-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 базы данных «Склад»**

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

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

**Москва 2024**

**СОДЕРЖАНИЕ**

**Стр.**

**ВВЕДЕНИЕ** 3

**1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 4

1.1 Описание предметной области и процессы склада кофейни 4

1.2 Важность учета и анализа данных для кофейни 5

1.3 Задачи анализа данных и управления запасами 5

**2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ** 7

2.1 Построение диаграммы прецедентов и классов 7

2.2 Выбор инструментов 9

2.3 Описание модулей 11

**3. ТЕСТИРОВАНИЕ И ОТЛАДКА СИСТЕМЫ** 16

3.1 Подходы к тестированию системы 16

3.2 Процесс отладки и оптимизации кода 16

3.3 Анализ результатов тестирования и исправление ошибок 17

**4. ПЛАНЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА** 18

4.1 Дополнительные функции для улучшения функционал 18

4.2 Развитие пользовательского интерфейса и добавление новых графиков 22

4.3 Реализация обновления данных 22

**ЗАКЛЮЧЕНИЕ** 24

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 25

**ПРИЛОЖЕНИЕ 1**  27

**ПРИЛОЖЕНИЕ 2**  29

**ПРИЛОЖЕНИЕ 3**  30

**ПРИЛОЖЕНИЕ 4**  31

**ПРИЛОЖЕНИЕ 5**  32

**ПРИЛОЖЕНИЕ 6**  33

**ПРИЛОЖЕНИЕ 7**  36

**ПРИЛОЖЕНИЕ 8**  38

**ВВЕДЕНИЕ**

Управление складом — ключевая часть работы кофейни, так как правильный учет запасов напрямую влияет на эффективность заведения. Важно не только отслеживать количество напитков, но и прогнозировать потребности в пополнении склада. Анализ данных о продажах помогает избежать излишков и дефицита товаров, что способствует бесперебойной работе кофейни и снижению затрат.

Актуальность темы заключается в том, что многие кофейни сталкиваются с проблемами неэффективного управления запасами, что может привести к дефициту популярных напитков или излишку менее востребованных. Разработка системы для учета и прогнозирования потребности в пополнении запасов повысит эффективность работы кофейни.

Цель проекта — разработать систему классов для базы данных, автоматизирующую учет продаж напитков и анализ потребности в пополнении запасов. Задачи включают анализ методов учета и анализа данных, создание структуры базы данных, реализацию прогнозирования потребности в пополнении запасов, разработку интерфейса для отслеживания остатков и тестирование системы.

Объектом исследования является процесс учета и управления запасами кофейни, предметом — проектирование системы классов для автоматизации этих процессов. В курсовой работе будут рассмотрены предметная область, подходы к проектированию системы и тестирование ее эффективности. В заключении предложены рекомендации по развитию системы.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

В данном курсовом проекте предметной областью для решения поставленных задач является склад кофейни. Это предприятие общественного питания, занимающееся продажей кофе, напитков и сопутствующих товаров. Ассортимент продукции кофейни включает различные виды кофе, чай, десерты и сладости, а также товары для приготовления напитков. Основные посетители кофейни – это любители кофе, жители и работники близлежащих районов, а также гости, которые ценят высокое качество напитков и уютную атмосферу.

Деятельность кофейни включает в себя не только приготовление напитков и обслуживание клиентов, но и организацию эффективного учета запасов на складе. Это включает в себя контроль за количеством напитков, расходных материалов, а также прогнозирование потребности в пополнении запасов на основе анализа продаж. Возникает необходимость в компьютерной обработке данных о запасах, продажах и пополнении товаров, с целью автоматизации учета и анализа информации для улучшения управления складом и повышения общей эффективности работы кофейни.

* 1. **Описание предметной области и процессы склада кофейни**

Предметной областью в рамках данного курсового проекта является управление запасами и анализ данных о продажах на складе кофейни. В кафе, как и в других общественных заведениях, существует необходимость постоянного учета товаров, включая ингредиенты для напитков, упаковку, расходные материалы и готовую продукцию. Все эти товары необходимо отслеживать для того, чтобы поддерживать оптимальный уровень запасов, минимизируя как излишки, так и дефицит.

Процесс склада кофейни можно разделить на несколько ключевых этапов:

1. **Приемка товара** – в этот момент происходит занесение поступивших товаров в систему и учет их количества.
2. **Хранение** – товары размещаются на складе, и их наличие регулярно проверяется.
3. **Продажа** – данные о продажах напитков автоматически обновляют информацию о расходах со склада, уменьшая остатки товаров.
4. **Прогнозирование потребностей** – на основе анализа данных о продажах система помогает прогнозировать потребности в пополнении запасов для удовлетворения спроса.
   1. **Важность учета и анализа данных для кофейни**

Учет запасов и анализ данных о продажах являются важнейшими аспектами эффективного функционирования кофейни. Система учета запасов помогает избежать ситуаций, когда на складе не хватает популярных товаров или, наоборот, накоплены излишки, которые не используются в нужный момент. Анализ данных о продажах, в свою очередь, позволяет выявить тренды в потребительских предпочтениях, что помогает не только оптимизировать запасы, но и сделать более точные прогнозы для закупок.

Для малых предприятий, таких как кафе или кофейни, важно иметь инструмент, который позволит:

* **Минимизировать расходы** за счет точного учета и прогноза потребностей.
* **Упростить процессы учета** с минимальными усилиями со стороны персонала.
* **Повысить уровень обслуживания** клиентов, обеспечив наличие востребованных напитков в любое время.

Недостаток системы учета и анализа может привести к неэффективному использованию ресурсов, потерям и даже снижению уровня обслуживания клиентов.

* 1. **Задачи анализа данных и управления запасами**

Основной задачей анализа данных на складе кофейни является прогнозирование потребности в пополнении запасов и обеспечение бесперебойной работы кофейни. Для этого необходимо решить несколько ключевых задач:

1. **Анализ данных о продажах**: выявление популярных напитков, сезонных колебаний спроса, а также выявление менее востребованных товаров.
2. **Прогнозирование потребностей в пополнении запасов**: на основе данных о прошлых продажах система должна рассчитывать, какое количество товаров необходимо закупить, чтобы избежать дефицита или излишков.
3. **Учет товарных запасов**: постоянный мониторинг остатков на складе, с возможностью автоматического уведомления персонала о необходимости пополнения.
4. **Анализ эффективности работы склада**: оценка точности прогнозов и уровня обслуживания для оптимизации работы кофейни в будущем.

Решение этих задач с помощью программного обеспечения позволит кофейне оптимизировать работу склада, повысить точность прогнозов и минимизировать ошибки, которые могут быть связаны с ручным учетом.

1. **ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ**

Вторая глава посвящена процессу проектирования и разработки системы, которая предназначена для анализа данных о продажах и управления запасами на складе кофейни. В этом разделе будут рассмотрены ключевые аспекты, такие как построение диаграмм прецедентов и классов, выбор инструментов для реализации системы, а также описание модулей, из которых состоит проект.

Процесс проектирования будет включать описание использования диаграмм прецедентов для определения взаимодействия пользователей с системой, диаграмм классов для моделирования структуры данных и выбора технологий, необходимых для реализации всех требований проекта.

* 1. **Построение диаграммы прецедентов и классов**

Диаграмма прецедентов описывает взаимодействие пользователя (менеджера склада) с системой. В этой системе менеджер выполняет несколько ключевых операций:

* **Загрузка данных**: Менеджер загружает информацию о продуктах и продажах.
* **Анализ популярности продуктов**: Менеджер инициирует анализ для определения наиболее популярных товаров.
* **Анализ потребности в пополнении запасов**: Система предоставляет рекомендации по пополнению запасов.
* **Визуализация данных**: Менеджер получает графическое представление результатов анализа.

Эти процессы отображены на рисунке 1.



Рисунок 1 – Диаграмма прецедентов

Диаграмма классов отражает структуру классов в системе и их отношения. Классы имеют связи через использование объектов других классов.

Диаграмма классов приведена в рисунке 2.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 2 – Диаграмма классов

* Классы **Product** и **Sale** связаны между собой через product\_id, что позволяет сопоставить продажи с конкретными продуктами.
* Класс **SalesAnalyzer** использует данные классов Product и Sale для анализа популярности.
* Класс **InventoryManager** использует данные о продаже и остатках на складе для анализа потребности в пополнении.
* Класс **Visualizer** предоставляет визуализацию данных для анализа.
  1. **Выбор инструментов**

При выборе языка программирования для реализации данного проекта был выбран Python. Этот выбор обоснован его удобством, высокой продуктивностью разработки и наличием мощных библиотек для работы с данными. В этом разделе проведено сравнение Python и C по ключевым критериям, важным для данного проекта.

Для разработки использовалась интегрированная среда разработки PyCharm от компании JetBrains, которая предоставила удобные инструменты для написания, отладки и тестирования кода.

Среда разработки PyCharm является одной из лучших сред для разработки на Python, предлагая такие функции как автодополнение кода, отладчик, управление виртуальными окружениями. Эти возможности стали важным преимуществом в работе над проектом. В отличие от C, где настройка среды разработки и отладка могут занять больше времени, Python в PyCharm обеспечил удобство и быстроту разработки, что сделало его идеальным выбором для данного проекта. В таблице 1 произведено сравнение Python и C по ключевым критериям, что позволило обосновать выбор языка программирования для данного проекта.

|  |  |  |
| --- | --- | --- |
| 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, дебаггер) | Требует настройки окружения, поддержка ограничена |

Таблица 1 – Сравнение языков программирования Python и C

Для анализа данных и визуализации в проекте были выбраны библиотеки Pandas и Matplotlib, а для тестирования использовался Pytest.

Pandas был выбран за удобные структуры данных (DataFrame), которые позволяют эффективно обрабатывать и анализировать большие объемы данных, такие как информация о продажах и запасах. Это облегчает агрегацию, фильтрацию и манипуляцию данными.

Matplotlib используется для создания графиков, таких как гистограммы и столбчатые диаграммы. Она предоставляет широкие возможности для настройки графиков, что позволяет наглядно визуализировать результаты анализа.

Для автоматизированного тестирования был использован Pytest, так как он поддерживает простоту написания и выполнения тестов, позволяя быстро проверять корректность работы отдельных компонентов системы и находить ошибки.

* 1. **Описание модулей**

Проект состоит из нескольких модулей, каждый из которых выполняет свою функциональную задачу. Описание каждого модуля представлено ниже:

1. **main.py**

Главный модуль проекта, который выполняет последовательность операций от загрузки данных до их анализа и визуализации. Он загружает данные из файлов JSON, создает объекты для продуктов и продаж, а также выполняет анализ популярности продуктов и потребности в пополнении запасов. В конце генерируется график популярности продуктов и график пополнения запасов, которые приведены в рисунках 4 и 5 – соответственно. Данный модуль состоит из кода, который приведен в приложении 1. Блок-схема главного модуля представлена на рисунке 3.

Задачи:

* Загрузка данных о продуктах и продажах из JSON-файлов с использованием функции *load\_json\_data*.
* Создание экземпляров объектов классов Product и Sale на основе загруженных данных.
* Проведение анализа популярности продуктов с использованием модуля SalesAnalyzer.
* Анализ потребности в пополнении запасов товаров с применением модуля InventoryManager.
* Вывод результатов анализа и построение графиков с помощью модуля Visualizer.

Блок-схема главного модуля:

**Изображение выглядит как текст, диаграмма, зарисовка, рисунок

Автоматически созданное описание**

Рисунок 3 – Блок-схема работы главного модуля (main.py)

1. **product.py**

Модуль, содержащий класс Product, представляющий отдельный товар. Данный модуль состоит из кода, который приведен в приложении 2.

Задачи:

* Инициализация объекта продукта с атрибутами: product\_id, product\_name, stock\_quantity.
* Переопределение методов \_\_str\_\_ и \_\_repr\_\_, обеспечивающих вывод информации о продукте в читаемом виде.

1. **sale.py**

Модуль, реализующий класс Sale, представляющий отдельную продажу товара. Данный модуль состоит из кода, который приведен в приложении 3.

Задачи:

* Инициализация данных о продаже, таких как product\_id и sold\_units.
* Переопределение методов \_\_str\_\_ и \_\_repr\_\_, позволяющих удобно отображать информацию о продаже.

1. **inventory\_manager.py**

Модуль, осуществляющий управление запасами товаров. Данный модуль состоит из кода, который приведен в приложении 4.

Задачи:

* Анализ потребности в пополнении запасов товаров, на основе данных о текущих запасах и продажах.
* Вычисление необходимого количества для пополнения запасов каждого товара.
* Возвращение информации о товарах, требующих пополнения, в виде структуры данных DataFrame для дальнейшей визуализации.

1. **sales\_analyzer.py**

Модуль, предназначенный для анализа продаж. Данный модуль состоит из кода, который приведен в приложении 5.

Задачи:

* Анализ популярности продуктов, группировка данных по количеству проданных единиц.
* Сортировка продуктов по популярности (по количеству проданных единиц).
* Использование структуры данных DataFrame для представления и обработки данных о продажах.

1. **visualizer.py**

Модуль, предназначенный для визуализации данных. Данный модуль состоит из кода, который приведен в приложении 6.

Задачи:

* Построение графиков, отображающих популярность продуктов и потребность в пополнении запасов с использованием библиотеки matplotlib.
* Обеспечение настройки цветов и других параметров графиков для улучшения восприятия информации.

**Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание**

Рисунок 4 – График аналитики популярности продуктов

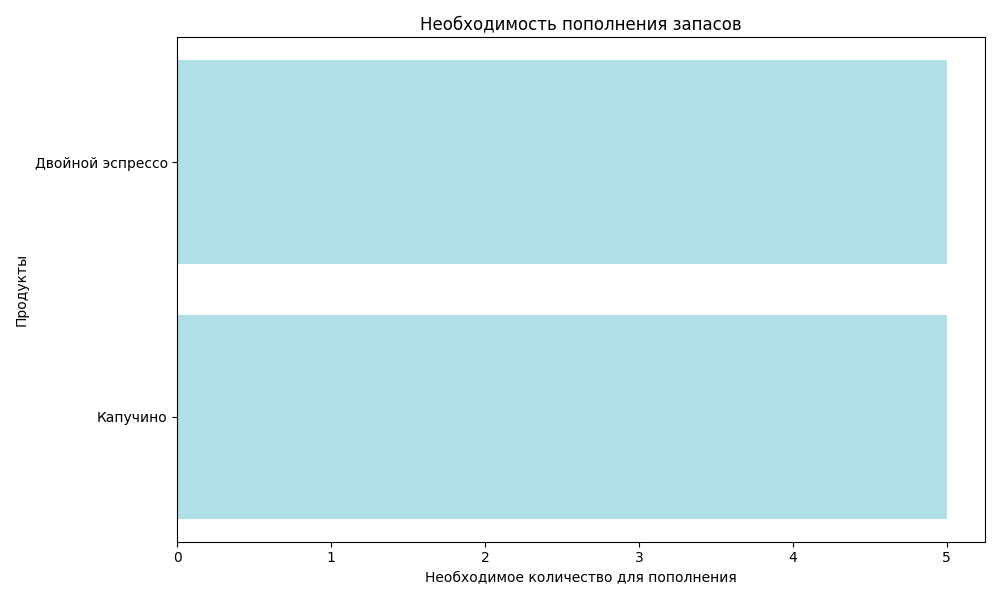
****

Рисунок 5 – График аналитики необходимости пополнения запасов

1. **ТЕСТИРОВАНИЕ И ОТЛАДКА СИСТЕМ**
   1. **Подходы к тестированию системы**

При создании проекта использовался ряд тестов для проверки правильности работы основных функциональных компонентов. Были проведены модульные тесты для классов Product, Sale, SalesAnalyzer и InventoryManager, а также проверка правильности работы методов обработки данных, анализа популярности продуктов и потребности в пополнении запасов.

Тем не менее, в ходе разработки могут возникать ошибки, которые влияют на корректность выполнения программы. Рассмотрим несколько примеров таких ошибок и способы их устранения.

* 1. **Процесс отладки и оптимизации кода**

1. Ошибка: Проблемы с загрузкой данных из JSON-файлов

На этапе тестирования возникла ошибка при попытке загрузить данные из файлов products.json и sales\_data.json. Программа не могла корректно обработать данные, если файл был пустым или содержал некорректные данные.

Способ устранения. Для предотвращения этой ошибки в функцию load\_json\_data была добавлена проверка на пустые файлы или некорректные данные:

def load\_json\_data(filename):

try:

with open(filename, 'r', encoding='utf-8') as file:

data = json.load(file)

if not data:

raise ValueError(f"Файл {filename} пуст.")

return data

except (json.JSONDecodeError, ValueError) as e:

print(f"Ошибка при загрузке данных: {e}")

return None

1. Ошибка: Ошибка визуализации данных

В процессе тестирования визуализации графиков возникала ошибка при отображении графиков с использованием matplotlib, если данные для отображения были пустыми или не соответствовали ожиданиям.

Способ устранения. Для решения проблемы в методах визуализации добавлена проверка на пустоту данных:

class Visualizer:

@staticmethod

def plot\_inventory(low\_stock\_df, color='powderblue'):

if low\_stock\_df.empty:

print("Все товары в достаточном количестве.")

return

plt.figure(figsize=(10, 6))

plt.barh(low\_stock\_df['Название'], low\_stock\_df['Необходимый запас'], color=color)

plt.title('Необходимость пополнения запасов')

plt.xlabel('Необходимое количество для пополнения')

plt.ylabel('Продукты')

plt.tight\_layout()

plt.show()

* 1. **Анализ результатов тестирования и исправление ошибок**

После проведения тестов было выявлено несколько критичных ошибок, которые касались загрузки данных, агрегации информации и визуализации. Все ошибки были устранены с помощью отладки и внесения изменений в код. Тесты показали, что система теперь корректно загружает данные, анализирует популярность продуктов и отображает графики с учетом всех проверок.

Кроме того, в процессе тестирования были проведены проверки на корректность выполнения операций с пустыми данными или некорректным форматом входных файлов, что также позволило повысить стабильность работы системы. Тесты приведены в приложениях 7 и 8.

1. **ПЛАНЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТ**

Проект продолжит развиваться, и в будущем будут добавлены новые функции, улучшения и возможности, направленные на оптимизацию работы с данными о продажах и запасах. Эти обновления охватят как технические аспекты, так и интерфейсные улучшения для более удобного и эффективного взаимодействия с системой. В этом разделе представлены ключевые направления для дальнейшего развития.

* 1. **Дополнительные функции для улучшения функционал**

В будущем проект будет дополнен новыми функциями, направленными на улучшение анализа данных и управления запасами. На основе макета из Figma уже определены дополнительные функциональные возможности, которые будут реализованы в следующей версии системы.

1. **График популярности за различные периоды времени**

Одним из ключевых дополнений станет возможность отображения графиков, показывающих популярность продуктов за разные временные периоды: день, неделю, месяц, год и за весь период. Эти графики будут динамически обновляться в зависимости от выбранного пользователем периода времени в соответствии с рисунком. График популярности за различные периоды времени представлен на рисунке 6.

* **Периоды для анализа**: Пользователи смогут выбирать, за какой период времени они хотят увидеть данные о продажах. Это поможет детально анализировать эффективность продаж в разные временные отрезки, например, за день, неделю, месяц или год.
* **Типы графиков**: Графики будут отображать количество проданных единиц для каждого продукта, что позволит пользователям быстро определить, какие продукты являются наиболее популярными за выбранный период.
* **Будущее расширение**: В дальнейших обновлениях планируется добавить интерактивные графики, которые позволят пользователю настраивать период времени самостоятельно, а также визуализировать данные в реальном времени. Такой подход повысит гибкость и функциональность системы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6 – Макет интерфейса с графиками популярности продуктов (графики, отображающие данные за разные периоды времени)

1. **Виджет сравнения продаж с предыдущим днем**

В систему будет добавлен виджет для сравнения продаж текущего дня с предыдущим. Этот функционал позволит пользователям оперативно отслеживать изменения в продажах и предпринимать необходимые действия для корректировки запасов и стратегии продаж. Виджет сравнения продаж представлен на рисунке 7.

* **Описание функционала**: виджет будет автоматически сравнивать количество проданных единиц каждого продукта с предыдущим днем, предоставляя статистику в виде текстовых отчетов или графиков.
* **Цель**: это поможет выявить тренды и оперативно реагировать на изменения в спросе, например, если продажи какого-либо продукта значительно увеличились или снизились по сравнению с предыдущим днем.
* **Пример статистики**: виджет будет отображать изменения в продажах по каждому продукту, например, "Продано на 20% больше, чем вчера". Это позволит менеджерам быстро адаптировать стратегию пополнения запасов.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7 – Макет виджета для сравнения продаж с предыдущим днем (сравнение показателей за текущий и предыдущий день).

1. **Виджет для просмотра информации о клиентах и их покупках**

В макете предусмотрен новый виджет, который позволит пользователям просматривать информацию о клиентах и их покупках. Этот функционал будет полезен для анализа поведения клиентов и принятия решений по улучшению стратегии продаж и маркетинга. Виджет для просмотра информации о клиентах и их покупках представлен на рисунке 8.

* **Описание функционала**: виджет будет отображать информацию о клиентах, включая их историю покупок, предпочтения, частоту покупок и другие параметры. Это даст возможность более точно анализировать поведение клиентов и разрабатывать персонализированные предложения.
* **Цель**: виджет поможет компаниям эффективно анализировать информацию о клиентах, выявлять тренды и разрабатывать стратегию улучшения продаж.
* **Пример использования**: Пользователи смогут видеть, какие продукты покупает конкретный клиент, как часто он совершает покупки и какие товары пользуются наибольшим спросом.

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

Автоматически созданное описание

Рисунок 8 — Макет виджета для просмотра информации о клиентах и их покупках.

* 1. **Развитие пользовательского интерфейса и добавление новых графиков**

Планируется продолжить работу над улучшением пользовательского интерфейса (UI), добавив новые возможности для более удобного и функционального взаимодействия с системой. Одним из важнейших шагов будет развитие графического отображения данных.

**Поддержка различных типов графиков**: в проект будет интегрирована поддержка дополнительных типов графиков, которые позволят пользователю более гибко настраивать визуализацию данных. Эти графики могут включать столбчатые, линейные, круговые диаграммы и другие варианты отображения, которые будут зависеть от типа данных.

**Интерактивность графиков**: одним из направлений для дальнейшего развития является внедрение интерактивных графиков, позволяющих пользователю самостоятельно выбирать временные интервалы для отображения данных, а также просматривать детализированную информацию по каждому продукту.

**Обновление интерфейса**: пользовательский интерфейс будет дополнен виджетами и фильтрами, которые обеспечат более удобное отображение и анализ данных. Интуитивно понятный интерфейс и улучшенная визуализация позволят пользователю быстрее ориентироваться в информации и принимать решения.

* 1. **Реализация обновления данных**

Одним из важнейших шагов в развитии проекта будет реализация механизма обновления данных в реальном времени. Это обеспечит пользователям постоянную актуальность информации и позволит оперативно реагировать на изменения.

**Динамическое обновление данных**: для обеспечения точности и своевременности данных система будет использовать механизмы частичных обновлений. Это значит, что информация будет обновляться без необходимости перезагрузки всей системы, что обеспечит более быструю работу.

**Автоматическое обновление**: система будет работать в фоновом режиме, автоматически обновляя данные о продажах и запасах. Менеджеры смогут получать актуальную информацию в любое время, что позволит им оперативно реагировать на изменения и корректировать стратегию пополнения запасов.

**Оповещения и уведомления**: в системе будут реализованы уведомления о потребности в пополнении запасов, что поможет избежать дефицита продукции и улучшить процессы логистики.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данного проекта была разработана система для анализа данных о продажах продуктов и потребности в пополнении их запасов. Основной целью проекта было создание инструмента, который позволяет эффективно управлять товарными запасами, анализируя популярность продуктов и определяя потребность в их пополнении. Система включает в себя несколько ключевых компонентов, таких как классы для работы с продуктами и продажами, анализ данных, а также визуализация результатов в виде графиков.

Особое внимание было уделено тестированию и отладке кода. В ходе тестирования были выявлены и исправлены ошибки, связанные с обработкой данных, агрегацией продаж, а также с отображением графиков. Внедрение модульных тестов позволило убедиться в корректности работы отдельных компонентов системы и повысить ее надежность. Все основные ошибки, выявленные в процессе разработки, были успешно устранены, что позволило системе работать стабильно и без сбоев.

Результаты работы проекта были продемонстрированы в виде графиков популярности продуктов и потребности в пополнении запасов, что позволяет принимать обоснованные решения о дальнейших закупках и корректировке товарных запасов. Разработанный инструмент может быть полезен для бизнес-анализа в различных сферах, связанных с продажами и логистикой.

В будущем проект может быть расширен за счет добавления дополнительных функциональных возможностей, таких как более сложные алгоритмы анализа, интеграция с внешними источниками данных и улучшение интерфейса визуализации. Также возможно расширение системы для обработки больших объемов данных и работы в реальном времени, что позволит еще более эффективно управлять запасами и продажами.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Стандарты и законодательные материалы

1. Федеральный закон от 13 июля 2015 года № 266-ФЗ «О бухгалтерском учете».
2. Федеральный закон от 27 ноября 2010 года № 312-ФЗ «О защите прав потребителей».
3. Постановление Правительства РФ от 10 октября 2018 года № 1169 «Об утверждении правил ведения учета товаров и запасов на складе».
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

1. Хаштамов Адиль Введение в pandas: анализ данных на Python // https://khashtamov.com/ru/pandas-introduction/
2. 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
3. SkillFactory Pandas // https://blog.skillfactory.ru/glossary/pandas/
4. SkillBox Библиотека Matplotlib для построения графиков // https://skillbox.ru/media/code/biblioteka-matplotlib-dlya-postroeniya-grafikov/
5. PyDocs Модуль Matplotlib // https://pydocs.ru/modul-matplotlib/
6. Tomicevik Dominik & BudiselićMarkoLearn Graph Analytics with Python // https://memgraph.com/blog/learn-graph-analytics-with-python
7. Балакирев С.В. Рисуем гистограммы, столбчатые и круговые диаграммы // https://proproprogs.ru/modules/matplotlib-risuem-gistogrammy-stolbchatye-i-krugovye-diagrammy
8. Балакирев С.В. Добрый, добрый Python с Сергеем Балакиревым // https://stepik.org/course/100707/syllabus
9. Росстат Розничная торговля и общественное питание // https://rosstat.gov.ru/statistics/roznichnayatorgovlya
10. Мустафин Ильдар Финансовый учёт для малого бизнеса // https://www.moedelo.org/club/upravlencheskiy-uchet/upravlencheskii-uchet-dlya-malogo-biznesa
11. Самойленко Наталья Работа с файлами в формате JSON // https://pyneng.readthedocs.io/ru/latest/book/17\_serialization/json.html
12. Климова Тамара Маркетинговая аналитика: ассортиментный анализ сети кофеен // https://place.lemma.ru/article/marketingovaya-analitika-assortimentnyi-analiz-seti-kofeen
13. Попов Евгений Объектно-ориентированное программирование. Классы и объекты // https://metanit.com/python/tutorial/7.1.php

**ПРИЛОЖЕНИЕ 1** Код модуля main.py

import json  
from product import Product  
from sale import Sale  
from inventory\_manager import InventoryManager  
from sales\_analyzer import SalesAnalyzer  
from visualizer import Visualizer  
  
  
# Загрузка данных из JSON  
def load\_json\_data(filename):  
 with open(filename, 'r', encoding='utf-8') as file:  
 return json.load(file)  
  
  
def main():  
 # Загрузка данных  
 products\_data = load\_json\_data('products.json')  
 sales\_data = load\_json\_data('sales\_data.json')  
  
 if not sales\_data or not products\_data:  
 print("Нет данных для анализа.")  
 return  
  
 # Создание объектов для продуктов и продаж  
 products = [Product(\*\*p) for p in products\_data]  
 sales = [Sale(\*\*s) for s in sales\_data]  
  
 # Анализ популярности продуктов  
 analyzer = SalesAnalyzer(sales, products)  
 product\_sales = analyzer.analyze\_popularity()  
 print("Анализ популярности продуктов:")  
 print(product\_sales)  
  
 # Визуалка популярности продуктов  
 visualizer = Visualizer()  
 visualizer.plot\_popularity(product\_sales)  
  
 # Анализ пополнения запасов  
 inventory\_manager = InventoryManager(products, sales)  
 low\_stock\_products = inventory\_manager.analyze\_inventory\_needs()  
 print("\nПродукты, которые требуют пополнения запасов:")  
 print(low\_stock\_products)  
  
 # Визуалка пополнения запасов  
 visualizer.plot\_inventory(low\_stock\_products)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**ПРИЛОЖЕНИЕ 2** Код модуля product.py

class Product:  
 def \_\_init\_\_(self, product\_id, product\_name, stock\_quantity):  
 self.product\_id = product\_id  
 self.product\_name = product\_name  
 self.stock\_quantity = stock\_quantity  
  
 def \_\_str\_\_(self):  
 return f"{self.product\_name} ({self.product\_id})"  
  
 def \_\_repr\_\_(self):  
 return f"""  
 Товары  
 (product\_id = {self.product\_id},  
 product\_name = {self.product\_name},  
 stock\_quantity = {self.stock\_quantity})  
 """

**ПРИЛОЖЕНИЕ 3** Код модуля sale.py

class Sale:  
 def \_\_init\_\_(self, product\_id, sold\_units):  
 self.product\_id = product\_id  
 self.sold\_units = sold\_units  
  
 def \_\_str\_\_(self):  
 return f"Продажи {self.product\_id}, всего продано: {self.sold\_units}"  
  
 def \_\_repr\_\_(self):  
 return f"""  
 Продажи  
 (product\_id = {self.product\_id},  
 sold\_units = {self.sold\_units})  
 """

**ПРИЛОЖЕНИЕ 4** Код модуля inventory\_manager.py

import pandas as pd  
  
  
class InventoryManager:  
 def \_\_init\_\_(self, products, sales):  
 self.products = products  
 self.sales = sales  
  
 def analyze\_inventory\_needs(self):  
 sales\_by\_product = {sale.product\_id: sale.sold\_units for sale in self.sales}  
  
 low\_stock = []  
  
 for product in self.products:  
 required\_stock = sales\_by\_product.get(product.product\_id, 0) + 10 - product.stock\_quantity  
 if required\_stock > 0:  
 low\_stock.append((product.product\_name, required\_stock))  
  
 df = pd.DataFrame(low\_stock, columns=['Название', 'Необходимый запас'])  
 return df

**ПРИЛОЖЕНИЕ 5** Код модуля sales\_analyzer.py

import pandas as pd  
  
  
class SalesAnalyzer:  
 def \_\_init\_\_(self, sales, products):  
 self.sales = sales  
 self.products = products  
  
 def analyze\_popularity(self):  
 sales\_df = pd.DataFrame([vars(sale) for sale in self.sales])  
 product\_sales = sales\_df.groupby('product\_id')['sold\_units'].sum().reset\_index()  
 product\_sales = product\_sales.merge(pd.DataFrame([vars(product) for product in self.products]),  
 on='product\_id')  
 return product\_sales.sort\_values(by='sold\_units', ascending=False)

**ПРИЛОЖЕНИЕ 6** Код модуля visualizer.py

import matplotlib.pyplot as plt  
  
  
class Visualizer:  
 @staticmethod  
 def plot\_popularity(product\_sales, sort\_by='sold\_units', color='plum'):  
 *"""  
 График популярности продуктов  
 product\_sales: данные о продажах продуктов в виде DataFrame  
 sort\_by: сортировка по столбц  
 color: цвет столбцов  
 """* product\_sales = product\_sales.sort\_values(by=sort\_by, ascending=False)  
  
 plt.figure(figsize=(10, 6))  
 bars = plt.barh(product\_sales['product\_name'], product\_sales['sold\_units'], color=color)  
 plt.title('Популярность продуктов')  
 plt.xlabel('Количество проданных единиц')  
 plt.ylabel('Продукты')  
  
 for bar in bars:  
 plt.text(bar.get\_width(), bar.get\_y() + bar.get\_height() / 2,  
 f'{bar.get\_width()}', va='center', ha='left', color='black')  
  
 plt.tight\_layout()  
 plt.show()  
  
 @staticmethod  
 def plot\_inventory(low\_stock\_df, color='powderblue'):  
 *"""  
 График потребности в пополнении запасов  
 low\_stock\_df: данные о товарах, требующих пополнения в виде DataFrame  
 color: цвет столбцов  
 """* if low\_stock\_df.empty:  
 print("Все товары в достаточном количестве.")  
 return  
  
 plt.figure(figsize=(10, 6))  
 bars = plt.barh(low\_stock\_df['Название'], low\_stock\_df['Необходимый запас'], color=color)  
 plt.title('Необходимость пополнения запасов')  
 plt.xlabel('Необходимое количество для пополнения')  
 plt.ylabel('Продукты')  
  
 for bar in bars:  
 plt.text(bar.get\_width(), bar.get\_y() + bar.get\_height() / 2,  
 f'{bar.get\_width()}', va='center', ha='left', color='black')  
  
 plt.tight\_layout()  
 plt.show()

**ПРИЛОЖЕНИЕ 7** Код теста test\_product.py

import pytest  
from product import Product  
  
# Параметризуем данные для инициализации продукта  
@pytest.mark.parametrize(  
 "product\_id, product\_name, stock\_quantity",  
 [  
 (1, "Латте", 90),  
 (2, "Капучино", 75),  
 (3, "Черный чай", 50),  
 (4, "Двойной эспрессо", 10)  
 ]  
)  
def test\_product\_initialization(product\_id, product\_name, stock\_quantity):  
 product = Product(product\_id, product\_name, stock\_quantity)  
 assert product.product\_id == product\_id  
 assert product.product\_name == product\_name  
 assert product.stock\_quantity == stock\_quantity  
  
# Параметризуем данные для метода \_\_str\_\_  
@pytest.mark.parametrize(  
 "product\_id, product\_name, expected\_str",  
 [  
 (1, "Латте", "Латте (1)"),  
 (2, "Капучино", "Капучино (2)"),  
 (3, "Черный чай", "Черный чай (3)"),  
 (4, "Двойной эспрессо", "Двойной эспрессо (4)")  
 ]  
)  
def test\_product\_str\_method(product\_id, product\_name, expected\_str):  
 product = Product(product\_id, product\_name, 90) # stock\_quantity не влияет на \_\_str\_\_  
 assert str(product) == expected\_str

**ПРИЛОЖЕНИЕ 8** Код теста test\_sales\_analyzer.py

import pytest  
from sales\_analyzer import SalesAnalyzer  
from product import Product  
from sale import Sale  
  
# Параметризуем данные для продуктов и продаж  
@pytest.mark.parametrize(  
 "products, sales, expected\_first, expected\_second, expected\_last",  
 [  
 (  
 [  
 Product(1, "Латте", 90),  
 Product(2, "Капучино", 75),  
 Product(3, "Черный чай", 50),  
 Product(4, "Двойной эспрессо", 10),  
 Product(5, "Американо", 30)  
 ],  
 [  
 Sale(1, 50),  
 Sale(2, 70),  
 Sale(3, 30),  
 Sale(4, 5),  
 Sale(5, 15)  
 ],  
 "Капучино",  
 "Латте", # Ожидаемые элементы  
 "Двойной эспрессо"  
 )  
 ]  
)  
def test\_analyze\_popularity(products, sales, expected\_first, expected\_second, expected\_last):  
 analyzer = SalesAnalyzer(sales, products)  
 result = analyzer.analyze\_popularity()  
  
 # Проверка, что результат отсортирован по проданным единицам (ascending='False')  
 assert result.iloc[0]['product\_name'] == expected\_first  
 assert result.iloc[1]['product\_name'] == expected\_second  
 assert result.iloc[4]['product\_name'] == expected\_last  
  
 # Проверка правильности данных  
 assert result.iloc[0]['sold\_units'] == 70  
 assert result.iloc[1]['sold\_units'] == 50