Компьютерные инструменты в образовании, 2021
№ 3: 41–56
УДК: 004.056
http://cte.eltech.ru
doi:10.32603/2071-2340-2021-3-41-56

ТЕХНОЛОГИИ ПАРСИНГА NODE.JS В ЗАДАЧЕ АГРЕГАЦИИ СВЕДЕНИЙ И ОЦЕНКИ ПАРАМЕТРОВ ГРУЗОВЫХ МАРШРУТОВ ПОСРЕДСТВОМ ИЗВЛЕЧЕНИЯ ДАННЫХ ИЗ ОТКРЫТЫХ ИСТОЧНИКОВ*

Корепанова А. А.[1], младший научный сотрудник, aak@dscs.pro
Бушмелев Ф. В.[1], младший научный сотрудник, ✉ fvb@dscs.pro
Сабреков А. А.[1], младший научный сотрудник, sabrekov2@gmail.com

[1]Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (СПб ФИЦ РАН), 14 линия В.О., 39, 199178, Санкт-Петербург, Россия

Аннотация

Данная статья посвящена технологиям веб-скрейпинга (парсинга сайтов) для Node.js, применяемые в задаче агрегации сведений и оценки параметров грузовых маршрутов посредством извлечения данных из открытых источников. Задача веб-скрейпинга возникает во множестве различных контекстов как научных, так и промышленных. Задачи веб-скрейпинга имеют как широкое практическое применение, так и значительный образовательный аспект. Тем не менее, существующие материалы, посвящённые веб-скрейпингу разрозненны и не структурированы. В данной работе на примере решения научно-технической задачи агрегации сведений и оценки параметров грузовых маршрутов посредством извлечения данных из открытых источников представлен обзор технологий парсинга сайтов на Node.js, описана классификация сайтов по сложности, приведена систематизация особенностей сайтов, которые являются препятствием для парсинга, и возможные пути их обхода. Таким образом, достигается дидактическая цель данной статьи систематизировать материал по парсингу веб-сайтов.

Ключевые слова: *веб-скрейпинг, парсинг, веб-технологии, Node.js, HTML.*

Цитирование: Корепанова А. А., Бушмелев Ф. В., Сабреков А. А. Технологии парсинга Node.js в задаче агрегации сведений и оценки параметров грузовых маршрутов посредством извлечения данных из открытых источников // Компьютерные инструменты в образовании. 2021. № 3. С. 41–56. doi: 10.32603/2071-2340-2021-3-41-56

1. ВВЕДЕНИЕ

«Веб-скрейпинг» или *«парсинг»* сайтов (в русскоязычном пространстве и данной статье термины используются как синонимы) — это процесс автоматизированного сбора информации с веб-ресурсов [1]. Задача веб-скрейпинга возникает во множестве различных контекстов как научных, так и промышленных. Например, при необходимости

* Работа выполнена в рамках проекта по государственному заданию СПб ФИЦ РАН № 0073-2019-0003 и при поддержке ООО «ТАЗМАР».

настроить взаимодействие с сайтом-партнёром, который не имеет API, при сборе информации об ассортименте и ценах конкурентов для создания своего интернет-магазина и предсказания спроса [2], для формирования наборов данных в задачах машинного обучения [3–5] — среди них множество задач извлечения данных из социальных сетей [6–10], в рамках OSINT (разведка по открытым источникам) [11] — или при разработке сайта-агрегатора, собирающего информацию из разных источников на одном ресурсе (aviasales.ru, skyscanner.ru и т. п.). Индексация сайтов поисковыми системами тоже происходит с помощью веб-скрейпинга: специальный кроулер (*«crawler», «spider», «spiderbot»*) проходит по сайтам, размещённым в интернете, и извлекает их содержимое, чтобы добавить в свою выдачу. Кроме того, решение задач веб-скрейпинга позволяет глубже погрузиться в веб-технологии, а также изучить структуру работы чужих веб-ресурсов, что расширяет кругозор разработчика. Таким образом, задачи веб-скрейпинга имеют как широкое практическое применение, так и значительный образовательный аспект.

Веб-скрейпинг, хотя и является законным [12], часто не поощряется сайтами, с которых информацию выгружают, так как одновременная выгрузка больших объёмов данных с ресурса может навредить ему, например, значительно понизить скорость работы. Поэтому при извлечении данных важно изучать пользовательское соглашение сайта, делать интервалы между запросами, прибегать к иным дополнительным мерам, позволяющим не наносить сайтам и их владельцам ущерба.

Таким образом, рынок различных агрегаторов и других ресурсов, которые решают задачи веб-скрейпинга, довольно широкий, востребованность в специалистах растет. Сложность парсинга возрастает, поскольку развиваются технологии защиты от него. Вместе с тем отсутствуют курсы в рамках учебного процесса, позволяющие освоить данный инструментарий, информация не структурирована, разрознена. Таким образом, основная цель данной статьи — дидактическая — систематизировать материал по парсингу веб-сайтов и изложить его в структурированном виде, рассмотрев применение на примере научно-технической задачи агрегации сведений и оценки параметров грузовых маршрутов посредством извлечения данных из открытых источников.

Кроме дидактической цели статья имеет научно-техническую цель, которая заключается в автоматизации агрегации сведений и оценке параметров грузовых маршрутов на основе данных о доставке грузовых контейнеров. Актуальность данной проблематики подчеркивается тем, что в настоящее время наблюдается стремительный рост объёма грузоперевозок. В первом квартале 2021 года на рынке розничной электронной торговли в России было доставлено 290 миллионов посылок, что на 56 % выше, чем за аналогичный период в 2020 году [13]. Эпидемия коронавируса [14] привела к тому, что объёмы грузоперевозок и курьерских доставок выросли. В этих условиях актуальными становятся исследования, направленные на повышение качества услуг перевозчиков [15–17]. Многие грузовые компании, которые занимаются перевозкой контейнеров, предлагают клиентам возможность отследить текущее местоположение контейнера по его уникальному номеру и пишут предположительную дату его прибытия в пункт назначения. Эти временные оценки зачастую оказываются неточными. Актуальной, таким образом, является задача агрегации сведений о предположительном и итоговом времени доставки грузовых контейнеров для дальнейшего анализа качества предсказаний различных компаний, а также причин, возникающих в пути задержек. Для проведения дальнейшего анализа на первом этапе необходимо использовать технологии веб-скрейпинга для реализации сервиса агрегации данных. Научная значимость общего исследования

лежит в разработке моделей оценки и анализа грузовых маршрутов. Практическая значимость данного этапа исследования состоит в разработке инструментария для агрегации данных о статусах контейнеров.

Таким образом, статья посвящена инструментам и подходам, которые могут быть использованы при реализации одной из задач скрейпинга, а именно при создании сайта-агрегатора. Сайты-агрегаторы позволяют пользователям получать различную информацию из множества ресурсов в одном месте. Примерами могут послужить сайты — новостные агрегаторы, сайты, представляющие на своих страницах цены на товары или услуги разных компаний одной направленности. В данной статье рассмотрены технологии, которые используются для разработки агрегаторов выполняющих переадресацию запросов пользователя на сторонние ресурсы и представляющие полученную информацию на своем ресурсе. Рассмотрены основные модули Node.js, использующиеся при парсинге, проведен обзор основных методов защиты сайтов от парсинга и методов их обхода. Практическая значимость статьи состоит в подробном обзоре инструментов и подходов, использующихся в задаче веб-скрейпинга, которая является многоаспектной задачей, возникающей во множестве различных контекстов, а её решение способствует освоению веб-технологий.

Статья построена следующим образом: в разделе «Релевантные работы» представлено описание существующих работ по данной теме; дальше следует описание HTTP(S)-протокола, с помощью которого происходит взаимодействие с сайтами; разделы «Классификация сайтов по сложности», «Парсинг простых сайтов» и «Парсинг сложных сайтов» посвящены различным методам защиты сайтов от парсинга; в разделе «Инструменты» предложены основные технологии и инструменты (в том числе модули Node.js), необходимые для написания парсера, в разделе «Сервис агрегации сведений о статусе грузовых контейнеров» представлено описание сервиса, написанного с применением описанных подходов.

1.1. Релевантные работы

В интернете представлено множество ресурсов, в которых описаны подходы к парсингу сайтов [18–26]. Так, например, в статьях [18–20] представлены некоторые методы парсинга на языке Python, дан обзор используемых библиотек. В таких статьях по парсингу сайтов преобладают материалы, освещающие особенности решения задачи на языке Python. Этот язык является довольно простым, но не самым быстрым и может проигрывать в производительности другим решениям, поэтому в данной статье проводится обзор инструментов платформы Node.js, которая имеет ряд преимуществ, описанных ниже в разделе «Инструменты», в частности, в скорости. Существующие материалы по написанию парсеров на Node.js либо обозревают исключительно основы парсинга самых простых сайтов [21–24], либо, наоборот, только узкоспецифичные библиотеки [25].

Также существуют готовые ресурсы, реализующие парсинг за пользователя, которому остаётся только указать этим ресурсам задачу в некоторой форме [27–29]. Недостатком таких решений является то, что они, если и подходят для парсинга простых страниц, зачастую не способны справиться с более нетривиальной задачей, для решения которой всё равно необходимо самостоятельное погружение в материал.

2. HTTP(S)-ЗАПРОС

«HTTP-протокол» — это основной протокол передачи данных, который используется в интернете для взаимодействия с сайтами [30]. Чтобы получить необходимые данные

с сайта, требуется сформировать для него корректный HTTP(S)-запрос. Для этого нужно изучить HTTP(S)-запросы, которые посылает браузер, например с помощью консоли разработчика, имеющейся почти в любом браузере во вкладке «Network» (рис. 1).

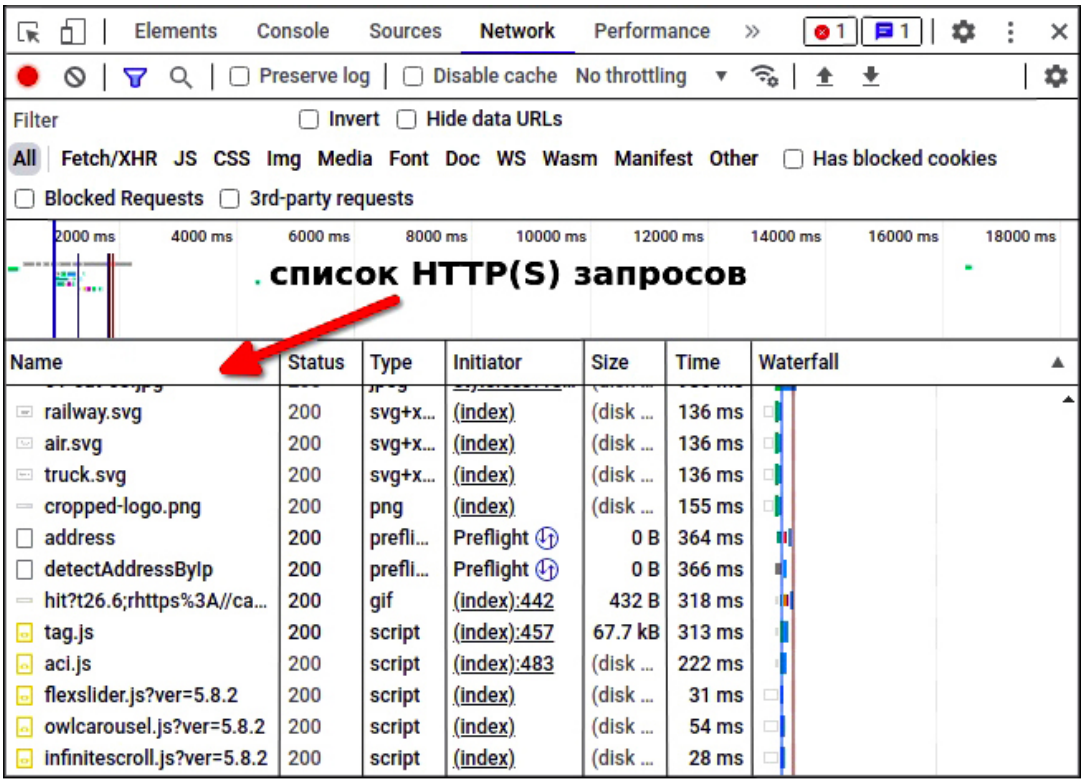

Рис. 1. Вкладка Network в консоли разработчика браузера Google Chrome

Ниже дано краткое описание, из чего состоит HTTP(S)-запрос [30]:

- **Метод запроса**. Он определяет операцию, которая должна выполниться на сервере. Их всего существует девять, в рамках задачи парсинга интерес представляют две: GET и POST, более подробно про остальные можно прочитать по ссылке [31]. Метод GET используется для получения данных с сервера, параметры запроса передаются в URL, сам запрос применяется для получения статической информации, которая может быть закэширована (то есть временно сохранена, чтобы предотвратить лишние обращения к серверу) браузером. Параметры, передаваемые методом GET, можно заметить в адресной строке браузера после вопросительного знака. Запись может выглядеть, например, следующим образом: «test.tu?p=1&p=кукуруза». Метод POST предназначен для загрузки данных на сервер, однако может быть использован для получения динамического контента (например постов на странице, комментариев и т. д.). Параметры, передаваемые методом POST в адресной строке не отображаются, они содержатся в теле запроса. GET-запросами информацию получить обычно гораздо проще, а разница в них со стороны клиента невелика, поэтому, не умаляя общности, в данной статье будет говориться про POST-запросы.
- **Путь к ресурсу** (*URL*).
- **Версия протокола** [32]. Версия протокола HTTP/2.0 обеспечивает более быстрое взаимодействие между клиентом и сервером, чем предыдущие, однако поддерживается не всеми сайтами.

- **Заголовки** (*Headers*). В них предоставляется дополнительная информация о клиенте, браузере клиента, запрошенной странице и т. д.
- **Тело запроса**. Тут могут быть параметры запроса или же файл.

Чтобы получить необходимую информацию к ресурсу, необходимо отправить корректный запрос, особое внимание уделив заголовкам и телу запроса. Стандартный набор заголовков, которые важны для получения желаемого ответа на запрос: *Content-Type* (формат содержимого), *Cookie* (фрагменты данных, которые сервер оставил на хранение на компьютере клиента для его аутентификации, хранения персональных настроек и т. д.), *User-Agent* (версия браузера). Однако данный перечень не является исчерпывающим. В POST-запросе после параметров идёт тело запроса, в котором обычно и передаются необходимые для получения данных параметры.

Прежде чем перейти к описанию того, какие составляющие HTTP(S)-запроса и как мы используем, приведем классификацию сайтов по сложности.

3. КЛАССИФИКАЦИЯ САЙТОВ ПО СЛОЖНОСТИ

Автоматизация запросов к сайту может быть для него опасна — оказывать слишком высокую нагрузку на сервер, из-за чего сильно снижать скорость обработки запросов. В связи с этим на сайтах зачастую присутствует защита от веб-скрейпинга, которая отличает автоматические запросы от человеческих и ограничивает доступ к сайту. Зачастую для скрейпинга информации с сайта эту защиту приходится обходить, но всегда это важно делать с высокой степенью аккуратности, соблюдая интервалы между запросами, чтобы не нарушать работу ресурса.

Некоторые защиты обойти очень просто, некоторые возможно только с приложением больших усилий. Условно можно разделить сайты на 3 уровня сложности:

1. **Легкий**. К нему относятся сайты, которым для получения необходимой информации достаточно одного HTTP(S)-запроса с любыми заголовками, изменяющийся параметр запроса просто подобрать.
2. **Средний**. Здесь может быть несколько изменяющихся обязательных параметров у запроса, могут быть обязательные переменные заголовки запроса. Информацию с сайта этого уровня сложности все еще можно получить обычными HTTP(S)-запросами.
3. **Сложный**. Сюда входят сайты с обязательным вводом текстовой или нетекстовой CAPTCHA (например выбор объектов на фото, совмещение пазлов, аудио-CAPTCHA и т. д.), сайты, в которых один из обязательных параметров запроса получается в результате выполнения JavaScript-кода на странице — простыми HTTP(S)-запросами этого сделать невозможно.

Провести первичную оценку, к какому классу относится страница, можно визуально. Для этого необходимо изучить запросы, которые посылает браузер для получения необходимой информации, и HTML-код страницы. Изучить запросы можно, например, с помощью вкладки «Network» консоли разработчика в браузере (рис. 1).

1. Первым делом нужно определить, в ответ на какой запрос приходит необходимая информация. Если необходимая информация получается после заполнения формы и нажатия на кнопку, нужно это сделать с открытой консолью разработчика. В консоли отобразятся все запросы, которыми обменялись клиент с сервером. С помощью поиска по консоли или простого перебора запросов можно определить, тело ответа какого запроса содержит нужную информацию.

2. Далее следует посмотреть, какие параметры используются в теле запроса. Параметры могут иметь различную для парсинга сложность обработки. Они могут быть достаточно простыми (фиксированными или получаемыми через форму от клиента), либо более сложными (например динамически генерируемыми с помощью выполнения скриптов). Если параметры простые, то, вероятно, при соблюдении других условий сайт принадлежит к первой группе, иначе является более сложным.
   Приведём пример простых параметров, они обычно недлинные и могут иметь понятный для пользователя смысл, пример представлен на рисунке 2: параметр *«type»* короткий и, скорее всего, является фиксированным, *«action»* означает команду *«поиск»* и в рамках поиска тоже будет скорее всего фиксированным, параметр *«number1»* вводится в форму на клиенте:

Рис. 2. Пример простых параметров

   Рассмотрим пример сложного параметра:
   «_csrf: n7h7LFK-et5k38yDXhIalfG77XOkDQSA16cW0E»:
   `_csrf: n7h7LFK-et5k38yDXhIalfG77XOkDQSA16cW0Euo-UXKwSpFIMYCvxLrgbokZn7SyIqCP-pDba2j_3GiH-uhcw`.
   Этот параметр выглядит как сгенерированный случайным образом, параметры такого вида практически всегда будут переменными.
3. И в конечном итоге стоит обратить внимание на заголовок, который содержит Cookie. В случае отсутствия у веб-сайта этого заголовка и простоты параметров, оцениваемых на предыдущем шаге, сайт является простым для парсинга. Если присутствует хотя бы один Cookie, то необходимо определить, является ли он обязательным, а также как часто меняется. Если все Cookie необязательные, то есть запросы получают корректные ответы и без них, то нет необходимости определять алгоритм их получения. Если Cookie являются бессрочными, то можно использовать их как статический параметр.

Если в коде атрибут src какого-либо html-тега на странице сайта начинается следующим образом: *«src="/_Incapsula_Resource?...»* или один из промежуточных запросов уходит на сервис *«Google Recaptcha»*, то сайты точно сложные и имеют высокий уровень защиты от автоматического веб-скрейпинга.

Если по одному из вышеперечисленных параметров сайт можно отнести к сложным для парсинга (есть текстовая или нетекстовая CAPTCHA или видно, что есть много обязательных запросов и где-то в промежутке требуется выполнение javascript-кода), то одними HTTP(S)-запросами обойтись не получится. Как писать парсеры таких сайтов, представлено ниже.

Ниже представлены основные методы защиты сайтов от автоматизированного получения информации:

- Обязательные заголовки.
- Cookie.
- Параметры в теле запроса:

— токены и альтернативы,
— скрытые инпуты (hidden inputs).

- Реализация JavaScript-кода.
- CAPTCHA.

Далее рассмотрим более подробно каждый из перечисленных методов.

4. ПАРСИНГ ПРИ РАЗЛИЧНЫХ МЕТОДАХ ЗАЩИТЫ

Начать поиск решения можно с попытки повторения HTTP(S)-запроса, который сайт отправляет для получения необходимой информации. Для этого в консоли разработчика есть команда «*Replay*», которая посылает такой же запрос из браузера, который был послан раньше (рис. 3).

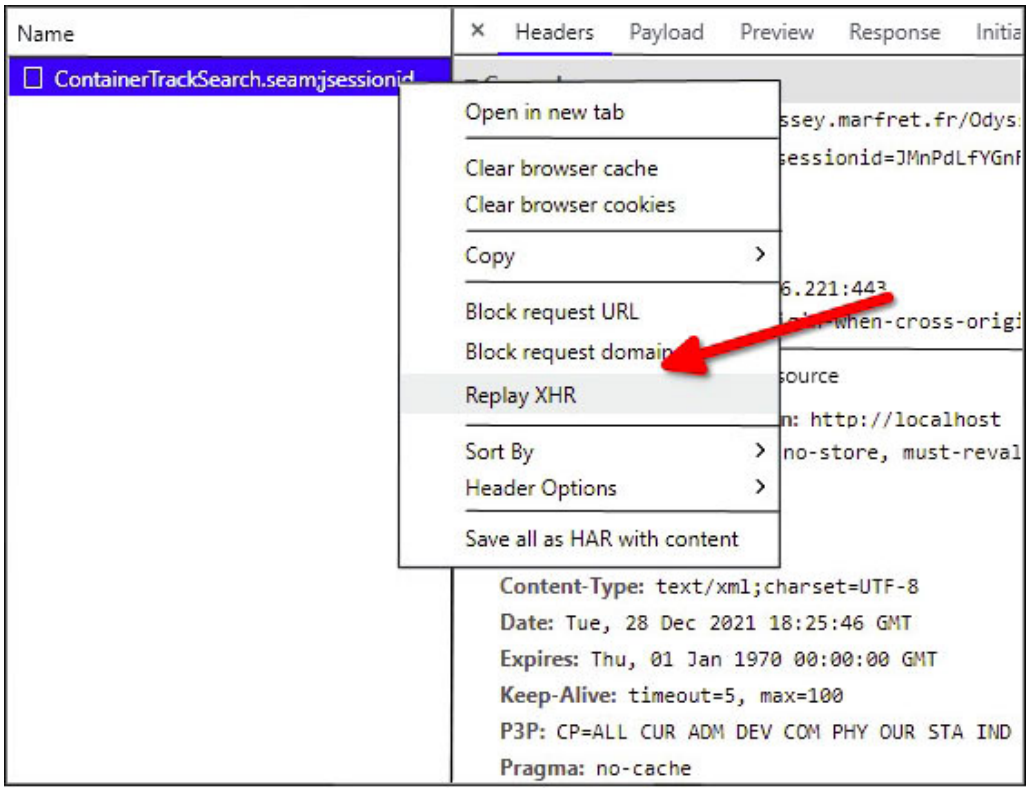

Рис. 3. Повторение запроса из консоли разработчика через команду Replay

В качестве альтернативы можно воспользоваться инструментом отладки *«Fidler»*, где есть функция *«reissue request()»*.

4.1. Обязательные заголовки

Большинство серверов не пропускают запросы, если в них отсутствуют обязательные заголовки. Обычно это заголовки *«User-Agent»*, *«Cookie»*, *«Content-Type»*, но иногда бывают и другие. Поэтому важно внимательно исследовать, какие заголовки являются обязательными, а какие нет.

4.2. Cookie

«Cookie» — это фрагменты данных, которые сервер оставил на хранение на компьютере клиента для его аутентификации, хранения персональных настроек и т. д. [33] Сначала необходимо убедиться, что ответ сервера зависит от Cookie. Можно повторить запрос браузера, затем, если запрос прошел, убрать заголовок Cookie. Если ответ отличный от ожидаемого — значит, запрос зависит от Cookie. Если не изменился — Cookie можно не использовать.

Затем нужно понять, какой конкретно Cookie требуется. Чтобы в этом разобраться, можно убирать из запроса по одному Cookie и смотреть, продолжает ли сервер возвращать корректную информацию. В результате будет список из необходимых Cookie, получение значений которых нужно далее автоматизировать.

Следующим шагом необходимо найти в поиске запрос, в котором присутствует заголовок ответа *«Set-Cookie»* с названием этого Cookie. Этот запрос нужно будет повторять для получения значения Cookie.

Бывает затруднительно найти запрос с заголовком Set-Cookie, если сервер выдает его 1 раз на долгое время. В таком случае можно очистить Cookie сайта средствами браузера и обновить страницу.

4.3. Параметры в теле запроса

Под параметрами в теле запроса здесь понимается совокупность всевозможных переменных параметров, которые не зависят от ввода пользователя, каким-то образом получаются с сервера или генерируются браузером и без которых невозможно получить информацию с сайта.

- **Токены**. Некоторые сайты могут использовать для получения данных целую цепочку обязательных запросов. В ответе на каждый запрос выдаётся некоторый обязательный токен, который нужно подставлять в параметры к следующему запросу.
- **Скрытые поля ввода**. Чтобы защитить сайт от автоматического парсинга, на веб-странице, в html-коде которой размещаются скрытые html-теги, идентификаторы (*id*) которых обычно совпадают с id параметра в запросе, а значение совпадает, соответственно, со значением параметра. Обычно эти параметры автоматически генерируются на сервере, называются «__VIEWSTATE», «__VIEWSTATEGENERATOR», «__EVENTVALIDATION» и т. п. Браузер при составлении тела запроса к странице заполняет значения этих параметров с помощью выполнения JavasSript-кода. Таким образом, при формировании запроса на получение нужной информации, необходимо сделать предварительный запрос к странице для получения html-документа и извлечения из него значения этих скрытых полей ввода.

4.4. Реализация Javascript-кода

Иногда сам url-адрес, по которому сайт делает запрос на получение информации, генерируется через выполнение JavaScript-кода. По запросу выдается сам скрипт и предполагается, что браузер его выполнит и перейдет по ссылке. Также какой-то из обязательных параметров запроса может генерироваться через выполнение JavaScript-кода. В любом случае, используя лишь обычные HTTP(S)-запросы, получить необходимую информацию становится невозможным. В этом случае можно использовать инструменты *«Phantomjs»* или *«Puppeteer»*, которые являются браузерами без визуального отображения, то есть «безголовыми бразуерами». С их помощью можно не только делать простые

запросы, но и выполнять JavaScript-код на странице, а также заполнять формы, делать имитацию кликов и многое другое.

Альтернативным решением является выполнение части JavaScript-кода самостоятельно. Если удаётся определить, какой скрипт генерирует необходимый параметр, можно повторить его на Node.js и попытаться выполнить.

Эмуляция браузера не всегда срабатывает: некоторые сайты могут вычислять, что манипуляции проводятся не с реального браузера.

Если сайт кажется простым или средним по сложности, желательно попробовать решить его простыми запросами а не прибегать сразу к использованию «безголовых» браузеров, так как они являются очень ресурсоёмкими в использовании, что может быть критичным при написании сайта-агрегатора.

4.5. CAPTCHA

На некоторых сайтах для отправки запроса к особенно защищаемым данным или данным, к которым часто обращаются, требуется решить CAPTCHA. Это может быть как текстовая CAPTCHA, так и слайдер или Google Recaptcha с выбором картинок. Для решения CAPTCHA самым простым вариантом является использование внешних сервисов, которые либо вручную решают присланные CAPTCHA, либо применяют автоматизацию. Самостоятельная автоматизация зачастую требует погружения в область машинного обучения и компьютерного зрения.

5. ИНСТРУМЕНТЫ

Программу для задач веб-скрейпинга можно писать потенциально на любом языке. Однако большинство написанных интрукций и руководств по веб-скрейпингу описывают инструменты языка Python. Язык Python имеет как множество преимуществ, среди которых важное место занимают низкий порог вхождения и простота синтаксиса, так и ряд недостатков, к которым относится скорость выполнения. Поэтому в данном обзоре предложены инструменты платформы Node.js. Преимуществом Node.js в данном контексте также является её ориентированность на асинхронное программирование [34], в данном контексте это значит удобный инструментарий для написания программы таким образом, чтобы она не простаивала в ожидании ответа на HTTP-запрос от какого-либо сайта, а была способна продолжать выполнение. Это позволяет запустить несколько парсеров одновременно и максимально эффективно использовать время. Используемые в веб-скрейпинге модули Node.js:

— *«node-fetch»* — используется для простой отправки HTTP(S)-запросов;
— *«http-proxy-agent»* — используется для отправления запросов через прокси;
— *«cheerio»* — используется для парсинга html-документа;
— *«phantomjs»* — «безголовый» браузер, то есть программа, выполняющая функции браузера, но не отрисовывающая содержимое сайтов на экране. Данная программа позволяет автоматизировать взаимодействие с веб-страницей, и, в частности, позволяет автоматически выполнять JavaScript-код сайта, который сервер посылает на клиент для выполнения в браузере.
— *«puppeteer»* — позволяет использовать для эмуляции браузера не только phantomjs, который является несколько упрощённым по сравнению с полноценными браузерами, но и такие популярные браузеры как Chrome, Mozilla Firefox и т. д.

Удобными инструментами анализа интернет-трафика в веб-скрейпинге являются консоль разработчика, доступная в большинстве браузеров по нажатию клавиши «F12» на клавиатуре.

Также необходимо на базовом уровне освоить Fidler [35]. Он позволяет перехватывать трафик взаимодействия браузера и сервера. Зачастую в нем можно получить больше информации, чем во вкладке Network консоли разработчика браузеров.

6. СЕРВИС АГРЕГАЦИИ СВЕДЕНИЙ О СТАТУСЕ ГРУЗОВЫХ КОНТЕЙНЕРОВ

Данные инструменты были использованы в рамках разработки сервиса агрегации информации с сайтов морских перевозчиков, а именно данных о статусе и местоположении грузового контейнера (ISO-контейнера) [36]. Определение статуса грузового контейнера — актуальная задача в работе клиентов грузовых компаний. ISO-контейнер — стандартизированная тара, предназначенная для перевозки грузов несколькими видами транспорта, позволяющая удобную механизированную перегрузку с одного транспортного средства на другое. Каждый ISO-контейнер имеет свой уникальный номер, по которому его можно идентифицировать. Он состоит из 4 буквенных символов, идентифицирующих компанию перевозчика, и 7 цифровых символов, например, XXXX1234567, где XXXX — это префикс морской линии, 1234567 — семизначный цифровой идентификатор. На многих сайтах грузовых компаний существует возможность запросить по номеру контейнера информацию о его местоположении, статусе, планируемом времени доставки и т. д. Клиенты, перевозящие большие объёмы грузов разными компаниями, вынуждены запрашивать текущий статус множества контейнеров на разных сайтах грузовых компаний вручную. Для ускорения этого процесса был разработан агрегатор информации о статусе грузовых контейнеров, который автоматически определяет, какой компанией перевозится контейнер и при запросе пользователя совершает запросы к сайтам различных грузовых компаний.

Основной алгоритм работы:

1. После того как пользователь вводит номер контейнера, он отправляется на сервер сайта вместе со случайно сгенерированным идентификатором запроса.
2. По префиксу номера контейнера определяются вероятные компании-перевозчики и посылаются запросы на их сервисы трекинга контейнеров. Соответствие между префиксами и компаниями-перевозчиками не взаимно-однозначное, поэтому зачастую определить компанию сразу оказывается невозможным, и приходится посылать запросы во множество компаний. Наиболее вероятные компании-перевозчики – это те, в которых контейнеры с данным префиксом находились ранее. Запросы во все компании не посылаются, чтобы уменьшить нагрузку на сервера компаний и не посылать заведомо ложные запросы.
3. Если в какой-то из компаний нашёлся контейнер с запрашиваемым номером, то статус контейнера считается определенным, если нет — посылаются запросы в остальные компании. Если контейнер найден на их сайтах, то компания добавляется в список возможных перевозчиков контейнеров с данным префиксом, результат отправляется пользователю.

Сервис находится на этапе активной разработки и роста, поэтому для быстрого добавления и развёртывания новых функциональностей была выбрана бессерверная архитектура (рис. 4), сервис развёрнут на платформе Yandex.Cloud [37].

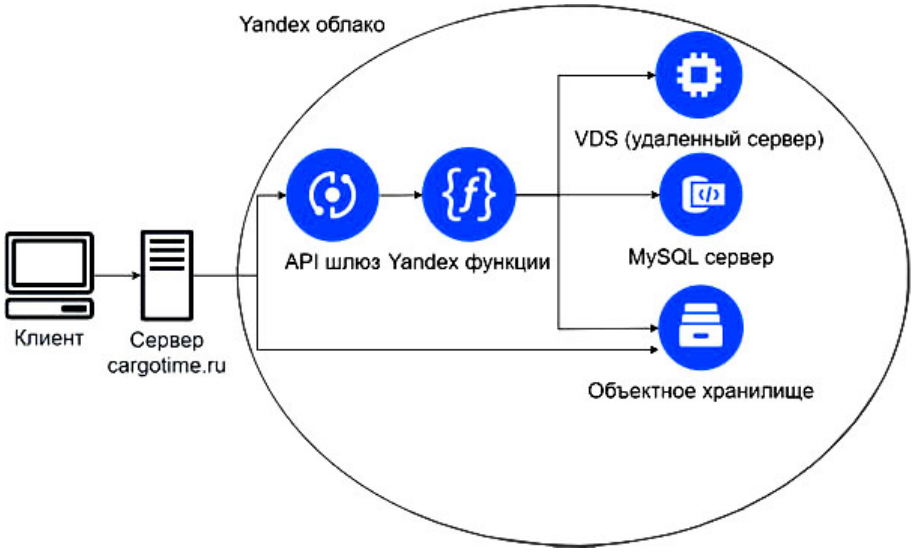

Рис. 4. Архитектура сервиса трекинга контейнеров

Ниже представлен рейтинг по частоте использования защит от парсинга среди сайтов грузовых компаний, начиная с самой часто встречающейся:

1. Отсутствие какой-либо защиты.
2. Скрытые поля ввода.
3. Заголовки Cookie.
4. Скрытые поля ввода вместе с заголовками Cookie.
5. CAPTCHA разных видов (обычно этот метод защиты сопровождается обязательными заголовками Cookie).
6. Выполнение JavaScript кода.

7. ЗАКЛЮЧЕНИЕ

Задача веб-скрейпинга, то есть автоматизированного извлечения данных с сайтов, широко распространена и имеет множество приложений. В данной статье представлен обзор подходов и инструментария для задачи скрейпинга информации с сайтов и написания своего ресурса-агрегатора на Node.js. Представлено описание модулей Node.js, подходящих для решения данной задачи, рассмотрена классификация сайтов по уровню сложности, а также описаны часто встречающиеся методы защиты сайтов от парсинга и способы её обхода. Описанные в статье подходы являются рабочими и применяются в реальном проекте в рамках реализации ресурса https://cargotime.ru/, в статье дано описание сервиса агрегации сведений о статусе ISO-контейнеров.

Таким образом, выполнена основная дидактическая цель данной статьи, а именно изложен материал по парсингу веб-сайтов в структурированном виде на примере создания платформы для агрегации сведений о морских перевозках и калькуляторах доставки, а также дано описание выполнения научно-технической задачи. В дальнейших публикациях планируется более подробно исследовать вопрос законности и допустимости парсинга отдельных сайтов.

Список литературы

1. *Москаленко А. А., Лапонина О. Р., Сухомлин В. А.* Разработка приложения веб-скрапинга с возможностями обхода блокировок // Современные информационные технологии и ИТ-образование. 2019. Т. 5, № 2. С. 413–420. doi: 10.25559/SITITO.15.201902
2. *Chong A.Y.L., Ch'ng E., Liu M.J., Li B.* Predicting consumer product demands via Big Data: the roles of online promotional marketing and online reviews // International Journal of Production Research. 2017. № 55 (17). P. 5142–5156.
3. *Басалаева А. Ю., Гареева Г. А., Григорьева Д. Р.* Web-scraping и классификация текстов методом наивного Байеса // Инновационная наука. 2018. № 5. С. 11–14.
4. *Barbado R., Araque O., Iglesias C.A.* A framework for fake review detection in online consumer electronics retailers // Information Processing and Management. 2019. № 56 (4). P. 1234–1244.
5. *De Mauro A., Greco M., Grimaldi M., Ritala P.* Human resources for Big Data professions: A systematic classification of job roles and required skill sets // Information Processing and Management. 2018. № 54 (5). P. 807–817.
6. *Корепанова А. А., Абрамов М. В.* Применение случайного леса в выборе метода восстановления возраста пользователя социальной сети // Искусственный интеллект и принятие решений. 2021. № 2. С. 66–77.
7. *Oliseenko V. D., Abramov M. V., Tulupyev A. L.* Identification of user accounts by image comparison: The phash-based approach // Scientific and Technical Journal of Information Technologies, Mechanics and Opticsthis. 2021. № 21 (4). P. 562—570.
8. *Bushmelev F., Abramov M., Tulupyeva T.* Adaptive method of color selection in application to social media images. // CEUR Workshop Proceedings. 2020. Vol. 2782. P. 252–257.
9. *Торопова А. В.* Сбор данных о последних эпизодах и интенсивности постинга в социальной сети ВКонтакте // Региональная информатика и информационная безопасность. Сборник трудов. Вып. 9. СПб.: СПОИСУ, 2020. С. 228–230.
10. *Stoliarova, V. F., Tulupyev, A. L.* Regression Model for the Problem of Parameter Estimation in the Gamma Poisson Model of Behavior: An Application to the Online Social Media Posting Data // Proceedings of 2021 24th International Conference on Soft Computing and Measurements, SCM 2021. 2021. P. 24–27.
11. *Seitz J.* Building a Keyword Monitoring Pipeline with Python, Pastebin and Searx [Электронный ресурс]. 2017. URL: https://www.bellingcat.com/resources/2017/04/21/building-keyword-monitoring-pipeline-python-pastebin-searx/ (дата обращения: 23.09.2021).
12. *Кульгин М.* Парсинг сайтов. Россия и мир. Как с точки зрения закона выглядит один из самых полезных инструментов? [Электронный ресурс]. 2019. URL: https://vc.ru/legal/64328-parsing-saytov-rossiya-i-mir-kak-s-tochki-zreniya-zakona-vyglyadit-odin-iz-samyh-poleznyh-instrumentov (дата обращения: 23.09.2021).
13. Последняя миля для интернет-торговли [Электронный ресурс]. 2021. URL: http://logistics.datainsight.ru/poslednyaya-milya-dlya-internet-torgovli (дата обращения: 23.09.2021).
14. *Ворошилов Д.* РБК. Эксперты назвали объем выполненных российскими курьерами за год заказов [Электронный ресурс]. 2021. URL: https://www.rbc.ru/business/24/11/2021/619cffaa9a79473c95fc20d0 (дата обращения: 23.09.2021).
15. *Balster A., Hansen O., Friedrich H., Ludwig A.* An ETA Prediction Model for Intermodal Transport Networks Based on Machine Learning // Business and Information Systems Engineering. 2020. № 62 (5). P. 403–416.
16. *Atak Ü., Arslanoğlu Y.* Machine learning methods for predicting marine port accidents: a case study in container terminal // Ships and Offshore Structures. 2021. doi: 10.1080/17445302.2021.2003067
17. *Jurdana I., Krylov A., Yamnenko J.* Use of artificial intelligence as a problem solution for maritime transport // Journal of Marine Science and Engineering. 2021. № 8 (3): № 201. doi: 10.3390/jmse8030201
18. *Gal El. Al. Web Scraping With Python: A Beginner's Guide* [Электронный ресурс]. 2020. URL: https://brightdata.com/blog/how-tos (дата обращения: 23.09.2021).
19. Web Parsing. Основы на Python [Электронный ресурс]. 2020. URL: https://vc.ru/newtechaudit/109368-web-parsing-osnovy-na-python (дата обращения: 23.09.2021).
20. Нагайкин А. Web Scraping [Электронный ресурс]. 2020. URL: https://habr.com/ru/post/488720/ (дата обращения: 23.09.2021).

21. Парсинг сайтов при помощи Node.js: краткое руководство с примерами [Электронный ресурс]. 2020. URL: https://tproger.ru/translations/web-scraping-node-js/ (дата обращения: 23.09.2021).
22. *Rawat A.* Web Scraping in Node.js with Multiple Examples [Электронный ресурс]. 2017. URL: https://hackprogramming.com/web-scraping-in-node-js-with-multiple-examples/ (дата обращения: 23.09.2021).
23. *Ni D.* The Ultimate Guide to Web Scraping with Node.js [Электронный ресурс]. 2018.URL: https://www.freecodecamp.org/news/the-ultimate-guide-to-web-scraping-with-node-js-daa2027dcd3/ (дата обращения: 23.09.2021).
24. *Джейн А.* Веб-скрапинг с помощью Node.js — Часть 1 [Электронный ресурс]. 2019. URL: https://blog.bitsrc.io/https-blog-bitsrc-io-how-to-perform-web-scraping-using-node-js-5a96203cb7cb (дата обращения: 23.09.2021).
25. Levada E. Puppeteer: парсинг сайтов с JavaScript [Электронный ресурс]. 2020. URL: https://proglib.io/p/puppeteer-parsing-saytov-na-javascript-2020-06-16 (дата обращения: 23.09.2021).
26. *Kumar M., Bhatia R., Rattan D.* A survey of Web crawlers for information retrieval // Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2017. № 7 (6): № e1218. doi: 10.1002/widm.1218
27. Scrapy [Электронный ресурс]. 2021. URL: https://scrapy.org/ (дата обращения: 23.09.2021).
28. Cloudflare. The easiest way to accelerate and secure your website [Электронный ресурс]. 2021. URL: https://www.cloudflare.com/ (дата обращения: 23.09.2021).
29. ZennoPoster — Автоматизируйте любые задачи в интернете [Электронный ресурс]. 2021. URL: https://zennolab.com/ru/products/zennoposter/ (дата обращения: 23.09.2021).
30. Обзор протокола HTTP [Электронный ресурс]. 2021. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Overview (дата обращения: 23.09.2021).
31. Методы HTTP запроса [Электронный ресурс]. 2021. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Methods
32. Evolution of HTTP [Электронный ресурс]. 2021. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP (дата обращения: 23.09.2021).
33. Как Google использует файлы cookie [Электронный ресурс]. 2021. URL: https://policies.google.com/technologies/cookies?hl=ru (дата обращения: 23.09.2021).
34. Основные понятия асинхронного программирования [Электронный ресурс]. 2021. URL: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Asynchronous/Concept (дата обращения: 23.09.2021).
35. Bugs. Happen. Debugging and Troubleshooting Reimagined [Электронный ресурс]. 2021. URL: https://www.telerik.com/fiddler (дата обращения: 23.09.2021).
36. *Lewandowski Kr.* Growth in the Size of Unit Loads and Shipping Containers from Antique to WWI // Packaging Technology and Science. 2016. Vol. 29, № 8–9. P. 451—478. doi: 10.1002/pts.2231.
37. Yandex.Cloud [Электронный ресурс]. 2021. URL: https://cloud.yandex.ru/ (дата обращения: 23.09.2021).

Поступила в редакцию 20.08.2021, окончательный вариант — 23.09.2021.

Корепанова Анастасия Андреевна, младший научный сотрудник лаборатории теоретических и междисциплинарных проблем информатики, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (СПб ФИЦ РАН), aak@dscs.pro

Бушмелев Федор Витальевич, младший научный сотрудник лаборатории теоретических и междисциплинарных проблем информатики, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (СПб ФИЦ РАН), ✉ fvb@dscs.pro

Сабреков Артём Азатович, младший научный сотрудник лаборатории теоретических и междисциплинарных проблем информатики, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (СПб ФИЦ РАН), sabrekov2@gmail.com

Computer tools in education, 2021
№ 3: 41–56
http://cte.eltech.ru
doi:10.32603/2071-2340-2021-3-41-56

Node.js Parsing Technologies in the Task of Aggregating Information and Evaluating the Parameters of Cargo Routes by Extracting Data from Open Sources

Korepanova A. A.[1], Junior Researcher, aak@dscs.pro
Bushmelev F. V.[1], Junior Researcher, ✉ fvb@dscs.pro
Sabrekov A. A.[1], Junior Researcher, sabrekov2@gmail.com

[1]St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS), 14-th Linia, VI, 39, 199178, Saint Petersburg, Russia

Abstract

This article is devoted to the technologies of web scraping (web crawling) for Node.js, used in the task of aggregating information and estimating the parameters of cargo routes by extracting data from open sources. The challenge of web scraping occurs in many different contexts, both scientific and industrial. The tasks of web scraping have both wide practical applications and a significant educational aspect. However, the existing material on web scraping is scattered and unstructured. In this paper, using the example of solving the scientific and technical problem of aggregating information and evaluating the parameters of cargo routes by extracting data from open sources, an overview of the technologies for web scraping on Node.js is presented, the classification of sites by complexity is described, the systematization of the features of sites that are an obstacle to scrapinf is given, and possible ways to bypass them. Thus, the didactic goal of this article is achieved to systematize the material on parsing websites.

Keywords: *web scraping, web crawling, web technologies, Node.js, HTML.*

Citation: A. A. Korepanova, F. V. Bushmelev, and A. A. Sabrekov, "Node.js Parsing Technologies in the Task of Aggregating Information and Evaluating the Parameters of Cargo Routes by Extracting Data from Open Sources," *Computer tools in education*, no. 3, pp. 41–56, 2021 (in Russian); doi:10.32603/2071-2340-2021-3-41-56

Acknowledgements: *The paper is prepared with the support of the St. Petersburg Federal Research Center of the Russian Academy of Sciences, the government task № 0073-2019-0003 and LLC «TAZMAR».*

References

1. A. A. Moskalenko, O. R. Laponina, and V. A. Sukhomlin, "Developing a Web Scraping Application with Bypass Blocking," *Sovremennye informacionnye tehnologii i IT-obrazovanie*, vol. 15, no. 2, pp. 413-420, 2019 (in Russian); doi: 10.25559/SITITO.15.201902.413-420
2. A. Y. L. Chong, E. Ch'ng, M. J. Liu, and B. Li, "Predicting consumer product demands via Big Data: the roles of online promotional marketing and online reviews," *International Journal of Production Research*, vol. 55, no. 17, pp. 5142–5156, 2017; doi: 10.1080/00207543.2015.1066519
3. A. Yu. Basalaeva, G. A. Gareeva, and D. R. Grigor'eva, "Web-scraping i klassifikatsiya tekstov metodom naive Bayes," *Innovation Science*, no. 5 (1), pp. 11–14, 2018 (in Russian).

4. R. Barbado, O. Araque, and C. A. Iglesias, “A framework for fake review detection in online consumer electronics retailers,” *Information Processing and Management*, vol. 56, no. 4, pp. 1234–1244, 2019; doi: 10.1016/j.ipm.2019.03.002
5. A. De Mauro, M. Greco, M. Grimaldi, and P. Ritala, “Human resources for Big Data professions: A systematic classification of job roles and required skill sets,” *Information Processing and Management*, vol. 54, no. 5, pp. 807–817, 2018; doi: 10.1016/j.ipm.2017.05.004
6. A. A. Korepanova and M. V. Abramov, “Application Of Random Forest In Choosing A Method For The Age Of A Social Network User Recovery,” *Artificial Intelligence and Decision Making*, no. 2., pp. 66–77, 2021 (in Russian); doi: 10.14357/20718594210207
7. V. D. Oliseenko, M. V. Abramov, and A. L. Tulupyev, “Identification of user accounts by image comparison: The phash-based approach,” *Scientific and Technical Journal of Information Technologies, Mechanics and Opticsthis*, vol. 21, no. 4, pp. 562–570, 2021; doi: 10.17586/2226-1494-2021-21-4-562-570
8. F. Bushmelev, M. Abramov, and T. Tulupyeva, “Adaptive method of color selection in application to social media images,” *CEUR Workshop Proceedings*, vol. 2782, pp. 252–257, 2020.
9. A. V. Toropova, “Collecting data about the latest episodes and the rate of posting on the social network Vkontakte,” in *Proc. Regional informatics and information security, St. Peterburg, Russia, 2020*, vol. 9, St. Peterburg, Russia: SPOISU, 2020, pp. 228–230 (in Russian).
10. V. F. Stoliarova and A. L. Tulupyev, “Regression Model for the Problem of Parameter Estimation in the Gamma Poisson Model of Behavior: An Application to the Online Social Media Posting Data,” in *Proc. of 2021 24th International Conference on Soft Computing and Measurements, SCM, 2021*, 2021 pp. 24–27; doi: 10.1109/SCM52931.2021.9507187
11. J. Seitz, “Building a Keyword Monitoring Pipeline with Python, Pastebin and Searx,” in *Bellingcat*, 21 Apr. 2017. [Online]. Available: https://www.bellingcat.com/resources/2017/04/21/building-keyword-monitoring-pipeline-python-pastebin-searx/
12. M. Kul’gin, “Site parsing. Russia and the world. What does one of the most useful tools look like from a legal point of view?,” in *vc.ru*, 30 Jul. 2019. [Online] (in Russian). Available: https://vc.ru/legal/64328-parsing-saytov-rossiya-i-mir-kak-s-tochki-zreniya-zakona-vyglyadit-odin-iz-samyh-poleznyh-instrumentov
13. “The last mil for internet trading,” in *logistics.datainsight*, 26 Aug. 2021. [Online] (in Russian). Available: http://logistics.datainsight.ru/poslednyaya-milya-dlya-internet-torgovli
14. D. Voroshilov, “Experts called the volume of orders executed by Russian couriers for the year,” in *RBC*, 24 Nov. 2021. [Online] (in Russian). Available: https://www.rbc.ru/business/24/11/2021/619cffaa9a79473c95fc20d0
15. A. Balster, O. Hansen, H. Friedrich, and A. Ludwig, “An ETA Prediction Model for Intermodal Transport Networks Based on Machine Learning,” *Business and Information Systems Engineering*, vol. 62, no. 5, pp. 403–416, 2020; doi: 10.1007/s12599-020-00653-0
16. Ü. Atak and Y. Arslanoğlu, “Machine learning methods for predicting marine port accidents: a case study in container terminal,” *Ships and Offshore Structures*, pp. 1–8, 2021; doi: 10.1080/17445302.2021.2003067
17. I. Jurdana, A. Krylov, and J. Yamnenko, “Use of artificial intelligence as a problem solution for maritime transport,” *Journal of Marine Science and Engineering*, vol. 8, no. 3, article 201, 2021; doi: 10.3390/jmse8030201
18. El. Al. Gal, “Web Scraping With Python: A Beginner’s Guide,” in *Brightdata*, 10 Sep. 2020. [Online]. Available: https://brightdata.com/blog/how-tos
19. NTA, “Web Parsing. Basics in Python,” in *vc.ru*, 27 Feb. 2020. [Online] (in Russian). Available: https://vc.ru/newtechaudit/109368-web-parsing-osnovy-na-python
20. A. Nagaikin, “Web Scrapin,” in *vc.ru*, 17 Feb. 2020. [Online] (in Russian). Available: https://habr.com/ru/post/488720/
21. A. Rawat, “Web Scraping in Node.js with Multiple Examples,” in *Tproger*, 29 Apr. 2017. [Online] (in Russian). Available: https://tproger.ru/translations/web-scraping-node-js/
22. A. Rawat, “Web Scraping in Node.js with Multiple Examples,” in *hackprogramming.com*, 08 Mar. 2017. [Online]. Available: https://hackprogramming.com/web-scraping-in-node-js-with-multiple-examples/
23. D. Ni, “The Ultimate Guide to Web Scraping with Node.js,” in *freecodecamp.org*, 08 Aug. 2018. [Online]. Available: https://www.freecodecamp.org/news/the-ultimate-guide-to-web-scraping-with-node-js-daa2027dcd3/
24. A. Jain, “How to Perform Web-Scraping using Node.js,” in *Bits and Pieces*, 25 Dec. 2018. [Online]. Available: https://blog.bitsrc.io/https-blog-bitsrc-io-how-to-perform-web-scraping-using-node-js-5a96203cb7cb
25. E. Levada, “Puppeteer: site parsing with JavaScript,” in *Proglib*, 16 Jun. 2020. [Online] (in Russian). Available: https://proglib.io/p/puppeteer-parsing-saytov-na-javascript-2020-06-16
26. M. Kumar, R. Bhatia, and D. Rattan, “A survey of Web crawlers for information retrieval,” *WIREs Data Mining Knowl Discov*, vol. 7, no. 6, p. e1218, 2017; doi: 10.1002/widm.1218
27. *Scrapy*, [Official Site], 2022. [Online]. Available: https://scrapy.org/
28. *Cloudflare*, [Official Site], 2022. [Online]. Available: https://www.cloudflare.com/

29. *ZennoPoster. Automate any tasks on the Internet*, [Official Site], 2022. [Online]. Available: https://zennolab.com/en/products/zennoposter/
30. "An overview of HTTP," in *developer.mozilla.org*, 04 Dec. 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview
31. "HTTP request methods," in *developer.mozilla.org*, 03 Oct. 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
32. "Evolution of HTTP," in *developer.mozilla.org*, 21 Nov. 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP
33. "How Google uses cookies," in *policies.google.com*, 2022. [Online]. Available: https://policies.google.com/technologies/cookies?hl=en
34. "General asynchronous programming concepts," in *developer.mozilla.org*, 22 Jan. 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Concepts
35. "Bugs. Happen. Debugging and Troubleshooting Reimagined," in *telerik.com*, 2022. [Online]. Available: https://www.telerik.com/fiddler
36. Kr. Lewandowski, "Growth in the Size of Unit Loads and Shipping Containers from Antique to WWI," *Packag. Technol. Sci.*, vol. 29, no. 8–9, pp. 451–478, 2016; doi: 10.1002/pts.2231
37. *Yandex.Cloud*, [Official Site], 2022. [Online]. Available: URL: https://cloud.yandex.com/en/

Received 20-08-2021, the final version — 23-09-2021.

Anastasia Korepanova, Junior Researcher, Laboratory of Theoretical and Interdisciplinary Problems of Informatics, SPC RAS, aak@dscs.pro

Fedor Bushmelev, Junior Researcher, Laboratory of Theoretical and Interdisciplinary Problems of Informatics, SPC RAS, ✉ fvb@dscs.pro

Artem Sabrekov, Junior Researcher, Laboratory of Theoretical and Interdisciplinary Problems of Informatics, SPC RAS, sabrekov2@gmail.com