

Федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»
Кафедра «Автоматизированные системы обработки информации и
управления»

ОТЧЕТ

по созданию дипломного проекта с помощью ChatGPT
по дисциплине «Современные инструментальные средства разработки
программного обеспечения»

Работу выполнил:
студент гр. ПИН-201
А. Д. Дергачев

(подпись, дата)

Работу проверил:
ст. преподаватель
А.А. Кабанов

(подпись, дата)

Омск 2024

Для диплома я планирую создать приложение на Python для работы с маркетплейсами, я написал ChatGPT какие темы можно выбрать в качестве дипломного проекта.

Конечно, вот несколько потенциальных тем для дипломной работы, связанные с разработкой приложения для работы с маркетплейсами:

1. "Разработка мобильного приложения для управления аккаунтом продавца на маркетплейсе"
2. "Исследование и разработка алгоритмов рекомендации товаров в приложении маркетплейса"
3. "Анализ и оптимизация процесса поиска и фильтрации товаров в мобильном приложении маркетплейса"
4. "Разработка функционала реализации покупок через мобильное приложение маркетплейса"
5. "Интеграция различных платёжных систем в мобильном приложении для маркетплейса"
6. "Исследование и применение технологий машинного обучения для улучшения пользовательского опыта в приложении маркетплейса"

Эти темы можно дополнить, углубить и адаптировать под ваши интересы и специализацию. Выбирая тему для дипломной работы, важно учитывать свои знания, интересы и возможности для реализации проекта.

Далее была выбрана тема “Разработка автоматизированной системы анализа и обработки данных для маркетплейсов”

1. Определение функциональных требований:

- Определение основных целей и функций приложения для работы с маркетплейсами, таких как ABC анализ товара, расчет поставок и т.п.
- Формирование Excel документов для дальнейшей обработки операторами.

2. Проектирование пользовательского интерфейса (UI):

- Разработка интуитивно понятного и привлекательного пользовательского интерфейса для удобного взаимодействия с приложением.
- Создание макетов и схем пользовательского интерфейса с учетом потребностей пользователей.

3. Разработка бэкенда:

- Реализация работы с маркетплейсом через API.

4. Разработка десткоп приложения:

- Реализация всех необходимых функций, таких как загрузка отчетов, анализ продаж и остатков, расчет поставок.

5. Реализация аналитики и отчетности:

- Разработка алгоритмов для анализа данных о заказах и продажах.
- Создание отчетов и графиков для анализа продаж.

6. Управление пользователями:

- Реализация системы аутентификации и авторизации пользователей.

7. Тестирование и отладка:

- Проведение тестирования приложения на различных платформах и устройствах для обнаружения и устранения ошибок и несоответствий.

8. Поддержка и обновление приложения:

- Оказание технической поддержки, реагирование на обратную связь пользователей и исправление выявленных проблем.
- Регулярное обновление и улучшение функций приложения.

Далее я начал отправлять запросы с моими требованиями:

Представь, что мы сейчас разрабатываем приложение для анализа и обработки данных для маркетплейсов, определи функциональные требования приложения

Конечно, при разработке приложения для анализа и обработки данных для маркетплейсов было бы важно определить следующие функциональные требования:

1. Автоматизированный анализ данных:

- Возможность автоматической загрузки и обработки данных из различных источников на маркетплейсе.
- Реализация различных алгоритмов анализа данных для выявления трендов, популярности товаров, поведения пользователей и прочего.

2. Гибкий отчетный функционал:

- Возможность создания персонализированных отчетов и дашбордов на основе анализируемых данных.
- Встроенные инструменты для визуализации данных, такие как диаграммы, графики, тепловые карты и т.д.

3. Управление контентом:

- Возможность редактирования и планирования контента на маркетплейсе, например, публикация товаров, акций, описаний и т.д.
- Механизмы управления каталогами товаров, категориями, фильтрами и т.д.

4. Инструменты мониторинга:

- Возможность отслеживания показателей эффективности и продаж, мониторинга активности пользователей и действий конкурентов.

- Уведомления и алерты для оперативного реагирования на изменения на маркетплейсе.

5. Интеграция с внешними системами:

- Возможность интеграции с платежными системами, поставщиками, системами логистики и другими сторонними сервисами для полноценного функционирования маркетплейса.

Это лишь общий перечень функциональных требований, который можно дополнить или уточнить в зависимости от конкретных потребностей и целей разрабатываемого приложения.

Хорошо, а теперь напиши, как будет проектироваться пользовательский интерфейс

При проектировании пользовательского интерфейса для приложения анализа и обработки данных для маркетплейсов, следует уделить внимание следующим аспектам:

1. Интуитивно понятный дизайн:

- Разработка простого и интуитивно понятного интерфейса для удобного взаимодействия пользователя с приложением.
- Максимальное снижение числа кликов для доступа к ключевым функциям приложения.

2. Понятная навигация:

- Четкая структура навигации, позволяющая пользователям легко перемещаться между различными разделами и функциями приложения.
- Использование меню, вкладок, кнопок и других элементов для удобства перемещения по приложению.

3. Визуализация данных:

- Использование графиков, диаграмм, тепловых карт и других элементов визуализации данных для наглядного представления информации.
- Удобство фильтрации и сортировки данных для анализа и сравнения.

4. Настройка отображения:

- Возможность настройки отображения информации в соответствии с потребностями пользователя, например, выбор типа графика, параметров отчетов и прочего.
- Предоставление функционала для сохранения настроек и создания персонализированных видов данных.

5. Безопасность и конфиденциальность:

- Обеспечение безопасности данных пользователей и конфиденциальности персональной информации.
- Внедрение механизмов аутентификации, шифрования данных и других мер для защиты информации.

6. Отзывчивый дизайн (Responsive Design):

- Учет адаптивности интерфейса под различные устройства и разрешения экранов, чтобы обеспечить оптимальный пользовательский опыт на всех устройствах.

Эти принципы помогут создать удобный, функциональный и привлекательный пользовательский интерфейс для приложения анализа и обработки данных для маркетплейсов.

Напиши код для разработки интерфейса на Python для десктоп приложения для анализа и обработки данных для маркетплейсов

Для разработки интерфейса на Python для десктоп приложения, предназначенного для анализа и обработки данных маркетплейсов, мы можем воспользоваться библиотеками Tkinter для создания GUI и pandas для работы с данными. Ниже приведен пример кода, который демонстрирует простой интерфейс для загрузки и анализа CSV файла с помощью pandas:

```
import tkinter as tk
from tkinter import filedialog
import pandas as pd

# Функция для загрузки и анализа данных из CSV файла
def analyze_data():
    file_path = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])

    if file_path:
        data = pd.read_csv(file_path)
        data_analysis = data.describe()

        # Отображение результатов анализа данных
        text_widget.delete(1.0, tk.END) # Очистить текстовый виджет
        text_widget.insert(tk.END, data_analysis)

# Создание главного окна
root = tk.Tk()
root.title("Анализ данных маркетплейсов")

# Кнопка для загрузки и анализа данных
load_button = tk.Button(root, text="Загрузить и проанализировать данные",
command=analyze_data)
load_button.pack()

# Текстовый виджет для отображения результатов анализа
text_widget = tk.Text(root)
text_widget.pack()

# Запуск главного цикла обработки событий
root.mainloop()
```

Этот код создает простой GUI, который позволяет пользователю загрузить CSV файл, проанализировать его с помощью pandas и отобразить результаты анализа в текстовом виджете. Вы можете доработать этот пример, добавив другие функции анализа данных и улучшив интерфейс в соответствии

с требованиями вашего проекта.

А как будет разрабатываться бэкенд приложения с получением данных через API

Для разработки бэкенд части десктоп приложения на Python, которое будет обрабатывать данные для анализа маркетплейсов через API, вы можете использовать такие инструменты как фреймворк Flask, библиотеку requests для выполнения HTTP запросов к API и библиотеку pandas для работы с данными.

Примерный план разработки мог бы включать следующие шаги:

1. **Установка необходимых библиотек:** установите необходимые библиотеки, такие как Flask, requests и pandas, используя pip:
pip install Flask requests pandas
2. **Создание Flask приложения:** Создайте Flask приложение, которое будет обрабатывать запросы к вашему API и выполнять анализ данных. Примерно это могло бы выглядеть так:

```
from flask import Flask, jsonify
import requests
import pandas as pd

app = Flask(__name__)

@app.route('/get_marketplace_data')
def get_marketplace_data():
    api_url = 'https://api.marketplace.com/data'
    response = requests.get(api_url)

    if response.status_code == 200:
        data = response.json()
        # Анализ данных с использованием pandas
        df = pd.DataFrame(data)
        # Дальнейшая обработка данных...

        return jsonify({'message': 'Data analyzed successfully'})
    else:
        return jsonify({'error': 'Failed to fetch data'})
```

```
if __name__ == '__main__':  
    app.run(debug=True)
```

3. Обработка данных и анализ: внутри вашего Flask приложения используйте библиотеку `pandas` для анализа данных, полученных через API. Вы можете выполнять различные манипуляции с данными, агрегировать их, строить графики и применять различные методы анализа.

4. Дальнейшая обработка данных и функциональность: В зависимости от ваших потребностей, вы можете добавить дополнительные функции в ваше бэкенд приложение, такие как сохранение результатов анализа, генерация отчетов, визуализация данных и другие.

Не забудьте, что для реального проекта вам следует уделить внимание безопасности передачи данных, обработке ошибок, масштабируемости и другим важным аспектам разработки программного обеспечения.

А как будет происходить разработка приложения, его основных функций, таких как анализ продаж и остатков, загрузка отчетов и расчет поставок

Для разработки приложения, осуществляющего анализ продаж и остатков, загрузку отчетов и расчет поставок, вам потребуется хорошо спланировать функциональность и процессы, которые будет выполнять ваше приложение. Вот общий план, который вы можете использовать для разработки такого приложения:

1. Определение требований: Начните с определения основных требований к вашему приложению. Это может включать в себя функции, типы данных, используемые источники данных, требования к безопасности и т.д.

2. Проектирование базы данных: Разработайте структуру базы данных для хранения данных о продажах, остатках, отчетах и поставках. Определите таблицы, их поля, связи между ними и т.д.

3. Разработка бэкенд части: Напишите бэкенд приложения, которое будет обрабатывать запросы, выполнять анализ данных, загружать отчеты, рассчитывать поставки и взаимодействовать с базой данных. Для этого вы можете использовать язык программирования и фреймворк, на ваш выбор, например Python с Flask или Django.

4. Разработка фронтенд части: Создайте пользовательский интерфейс, который позволит пользователям взаимодействовать с вашим приложением. Разработайте страницы для просмотра отчетов, ввода данных, просмотра аналитики и т.д.

5. Интеграция с внешними сервисами: Если ваше приложение необходимо будет взаимодействовать с внешними сервисами для получения данных или выполнения дополнительных действий, убедитесь, что вы правильно настроили их интеграцию.

6. Тестирование и отладка: Протестируйте ваше приложение, убедитесь, что все функции работают корректно и выполняются без ошибок. Исправьте любые обнаруженные проблемы.

7. Внедрение и мониторинг: Разверните ваше приложение, чтобы оно было доступно пользователям. Убедитесь, что у вас есть механизмы для мониторинга производительности и доступности.

8. Поддержка и обновление: Обеспечьте поддержку вашего приложения, реагируйте на обратную связь пользователей, выпускайте регулярные обновления для улучшения функционала и исправления ошибок.

Это общий план, который поможет вам разработать приложение для анализа продаж и остатков, загрузки отчетов и расчета поставок. Не забывайте о том, что в процессе разработки могут возникнуть дополнительные задачи и требования, которые также нужно учитывать.

Хорошо, а как будет происходить реализация алгоритмов и отчетности, разработка алгоритмов для анализа продаж и заказов и создание отчетов и графиков

Для реализации алгоритмов анализа продаж и заказов, а также для создания отчетов и графиков вам может потребоваться следующий план действий:

1. Выбор инструментов и технологий: Определите необходимые инструменты для работы с данными и аналитики. Это может включать в себя использование языков программирования (например, Python, R), библиотек и фреймворков для анализа данных (например, Pandas, NumPy, SciPy), инструментов визуализации данных (например, Matplotlib, Seaborn) и СУБД для хранения данных.

2. Создание алгоритмов анализа данных: Разработайте алгоритмы для анализа данных о продажах и заказах. Это может включать в себя вычисление ключевых метрик (например, оборот, прибыль, соотношение себестоимости), проведение сезонного анализа, определение трендов и паттернов в данных.

3. Разработка отчетов и графиков: Создайте отчеты и графики на основе результатов анализа данных. Это может быть сводная таблица с основными показателями продаж, диаграммы для визуализации динамики продаж, гистограммы для анализа распределения данных и т.д. Используйте выбранные инструменты для визуализации данных, чтобы создать информативные отчеты и графики.

4. Тестирование и оптимизация: Протестируйте разработанные алгоритмы и отчеты, убедитесь, что результаты соответствуют

ожиданиям. Проведите оптимизацию алгоритмов для улучшения производительности и эффективности работы приложения.

5. Интеграция с приложением: Интегрируйте разработанные алгоритмы и отчеты с вашим приложением для анализа продаж и заказов. Убедитесь, что данные передаются корректно между различными модулями приложения.

6. Обучение пользователей: Обучите пользователей вашего приложения работать с разработанными алгоритмами и отчетами. Предоставьте необходимую документацию и обучающие материалы для комфортной работы с новыми функциями.

7. Поддержка и дальнейшее развитие: Обеспечьте поддержку пользователям в работе с созданными отчетами и алгоритмами, а также проводите дальнейшее развитие функционала на основе обратной связи пользователей.

А как будет происходить управление пользователями и как будет реализована система аутентификации и авторизации

Для управления пользователями и реализации системы аутентификации и авторизации в приложении можно использовать следующие методы и технологии:

1. База данных пользователей: Создание базы данных для хранения информации о пользователях, такой как логины, пароли, роли пользователей и другие связанные данные. Для этого можно использовать реляционные базы данных (например, MySQL, PostgreSQL) или NoSQL базы данных (например, MongoDB).

2. Хэширование паролей: Для обеспечения безопасности паролей пользователей следует использовать хэширование паролей при их хранении в базе данных. Рекомендуется применять современные хэширования, такие как bcrypt или Argon2.

3. Аутентификация: Реализация процесса аутентификации пользователей при помощи логина и пароля, токенов или других методов. Обычно это включает проверку соответствия введенных пользователем учетных данных данным в базе данных.

4. Сессии и токены: Работа с сессиями и токенами для сохранения состояния аутентификации пользователя в приложении. Сессия может быть хранена на сервере или в куках браузера, а также можно использовать токены доступа для авторизации запросов к API.

5. Ролевая модель доступа: Определение ролей пользователей и реализация соответствующих прав доступа к различным

функциональным возможностям приложения. Это позволяет ограничить доступ к определенным ресурсам в зависимости от роли пользователя.

6. JWT (JSON Web Tokens): Использование JWT для создания и проверки токенов доступа, которые могут содержать информацию об аутентификации и авторизации пользователя. JWT может быть эффективным способом управления сессиями пользователей и обеспечения безопасности.

7. SSL/TLS шифрование: Обеспечение безопасной передачи данных между клиентом и сервером путем использования протоколов шифрования SSL/TLS. Это защитит данные пользователя от перехвата или изменения во время передачи.

8. Логирование и мониторинг: Ведение журналов действий пользователей и администраторов в приложении, что позволяет отслеживать активность пользователей и обнаруживать ненормальное поведение или попытки несанкционированного доступа.

Реализация системы управления пользователями и безопасности в приложении важна для обеспечения конфиденциальности и целостности данных, а также защиты от киберугроз и несанкционированного доступа.

А как провести тестирование такого приложения

Для тестирования приложения с системой управления пользователями и авторизацией следует учитывать несколько аспектов, чтобы гарантировать безопасность, надежность и правильную работу функциональности. Вот некоторые шаги, которые можно выполнить в процессе тестирования подобного приложения:

1. Тестирование функциональности:

- Проверка регистрации новых пользователей с использованием различных комбинаций данных.
- Проверка аутентификации пользователя с корректными и недействительными учетными данными.
- Проверка функциональности смены пароля и сброса пароля.
- Проверка функциональности управления ролями и правами доступа.

2. Тестирование безопасности:

- Проверка на наличие уязвимостей входа, таких как подбор пароля, CSRF-атаки, инъекции SQL.
- Тестирование стойкости паролей и механизмов хэширования паролей.
- Проверка правильности управления сессиями и токенами доступа.
- Проверка наличия SSL/TLS шифрования для безопасной передачи данных.

3. Тестирование производительности:

- Проверка нагрузочной стойкости системы при большом количестве одновременных пользователей.

- Оценка времени ответа системы при различных действиях пользователей.

4. Тестирование совместимости:

- Проверка работы приложения на различных браузерах и устройствах.
- Тестирование на различных операционных системах (Windows, MacOS, Linux).

5. Тестирование на безопасность:

- Проверка на наличие утечек информации.
- Проверка на наличие возможности злоумышленников проникнуть в систему.
- Анализ возможности атак на приложение, таких как переполнение буфера, межсайтовый скриптинг (XSS) и другие.

Важно также уделить внимание автоматизации тестирования с использованием инструментов, таких как Selenium, Postman, OWASP ZAP и другие, чтобы повысить эффективность и повторяемость тестов. Кроме того, важно проводить регулярные проверки на безопасность и обновлять зависимости приложения для минимизации рисков.

Как будет реализована поддержка актуальной версии и обновление приложения

Поддержка актуальной версии и регулярное обновление приложения играют важную роль в обеспечении его безопасности, стабильности и функциональности. Вот несколько методов, которые могут быть использованы для реализации поддержки актуальной версии и обновления приложения:

1. Автоматическое обновление: Реализация механизма автоматического обновления позволит пользователям получать доступ к последним версиям приложения без необходимости ручных действий. Приложение может регулярно проверять наличие обновлений и загружать их автоматически.

2. Уведомления о доступных обновлениях: Реализация функциональности, позволяющей пользователям получать уведомления о наличии новых версий приложения, поможет им быть в курсе изменений и обновлений.

3. Централизованный управляемый процесс обновления: Создание централизованного механизма управления обновлениями приложения, который позволит администраторам безопасно и эффективно устанавливать и контролировать процесс обновления на всех устройствах и серверах.

4. Публикация обновлений через официальные магазины: Размещение приложения в официальных магазинах приложений (например, App Store, Google Play) и публикация обновлений через них обеспечивает простой доступ к обновлениям для пользователей и дает

возможность привлечения новых пользователей.

5. Тестирование перед выпуском: Перед выпуском обновлений важно проводить тщательное тестирование, чтобы убедиться в их стабильности, безопасности и совместимости с различными устройствами и операционными системами.

6. Создание регулярного графика обновлений: Разработка графика регулярных обновлений поможет пользователям ожидать и готовиться к появлению новых версий приложения, а также позволит вам планировать ресурсы для разработки и тестирования обновлений.

7. Обратная связь и мониторинг: Важно учитывать отзывы пользователей об обновлениях и мониторить процесс обновлений, чтобы быстро реагировать на проблемы и улучшать процесс обновления в будущем.

Применение этих методов позволит обеспечить бесперебойную работу и безопасность вашего приложения, что в свою очередь способствует удовлетворенности пользователей и успеху приложения на рынке.

Заключение

В целом, нейросеть объяснила основные моменты разработки моего приложения для дипломной работы и подсказала чем можно воспользоваться для правильной реализации задуманного функционала.