Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Тема: Разработка системы классов для приложения «Склад»**

**Специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы 302ИС-22** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **А.С. Маршинина** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Л.Б. Гусятинер** |

**Москва 20****24**

Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**УТВЕРЖДАЮ**

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Специальность 09.02.07 «Информационные системы и программирование»**

**Студент группы 302ИС-22 Алиса Маршинина**

**ТЕМА: Разработка системы классов для приложения «Склад»**

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Срок сдачи проекта «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

**Москва 2024**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc185467722)

[1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc185467723)

[1.1 Введение в предметную область 5](#_Toc185467724)

[1.2 Анализ готовых решений 6](#_Toc185467725)

[2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 7](#_Toc185467726)

[2.1 Общее назначение системы 7](#_Toc185467727)

[2.2 Требования к функциям, выполняемым системой 7](#_Toc185467728)

[2.3 Диаграмма прецедентов 8](#_Toc185467729)

[2.4 Диаграмма классов 8](#_Toc185467730)

[2.5 Проектирование сценария 10](#_Toc185467731)

[2.6 План тестирования 10](#_Toc185467732)

[3 ОПИСАНИЕ СРЕДЫ РАЗРАБОТКИ 11](#_Toc185467733)

[3.1 Выбор и обоснование программных инструментов 11](#_Toc185467734)

[3.1.1 Выбор программных инструментов 11](#_Toc185467735)

[3.1.2 Обоснование выбора инструментария разработки 12](#_Toc185467736)

[3.2 Реализация проекта системы 13](#_Toc185467737)

[3.2.1 Описание работы модулей 13](#_Toc185467738)

[3.2.2 Реализация пользовательского интерфейса программы 14](#_Toc185467739)

[3.2.3 Результаты работы и тестирования 18](#_Toc185467740)

[3.3 To-Do Лист 19](#_Toc185467741)

[ЗАКЛЮЧЕНИЕ 22](#_Toc185467742)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc185467743)

[ПРИЛОЖЕНИЕ 1 25](#_Toc185467744)

[ПРИЛОЖЕНИЕ 2 27](#_Toc185467745)

[ПРИЛОЖЕНИЕ 3 30](#_Toc185467746)

[ПРИЛОЖЕНИЕ 4 39](#_Toc185467747)

[ПРИЛОЖЕНИЕ 5 40](#_Toc185467748)

[ПРИЛОЖЕНИЕ 6 41](#_Toc185467749)

[ПРИЛОЖЕНИЕ 7 43](#_Toc185467750)

[ПРИЛОЖЕНИЕ 8 44](#_Toc185467751)

[ПРИЛОЖЕНИЕ 9 46](#_Toc185467752)

[ПРИЛОЖЕНИЕ 10 50](#_Toc185467753)

**ВВЕДЕНИЕ**

В современном мире предъявляются высокие требования к качеству и скорости, а также эффективности управления бизнес-процессами. Особенно это актуально для предприятий в сфере общественного питания, где эффективность управления складскими запасами является ключевым элементом для обеспечения постоянной работы предприятия и удовлетворения потребности посетителей. Однако малые предприятия часто сталкиваются с проблемами ручного учёта, низкой эффективностью и прозрачностью процессов, а также отсутствием доступных систем автоматизации управления запасами.

Актуальность данной темы обусловлена необходимость разработки простой и функциональной системы, которая способна автоматизировать учёт складских операций, минимизировать ошибки, связанные с человеческим фактором, а также предоставлять аналитические данные для улучшения эффективности работы предприятия. Автоматизация процессов управления складом позволит сократить время на обработку данных и обеспечить возможность быстрого реагирования на изменения в потребностях клиентов.

Объектом исследования является склад кофейни, где хранятся ингредиенты для приготовления напитков. Предметом исследования являются процессы учета, анализа и прогнозирования данных о запасах и продажах.

Целью данного курсового проекта является разработка системы классов для приложения склада, которая обеспечит предприятию автоматизацию учета складских запасов и анализ продаж, а также прогнозирование потребности в пополнении запасов склада. Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Изучить особенности складов в кофейнях и провести анализ существующих решений в данной области.
2. Разработать структуру хранилища данных, которая отражает главные аспекты учета запасов и продаж.
3. Реализовать функционал для анализа данных о продажах и прогнозирования потребности в пополнении запасов.
4. Создать пользовательский интерфейс, который обеспечит доступ к аналитическим данным и отчетам для различных ролей пользователей (администратор, менеджер, кассир).
5. Провести тестирование разработанной системы, чтобы убедиться в правильности ее работы и надежности использования.

Результаты данной работы позволят создать инструмент, который способен повысить эффективность управления складскими запасами кофейни, снизить возможность дефицита или профицита запасов, а также предоставить пользователям удобные инструменты для анализа данных и принятия решений. Таким образом, разработка системы классов для приложения «Склад» является ключевым шагом в развитии бизнес-процессов и повышению общей производительности предприятия.

# **1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Введение в предметную область**

Управление складскими запасами составляет основу работы каждого предприятия в сфере общественного питания. Для кофейни правильное ведение учета склада существенно влияет на качество обслуживания клиентов, избежание дефицита и профицита запасов.

История автоматизации складских операций связана с развитием ERP-систем, но эти решения зачастую недоступны малым предприятиям из-за высокой стоимости использования этих систем и сложности их внедрения.

Современные инструменты и подходы к автоматизации складов включают в себя использование специализированных баз данных, программ учёта и анализа, а также интерфейсов, которые подходят для различных ролей пользователей.

Законодательная база, регулирующая разработку программных продуктов в этой области, включает:

1. Федеральный закон «Об информации, информационных технологиях и о защите информации» от 27.07.2006 №149-ФЗ: обеспечивает правовые основы работы с информацией и ее защиты. [1]
2. ГОСТ Р ИСО/МЭК 12207-2010: описывает процессы жизненного цикла программного обеспечения, включая этапы разработки, тестирования и сопровождения программного продукта. [2]
3. Федеральный закон «О персональных данных» от 27.07.2006 №152-ФЗ: регулирует обработку и защиту персональных данных. [3]

Методология разработки системы складского учета включает такие этапы, как:

1. Анализ предметной области и требований пользователей.
2. Разработка системы классов для базы данных склада.
3. Создание функционального интерфейса для пользователей с различными ролями (администратор, менеджер, кассир).
4. Реализация системы анализа и отчетности.
5. Тестирование и отладка.

## **1.2 Анализ готовых решений**

На рынке представлено множество систем автоматизации управления складом, включая ERP-системы и специализированные программы:

1. ЕКАМ – облачная система управления торговлей, помогает предпринимателям автоматизировать и оптимизировать процессы в бизнесе. Спектр инструментов и функций подходит для предприятий розничной торговли, общественного питания и сферы услуг. Минусы: необходимость обучения работе с системой, возникновение технических проблем, особенно на бесплатных тарифах, узкий функционал бесплатных тарифов, невозможно управлять доставкой.
2. LiteBox – облачное программное обеспечение для учёта и оптимизации торговли. Подходит малому бизнесу в сфере розничной торговли. Минусы: нет ведения клиентской базы в приложении, сложность освоения и первичной настройки.
3. МойСклад – облачное программное обеспечение для управления складом и торговлей оптом и в розницу. Подходит малому и среднему бизнесу, включая производство, торговлю и услуги. Минусы: сложности с технической поддержкой в случае обращения.

Проведенный анализ показывает, что готовые решения часто оказываются сложны в освоении или не адаптированы под специфику кофейного бизнеса. Это является обоснованием необходимости разработки системы, которая будет учитывать особенности управления складом кофейни и обеспечит удобный интерфейс для всех ролей пользователей.

# **2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ**

## **2.1 Общее назначение системы**

Разрабатываемая система классов для приложения «Склад» предназначена для автоматизации управления запасами склада кофейни, анализа данных о продажах, прогнозирования необходимости пополнения запасов на складе и формирования отчетных графиков. Система предоставляет пользователям с различными ролями доступ к соответствующей информации о складе.

Основными целями системы являются:

1. Улучшение эффективности управления складскими запасами.
2. Упрощение аналитики продаж и остатков.
3. Создание простого и удобного интерфейса для взаимодействия пользователей с системой, в зависимости от их роли.
4. Возможность экспорта данных в Excel для их дальнейшего использования в составлении отчетов.

Система ориентирована на три категории пользователей: администратора, кассира и менеджера. Каждая роль имеет доступ только к определенным функциям, которые ограничивает система.

## **2.2 Требования к функциям, выполняемым системой**

В соответствии с анализом предметной области выделены специальные функциональные возможности системы:

* Работа с хранилищем данных складских запасов и напитков: добавление, обновление и удаление данных.
* Отображение аналитических графиков, создаваемых на основе данных о запасах и продажах.
* Экспорт данных в файлы Excel в виде таблиц DataFrame.

## **2.3 Диаграмма прецедентов**

В данном разделе на рисунке 1 (Приложение 1) представлена диаграмма прецедентов, которая отображает основные функции для каждой роли и их взаимосвязи.

Функции для каждой роли:

1. Администратор (admin)

* Регистрация новых пользователей и назначение ролей для них
* Удаление существующих пользователей
* Доступ ко всем аналитических графикам (график популярности напитков и график необходимости пополнения запасов на складе)
* Не может: экспортировать данные в Excel для дальнейшего ведения отчетности

1. Кассир (cashier)

* Доступ к графику необходимости пополнения запасов на складе
* Экспорт данных в Excel
* Не может: регистрировать и удалять пользователей, назначать роли, а также просматривать график популярности напитков

1. Менеджер (manager)

* Доступ к графику популярности напитков
* Экспорт данных в Excel
* Не может: регистрировать и удалять пользователей, назначать роли, а также просматривать график необходимости пополнения запасов на складе

## **2.4 Диаграмма классов**

В данном разделе на рисунке 2 (Приложение 1) представлены классы системы и их взаимодействие.

Основные сущности:

1. Класс Ingredient:

* Представляет запасы на складе
* Атрибуты: ingredient\_id, ingredient\_name, stock\_quantity
* Методы: update\_stock(amount), to\_dict()

1. Класс Product

* Представляет готовые товары (напитки)
* Атрибуты: product\_id, product\_name, stock\_quantity, ingredient\_requirements
* Методы: update\_stock(amount), to\_dict()

1. Класс Sale

* Представляет данные о продажах
* Атрибуты: product\_id, sold\_units
* Методы: to\_dict()

1. Класс User:

* Представляет пользователей системы
* Атрибуты: username, password, role
* Методы: to\_dict()

1. Класс InventoryManager:

* Представляет управление складом
* Атрибуты: products, ingredients, sales
* Методы: analyze\_inventory\_needs()

1. Класс SalesAnalyzer:

* Представляет анализ продаж товаров
* Атрибуты: sales, products
* Методы: analyze\_popularity()

1. Класс Visualizer:

* Представляет визуализацию графиков
* Методы: plot\_popularity(product\_sales, sort\_by, coolor, show), plot\_inventory(low\_stock\_df, color, show)

## **2.5 Проектирование сценария**

Ниже описаны примеры взаимодействия пользователей с программой.

Сценарий 1: добавление нового пользователя.

1. Администратор запускает приложение и вводит свои учетные данные

* Логин: admin
* Пароль: admin

1. После успешного входа открывается главная страница администратора
2. Администратор выбирает опцию «Добавить нового пользователя»

* Указывает логин нового пользователя, например: manager1
* Задает пароль, например: password123
* Вводит/выбирает роль нового пользователя, например: manager

1. После ввода данных нажимает «Добавить»

Результат: новый пользователь успешно добавлен, и информация сохраняется в файл user.json. Администратор видит подтверждение успешного выполнения операции.

## **2.6 План тестирования**

Цель тестирования – убедиться, что разработанная система работает правильно и соответствует всем требованиям. Это важно для того, чтобы минимизировать ошибки в процессе эксплуатации и обеспечить стабильную работу для всех пользователей с различными правами доступа. Для тестирования был составлен план, в котором подробно описаны основные проверки (Приложение 2).

# **3 ОПИСАНИЕ СРЕДЫ РАЗРАБОТКИ**

## **3.1 Выбор и обоснование программных инструментов**

Проект разработки системы классов для приложения «Склад» был реализован при помощи современных технологий и инструментов, которые обеспечивают надежность и удобство использования приложения. В процессе разработки системы была выбрана интегрированная среда разработки, а также использование специальных библиотек для работы с аналитическими графиками и создания пользовательского интерфейса. Такой подход помог добиться высокой производительности, а также упростить реализацию главных функций приложения.

### **3.1.1 Выбор программных инструментов**

Для разработки системы классов для приложения «Склад» был выбран следующий инструментарий:

1. PyCharm: эта среда разработки предоставляет удобный интерфейс для написания и отладки кода. Кроме того, она содержит встроенные инструменты для анализа кода, что способствует быстрому выявлению ошибок и улучшению качества разрабатываемой системы.
2. Библиотека Tkinter: позволяет создавать окна, кнопки, поля ввода и другие элементы пользовательского интерфейса, обеспечивая простой и удобный способ взаимодействия пользователя с приложением.
3. Библиотека Pandas: ее функционал обеспечивает удобное управление информацией о запасах, продажах и аналитических отчетах, а также позволяет экспортировать данные в Excel.
4. Библиотека Matplotlib: позволяет строить визуальные отчеты, такие как графики популярности товаров или анализа необходимости пополнения запасов.
5. JSON-файлы: используются для хранения и обработки данных. Этот формат обеспечивает удобство и простоту работы с данными, а также их переносимость между различными системами.

### **3.1.2 Обоснование выбора инструментария разработки**

В качестве языка программирования для реализации проекта, посвященному разработке классов для приложения «Склад» рассматривались Python и C. Выбор языка Python обусловлен простотой его синтаксиса и наличием мощных библиотек для работы с данными. Сравнение языков программирования по ключевым критериям приведено в таблице 1.

|  |  |  |
| --- | --- | --- |
| Критерий | Python | C |
| Синтаксис | Простой и читаемый | Более сложный и требует внимательности |
| Продуктивность разработки | Высокая, быстрое написание кода | Низкая, требует больше времени на реализацию |
| Библиотеки для работы с данными | Широкий выбор готовых библиотек | Отсутствие готовых библиотек для работы с данными |
| Работа с JSON | Встроенная поддержка | Требуется сторонняя библиотека |
| Производительность | Удовлетворительна для большинства задач | Высокая |
| Графическая визуализация | Простая реализация с помощью библиотеки Matplotlib | Требуются дополнительные усилия для реализации |
| Среда разработки | PyCharm (удобная настройка, встроенный дебаггер) | Требует настройки окружения, поддержка ограничена |

Таблица 1 – Сравнение языков программирования Python и C

## **3.2 Реализация проекта системы**

Разработанная система состоит из нескольких модулей, реализующих основные функциональные блоки. Эти модули включают: управление пользователями, работу со складом, генерацию аналитических данных, а также пользовательский интерфейс.

### **3.2.1 Описание работы модулей**

Программа состоит из следующих модулей, которые обеспечивают функциональность системы управления складом кофейни:

1. main.py: модуль описывает класс CoffeeShopApp, представляющий основу работы программы с внедрением в него пользовательского интерфейса через Tkinter (см. Листинг 1, Приложение 3).
2. ingredient.py: модуль описывает класс Ingredient, представляющий ингредиенты на складе (см. Листинг 2, Приложение 3). Основные методы:

* update\_stock(amount): обновляет количество ингредиента на складе.
* to\_dict(): преобразует объект в словарь для удобства сохранения данных.

1. product.py: модуль содержит класс Product, который представляет готовые продукты (см. Листинг 3, Приложение 4). Основные методы:

* update\_stock(amount): обновляет количество продукта на складе.
* to\_dict(): преобразует объект в словарь.

1. sale.py: класс Sale используется для хранения информации о продажах продуктов, включая количество проданных единиц (см. Листинг 4, Приложение 4). Основной метод:

* to\_dict(): преобразует объект в словарь.

1. inventory\_manager.py: модуль реализует анализ текущих запасов и их потребностей на складе (см. Листинг 5, Приложение 5). Основные функции:

* analyze\_inventory\_needs(): анализирует потребности в ингредиентах на основе текущих продаж и возвращает таблицу с результатами.

1. sales\_analyzer.py: модуль отвечает за анализ популярности продуктов (см. Листинг 6, Приложение 6). Основной метод:

* analyze\_popularity(): строит рейтинг продуктов по количеству проданных единиц.

1. user.py: модуль используется для хранения информации о данных пользователей и их ролей, зарегистрированных в системе (см. Листинг 7, Приложение 7)
2. utils.py: имеет вспомогательные функции для загрузки и сохранения данных в формате JSON (см. Листинг 8, Приложение 7):

* load\_json\_data(filename) — загрузка данных.
* save\_json\_data(filename, data) — сохранение данных.

1. visualizer.py: реализует построения графиков, которые приведены на рисунке 5 и 6. Код модуля приведен в листинге 9 (Приложение 8). Основной метод:

* plot\_popularity(product\_sales): строит график популярности продуктов.
* plot\_inventory(data): визуализирует потребности в складских запасах.

Помимо этого, наборы данных в системе хранятся в виде файлов JSON, все из них приведены в Листинге 10-13 (Приложение 9).

### **3.2.2 Реализация пользовательского интерфейса программы**

Интерфейс программы реализован с использованием библиотеки Tkinter. Приложение разделено на несколько интерфейсных окон в зависимости от роли пользователя:

Окно входа: при запуске приложения пользователю отображается окно авторизации с полями ввода логина и пароля, а также кнопкой выхода из программы (см. рисунок 1)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1 – Окно авторизации пользователей

Интерфейс администратора: после авторизации под логином и паролем администратора, пользователю отображается окно интерфейса администратора с такими возможностями, как добавление нового пользователя, просмотр графиков продажи товаров и необходимости пополнения запасов, удаление пользователей, а также кнопка выхода в окно авторизации (см. рисунок 2)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2 – Интерфейсное окно администратора

Интерфейс менеджера: после авторизации под логином и паролем менеджера, пользователю отображается окно интерфейса менеджера с такими возможностями, как просмотр графика продажи товаров и экспорт данных в файл Excel, а также кнопка выхода в окно авторизации (см. рисунок 3)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3 – Интерфейсное окно менеджера

Интерфейс кассира: после авторизации под логином и паролем кассира, пользователю отображается окно интерфейса кассира с такими возможностями, как график необходимости пополнения запасов и экспорт этих данных в файл Excel, а также кнопка выхода в окно авторизации (см. рисунок 4)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4 – Интерфейсное окно кассира

Отображение графиков: после успешного входа пользователю представляется отображение графиков. В зависимости от роли пользователя, отображается график продажи товаров и/или необходимости пополнения запасов (см. рисунок 5 и рисунок 6)

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

Рисунок 5 – График аналитики продаж товаров

Изображение выглядит как текст, снимок экрана, дисплей, диаграмма

Автоматически созданное описание

Рисунок 6 – График необходимости пополнения запасов склада

### **3.2.3 Результаты работы и тестирования**

Программа была протестирована с различными сценариями использования. Помимо тест-кейсов в ходе разработки системы классов для приложения были разработаны тесты для некоторых модулей (см. Листинг 14-16, Приложение 10), чтобы убедиться в правильности их работы и отладить возникшие проблемы:.

1. Тестирование авторизации пользователей: тесты из модуля test\_coffeeshop.py проверяют корректность обработки правильных учетных данных и наличие ошибок при вводе некорректных данных.
2. Тестирование управления продуктами: тесты из модуля test\_product.py включают инициализацию объектов продуктов и конвертацию данных в словарь.
3. Тестирование управления запасами и аналитики: тесты из модуля test\_system.py проверяют анализ потребностей в запасах и анализ популярности продуктов.

Основные результаты:

1. Тестирование авторизации: Успешная проверка ввода корректного и некорректного логина/пароля. Ограничение доступа к интерфейсу в зависимости от роли пользователя.
2. Тестирование анализа запасов: Корректное отображение таблицы потребностей ингредиентов при низком уровне запасов. Возможность экспорта данных в Excel (inventory\_report.xlsx).
3. Тестирование анализа продаж: Корректная сортировка продуктов по популярности. Генерация графика популярности и его отображение.
4. Тестирование регистрации и удаления пользователей: Администратор может добавлять/удалять пользователей с проверкой корректности ввода.

## **3.3 To-Do Лист**

На основе анализа нынешней версии программы сформулированы задачи для её дальнейшего улучшения:

1. Реализовать автоматическую отправку уведомлений о необходимости пополнения запасов пользователям с доступными для этого возможностями.
2. Добавить возможность изменения данных о пользователях и их ролях через интерфейс.
3. Добавить возможность изменения данных о продуктах и товарах.
4. Расширить функционал аналитики (например, добавить отчёты по категориям товаров).
5. Интегрировать систему с внешними базами данных для обработки больших объёмов данных.
6. Обновление пользовательского интерфейса и добавление новых виджетов (см. рисунок 7-9)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 7 – Макет популярности продуктов, отсортированных по времени

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8 – Макет для сравнения продаж

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 9 – Макет для просмотра информации о клиентах и их покупках

# **ЗАКЛЮЧЕНИЕ**

В рамках этого проекта была создана система классов для приложения "Склад", предназначенная для анализа данных о продажах товаров и необходимости их пополнения. Основная задача разработки заключалась в создании удобного инструмента, который помогает эффективно управлять запасами, анализировать популярность товаров и определять потребность в их пополнении. Система включает в себя несколько важных компонентов, таких как классы для работы с товарами и продажами, модули для анализа данных и визуализации результатов с помощью графиков.

Особое внимание было уделено тестированию и отладке программы. В процессе тестирования удалось выявить и устранить ошибки, связанные с обработкой данных, расчетом продаж и отображением графиков. Благодаря модульному тестированию разработчики убедились в правильной работе отдельных частей системы, что существенно повысило её надежность. Все выявленные проблемы были исправлены, что обеспечило стабильную и бесперебойную работу приложения.

Результаты проекта представлены в виде наглядных графиков, отражающих популярность товаров и потребность в их пополнении. Это позволяет принимать более обоснованные решения по закупкам и управлению запасами. Разработанный инструмент может быть полезен для анализа данных в бизнесе, связанного с продажами и логистикой.

В будущем систему можно доработать, добавив новые возможности, такие как более сложные алгоритмы анализа, интеграция с внешними базами данных и улучшение пользовательского интерфейса для работы с визуализацией.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Стандарты и законодательные материалы

1. Федеральный закон «Об информации, информационных технологиях и о защите информации» от 27.07.2006 №149-ФЗ

2. Федеральный закон «О персональных данных» от 27.07.2006 №152-ФЗ: регулирует обработку и защиту персональных данных

3. ГОСТ Р ИСО/МЭК 12207-2010 – Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств

4. ГОСТ 19.701-90 (ИСО 5807-85) – «Единая система программной документации. Схемы алгоритмов, программ, данных и систем»

5. ГОСТ Р 50779.11-2000 — «Системы управления запасами. Термины и определения»

6. ГОСТ 7.32-2001 — «Системы обработки информации. Требования к проектированию и разработке баз данных»

7. ГОСТ Р 51901.2-2006 — «Системы менеджмента качества. Требования»

Интернет-источники

8. Хаштамов Адиль Введение в pandas: анализ данных на Python // https://khashtamov.com/ru/pandas-introduction/

9. SkillFactory Pandas // https://blog.skillfactory.ru/glossary/pandas/

10. Балакирев С.В. Рисуем гистограммы, столбчатые и круговые диаграммы // https://proproprogs.ru/modules/matplotlib-risuem-gistogrammy-stolbchatye-i-krugovye-diagrammy

11. PyDocs Модуль Matplotlib // https://pydocs.ru/modul-matplotlib/

12. Балакирев С.В. Добрый, добрый Python с Сергеем Балакиревым // https://stepik.org/course/100707/syllabus

13. Росстат Розничная торговля и общественное питание // https://rosstat.gov.ru/statistics/roznichnayatorgovlya

14. Самойленко Наталья Работа с файлами в формате JSON // https://pyneng.readthedocs.io/ru/latest/book/17\_serialization/json.html

15. Климова Тамара Маркетинговая аналитика: ассортиментный анализ сети кофеен // https://place.lemma.ru/article/marketingovaya-analitika-assortimentnyi-analiz-seti-kofeen

16. inSales 35 лучших программ для складского учёта в 2024 году // https://www.insales.ru/blogs/university/programmy-skladskogo-ucheta

17. Попов Евгений Объектно-ориентированное программирование. Классы и объекты // https://metanit.com/python/tutorial/7.1.php

18. Наталия (authoress на GitHub). Как выбрать подмножество из DataFrame? // http://pandas.geekwriter.ru/getting\_started/intro\_tutorials/03\_subset\_data.html#min-tut-03-subset

19. Tomicevik Dominik & BudiselićMarkoLearn Graph Analytics with Python // https://memgraph.com/blog/learn-graph-analytics-with-python

20. Мустафин Ильдар Финансовый учёт для малого бизнеса // https://www.moedelo.org/club/upravlencheskiy-uchet/upravlencheskii-uchet-dlya-malogo-biznesa

# **ПРИЛОЖЕНИЕ 1**



Рисунок 1 – Диаграмма прецедентов

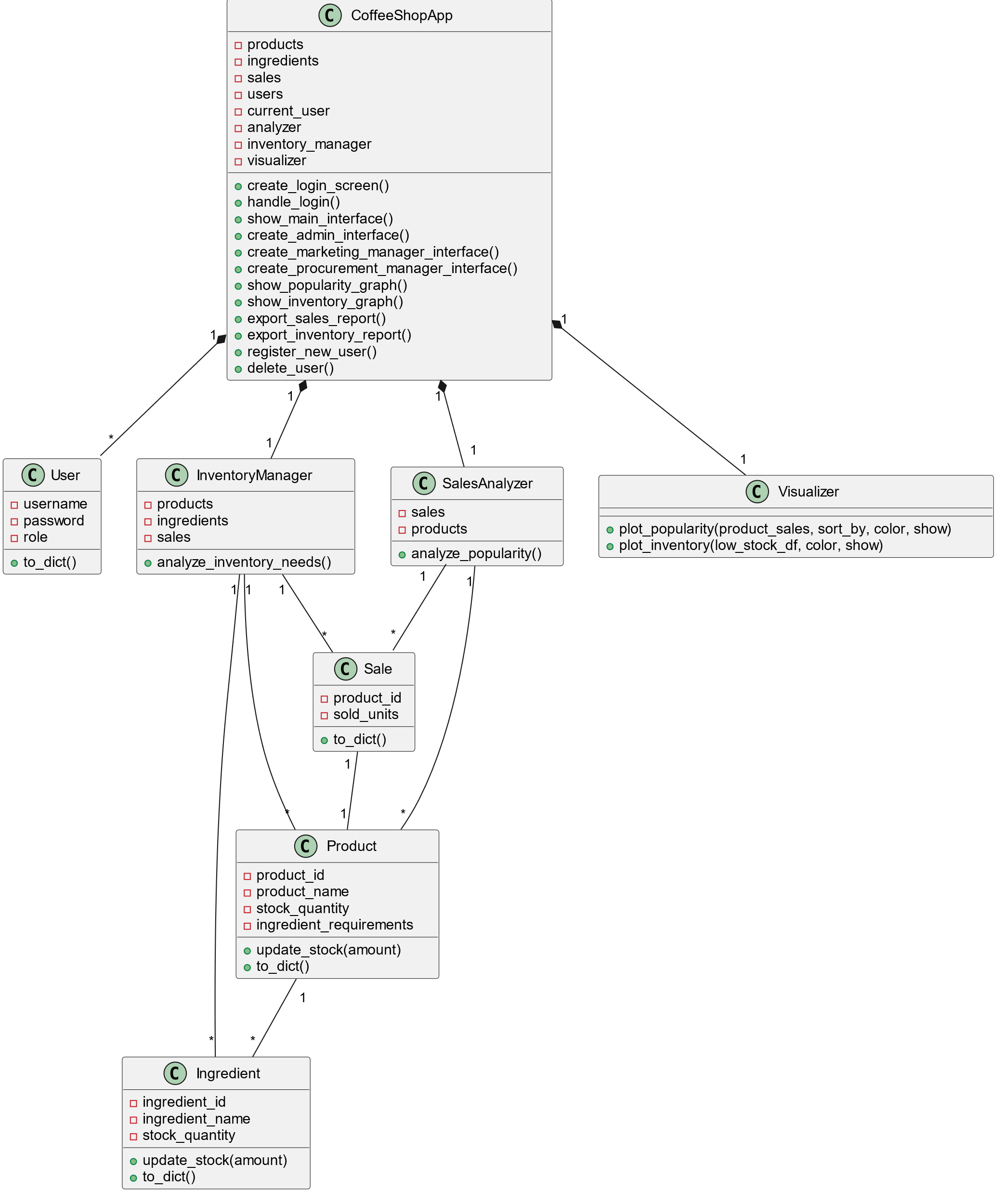


Рисунок 2 – Диаграмма классов

# **ПРИЛОЖЕНИЕ 2**

Тест-кейсы

Таблица 1 – тест-кейс №1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование функциональности | Наименование поля | Тестовый набор | Ожидаемый результат |
| Т-01 | Авторизация администратора с корректными данными | Логин | admin | Успешная авторизация в системе |
| Пароль | admin |
| Фактический результат. | | | | |

Таблица 2 – тест-кейс №2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование функциональности | Наименование поля | Тестовый набор | Ожидаемый результат |
| Т-02 | Регистрация нового пользователя с некорректными данными | Логин | manager2 | Успешное добавление нового пользователя |
| Пароль | 5\_password\_5 |
| Роль | manager |
| Фактический результат  . | | | | |

Таблица 3 – тест-кейс №3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование функциональности | Наименование поля | Тестовый набор | Ожидаемый результат |
| Т-03 | Удаление пользователя с некорректными данными | Логин пользователя | admin | Ошибка. Невозможно удалить аккаунт администратора |
| Фактический результат. | | | | |

# **ПРИЛОЖЕНИЕ 3**

Листинг 1 Модуль main.py

import tkinter as tk  
from tkinter import messagebox  
from tkinter import ttk  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from visualizer import Visualizer  
from utils import load\_json\_data, save\_json\_data  
from sales\_analyzer import SalesAnalyzer  
from inventory\_manager import InventoryManager  
from product import Product  
from ingredient import Ingredient  
from sale import Sale  
from user import User  
  
  
class CoffeeShopApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Система склада кофейни")  
  
 # Загрузка данных  
 self.products = [Product(\*\*p) for p in load\_json\_data('products.json')]  
 self.ingredients = [Ingredient(\*\*i) for i in load\_json\_data('ingredients.json')]  
 self.sales = [Sale(\*\*s) for s in load\_json\_data('sales\_data.json')]  
 self.users = load\_json\_data('users.json')  
  
 # Инициализация атрибутов  
 self.current\_user = None  
 self.analyzer = SalesAnalyzer(self.sales, self.products)  
 self.inventory\_manager = InventoryManager(self.products, self.ingredients, self.sales)  
 self.visualizer = Visualizer()  
  
 self.create\_login\_screen()  
  
 def create\_login\_screen(self):  
 for widget in self.root.winfo\_children():  
 widget.destroy()  
  
 tk.Label(self.root, text="Система склада кофейни", font=("Verdana", 18)).pack(pady=20)  
  
 tk.Label(self.root, text="Имя пользователя:").pack()  
 self.username\_entry = tk.Entry(self.root) # Сохраняем ссылку на новый объект Entry  
 self.username\_entry.pack()  
  
 tk.Label(self.root, text="Пароль:").pack()  
 self.password\_entry = tk.Entry(self.root, show="\*") # Сохраняем ссылку на новый объект Entry  
 self.password\_entry.pack()  
  
 tk.Button(self.root, text="Войти", command=self.handle\_login).pack(pady=10)  
 tk.Button(self.root, text="Выход", command=self.root.quit).pack(pady=10)  
  
 def handle\_login(self):  
 username = self.username\_entry.get().strip() # Убираем лишние пробелы  
 password = self.password\_entry.get().strip()  
  
 for user in self.users:  
 if user['username'] == username and user['password'] == password:  
 print(f"Найден пользователь: {user}") # Отладка  
 self.current\_user = User(\*\*user)  
 self.show\_main\_interface()  
 return  
  
 messagebox.showerror("Ошибка", "Неправильный логин или пароль!")  
  
 def show\_main\_interface(self):  
 for widget in self.root.winfo\_children():  
 widget.destroy()  
  
 tk.Label(self.root, text=f"Добро пожаловать, {self.current\_user.username}", font=("Arial", 16)).pack(  
 pady=10)  
  
 if self.current\_user.role == "admin":  
 self.create\_admin\_interface()  
 elif self.current\_user.role == "manager":  
 self.create\_marketing\_manager\_interface()  
 elif self.current\_user.role == "cashier":  
 self.create\_procurement\_manager\_interface()  
 else:  
 messagebox.showerror("Ошибка", "Неизвестная роль!")  
 self.create\_login\_screen()  
  
 def create\_admin\_interface(self):  
 tk.Button(self.root, text="График популярности", command=self.show\_popularity\_graph).pack(  
 pady=5)  
 tk.Button(self.root, text="График потребностей в инвентаре",  
 command=self.show\_inventory\_graph).pack(pady=5)  
 tk.Button(self.root, text="Добавить нового пользователя", command=self.register\_new\_user).pack(  
 pady=5)  
 tk.Button(self.root, text="Удалить пользователя", command=self.delete\_user).pack(pady=5)  
 tk.Button(self.root, text="Выход", command=self.create\_login\_screen).pack(pady=10)  
  
 def create\_marketing\_manager\_interface(self):  
 tk.Button(self.root, text="График популярности", command=self.show\_popularity\_graph).pack(  
 pady=5)  
 tk.Button(self.root, text="Экспорт отчета о продажах", command=self.export\_sales\_report).pack(  
 pady=5)  
 tk.Button(self.root, text="Выход", command=self.create\_login\_screen).pack(pady=10)  
  
 def create\_procurement\_manager\_interface(self):  
 tk.Button(self.root, text="График потребностей в инвентаре",  
 command=self.show\_inventory\_graph).pack(pady=5)  
 tk.Button(self.root, text="Экспорт отчета запасов", command=self.export\_inventory\_report).pack(  
 pady=5)  
 tk.Button(self.root, text="Выход", command=self.create\_login\_screen).pack(pady=10)  
  
 def show\_popularity\_graph(self):  
 if self.current\_user.role not in ["admin", "manager"]:  
 messagebox.showerror("Доступ запрещен", "У вас нет разрешения на просмотр этого графика.")  
 return  
  
 product\_sales = self.analyzer.analyze\_popularity()  
 print(product\_sales) # Выводим DataFrame в консоль  
  
 # Убедитесь, что здесь вы передаете правильные данные для визуализации  
 fig = self.visualizer.plot\_popularity(product\_sales, show=False)  
 self.display\_graph(fig)  
  
 def show\_inventory\_graph(self):  
 if self.current\_user.role not in ["admin", "cashier"]:  
 messagebox.showerror("Доступ запрещен", "У вас нет разрешения на просмотр этого графика.")  
 return  
  
 low\_stock\_df = self.inventory\_manager.analyze\_inventory\_needs()  
 print(low\_stock\_df) # Выводим DataFrame в консоль  
 if low\_stock\_df.empty:  
 messagebox.showinfo("Инфо", "Все ингредиенты в достаточном количестве.")  
 return  
  
 fig = self.visualizer.plot\_inventory(low\_stock\_df, show=False)  
 if fig is not None:  
 self.display\_graph(fig)  
  
 def display\_graph(self, fig):  
 graph\_window = tk.Toplevel(self.root)  
 graph\_window.title("Граф")  
  
 canvas = FigureCanvasTkAgg(fig, master=graph\_window)  
 canvas.draw()  
 canvas.get\_tk\_widget().pack()  
  
 def export\_sales\_report(self):  
 product\_sales = self.analyzer.analyze\_popularity()  
 print(product\_sales) # Выводим DataFrame в консоль  
 if not product\_sales.empty: # Проверяем, есть ли данные для экспорта  
 self.export\_to\_excel(product\_sales, "sales\_report.xlsx")  
 else:  
 messagebox.showwarning("Предупреждение", "Нет данных для экспорта отчета о продажах.")  
  
 def export\_inventory\_report(self):  
 low\_stock\_df = self.inventory\_manager.analyze\_inventory\_needs()  
 print(low\_stock\_df) # Выводим DataFrame в консоль  
 if not low\_stock\_df.empty: # Проверяем, есть ли данные для экспорта  
 self.export\_to\_excel(low\_stock\_df, "inventory\_report.xlsx")  
 else:  
 messagebox.showwarning("Предупреждение", "Нет данных для экспорта отчета о пополнении запасов.")  
  
 @staticmethod  
 def export\_to\_excel(df, filename):  
 try:  
 df.to\_excel(filename, index=False)  
 messagebox.showinfo("Успех", f"Отчет экспортирован в {filename}")  
 except Exception as e:  
 messagebox.showerror("Ошибка", f"Не удалось экспортировать отчет: {e}")  
  
 def register\_new\_user(self):  
 def save\_new\_user():  
 username = username\_entry.get()  
 password = password\_entry.get()  
 role = role\_combobox.get()  
  
 if not username or not password or not role:  
 messagebox.showerror("Ошибка", "Все поля должны быть заполнены!")  
 return  
  
 self.users.append({"username": username, "password": password, "role": role})  
 save\_json\_data('users.json', self.users)  
 messagebox.showinfo("Успешно", "Новый пользователь добавлен!")  
 register\_window.destroy()  
  
 register\_window = tk.Toplevel(self.root)  
 register\_window.title("Добавить нового пользователя")  
  
 tk.Label(register\_window, text="Логин:").grid(row=0, column=0, padx=5, pady=5)  
 username\_entry = tk.Entry(register\_window)  
 username\_entry.grid(row=0, column=1, padx=5, pady=5)  
  
 tk.Label(register\_window, text="Пароль:").grid(row=1, column=0, padx=5, pady=5)  
 password\_entry = tk.Entry(register\_window, show="\*")  
 password\_entry.grid(row=1, column=1, padx=5, pady=5)  
  
 tk.Label(register\_window, text="Роль:").grid(row=2, column=0, padx=5, pady=5)  
 role\_combobox = ttk.Combobox(register\_window, values=["admin", "manager", "cashier"])  
 role\_combobox.grid(row=2, column=1, padx=5, pady=5)  
  
 tk.Button(register\_window, text="Добавить", command=save\_new\_user).grid(row=3, column=0, columnspan=2, pady=10)  
  
 def delete\_user(self):  
 if self.current\_user.role != "admin":  
 messagebox.showerror("Доступ запрещён", "У вас недостаточно прав, чтобы удалить пользователя из системы.")  
 return  
  
 delete\_window = tk.Toplevel(self.root)  
 delete\_window.title("Удаление пользователя")  
  
 tk.Label(delete\_window, text="Логин пользователя:").grid(row=0, column=0, padx=5, pady=5)  
 username\_entry = tk.Entry(delete\_window)  
 username\_entry.grid(row=0, column=1, padx=5, pady=5)  
  
 def confirm\_delete():  
 username = username\_entry.get()  
 for user in self.users:  
 if user['username'] == username:  
 if user['role'] == "admin":  
 messagebox.showerror("Ошибка", "Невозможно удалить аккаунт администратора.")  
 return  
 self.users.remove(user)  
 save\_json\_data('users.json', self.users)  
 messagebox.showinfo("Успешно", f"Пользователь '{username}' успешно удален.")  
 delete\_window.destroy()  
 return  
 messagebox.showerror("Ошибка", f"Пользователь '{username}' не найден.")  
  
 def cancel\_action():  
 delete\_window.destroy() # Закрываем окно удаления пользователя  
  
 tk.Button(delete\_window, text="Удалить", command=confirm\_delete).grid(row=1, column=0, pady=10, padx=5)  
 tk.Button(delete\_window, text="Отмена", command=cancel\_action).grid(row=1, column=1, pady=10, padx=5)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = CoffeeShopApp(root)  
 root.mainloop()

Листинг 2 Модуль ingredient.py

class Ingredient:  
 def \_\_init\_\_(self, ingredient\_id, ingredient\_name, stock\_quantity):  
 self.ingredient\_id = ingredient\_id  
 self.ingredient\_name = ingredient\_name  
 self.stock\_quantity = stock\_quantity  
  
 def update\_stock(self, amount):  
 self.stock\_quantity += amount  
  
 def to\_dict(self):  
 return {  
 "ingredient\_id": self.ingredient\_id,  
 "ingredient\_name": self.ingredient\_name,  
 "stock\_quantity": self.stock\_quantity  
 }

# **ПРИЛОЖЕНИЕ 4**

Листинг 3 Модуль product.py

class Product:  
 def \_\_init\_\_(self, product\_id, product\_name, stock\_quantity, ingredient\_requirements):  
 self.product\_id = product\_id  
 self.product\_name = product\_name  
 self.stock\_quantity = stock\_quantity  
 self.ingredient\_requirements = ingredient\_requirements  
  
 def update\_stock(self, amount):  
 self.stock\_quantity += amount  
  
 def to\_dict(self):  
 return {  
 "product\_id": self.product\_id,  
 "product\_name": self.product\_name,  
 "stock\_quantity": self.stock\_quantity,  
 "ingredient\_requirements": self.ingredient\_requirements  
 }

Листинг 4 Модуль sale.py

class Sale:  
 def \_\_init\_\_(self, product\_id, sold\_units):  
 self.product\_id = product\_id  
 self.sold\_units = sold\_units  
  
 def to\_dict(self):  
 return {  
 "product\_id": self.product\_id,  
 "sold\_units": self.sold\_units,  
 }

# **ПРИЛОЖЕНИЕ 5**

Листинг 5 Модуль inventory\_manager.py

import pandas as pd  
class InventoryManager:  
 def \_\_init\_\_(self, products, ingredients, sales):  
 self.products = products  
 self.ingredients = ingredients  
 self.sales = sales  
  
 def analyze\_inventory\_needs(self):  
 sales\_by\_product = {sale.product\_id: sale.sold\_units for sale in self.sales}  
 ingredient\_needs = {ingredient.ingredient\_id: 0 for ingredient in self.ingredients}  
 for product in self.products:  
 product\_sales = sales\_by\_product.get(product.product\_id, 0)  
 for ingredient\_id, amount\_needed in product.ingredient\_requirements.items():  
 ingredient\_needs[ingredient\_id] += amount\_needed \* product\_sales  
 low\_stock\_ingredients = []  
 for ingredient in self.ingredients:  
 required\_stock = ingredient\_needs[ingredient.ingredient\_id] - ingredient.stock\_quantity  
 if required\_stock > 0: low\_stock\_ingredients.append((ingredient.ingredient\_name, required\_stock))  
 df = pd.DataFrame(low\_stock\_ingredients, columns=['Ингредиент', 'Необходимое количество'])  
 return df

# **ПРИЛОЖЕНИЕ 6**

Листинг 6 Модуль sales\_analyzer.py

import pandas as pd  
  
class SalesAnalyzer:  
 def \_\_init\_\_(self, sales, products):  
 self.sales = sales  
 self.products = products  
  
 def analyze\_popularity(self):  
 # Преобразуем данные о продажах в DataFrame  
 if not self.sales:  
 raise ValueError("Данные о продажах некорректны или отсутствуют.")  
  
 try:  
 sales\_df = pd.DataFrame([vars(sale) for sale in self.sales])  
 except Exception as e:  
 raise ValueError(f"Не удалось преобразовать данные о продажах в DataFrame: {e}")  
  
 # Проверяем наличие необходимых колонок  
 if 'product\_id' not in sales\_df.columns or 'sold\_units' not in sales\_df.columns:  
 raise KeyError("DataFrame не нашел ключевые столбцы: 'product\_id' или 'sold\_units'")  
  
 # Приведение типов  
 sales\_df['product\_id'] = sales\_df['product\_id'].astype(str)  
 product\_df = pd.DataFrame([vars(product) for product in self.products])  
 product\_df['product\_id'] = product\_df['product\_id'].astype(str)  
  
 # Группируем продажи по product\_id  
 product\_sales = sales\_df.groupby('product\_id')['sold\_units'].sum().reset\_index()  
  
 # Объединяем с DataFrame продуктов  
 product\_sales = product\_sales.merge(product\_df[['product\_id', 'product\_name']], on='product\_id')  
  
 # Выбираем только нужные колонки  
 product\_sales = product\_sales[['product\_name', 'sold\_units']]  
 product\_sales.columns = ['Товар', 'Проданных единиц']  
  
 # Возвращаем данные, отсортированные по количеству продаж  
 return product\_sales.sort\_values(by='Проданных единиц', ascending=False)

# **ПРИЛОЖЕНИЕ 7**

Листинг 7 Модуль user.py

class User:  
 def \_\_init\_\_(self, username, password, role):  
 self.username = username  
 self.password = password  
 self.role = role  
  
 def to\_dict(self):  
 return {  
 "username": self.username,  
 "password": self.password,  
 "role": self.role  
 }

Листинг 8 Модуль utils.py

import json  
  
def load\_json\_data(filename):  
 try:  
 with open(filename, 'r', encoding='utf-8') as file:  
 return json.load(file)  
 except (FileNotFoundError, json.JSONDecodeError):  
 print(f"Ошибка загрузки данных из {filename}")  
 return []  
  
  
def save\_json\_data(filename, data):  
 try:  
 with open(filename, 'w', encoding='utf-8') as file:  
 json.dump(data, file, indent=4, ensure\_ascii=False)  
 except Exception as e:  
 print(f"Ошибка сохранения данных в {filename}: {e}")

# **ПРИЛОЖЕНИЕ 8**

Листинг 9 Модуль visualizer.py

import matplotlib.pyplot as plt  
  
  
class Visualizer:  
 @staticmethod  
 def plot\_popularity(product\_sales, sort\_by='Проданных единиц', color='plum', show=True):  
 # Проверяем, что DataFrame не пустой  
 if product\_sales.empty:  
 print("Нет данных для отображения популярности продуктов.")  
 return None  
  
 # Сортируем данные для визуализации  
 product\_sales = product\_sales.sort\_values(by=sort\_by, ascending=False)  
  
 # Построение графика  
 fig, ax = plt.subplots(figsize=(10, 6))  
 bars = ax.barh(product\_sales['Товар'], product\_sales['Проданных единиц'], color=color)  
 ax.set\_title('Популярность продуктов')  
 ax.set\_xlabel('Количество проданных единиц')  
 ax.set\_ylabel('Продукты')  
  
 # Добавляем подписи на график  
 for bar in bars:  
 ax.text(bar.get\_width(), bar.get\_y() + bar.get\_height() / 2,  
 f'{bar.get\_width()}', va='center', ha='left')  
  
 plt.tight\_layout()  
 if show:  
 plt.show()  
 return fig # Возвращаем объект Figure для отображения в Tkinter  
  
 @staticmethod  
 def plot\_inventory(low\_stock\_df, color='powderblue', show=True):  
 if low\_stock\_df.empty or (low\_stock\_df['Необходимое количество'] <= 0).all():  
 print("Все ингредиенты в достаточном количестве.")  
 return None # Возвращаем None, если график не требуется.  
  
 fig, ax = plt.subplots(figsize=(10, 6))  
 bars = ax.barh(low\_stock\_df['Ингредиент'], low\_stock\_df['Необходимое количество'], color=color)  
 ax.set\_title('Необходимость пополнения ингредиентов')  
 ax.set\_xlabel('Требуемое количество')  
 ax.set\_ylabel('Ингредиенты')  
  
 for bar in bars:  
 ax.text(bar.get\_width(), bar.get\_y() + bar.get\_height() / 2,  
 f'{bar.get\_width()}', va='center', ha='left')  
  
 plt.tight\_layout()  
 if show:  
 plt.show()  
 return fig # Возвращаем объект Figure.

# **ПРИЛОЖЕНИЕ 9**

Листинг 10 Входные данные ingredients.json

[  
 {  
 "ingredient\_id": "1",  
 "ingredient\_name": "Кофе",  
 "stock\_quantity": 50  
 },  
 {  
 "ingredient\_id": "2",  
 "ingredient\_name": "Молоко",  
 "stock\_quantity": 20  
 },  
 {  
 "ingredient\_id": "3",  
 "ingredient\_name": "Сахар",  
 "stock\_quantity": 10  
 },  
 {  
 "ingredient\_id": "4",  
 "ingredient\_name": "Сливки",  
 "stock\_quantity": 12  
 },  
 {  
 "ingredient\_id": "5",  
 "ingredient\_name": "Вода",  
 "stock\_quantity": 8  
 },  
 {  
 "ingredient\_id": "6",  
 "ingredient\_name": "Сироп",  
 "stock\_quantity": 15  
 }  
]

Листинг 11 Входные данные products.json

[  
 {  
 "product\_id": "1",  
 "product\_name": "Капучино",  
 "stock\_quantity": 16,  
 "ingredient\_requirements": {  
 "1": 1,  
 "2": 2  
 }  
 },  
 {  
 "product\_id": "2",  
 "product\_name": "Латте",  
 "stock\_quantity": 10,  
 "ingredient\_requirements": {  
 "1": 1,  
 "2": 2,  
 "3": 2  
 }  
 },  
 {  
 "product\_id": "3",  
 "product\_name": "Раф",  
 "stock\_quantity": 13,  
 "ingredient\_requirements": {  
 "1": 1,  
 "3": 2,  
 "6": 1  
 }  
 },  
 {  
 "product\_id": "4",  
 "product\_name": "Двойной эспрессо",  
 "stock\_quantity": 30,  
 "ingredient\_requirements": {  
 "1": 2  
 }  
 },  
 {  
 "product\_id": "5",  
 "product\_name": "Американо",  
 "stock\_quantity": 20,  
 "ingredient\_requirements": {  
 "1": 1,  
 "5": 3  
 }  
 }  
]

Листинг 12 Входные данные sales\_data.json

[  
 {  
 "product\_id": "1",  
 "sold\_units": 15  
 },  
 {  
 "product\_id": "2",  
 "sold\_units": 7  
 },  
 {  
 "product\_id": "3",  
 "sold\_units": 3  
 },  
 {  
 "product\_id": "4",  
 "sold\_units": 2  
 },  
 {  
 "product\_id": "5",  
 "sold\_units": 10  
 }  
]

Листинг 13 Входные данные sales\_data.json

[  
 {  
 "username": "admin",  
 "password": "admin",  
 "role": "admin"  
 },  
 {  
 "username": "manager1",  
 "password": "password123",  
 "role": "manager"  
 },  
 {  
 "username": "cashier1",  
 "password": "password321",  
 "role": "cashier"  
 }  
]

# **ПРИЛОЖЕНИЕ 10**

Листинг 14 Модуля test\_coffeeshop.py:

import pytest  
from tkinter import Tk  
from main import CoffeeShopApp  
  
  
@pytest.fixture  
def app():  
 root = Tk()  
 app = CoffeeShopApp(root)  
 return app  
  
  
def test\_login\_success(app):  
 app.username\_entry.insert(0, "admin")  
 app.password\_entry.insert(0, "admin")  
 app.handle\_login()  
 assert app.current\_user is not None # Проверка успешного входа  
  
  
def test\_login\_failure(app):  
 app.username\_entry.insert(0, "wrong\_user")  
 app.password\_entry.insert(0, "wrong\_password")  
 app.handle\_login()  
 assert app.current\_user is None # Проверка на отсутствие пользователя

Листинг 15 Модуля test\_product.py

import pytest  
from product import Product  
  
  
@pytest.fixture  
def product():  
 return Product(product\_id="1", product\_name="Капучино", stock\_quantity=10, ingredient\_requirements={"1": 1, "2": 2})  
  
  
def test\_initialization(product):  
 assert product.product\_id == "1"  
 assert product.product\_name == "Капучино"  
 assert product.stock\_quantity == 10  
  
  
def test\_update\_stock(product):  
 product.update\_stock(5)  
 assert product.stock\_quantity == 15  
  
  
def test\_to\_dict(product):  
 expected\_dict = {  
 "product\_id": "1",  
 "product\_name": "Капучино",  
 "stock\_quantity": 10,  
 "ingredient\_requirements": {"1": 1, "2": 2}  
 }  
 assert product.to\_dict() == expected\_dict

Листинг 16 Модуля test\_system.py

import pytest  
from inventory\_manager import InventoryManager  
from sales\_analyzer import SalesAnalyzer  
from product import Product  
from sale import Sale  
from ingredient import Ingredient  
  
  
@pytest.fixture  
def setup\_inventory\_system():  
 products = [Product("1", "Капучино", 10, {"1": 1, "2": 2})]  
 ingredients = [Ingredient("1", "Кофе", 50), Ingredient("2", "Молоко", 20)] # Добавляем ингредиенты  
 sales = [Sale("1", 5)]  
 inventory\_manager = InventoryManager(products, ingredients, sales) # Передаем ингредиенты  
 sales\_analyzer = SalesAnalyzer(sales, products)  
 return inventory\_manager, sales\_analyzer  
  
  
def test\_inventory\_needs\_analysis(setup\_inventory\_system):  
 inventory\_manager, \_ = setup\_inventory\_system  
 low\_stock\_df = inventory\_manager.analyze\_inventory\_needs()  
 assert len(low\_stock\_df) == 0  
  
  
def test\_sales\_analysis(setup\_inventory\_system):  
 \_, sales\_analyzer = setup\_inventory\_system  
 product\_sales = sales\_analyzer.analyze\_popularity()  
 assert product\_sales['Товар'][0] == "Капучино" # Проверка на популярность товара