Нужно создать Веб приложение парсер на Python.

После перехода на адрес, например www.my-parser8674.com мы авторизируемся и можем начинать управлять профилями парсера.

Начальная область работы - это список профилей всех парсеров. Можно создать новый профиль, удалить профиль, редактировать профиль, запустить сбор данных.

При редактировании профиля нам отображается карточка выбранного профиля, лог всех прошедших ранее операций и результаты сбора данных в выбранном профиле.

В карточке профиля у нас есть возможность настроить условия сбора данных, а именно:

пример алгоритма создания профиля:

оператор указывает адрес, который его интересует, подтверждает и в этом же окне открывается встроенный в парсер браузер с анализатором всех DOM узлов страницы. Браузер визуально должен отображать 1-в-1 то, что отображается в обычном браузере. Дальше клиент добавляет новую сущность в справочник профиля: например, создает свойство с кодом "Price", название "Цена", в браузере визуально кликает на цене, ему отображаются доп настройки парсинга этого поля - тип (число, текст, html, картинка, ссылка, массив), удалять слова (возможно удалять перечень слов), условия (если равно/не равно/содержит/не содержит тогда ставить какое-то свое значение, например "-1"). Для примера рассмотрим еще такой тип как "массив". После выбора данного типа у нас появляется два доп. поля настройки элементов: название и значение - для них мы аналогично мышкой в браузере указываем все точно так же. В результате, основная настройка будет читать таблицу или список как массив, а доп настройки будут разбивать ее на массив с набором нужных значений.

Оператор должен иметь возможно вручную задавать УРЛЫ для сбора данных, загружать XML файл с урлами, или указать ссылку на файл sitemap.xml, научить программу самостоятельно определять тип страницы и собирать данные со всех аналогичных типов страниц на сайте, это больше актуально для многоуровневой настройки, описано ниже.

В программе должна быть возможность созданий многоуровневого сбора данных, пример:

оператор задает ссылку на главную страницу сайта донора, ему должен открыться виртуальный браузер и в какой-то части отобразятся стандартные данные страницы, например тег h1, все значения тегов meta и другие информативные теги которые визуально нельзя увидеть в браузере.

оператор выбирает дерево категорий, настраивает сбор как массив - название, ссылка. дальше в этот же профиль оператор добавляет еще один уровень - для настройки и сбора данных по этим же ссылкам (категориям), переходит в категорию аналогично категории настраивает сбор товаров (название, ссылка) - нужно обязательно учесть пагинацию, или оставить автоматическое определение или дать оператору указать где искать ссылки, добавляет еще один уровень вложенности и настраивает сбора данных по карточке товара.

В итоге мы должны получить примерно такие свойства: метатеги главной страницы, список категорий, список ссылок категорий, список товаров, список свойств товаров.

На следующем этапе мы настраиваем как сохранять данные. Нам должны быть доступны все свойства профиля (включая все уровни) и их значения (со страниц, которые мы просматривали для настройки, имеется ввиду предпросмотр готовых результатов), оператор указывает тип выгрузки (достаточно CSV, XLSX или XML с возможность указать названия тегов). Например, мы хотим собрать дерево категорий и товары, оператор должен выбрать вариант XML, выбрать первый уровень вложенности профиля парсера, выбрать свойства, прописать им название для файла и указать какое свойство выбирать, дальше второй уровень вложенности - товары, должна быть возможность выгружать или отдельным тегом XML, или вложенным в тег из первого уровня вложенности.

Примеры:

1й уровеь:

<cats>

<cat id="{id}">

<id>

{id}

</id>

<name>

{name}

</name>

<link>

{name}

</link>

</cat>

</cats>

id - запись с БД или же определять со ссылки на категорию - по типу число.

2й уровень, после настроек 2го уровня наша выгрузка может выглядеть как один из вариантов:

<cats>

<cat id="{id}">

<id>

{id}

</id>

<name>

{name}

</name>

<link>

{name}

</link>

<prods>

<prod>

<id>

{id}

</id>

<name>

{name}

</name>

<link>

{name}

</link>

</prod>

</prods>

</cat>

</cats>

Или

<cats>

<cat id="{id}">

<id>

{id}

</id>

<name>

{name}

</name>

<link>

{name}

</link>

</cat>

</cats>

<prods>

<prod id="{id}">

<id>

{id}

</id>

<catid>

{catid}

</catid>

<catname>

{catname}

</catname>

<name>

{name}

</name>

<link>

{name}

</link>

</prod>

</prods>

Вторая часть рабочей области это логи.

Журнал логов по каждому профилю должен содержать список все запусков, даты, к-во собранных записей, ошибки, которые возникли в процессе сбора. Например, если сейчас идет сбор данных в логах должно быть написано: запуск {дата}, вложенность 1, сбор данных по ссылке {ссылка}, и так по каждой логической итерации данного профиля, в логах должен быть перечень абсолютно каждого выполненного этапа работы парсера с детальным перечнем, что конкретно было сделано.

Система сбора данных должна работать в фоновом режиме без привязки к сессии оператора. Должна быть возможность настраивать прокси, к-во потоков, максимальное к-во переходов по УРЛ (на случай цикличности во вложениях или огромного к-ва товаров), прокси должны тестироваться и если какие-то выпали - отключать и равномерно уменьшать к-во потоков. Например, мы настроили 1 поток = 4 прокси. Если у нас 20 прокси - 5 потоков, когда выпадет 1-4 прокси, станет 4 потока.

Базовые требования к функционалу:

- создание, настройка профилей (описано выше);

- просмотр, скачивание, удаление логов;

- сохранение, отправка на почту результатов сбора профиля;

- настройка количества потоков для каждого профиля;

- настройка количества прокси на каждый поток для каждого профиля;

- настройка автоматической остановки сбора данных по профилю если найдена ошибка (системная или разбора страницы) - с опциями: пропускать ошибку / останавливать сбор;

- автоматическое тестирование прокси перед каждым запуском и во время сбора данных, при обнаружении ошибки отключать прокси на данном запуске и уменьшать потоки учитывая настройку к-во прокси на поток;

- ручной запуск сбора данных по профилю;

- автоматический запуск сбора данных по профилю по расписанию (календарь, время / каждые n секунд) ;

- блокировка повторного запуска по каждому профилю;

- ручная остановка сбора данных по каждому профилю;

- отправлять отчеты (логи результатов сбора) на почту каждые n секунд;

- отправлять ошибки (во время сбора данных) на почту;

- возможность копирования настроек с одного профиля в другой;

- возможность копирования настроек с одного уровня профиля в другой в одном профиле и между профилями;

- требования для разных типов данных:

при выборе любого типа данных должны быть доступны следующие галочки - удалять лишние пробелы (табы, переходы), удалить html атрибути (с доп полем для исключений), удалить html теги (с доп полем для исключений) ;

число - преобразовать в тип int или float, округлять - в большую, меньшую сторону, по основным правилам.

текст - максимальный размер для блока

html - максимальный размер для блока, если для поля не выбрана опция "удалить html теги" парсер должен искать картинки в этом блоке и сохранять их на сервер для дальнейшего скачивания.

картинка - сохранять ссылку на картинку, сохранять картинку на сервер (для каждого профиля должна быть отдельная директория на сервере с вложенными директориями картинок - не сплошным списком картинок). Если выбрана опция "сохранять картинку на сервер" в карточке профиля должны появится кнопка сохранить, которая скачивает архив с картинками

ссылка - относительная / абсолютная

массив - пример настроек описан выше в ТЗ