|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **НПОУ «ЯКУТСКИЙ КОЛЛЕДЖ ИННОВАЦИОННЫХ ТЕХНОЛОГИЙ»** | | | | | | | |
| (наименование образовательной организации) | | | | | | | | | |
|  | | | | | | | | | |
|  | **ОТДЕЛЕНИЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** | | | | | | |
| (наименование структурного подразделения (кафедра / отделение)) | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
| **ОТЧЕТ** | | | | | | | | | |
| **по Языкам программирования** | | | | | | | | | |
| Веб-парсинг сайта Озон для сбора отзывов с использованием семантического анализа текста | | | | | | | | | |
| (наименование типа практики) | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | | |  | Выполнили: | | | | | |
|  | | |  | обучающиеся | | КИСП-23 | группы | | |
|  | | |  | Варламов Никита Андреевич  Исаков Илья Игоревич  Левин Артем Данилович  Огонерова Сардаана Васильевна  Сафаргалеев Владимир Викторович  Яковлева Евгения Николаевна | | | | | |
|  | | |  | (фамилия, имя, отчество (при наличии)) | | | | | |
|  | | |  | \_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_ 20\_\_\_ г. | | | | | |
|  | | |  | (личная подпись, дата) | | | | | |
|  | | |  |  | | | | | |
|  | | |  |  | | | | | |
| Проверил руководитель практики | | |  | Проверил: | | | | | |
|  | | |  | преподаватель | | | | | |
| (уч. степень, уч. звание, должность) | | |  | (уч. степень, уч. звание, должность) | | | | | |
|  | | |  | Федоров Дьулуур Андрианович | | | | | |
| (фамилия, имя, отчество (при наличии)) | | |  | (фамилия, имя, отчество (при наличии)) | | | | | |
| Оценка | |  |  | Оценка |  | | | | |
| \_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_ 20\_\_\_ г. | | |  | \_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_ 20\_\_\_ г. | | | | | |
| (личная подпись, дата) | | |  | (личная подпись, дата) | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
| Якутск, 2025 | | | | | | | | | |

Содержание

[ВВЕДЕНИЕ 3](#_Toc199586661)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc199586662)

[1.1. Терминология по проекту и глоссарий 4](#_Toc199586663)

[1.2. Распределение ролей и работы 7](#_Toc199586664)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 12](#_Toc199586665)

[2.1. Создание кода для парсинга 12](#_Toc199586666)

[2.2. Создание Telegram бота 15](#_Toc199586667)

[2.3. Создание аккаунта в Yandex Cloud 18](#_Toc199586668)

[2.4. Создание API-ключа 19](#_Toc199586669)

[2.5. Создание кода для базы данных 23](#_Toc199586670)

[ЗАКЛЮЧЕНИЕ 27](#_Toc199586671)

ВВЕДЕНИЕ

Ozon является одним из крупнейших маркетплейсов на территории СНГ, предлагающим широкий выбор товаров в различных категориях. Удобная система поиска и интуитивно понятный личный кабинет позволяют пользователям легко находить нужные товары и отслеживать свои покупки.

Актуальность: Ozon - крупнейший маркетплейс на территории СНГ. Отзывы являются неотъемлемой частью любого маркетплейса. Они помогают, более детально изучить товар и подтолкнуть к его покупке или избежать некачественного товара.

Семантический анализ текста — это процесс оценки содержательности текста и определения его значения. Этот метод анализа помогает понять, какие ключевые идеи и смыслы заложены в тексте.

Объектом исследования процесс веб-парсинга сайта Озон для сбора отзывов с использованием семантического анализа текста, а методом исследования -  веб-парсинг и семантический анализ.

Целью проекта является создание веб-парсера для сбора отзывов с платформы Ozon для анализа тональности.

Задачи:

* Подготовка
* Парсинг страниц с отзывами
* Семантический анализ текста
* Визуализация результатов

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

* 1. Терминология по проекту и глоссарий

Веб-парсинг (или парсинг данных) — это процесс автоматизированного извлечения информации с веб-сайтов. Этот метод позволяет собирать данные, которые обычно представлены в виде текста, изображений и других форматов, преобразуя их в структурированный вид для дальнейшего анализа или использования.

Для выполнения веб-парсинга используются специальные программы или скрипты, которые имитируют действия пользователя, загружают страницы сайта, анализируют HTML-код и извлекают нужные данные. Веб-парсинг может применяться для различных целей, таких как мониторинг цен, сбор новостей, создание баз данных и многое другое.

Для начала необходимо получить доступ к содержимому веб-страницы. Это делается с помощью HTTP-запросов. Популярные библиотеки для этого:

Python:

* Requests: самая популярная библиотека для отправки HTTP-запросов.
* Urllib: встроенная библиотека для работы с URL.
* Httpx: Современная асинхронная библиотека для работы с HTTP-запросами.
* Aiohttp: Асинхронная библиотека для работы с HTTP-запросами, которая использует asyncio.

Если же вы работаете над высоконагруженным проектом, где важна скорость и возможность параллельного выполнения запросов, стоит рассмотреть использование httpx или aiohttp.

Методы парсинга:

Ручной парсинг - это самый простой метод, когда вы вручную копируете данные с сайта. Подходит только для небольших объемов информации и не подходит для автоматизации.

Автоматический парсинг с использованием регулярных выражений

Регулярные выражения позволяют находить и извлекать текстовые шаблоны из HTML-кода страницы. Этот метод эффективен при работе с простыми структурами данных, но становится сложным при обработке сложных веб-страниц.

Парсинг с помощью CSS-селекторов

Многие библиотеки и фреймворки поддерживают работу с CSS-селекторами, что позволяет легко выбирать нужные элементы на странице. Например, document.querySelector в JavaScript или функции селекции в библиотеках вроде BeautifulSoup.

XPath

XPath – это язык запросов к XML-документам, который также используется для работы с HTML. С его помощью можно извлекать информацию из структурированных документов, включая сложные структуры страниц.

Если сайт предоставляет API, то использование этого метода будет наиболее предпочтительным, так как он обеспечивает доступ к данным напрямую без необходимости парсить HTML-код.

Терминология:

* Семантика: наука о значениях знаков и символов.
* Значимость: конкретные смыслы отдельных элементов текста.
* Контекст: окружающая ситуация, влияющая на понимание значения.
* Антецедент: исходный элемент, от которого зависит другое слово или выражение.
* Консистенция: согласованность смыслов внутри текста.
* Энтитизация: выделение сущностей (объектов, лиц, мест и др.).
* Инференция: вывод новых фактов из имеющихся данных.
* Кореференция: установление связи между разными частями текста, относящимися к одному объекту.
* Омонимия: наличие разных значений одного слова.
* Гипонимия/гиперонимия: отношения типа "род-вид" ("яблоко" гипоним "фрукта").
* Синтаксичность: взаимосвязь синтаксиса и семантики.
  1. Распределение ролей и работы

Проект был распределён среди участников команды следующим образом:

Левин Артем и Сафаргалеев Владимир занимались бэкэнд разработкой:

Архитектура:

* Модульный класс (запросы, парсинг, экспорт)
* Сессии с ротацией User-Agent и задержками

Парсинг:

* Поиск товаров: названия, цены, рейтинги, ссылки
* Детали товара: описание, характеристики, цены
* Обработка ошибок и fallback-значения

Экспорт:

* CSV (через Pandas) и JSON
* Автоназвания файлов

Интерфейс:

* Консольное меню (поиск/детали товара)
* Прогресс-парсинг страниц

Исаков Илья занимался созданием многофункционального Telegram-бота, призванного продемонстрировать практическое применение современных технологий программирования. Разработка осуществлялась на языке Python.

Бот реализует классическую парадигму командного взаимодействия:

1. Инициализация сеанса (/start)
2. Справочная система (/help)
3. Эхо-функция (/echo)
4. Работа с файловой системой (/readfile)

Яковлева Евгения и Огонерова Сардаана занималась созданием кода для базы данных. Код осуществлял подключение к СУБД Microsoft SQL Server Management Studio и создавал таблицу для хранения данных. Затем был разработан метод, который читает данные из .txt и .json файлов, разбирает строки и преобразует данные, после вставляет их в таблицу TestTable. После чего код сохраняет изменения и закрывает соединение.

Варламов Никита занимался адаптированием генеративно-промптовой модели YandexGPT для работы с Ozon:

* 1. Создал аккаунт в Yandex Cloud
  2. Создал API-ключ
  3. Создал метод для обращения к yandexgpt-lite
  4. Создал метод чтения txt файла с отзывами для товара и отправки промпта с файлом yandexgpt-lite
  5. Создал метод выдачи ответа от yandexgpt-lite в виде txt файла для дальнейшего использования.

1.3 Стек технологий

Основные технологии:

* Python - основной язык программирования
* Requests - библиотека для выполнения HTTP-запросов
* BeautifulSoup4 - библиотека для парсинга HTML/XML
* Pandas - библиотека для работы с данными и экспорта в CSV
* Fake-useragent - генерация случайных User-Agent строк

Вспомогательные библиотеки:

* Time - стандартная библиотека Python для работы со временем (используется для задержек между запросами)
* JSON - стандартная библиотека Python для работы с JSON форматом
* Random - может использоваться для генерации случайных задержек (хотя в текущей реализации не используется)
* python-telegram-bot==20.x - предоставляет удобный интерфейс для создания ботов в Telegram с использованием асинхронного программирования (asyncio).
* typing-extensions - предоставляет дополнительные возможности для работы с системой типов в Python, особенно для версий, где эти функции ещё не встроены в модуль typing из стандартной библиотеки.
* pyodbc - это инструмент для работы с базами данных через ODBC в Python. Она позволяет подключаться к различным СУБД и выполнять SQL-запросы напрямую из Python-кода.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Создание кода для парсинга

Этот код представляет собой функцию get\_product\_info, которая получает информацию о товаре с сайта Wildberries, обходя возможные блокировки. Создаётся URL для API Wildberries, который возвращает информацию о товаре urlencode кодирует параметры запроса: nm (артикул товара) и dest. Устанавливаются HTTP-заголовки, чтобы запрос выглядел как браузерный:

* User-Agent имитирует Chrome
* Accept указывает, что ожидается JSON
* Referer указывает страницу товара

В блоке try выполняется запрос к первому API. Если ответ успешный (статус 200), парсится JSON. Проверяется наличие данных о товаре (data['data']['products']). Если товар найден, возвращается первый из списка.

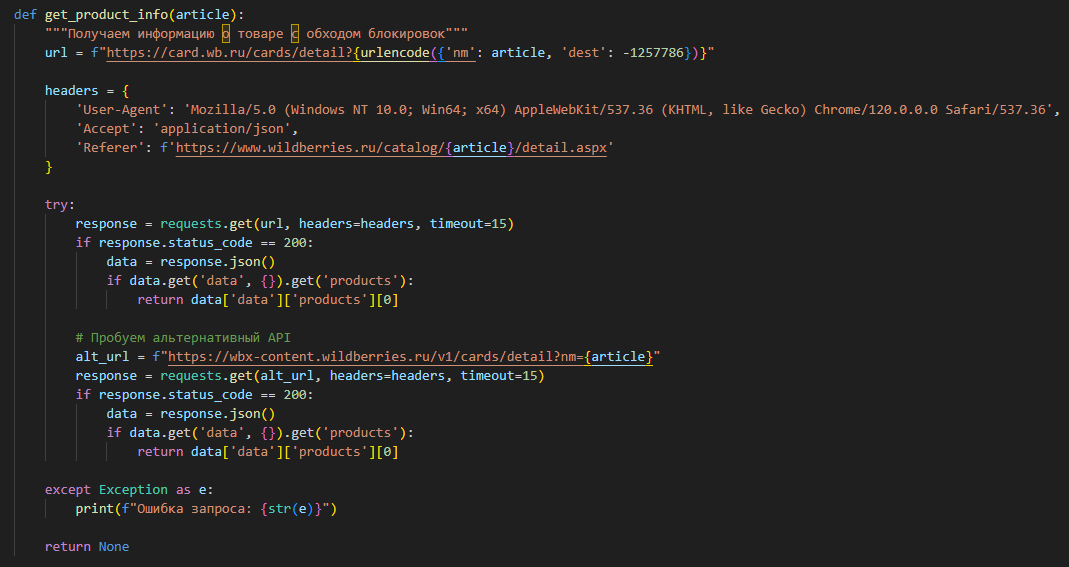


Рисунок 1 – Код функции get\_product\_info

Функция `get\_feedbacks` получает отзывы о товаре с Wildberries, обходя возможные блокировки через перебор нескольких резервных серверов. Она принимает два параметра: `product\_id` (артикул товара) и `limit` (максимальное количество отзывов, по умолчанию 10). Сначала функция вычисляет вспомогательные значения `vol` и `part` путём целочисленного деления артикула на 100000 и 1000 соответственно - это нужно для формирования правильного URL, так как Wildberries распределяет отзывы по разным серверам в зависимости от ID товара. Затем создаётся список из трёх возможных URL-адресов серверов с отзывами, использующих вычисленные `vol` и `part`. Функция последовательно перебирает эти серверы в цикле: для каждого отправляет GET-запрос с таймаутом 10 секунд, при успешном ответе (статус 200) извлекает из JSON-ответа список отзывов, обрезает его до указанного лимита и возвращает, если отзывы найдены. Если запрос к серверу fails (например, сервер недоступен или вернул ошибку), функция silently переходит к следующему серверу в списке благодаря обработке исключений. Если ни один из серверов не вернул valid данные, функция возвращает пустой список. Таким образом, реализован механизм отказоустойчивости через резервные серверы и обработку ошибок, что повышает надёжность получения отзывов даже при блокировках или проблемах с отдельными серверами Wildberries.

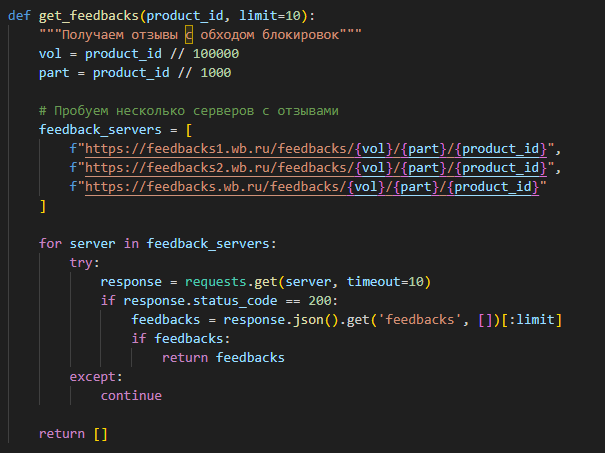


Рисунок 2 - get\_feedbacks

Функция save\_feedbacks сохраняет отзывы о товаре в текстовый файл reviews.txt в удобочитаемом формате. Она принимает два параметра: article (артикул товара) и feedbacks (список отзывов). Сначала функция открывает файл на запись с кодировкой UTF-8. Если список отзывов пуст, она записывает в файл сообщение об отсутствии отзывов для данного товара и завершает работу. Если отзывы есть, функция начинает запись с заголовка, указывающего артикул товара. Затем для каждого отзыва последовательно записывается следующая информация: рейтинг в виде звёздочек (сначала цифровая оценка, затем соответствующее количество символов "⭐"), дата создания отзыва, достоинства товара (плюсы), недостатки (минусы) и текст самого отзыва. Каждый отзыв отделяется от следующего горизонтальной линией из 50 дефисов для наглядности. В случае отсутствия каких-либо данных (например, если в отзыве не указаны плюсы или минусы) подставляются значения по умолчанию ("нет информации" или "без комментария"). Особенностью реализации является использование звёздочек для визуального отображения рейтинга: сначала выводится символ "★", а затем соответствующее оценке количество "⭐". В конце каждого отзыва добавляются два переноса строки для улучшения читаемости. Функция автоматически закрывает файл после завершения записи благодаря использованию конструкции with open(), что гарантирует сохранение данных даже в случае возникновения ошибок.

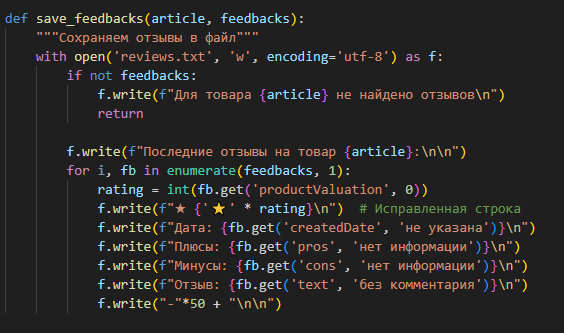


Рисунок 3 - save\_feedbacks

Функция main() является основной точкой входа в программу для сбора информации о товаре и отзывов с Wildberries. Программа начинается с запроса у пользователя артикула товара через стандартный ввод. После получения артикула выводится сообщение о начале процесса сбора данных с небольшой задержкой в 1 секунду для имитации работы. Затем вызывается функция get\_product\_info(), которая пытается получить данные о товаре по указанному артикулу. Если товар не найден или доступ ограничен, программа выводит сообщение об ошибке с возможными причинами (неверный артикул, необходимость использования VPN) и сохраняет соответствующее уведомление в файл reviews.txt, после чего завершает работу. При успешном получении информации о товаре программа выводит его название и переходит к сбору отзывов через функцию get\_feedbacks(), передавая идентификатор товара. Полученные отзывы сохраняются в файл с помощью функции save\_feedbacks(). По завершении процесса программа информирует пользователя о количестве сохраненных отзывов и пытается автоматически открыть файл с результатами через системный вызов os.startfile(). В случае невозможности автоматического открытия файла (например, на Linux-системах) выводится сообщение о расположении файла в текущей директории. Весь код заключен в условие if \_\_name\_\_ == "\_\_main\_\_":, что позволяет использовать этот файл как самостоятельную программу при непосредственном запуске, но не выполнять код при импорте модуля в другие проекты. Программа реализует простой и понятный интерфейс для конечного пользователя с визуализацией процесса через эмодзи и пошаговыми уведомлениями о статусе выполнения операций.

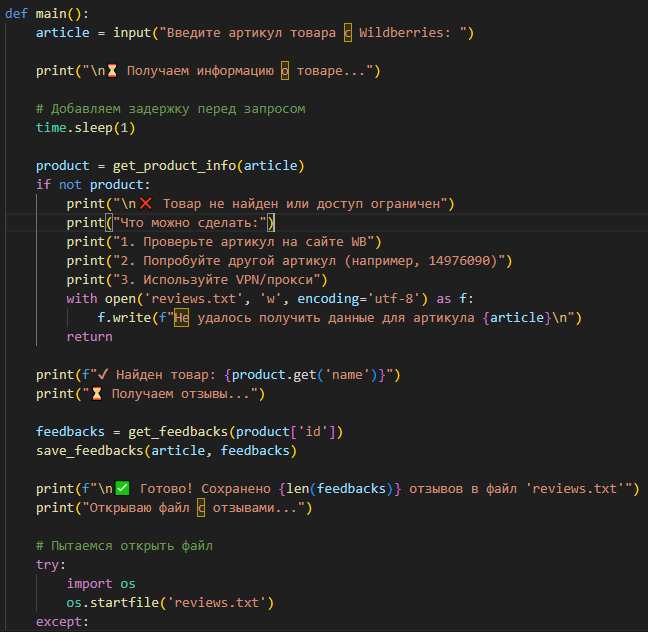
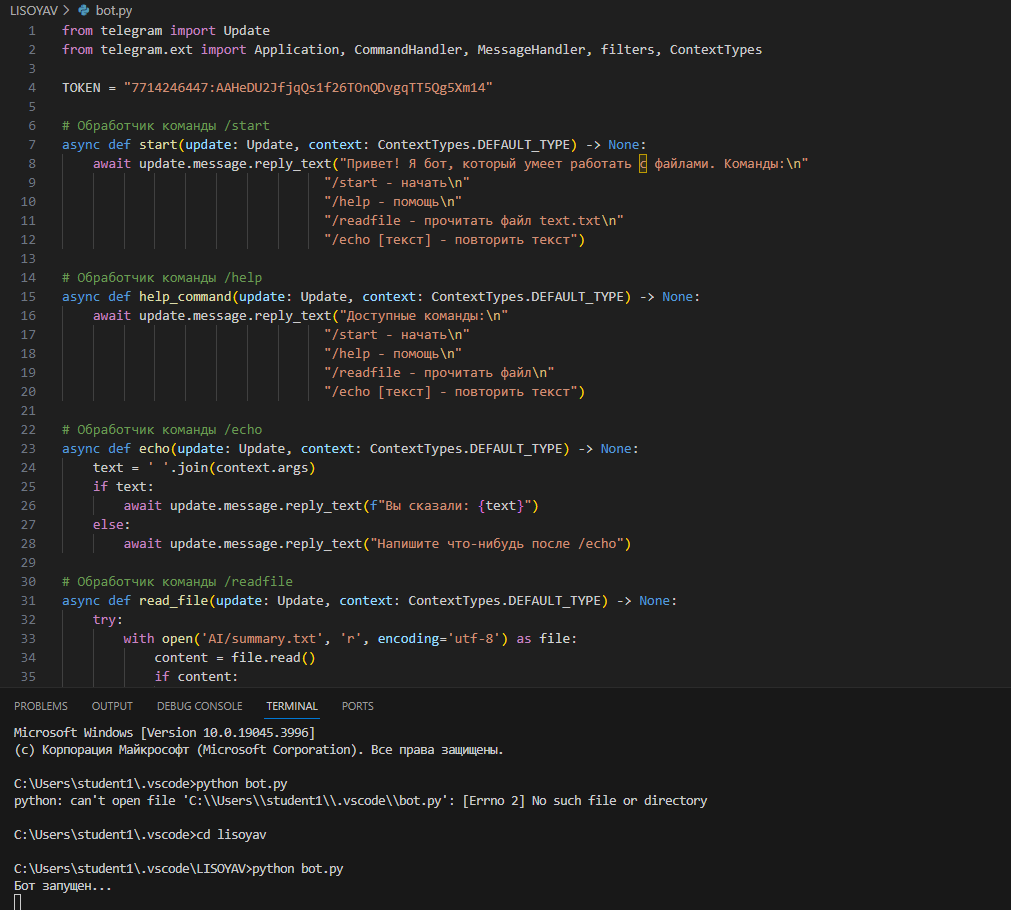
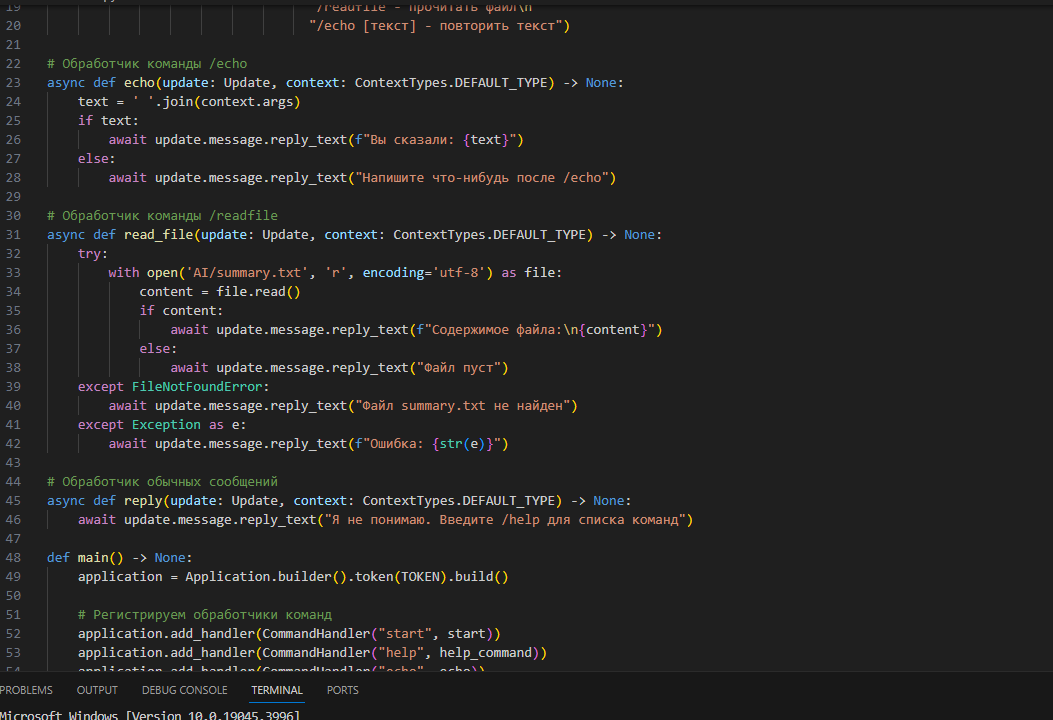
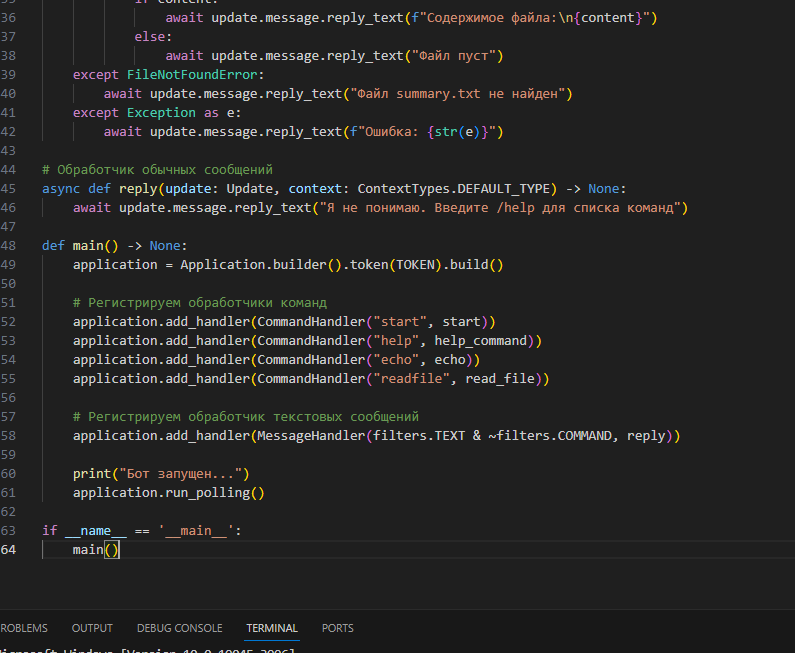


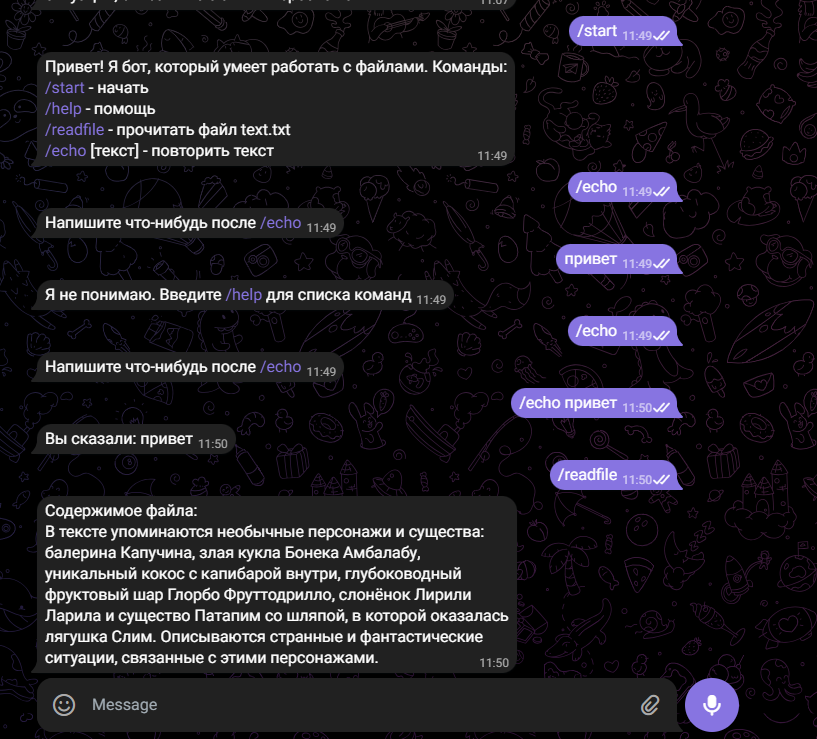
Рисунок 4 - main

## 2.2. Создание Telegram бота

  
Рисунок 5 Создание основного функционала бота

Рисунок 6 Создание основного функционала бота

  
Рисунок 7 Обработка ошибок

  
Рисунок 8 итог телеграмм бота

## 2.3. Создание аккаунта в Yandex Cloud

Перешёл на официальный сайт [Yandex Cloud](https://cloud.yandex.ru/).

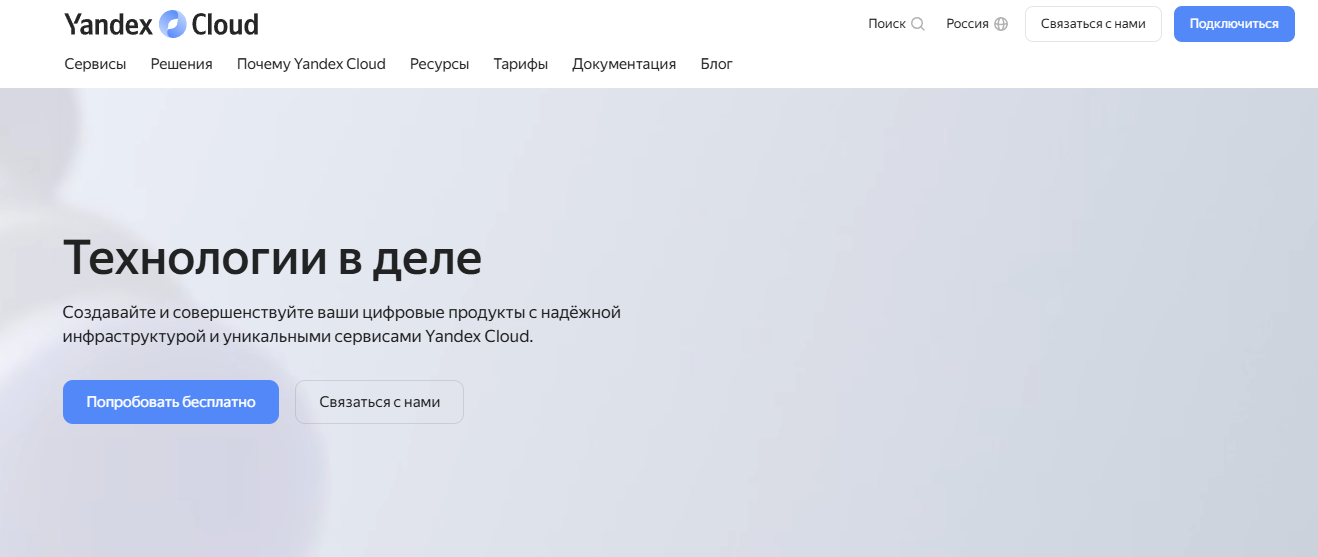


Рисунок 9 Официальный сайт Yandex Cloud

Зарегистрировал новый аккаунт, указав почту и номер телефона.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 10 Регистрация

Подтвердил личность и получил доступ к облачному сервису.

## 2.4. Создание API-ключа

В консоли Yandex Cloud перешёл в раздел **"Service accounts"**.

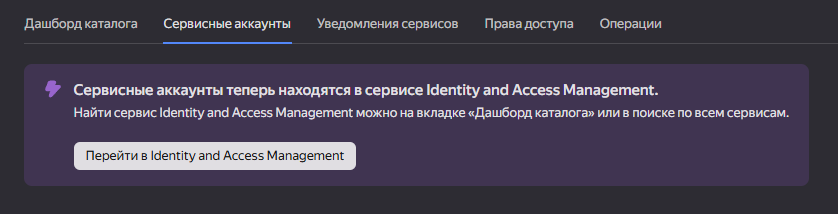


Рисунок 12 Раздел сервисные аккаунты

Создал новый сервисный аккаунт с правами доступа к **AI API**.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 13 Сервисный аккаунт

В настройках аккаунта сгенерировал **API-ключ** в формате JSON и сохранил его локально.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 Создание API ключа

Для работы с YandexGPT-Lite необходимо:

Получить **ID каталога** (Folder ID) из раздела настроек.



Рисунок 15 ID каталога

**Реализация Python-метода для отправки запроса к YandexGPT-Lite**

Создал скрипт на Python, который:

Инициализирует ключи и ID каталога

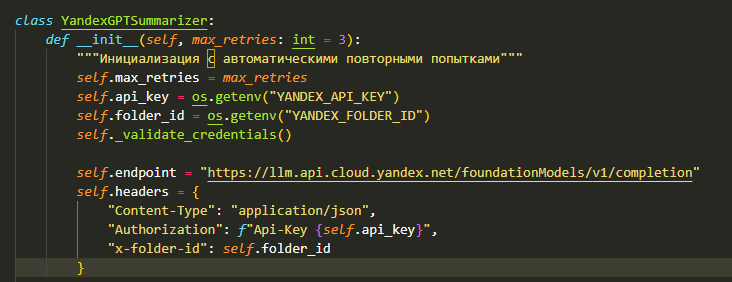


Рисунок 15 Инициализирует ключи и ID каталога

Проверяет данные

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 16 Проверяет данные

Отправляет запрос

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 17 Отправляет запрос

Формирует промпт для анализа.

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

Контент, сгенерированный ИИ, может содержать ошибки.

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 18 Формирует промпт для анализа.

Читает текстовый файл с отзывами и записывает выжимку в отдельный текстовый файл.

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 19 Текстовый файл примера

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 20 Чтение текстового файла

Изображение выглядит как снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 21 Итоговый текст

## 2.5. Создание кода для базы данных

Запуск скрипта вызывает функцию main(), который устанавливает соединение с SQL Server с помощью connect\_to\_sql(). Затем с ее помощью БД закрывается. В конце выводятся сообщения о статусе операций.

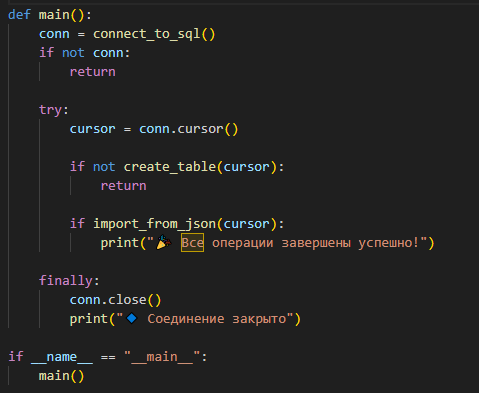


Рисунок 22 Установка соедиенение



Рисунок 23 Библиотеки и конфиграция

Функция ищет таблицу Products, если ее нет таблица создается с помошью функции create\_table. В конце выводятся сообщения о статусе операций.

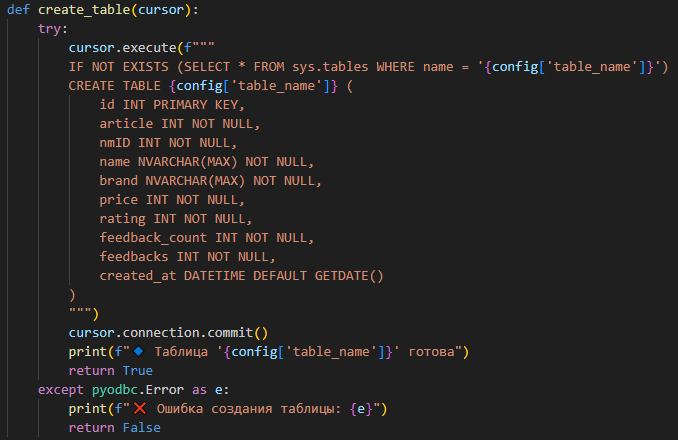


Рисунок 24 Create Table

С помщью import\_from\_json() читается JSON-файл. Его данные импортируются в созданную таблицу. В конце выводятся сообщения о статусе операций.

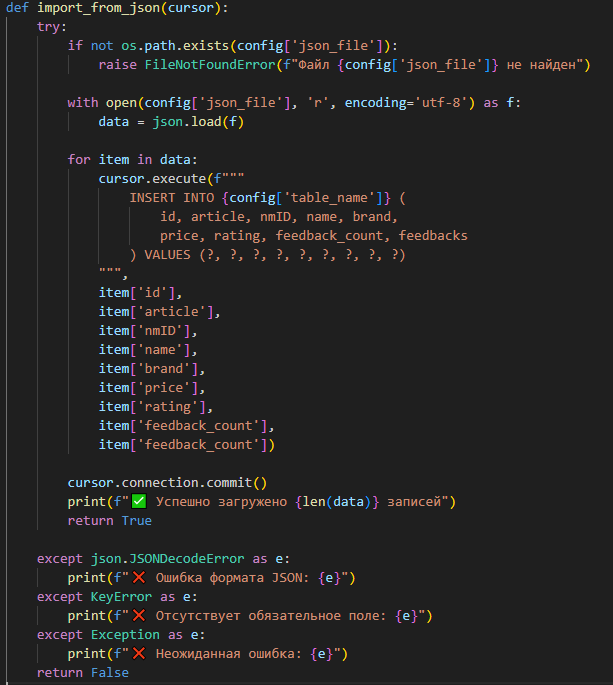


Рисунок 25 Добавление данных

ЗАКЛЮЧЕНИЕ

Использование результатов анализа отзывов открывает перед бизнесом широкие возможности для улучшения всех аспектов своей деятельности – от разработки новых продуктов до оптимизации внутренних процессов. Главное – внимательно слушать своих клиентов и своевременно реагировать на их обратную связь.