД

```
{
  "clients": [
    {
      "login": ""ewokasi"",
      "password": "100",
      "cart": [
        {
          "maker": "\u0041\u0040\u0030\u0041\u003d\u004b\u0039 \u001e\u0043\u0042\u004f\u0031\u0040\u004c",
          "taste": "\u0040\u003b\u0035\u003d\u0043\u004a\u0030"
        }
      ]
    },
    {
      "login": ""tabletka"",
      "password": "0",
      "cart": []
    }
  ]
}
```

3. Описание функций программы

Первая библиотека catalog_hub.

Библиотека представляет из себя инструмент управления базой данных excel, в которой организатор ведет учет товара. Исходный код приведен в приложении

Вторая библиотека client_hub

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

Исходный код приведен в приложении

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

3.1 Client_hub.registrate(login, psw);

Осуществляет запись нового пользователя в json файл для дальнейшей авторизации
На вход просит логин и пароль

3.2 Client_hub.sign_in()

Дает возможность зарегистрированному пользователю авторизоваться в системе
На вход просит логин и пароль

3.3 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.

3.4 Hash(password)

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

Управление структурой, в которой хранится информация о товаре

3.5 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

3.6 Catalog_hub.Catalog_del(maker, taste)

Реализует удаление товара по его ключам

В программе реализовано автоматическое управление базой данных клиентов

3.7 Добавление товара в корзину

На вход требует логин и информацию о товаре

3.8 Добавление брони товара для пользователя

На вход требует логин пользователя, для доступа к корзине

4. Описание пользовательского интерфейса

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

Catalog – для перехода в секцию товаров

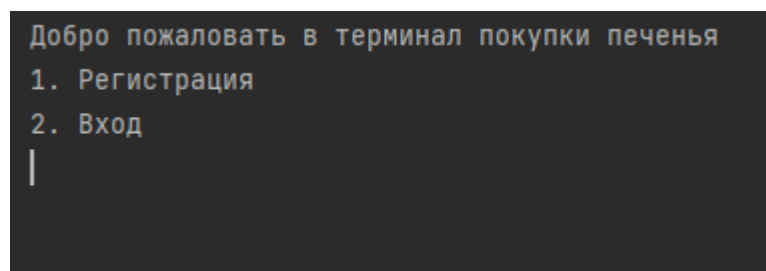
(название производителя) – для перебеода к товару производителя

Cart – для перехода в корзину покупателя

Book – забронировать товар

Cancel book -отменить бронь товара

5. Результат тестирования программы



```
Добро пожаловать в терминал покупки печенья
1. Регистрация
2. Вход
|
```

Рис.1 Уведомление на входе

```
You are signed in
Главное меню
-catalog перейти в каталог
-cart посмотреть корзину
to cart <product> - Чтобы добавить товар в корзину
book - сделать бронь вашей корзины
```

Рис.2 Главное меню

```
catalog
Выбери производителя
Милка
Яшкино
Красный Октябрь

Милка
Товары проиводителя в наличии:
Милка, Шоколад Шоколадный 140
Милка, Печенье клубничное 99

-catalog назад
```

Рис. 3 Каталог товаров

```
cart
Красный Октябрь Аленка 70р
Милка Шоколад Шоколадный 140р

Общая стоимость 210р
|
```

Рис.4 Корзина покупателя

```
Ваш номер заказа E-802. Сумма заказа: 210
|
```

Рис.5 Статус бронирования заказа

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., Матяш 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

Catalog_hub

```

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 количества, 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 xlsx (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 буюс

    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"])+ "p\n"
                pr = catalog_hub.find_product(item['maker'], item['taste'])
                price = price + pr["price"]
            answer = answer+f"\nОбщая стоимость {price}p"
        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)
```

