Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б07-мм

Реализация бота по парсингу файлов и дальнейшего получения данных через веб-сервис

Рыболовлев Алексей Александрович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
ст. преп. кафедры СП, И.В. Зеленчук

Санкт-Петербург
2024

Оглавление

Введение

В современном мире технологии играют значительную роль во всех сферах жизни. Автоматизация процессов существенно облегчает работу, экономит время и ресурсы. Примеры рутинных процессов, которые были автоматизированны, благодаря развитию программных продуктов.

- Бухгалтерия и финансовый учет: автоматизация бухгалтерского учёта с помощью программного обеспечения упростила процесс составления отчетов, обработки платежей и управления финансами для компаний.

- Управление запасами: автоматизированные системы управления запасами помогают компаниям оптимизировать хранение, инвентаризацию и отслеживание товаров, снижая потери и улучшая производительность.

- Управление проектами: программное обеспечение для управления проектами помогает командам организовывать задачи, отслеживать прогресс и контролировать сроки выполнения проектов.

- Управление человеческими ресурсами: автоматизация HR-процессов, таких как найм, управление персоналом и оплата труда, позволила компаниям лучше управлять своими сотрудниками и снижать затраты на эти процессы.

Именно поэтому разработка и внедрение информационных систем и программных продуктов является актуальной задачей. При этом удобство системы в первую очередь характеризуется удобством пользовательского интерфейса: чем он удобнее, тем лучше для конечного пользователя. Однако если целевыми пользователями конкретного продукта будут люди не особо компетентные в компьютерных технологиях, интерфейс должен стать ещё и интуитивно понятным и знакомым, не должен допускать возможности совершить ошибку.

Одним из таких процессов, который можно автоматизировать и улучшить с точки зрения пользовательского опыта, является создание

новых обработчиков файлов для конкретного автомобильного бренда, а также процесс добавления информации из новых файлов. Создание новых обработчиков файлов от каждого конкретного бренда лежит на разработчиках обслуживающей компании, а процесс добавления новых файлов не систематизирован и производится не централизованно.

1. Постановка задачи

Целью работы является автоматизация получения файлов, их обработка, централизованное хранение и предоставление информации с помощью API запросов. Для достижения данной цели были поставлены следующие задачи:

1. Выполнить обзор существующих решений.
2. Создать телеграмм бота, способного добавлять новые обработчики и получать файлы с данными.
3. Разработать функциональность по автоматическому определению бренда полученного файла, а также две базы данных: основную и служебную.
4. Создать универсальный обработчик файлов и собственное API для получения всей необходимой информации.
5. Протестировать полученное решение и провести его апробацию.

2. Background

Команда разработчиков, поддерживающая некоторое количество компаний, создаёт и настраивает базы данных под каждого клиента по отдельности. Компании специализируются на продаже автомобильных комплектующих и запчастей. В базах данных лежит информация о цене товаров, которая необходима для составления заказ-нарядов. Каждая компания работает с некоторым количеством брендов, которых всегда больше одного. Разные компании могут работать с одними и теми же брендами. Бренды передают компаниям цены на свою продукцию в специальных документах различных табличных форматов, но с заранее известной и строго соблюдаемой структурой. В них обязательно находятся оптовая и розничная цена товара, а также опционально может лежать некоторая дополнительная информация.

Одной из основных задач, которую необходимо решать ежедневно, является получение актуальной оптовой, розничной и рекомендованной розничной цены на конкретные комплектующие. Данная информация может постоянно меняться, поэтому скорость и удобство внесения изменений напрямую влияет на прибыль компаний. Чем больше времени будет потрачено на заполнение системой квитанции для конкретного потребителя, тем больше вероятность, что в будущем он предпочтёт обратиться в другую компанию.

Таким образом, необходимо создать продукт, который будет автоматизировать получение файлов с ценами на запасные части от различных автомобильных брендов, их обработку, загрузку в базу данных, централизованное хранение и предоставление информации с помощью API запросов.

3. Обзор

В данном разделе будет приведен обзор существующих аналогов и их недостатков.

3.1. Существующие аналоги

На данный момент существует два решения поставленной задачи.

- **Полуавтоматическое:** под каждый бренд программисты создают отдельный обработчик. Сотрудники компаний-клиентов кладут файлы с ценами в специальную папку соответствующего бренда, откуда с некоторой периодичностью происходит парсинг информации. При этом есть возможность производить загрузку частями.
- **Типовое решение, реализованное в конфигурации 1С Альфа Авто:** сотрудниками компаний-клиентов создаётся документ «изменение цен», настраиваются цены, которые необходимо загрузить. Далее в специальном окне необходимо выбрать настройки по загрузке и сам файл с ценами. При этом обработка запустится лишь в тот момент, когда информация потребуется для заполнения квитанций.

3.2. Недостатки аналогов

Для рассмотренных аналогов можно выделить следующие недостатки и их причины:

1. Из-за того что у каждой компании своя база данных происходит многократное повторение информации.
2. Создание новых или редактирование существующих обработчиков является ответственностью программистов, так как у клиентов нет необходимого для этого интерфейса.
3. Высокая сложность поддержания актуальности всех обработчиков. С каждым днём увеличивается количество брендов в том числе,

благодаря появлению большого количества китайских автопроизводителей, для каждого нужен новый обработчик. По мимо этого также возможно и изменение структуры обрабатываемых файлов от брендов, с которыми уже была налажена работа. Об этом отдел разработки узнаёт в последнюю очередь.

4. Сотрудники компаний тратят большое количество времени на добавление нового прайса, так как перед этим необходимо выполнить множество действий.

5. Обновление информации в базе данных возможно лишь с рабочего ПК. Решить данную проблему настройкой удалённого доступа невозможно из-за того, что есть разные политики фирм - у некоторых запрещён удалённый доступ.

6. Обработка прайсов по времени может занимать вплоть до 30 минут, так как целиком обрабатываются файлы размером до 100000 строк, но при этом из них может пригодиться не более 5-10.

7. Решение, поставляемое в конфигурации 1С Альфа Авто, позволяет загружать информацию только из .xls(x) - файлов. Этого недостаточно, так как некоторые бренды передают информацию в Access докуметах. В рамках 1С не предоставляется возможным конвертация Access в Excel, установка какого-то дополнительного ПО для решения данной проблемы может быть запрещена из-за политики компаний.

4. Требования к продукту

В результате обзора были выделены следующие требования к создаваемому продукту.

- **В базах данных должны быть актуальные сведения.** Чтобы не тратить время на обработку файлов с ценами, когда информация нужна прямо сейчас.

- **Одни и те же данные не должны повторяться по нескольку раз.** При загрузке файлов с ценами одной компанией, у остальных, работающих с этим же брендом, должна появиться возможность сразу воспользоваться этой информацией, а не загружать в свою базу тот же самый файл с ценами.

- **Необходимо дать ответственному за изменение цен человеку возможность выполнить обновление не только с рабочего места.** В конкретной компании вносить изменения в цены может очень небольшое количество сотрудников. Если быстрее всех может обновить данные человек, находящийся не на рабочем месте, то необходимо предоставить ему такую возможность.

- **Уменьшение временных затрат на обработку информации.** Уже готовые решения обрабатывают полученный массив данных целиком. Без возможности разбиения его на части или оптимизации этого процесса. Например, поступил файл, в котором 100000 позиций, из них нужна информация только по пяти. Сразу после нахождения указанных позиций можно вернуть необходимые данные пользователю, затем продолжить обработку остальной части файла.

- **Автоматизированная загрузка цен по новым товарным позициям.** Если в базу конкретной компании загрузить весь перечень товаров и каждый раз обновлять все изменения цен, то в базе получится много избыточной информации.

5. Используемые технологии

5.1. Язык программирования

В качестве языка программирования для данного проекта был выбран язык Python. Так как для него доступно огромное количество библиотек, которые могут быть использованы для решения различных задач. Это позволяет разработчикам сосредоточиться на своей основной задаче, не тратя время на написание базовых функций. Python используется в различных областях, таких как веб-разработка, анализ данных, автоматизация, научные вычисления и многое другое. В проекте планируется создание telegram-bot, работа с файлами различных типов и базами данных, а также создание API. На каждую из данных задач в Python существует множество библиотек с открытым исходным, поэтому он отлично подходит для реализации данного проекта и в будущем не будет проблем с лицензией конечного продукта.

5.2. Интерфейс

В качестве интерфейса было решено использовать telegram-bota. Telegram — основное средство связи внутри всех компаний, для которых создаётся данный продукт. Для сотрудников он знаком и удобен, а также установлен на каждом рабочем ПК, поэтому не придётся устанавливать дополнительное программное обеспечение. Также telegram-bot позволяет добавлять файлы с ценами не только лишь со своего рабочего места.

Для написания бота изначально из-за своей простоты и лёгкости в изучении была выбрана библиотека «pyTelegramBotApi». Однако в ней отсутствует поддержка асинхронности. А также стремительно развивающийся проект было бы труднее поддерживать на основе данной библиотеки. Поэтому она была заменена на библиотеку «AIOgram», которая поставляется с возможностью сразу писать асинхронный код, а также обладает в разы большим сообществом и более информативной документацией. Лицензия библиотеки — Apache 2.0[6].

5.3. Обработка данных

Для получения информации из поступающих файлов была выбрана библиотека «pandas». Это одна из самых популярных библиотек для обработки и анализа данных в Python. Она предоставляет набор инструментов для анализа и обработки данных. А также включает в себя методы для чтения из различных источников (excel-файлы, access-файлы), после чего производятся действия с одним классом «DataFrame». Такой подход исключает дублирования кода, так как работа с разными файлами отличается только в их чтении. Так же «pandas» оптимизирована для быстрого выполнения и использует векторизованные операции для увеличения производительности, особенно при работе с большими наборами данных. Лицензия библиотеки — BSD[7].

5.4. Сохранение данных

В проекте будет создано сразу две базы данных: вспомогательная и основная. Количество записей во вспомогательной базе данных соизмеримо с количеством брендов, поэтому для неё используется СУБД SQLite и одна из самых легких, простых и интуитивно понятных в использовании ORM библиотек — «peewee». На этапе создания прототипа для основной базы данных используется тот же набор инструментов, однако в будущем планируется смена СУБД на PostgreSQL и использование более мощной библиотеки, например «SQLAlchemy», из-за огромного массива данных и необходимости быстро с ним работать. Лицензия библиотек «peewee» и «SQLAlchemy» — MIT[8].

5.5. Получение данных

Получение данных реализовано посредством обращения к API, написанного на расширении «flask-restfull», одного из самых популярных фреймворков — «Flask», по мнению некоторых авторитетных изданий. Например, здесь[2] и здесь[5]. Лицензия фреймворка — BSD[7].

6. Реализация

В данном разделе будут представлены конкретные детали реализации и архитектура решения. Для удобства можно рассмотреть путь одного файла с информацией о ценах товаров, которую необходимо добавить в базу данных. Всё начинается с отправки сотрудником компании-клиента файла в telegram-bot.

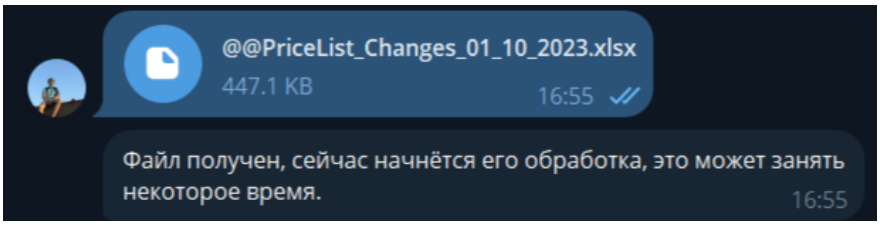

Рис. 1: Файл скачивается

6.1. Работа с Telegram API

При получении файла срабатывает обработчик, настроенный на получение документов с помощью специального декоратора, в функцию поступает информация необходимая для скачивания файла с серверов telegram. После этого файл загружается в pandas.DataFrame на основе его расширения.

6.2. Определение бренда полученного файла

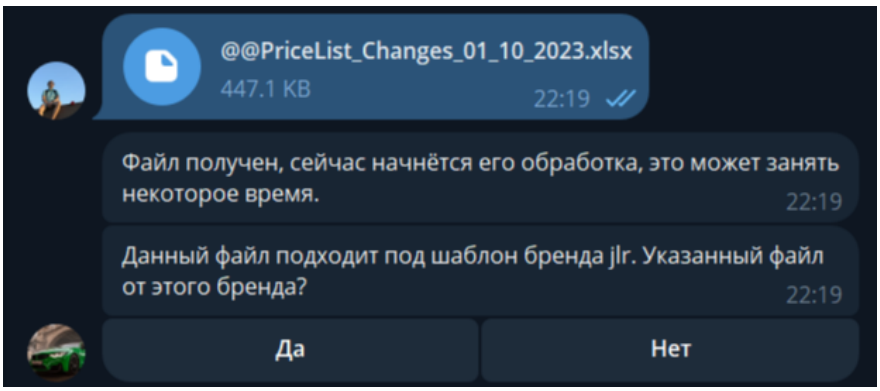

Рис. 2: Бренд полученного файла определён автоматически

Дата фрейм[1] передаётся в функцию check_current_document, которая проверяет его на соответствие одному из уже созданных обработчиков, если такой обработчик существует, то отправляется в телеграмм сообщение с подтверждением.

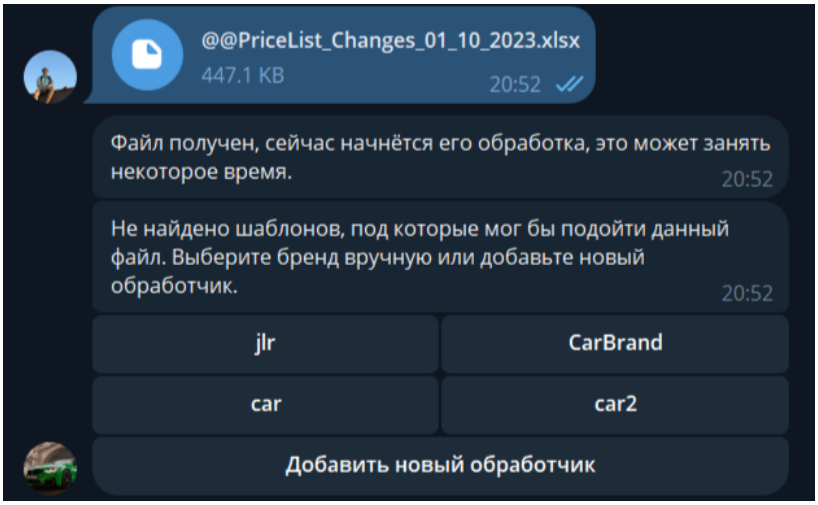

Рис. 3: Можно выбрать из уже созданных брендов или добавить новый

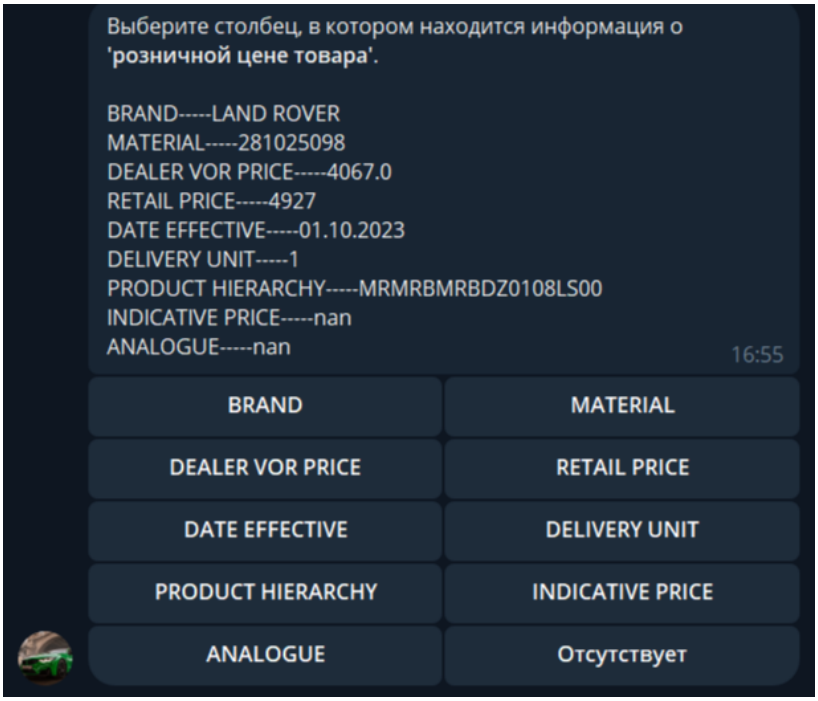

Рис. 4: Процесс добавления нового бренда

Если бренд файла не определён автоматически, то пользователю

[1]Двумерная структура данных, которая состоит из строк и столбцов и используется для хранения и организации табличных данных.

предоставляется возможность выбрать бренд вручную из уже сохранённых или создать новый.

Для добавления нового бренда пользователю отправляются названия столбцов и первая строка с данными из файла. Затем несколько сообщений с вопросами — какая информация находится в каком столбце, ответ производится нажатием на кнопки представляющие собой список названий столбцов. Один и тот же столбец не может быть ответом на вопросы бота более одного раза, поэтому выбранный столбец удаляется из списка возможных ответов.

Также есть возможность добавления нового обработчика без отправки в бота файла с ценами от этого бренда. Для этого в меню необходимо выбрать команду /add_new_brand, а затем пройти опрос, последовательно отправляемый ботом: бот присылает название столбца в базе данных, а пользователю необходимо отправить номер этого столбца в файле с ценами. Этот способ может показаться не самым удобным, но для ситуаций, когда нет необходимости или возможности отправить файл с ценами и на его основе создать новый обработчик, этот вариант является очень удобным.

6.3. Загрузка в базу данных

После определения бренда полученного файла или создания нового обработчика файл передаётся в функцию, которая на основе информации из вспомогательной базы данных берёт информацию только их тех столбцов, которые необходимы. А также предусмотрена перед загрузкой проверка на корректность полученных данных. Каждое сохранение в основную базу данных влечёт за собой соответствующую запись в csv-файл на случай появления некорректной информации. На основе истории из этого файла можно будет разобрать ошибочную ситуацию и не допустить её повторения в будущем.

Структура основной базы данных и вспомогательной полностью совпадают, за исключением поля upload_date (во вспомогательной его нет). В основную базу данных соответственно заполняется с помощью

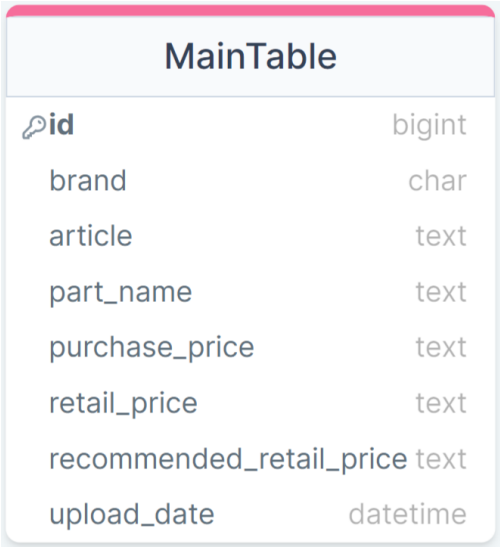

Рис. 5: Структура основной базы данных

обработчика информация из загружаемых файлов с ценами. Во вспомогательной базе, начиная с поля «article», находится соответствующее название столбца в поступаемых файлах у данного бренда.

6.4. Возникшие проблемы

В ходе реализации данного проекта, было встречено ряд проблем. Далее представлено их описание и инструкции по их решению.

- Проблема по работе сразу с несколькими пользователями, это оказалось невозможно из-за отсутствия поддержки асинхронности библиотекой «pyTelegramBotApi». Решение — переписать целиком работу с telegram на другую библиотеку. Сделать это было, не трудно, так как с самого начала я придерживался принципов проектирования SOLID: логика приложения отдельно, интерфейс пользователя отдельно. Наибольшую помощь в решении данной проблемы внесла официальная документация[4] данной библиотеки.

- Получение файлов с серверов telegram неожиданно вызвало ряд проблем: функции из библиотеки pandas не хотели обрабатывать файлы записанные в бинарном виде (только так их можно получить в своё распоряжение после отправки пользователем). Со-

здание временного файла из бинарного также не помогло, а вот загрузка напрямую в метод pandas.read_<тип файла> неожиданно сработала. Тип, размер и id файла можно получить после отправки пользователем, так что такое решение имеет право на использование в проекте. Для решения данной проблемы использовалась документация бибилотеки для работы с файлами при обращении к telegram API[1], чтобы понять какой объёкт можно получить и информацию о нём.

- Telegram не разрешает ботам скачивать файлы размером больше 50Мб, у пользователей же установлено ограничение в 1,5Гб. К счастью, на просторах интернета уже существуют решения данной проблемы[3].

7. Тестирование и апробация

Процесс тестирования и апробации был разделён на несколько основных частей:

1. **Тестирование всех частей по отдельности.** Тестирование всевозможных сценариев работы с telegram-ботом, проверка корректности получения информации из файлов с ценами. Тестирование загрузки в базу данных и получение из неё необходимых данных.

2. **Общее тестирование.** Проверка функционирования всего продукта, посредством отправки всевозможных файлов и получения из них информации. Также была проведена проверка безопасности системы от угроз внутренних — ошибка сотрудников при добавлении нового файла или создании нового обработчика, и внешних — несанкционированная загрузка посторонним пользователем информации в базу данных и получение из неё.

3. **Апробация.** Итоговая проверка продукта производилась вместе с сотрудниками компании-заказчицы. По итогам данной встречи было утверждено, что приложение выполняет основные поставленные задачи. Также значительно упрощает процесс добавления документов с ценами и новых обработчиков. Централизованная база данных позволяет сделать более удобной работу команды разработчиков. Раньше все данные о ценах хранились у компаний-клиентов, из-за передачи этой ответственности к команде разработчиков (ввиду централизации хранилища) в будущем возможно рассмотреть способы монетизирования получения необходимых данных для компаний, как отдельной услуги. Простота внедрения была одним из требований, поэтому сотрудники компаний клиентов уже получили возможность протестировать функциональность и удобство telegram-бота, который был запущен на собственном сервере команды разработчиков. Но есть ряд нюансов, которые необходимо доработать. К недочётам, которые удалось исправить в кратчайшие сроки можно отнести: несоответствие API изменившимся в

ходе работы требованиям (для решения была изменена структура запросов; удалены не нужные варианты запросов, например получение всех товаров, у которых цена находится в указанном диапазоне; добавлены новые варианты запросов, например получение товара по артиклю и бренду), отсутствие пользы от варианта создания нового обработчика без отправленного файла (такая возможность была удалена и не повлекла за собой никаких изменений в программном продукте). А также была проблема, решение которой планируется реализовать в следующем учебном семестре — скорость добавления новых записей в базу данных, по этому вопросу уже намечен план действий, но требуются более обширные знания в этой области.

Заключение

В ходе выполнения работы были получены следующие результаты:

1. Произведён обзор существующих решений.
2. Создан телеграмм бот автоматизирующий процесс добавления нового обработчика и способный получать файлы с данными для дальнейшей обработки.
3. Спроектированы необходимые базы данных и разработано автоматическое определение к какому бренду относится файл, а также универсальный обработчик, добавляющий все данные в централизованное хранилище.
4. Реализовано API для получения данных из БД.

Ссылка на репозиторий с проектом:
https://github.com/rybolovlevalexey/ParsingBotForSQL

Список литературы

[1] File. — 2023. — URL: https://docs.python-telegram-bot.org/en/stable/telegram.file.html.

[2] Khatri Vijay Singh. The Best Python Frameworks in 2023. — 2023. — URL: https://hackr.io/blog/python-frameworks.

[3] Telegram боты. Загружаем файлы больше 50мб. — 2018. — URL: https://habr.com/ru/articles/348234/.

[4] aiogram Documentation. — 2023. — URL: https://aiogram.readthedocs.io/_/downloads/en/latest/pdf/.

[5] Бартенев Евгений. 10 лучших фреймворков для веб-разработки на Python. — 2022. — URL: https://tproger.ru/articles/10-luchshih-frejmvorkov-dlja-veb-razrabotki-na-python.

[6] Лицензия Apache. — 2023. — URL: https://ru.wikipedia.org/wiki/Лицензия_Apache.

[7] Лицензия BSD. — 2023. — URL: https://ru.wikipedia.org/wiki/Лицензия_BSD.

[8] Лицензия MIT. — 2023. — URL: https://ru.wikipedia.org/wiki/Лицензия_MIT.