

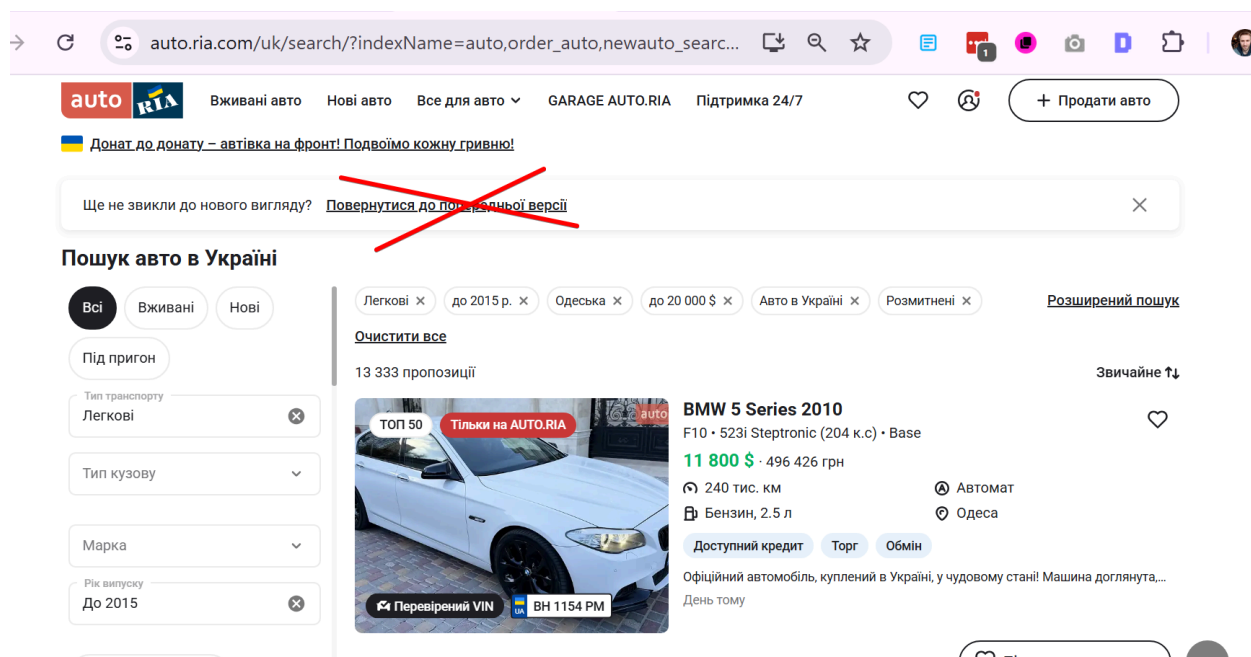
Реализуем парсер Autoria в виде Python скрипта

Входные данные

В качестве входных данных (файл input.txt лежит в корневой директории скрипта) для парсера вы подаете ссылку на выдачу сайта авториа. Например:

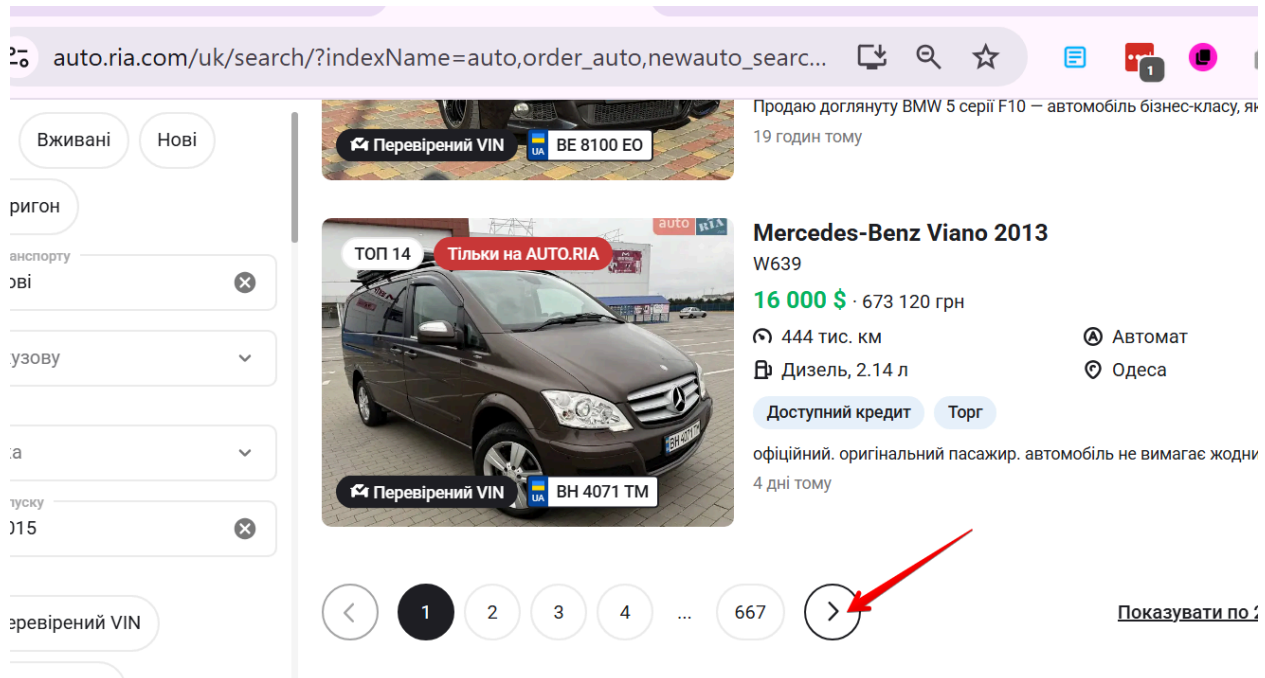
[https://auto.ria.com/uk/search/?search_type=1&category=1&all\[0\].any\[0\].state=2&all\[0\].any\[1\].state=8&all\[0\].any\[2\].state=6&all\[0\].any\[3\].state=1&all\[0\].any\[4\].state=16&all\[0\].any\[5\].state=20&all\[0\].any\[6\].state=24&price\[0\]=1&price\[1\]=3000&abroad=0&customs_cleared=1&order=7&republicshed_last=4](https://auto.ria.com/uk/search/?search_type=1&category=1&all[0].any[0].state=2&all[0].any[1].state=8&all[0].any[2].state=6&all[0].any[3].state=1&all[0].any[4].state=16&all[0].any[5].state=20&all[0].any[6].state=24&price[0]=1&price[1]=3000&abroad=0&customs_cleared=1&order=7&republicshed_last=4)

Важный момент - работаем именно с текущей (новой) версией сайта



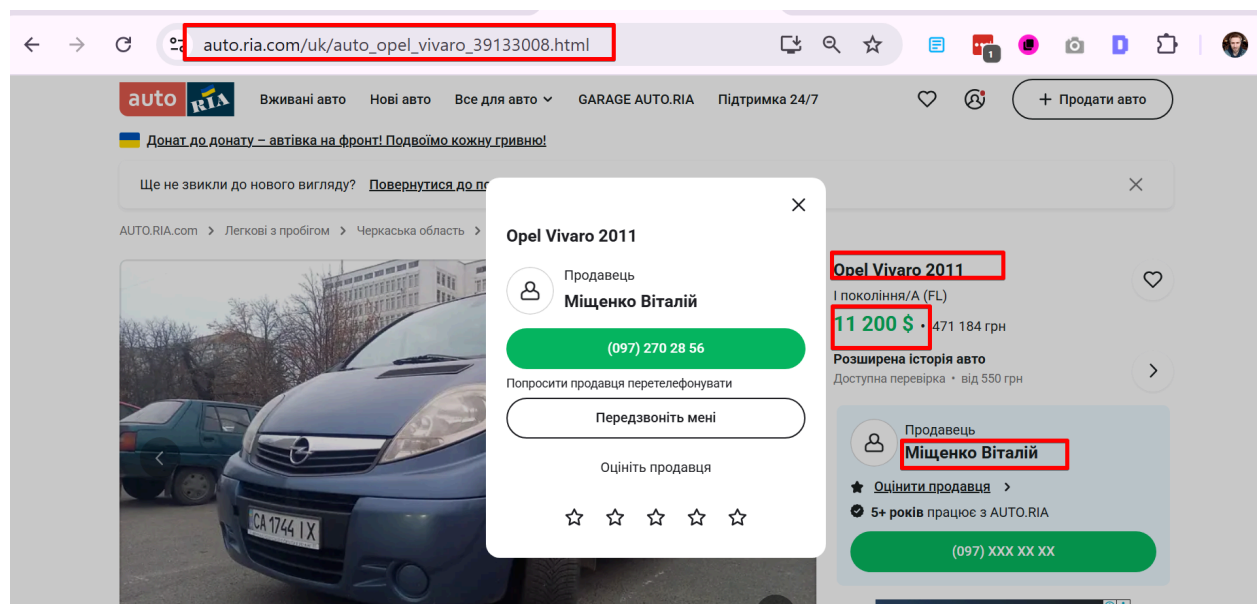
Сбор ссылок на товары со страницы поиска

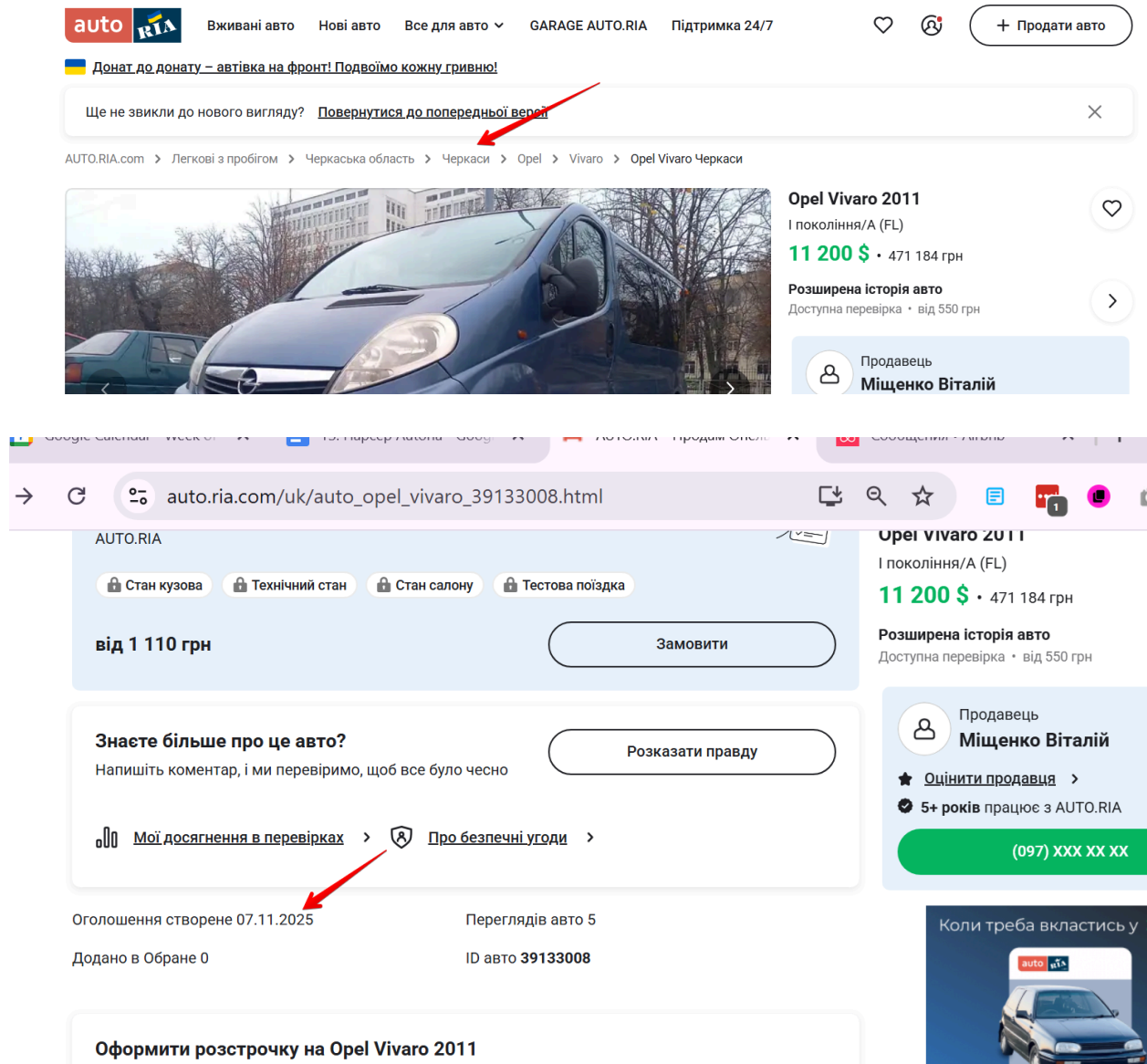
Парсер собирает ссылки на результаты (отдельные объявления) с первой страницы выдачи (каталога) объявлений, далее переходит на следующую страницу и так до конца. Ранее было замечено, что при загрузке страницы не всегда сразу прогружается блок с пагинацией. Соответственно, нужно дождаться его загрузки, чтобы не утерять данные, которые доступны на следующих страницах



Сбор данных о каждом товаре

Сбор данных происходит со страницы объявления. С каждого объявления выгружаем: заголовок объявления, телефон (если доступен), имя (если доступно), город, дату публикации, цену, ссылку на объявление. Телефон получаем кликом по плашке с телефоном.





Важный момент - перечень полей данных не фиксируем жестко в коде.

Отсеивание дублей по номерам телефона

При парсинге отсеиваем объявления, которые содержат номера, которые уже ранее попадались в процессе текущего парсинга.

Загрузка Веб-страниц

Для загрузки страниц будет использован браузерный компонент на основе Playwright. Основная цель решения - осуществлять загрузку вебстраниц максимально приближенно, как это делают браузеры разных пользователей.

После загрузки должен произойти клик по плашке с телефоном и нужно дождаться загрузки телефона. Если телефон не загрузился, то выдаем ошибку и добавляем текущий элемент в конец очереди на обработку (загрузится позже). Один элемент (страница объявления или каталога) повторно обрабатываем максимум N раз (параметры конфигурации `errorRetryTimes`). Ключевые особенности:

- **Поддержка прокси.** Решение будет поддерживать загрузку через прокси. Прокси для минимизации блокировок будут находиться в постоянной ротации.
- **Имитация загрузки с разных устройств:** Использование системы изменения отпечатка браузера для имитации естественной активности пользователя, что значительно снижает риск блокировки со стороны веб-сайтов. Проще говоря - каждая прокся будет иметь собственный кеш браузера, куки, разрешение экрана, UserAgent и т.п. Соответственно, папка с кешем каждого браузера сохраняется отдельно.
- **Кэширование загрузки страниц объявлений (но не страниц каталога):** (важно - не путаем с кешем браузера!) Решение включает встроенное кэширование загрузки объявлений в папку, сохраняя каждую URL-страницу в отдельный локальный файл. При первом посещении веб-страницы она загружается с сервера и кэшируется. Важный момент - со страницей также кешируем телефон (который получается позже при клике на плашке с телефоном). Соответственно, учесть это при сборе данных, что телефон может загружаться как с обычной, так и с кешированной версии. При последующих посещениях данные извлекаются из кэша. Это полезно, если вам потребуется повторно перепарсить те же страницы.

Сохранение собранных данных

Сохраняем данные в CSV файл (по ходу парсинга), который можно открыть в Excel для просмотра. Набор полей генерируем на основе текущих доступных полей сбора данных. Формат UTF8, разделитель “;”


```

"catalogXpaths": [
  "//section[@id='searchResults']//a[@class='m-link-ticket']",
  "//div[contains(@class, 'ticket-item')]/a[contains(@class, 'address')]",
  "//a[contains(@class, 'address') and contains(@href, '/auto/')]\"",
],
"paginationXpaths": [
  "//span[@class='page-item next']//a",
  "//a[contains(@class, 'page-link') and contains(text(), 'Далі')]",
  "//div[@class='pagination']//a[@class='next']",
  "//a[@class='page-link js-next']\"",
],
"phoneButtonXpaths": [
  "//button[contains(@class, 'phone')]",
  "//a[contains(@class, 'show-phone')]",
  "//div[@class='phone_show_link']",
  "//button[@class='show-phone']\"",
],
"dataFields": [
  {
    "name": "title",
    "xpathList": [
      "//h1[@class='head']",
      "//h1[contains(@class, 'heading')]",
      "//div[@class='head-title']//h1"
    ]
  }
]

```

```
]
},
{
  "name": "phone",
  "xpathList": [
    "//a[@class='phone bold']",
    "//div[contains(@class, 'seller-phones')]/a",
    "//a[contains(@class, 'show-phone')]",
    "//span[contains(@class, 'phone-number')]"
  ]
},
{
  "name": "name",
  "xpathList": [
    "//div[@class='seller_info_name']",
    "//div[contains(@class, 'seller-name')]",
    "//span[@class='seller-name'"
  ]
},
{
  "name": "city",
  "xpathList": [
    "//dd[@class='seller_info_location'",
    "//div[@class='seller_info_area'",
```

```
    "//span[contains(@class, 'location')]",
    "//div[contains(@class, 'seller-info-location')]"
  ]
},
{
  "name": "date",
  "xpathList": [
    "//div[@class='footer_box']//span[contains(@class, 'date')]",
    "//span[@class='date-created']",
    "//div[contains(@class, 'publication-date')]",
    "//span[contains(@class, 'auto-date')]"
  ]
},
{
  "name": "price",
  "xpathList": [
    "//div[@class='price_value']//strong",
    "//span[@class='price']",
    "//div[contains(@class, 'price-ticket')]",
    "//div[contains(@class, 'price_value')]"
  ]
}
],
"parsing": {
```

```
"threads": 3,

"delayBetweenRequests": {

  "min": 2,

  "max": 5

},

"pageLoadTimeout": 30000,

"waitForPaginationTimeout": 5000

},

"errorRetryTimes": 3,

"proxy": {

  "enabled": true,

  "rotation": true,

  "list": []

},

"cache": {

  "enabled": true,

  "directory": "./cache",

  "cacheListings": true,

  "cacheCatalog": false

},

"output": {

  "file": "output.csv",

  "encoding": "utf-8",

  "delimiter": ";"
```

}

}