Perbandingan Performa Tools Web Scraping pada Website dengan Data Statis dan Dinamis

Michael Levi, Henry Novianus Palit, Silvia Rostianingsih
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121-131, Surabaya 60236
Telp (031) – 2983455, Fax. (031) - 8417658
michael97levi@gmail.com, hnpalit@petra.ac.id, silvia@petra.ac.id

ABSTRAK

Dalam melakukan *scraping* suatu *website*, hal yang menjadi perhatian utama adalah tipe dari *website* tersebut apakah termasuk *website* statis atau dinamis, dan juga struktur data dari *website* tersebut. Dengan karakteristik *website* yang berbeda – beda dan *tools web scraping* yang beragam, akan membuat pengguna cukup kesulitan dalam memilih *tools* yang sesuai dengan kebutuhan.

penelitian ini bertujuan untuk membandingkan *tools web scraping* dalam *scraping* data dari karakteristik *website* yang berbeda, dan untuk memberikan rekomendasi *tools web scraping* untuk penelitian kedepannya dengan mengetahui *tools* yang tepat dalam menangani tiap karakteristik *website*.

Berdasarkan hasil pengujian yang sudah dilakukan, dapat disimpulkan *tools* mana yang lebih efektif dan efisien dalam kondisi – kondisi tertentu.

Kata Kunci: *Web Scraping, CURL, Scrapy, Cheerio, Headless Browser, Dynamic Web Content*

ABSTRACT

In scraping a website, the main concern is the type of website whether it is a static or dynamic website, and also the data structure of the website. With different website characteristics and diverse web scraping tools, it will make users quite difficult in choosing tools that suit their needs.

The purpose of this research is to compare web scraping tools from different website characteristics, and to provide recommendations for web scraping tools for future research by knowing the right tools in handling each website's characteristics.

Based on the results of tests that have been done, it can be concluded which tools are more effective and efficient in certain conditions.

Keywords: *Web Scraping, CURL, Scrapy, Cheerio, Headless Browser, Dynamic Web Content.*

1. PENDAHULUAN

Seiring perkembangan dari *World Wide Web*, pertumbuhan dari pengguna internet dan pertukaran data sangat tinggi. Banyak teknik – teknik baru yang diperkenalkan untuk meningkatkan fasilitas komputer dan jaringan dan berdampak pada pengurangan biaya pada perangkat keras dan biaya – biaya lain yang berkaitan dengan pengembangan suatu *website*. Penggunaan internet setiap hari menyebabkan data yang tersedia di internet sangat besar, peneliti, pengusaha, dan akademisi saling menyebarkan informasi di internet agar dapat menghubungkan informasi dengan orang dengan cepat dan mudah. Namun fenomena ini menimbulkan masalah baru yaitu bagaimana cara menangani data yang sangat banyak agar pengguna internet dapat mengakses informasi dengan mudah. Untuk menangani masalah ini para peneliti menemukan sebuah teknik yang dinamakan *web scraping* [9].

Web scraping merupakan teknik yang digunakan untuk mengekstrak data yang tidak terstruktur dari sebuah situs menjadi terstruktur. *Web scraping* banyak digunakan oleh pengguna internet dari berbagai bidang, di antaranya adalah untuk penelitian, *e-commerce*, dan juga pencari lowongan pekerjaan. Metode dari *web scraping* juga berkembang seiring dengan berkembangnya *World Wide Web*. Sejak tahun 2000, *Document Object Model* (DOM) menjadi populer pada *Dynamic* HTML, yang kemudian mengubah teknik dari HTML *Parsing* menjadi DOM *Parsing*. Contoh metode lainnya adalah *Application Programming Interface* (APIs). Metode ini merupakan metode yang paling baru diantara metode *web scraping* lainnya, pertumbuhan dari konten yang menyediakan API dari tahun 2005 hingga 2013 menurut "Programmable*Web*" yakni sebesar 10302. Beberapa metode *Web Scraping* lainnya di antaranya adalah, Manual *Parsing*, HTML *Parsing*, DOM *Parsing*, XPath, dan API [3].

Tidak hanya teknik *web scraping* yang berkembang, namun *tools* untuk *web scraping* juga banyak ditemukan dan dikembangkan. Mulai dari *tools* berupa aplikasi *desktop* yang siap pakai, untuk memberikan kemudahan pada pengguna dalam melakukan *scraping*, hingga *tools* yang berupa *library* untuk mendukung proses *scraping* yang terotomasi. *Tools web scraping* merupakan program yang digunakan untuk membantu proses *copy-paste* yang biasanya secara manual dapat dilakukan secara otomatis, dan juga mengumpulkan data dalam jumlah besar seperti informasi dari situs *real estate*, *directory*, dan situs lowongan pekerjaan dan disimpan pada *server* lokal. *Tools web scraping* bekerja sebagai *bot* atau *web crawler* untuk mengakses data dari *web* menggunakan protokol *Hypertext Transfer Protocol* (HTTP), atau dari *web browser*, dan mengekstrak data dan disimpan pada *database* lokal atau *file* JSON atau CSV untuk penggunaan data lebih lanjut [9].

Dalam melakukan *scraping* suatu *website*, hal yang menjadi perhatian utama adalah tipe dari *website* tersebut apakah termasuk *website* statis atau dinamis, dan juga struktur data dari *website* tersebut. Dengan karakteristik *website* yang berbeda – beda dan *tools web scraping* yang beragam, akan membuat pengguna cukup kesulitan dalam memilih *tools* yang sesuai dengan kebutuhan. Sebagai contoh untuk mengambil data dari *website* e-commerce dimana e-commerce termasuk dalam tipe *website* dinamis yang datanya dapat berubah – ubah dalam kurun waktu tertentu, pasti berbeda dengan mengambil data dari *website* statis seperti data profil perusahaan, artikel yang dipublikasikan secara online , dan lain sebagainya.

Penelitian tentang penggunaan *tools web scraping* pernah dilakukan oleh Ambre, Gaikwad, Pawar, & Patil, dengan judul penelitian "*Web* and Android Application for Comparison of E-Commerce Products". Penelitian ini bertujuan untuk mengambil data produk dari beberapa situs e-commerce berbeda dan membandingkan harga produk serta menampilkan spesifikasi tiap produk. *Tools* yang digunakan dalam penelitian ini adalah BeautifulSoup4 untuk mem-*filter* data yang didapat dan disimpan ke *database* [1].

Berdasarkan masalah yang disebutkan di atas, tujuan dari penelitian ini adalah untuk membandingkan *tools web scraping* dalam *scraping* data dari karakteristik *website* yang berbeda, dan untuk memberikan rekomendasi *tools web scraping* untuk penelitian kedepannya dengan mengetahui *tools* yang tepat dalam menangani tiap karakteristik *website*.

2. TINJAUAN PUSTAKA

2.1 Web Scraping

Web scraping merupakan suatu bentuk pengumpulan data dari internet secara terotomasi. Istilah dari *web scraping* bukan merupakan istilah yang baru, terdapat istilah – istilah lain yang biasa dikenal dengan *screen scraping*, *data mining*, *web harvesting*, dan lainnya. Secara teori, *web scraping* merupakan praktik pengumpulan data melalui cara apapun selain menggunakan program yang berinteraksi langsung dengan API. Secara praktik, *web scraping* mencakup teknik dan teknologi pemrograman secara luas, seperti analisis data dan keamanan informasi [7]. *web scraping* sangat baik digunakan untuk mengumpulkan dan memproses data dengan jumlah yang besar, seperti mengambil data analisa sentimen media sosial, produk – produk dari beberapa situs *e-commerce*, dan lain – lain [12].

2.2 Application Programming Interface

Application Programming Interface atau yang biasa disingkat API, merupakan sekumpulan kode pemrograman yang bertindak sebagai jembatan antara suatu aplikasi dengan aplikasi lainnya untuk saling berkomunikasi dan bertukar data. Selain sebagai jembatan antar aplikasi, API juga menyediakan lapisan keamanan antara *client* dengan *server*, dengan hanya mengirimkan paket data yang dibutuhkan dalam ukuran yang kecil [8]. Seiring berkembangnya waktu, parseAPI memiliki nilai jual yang dimanfaatkan oleh sebuah perusahaan, seperti Google, Amazon, eBay yang menjual service-API kepada siapapun yang hendak menggunakan layanan dari Google, Amazon, dan eBay secara lengkap.

2.3 JSON

JavaScript *Object* Notation atau JSON merupakan format pertukaran data yang ringan. Format JSON mudah bagi manusia untuk menulis maupun membaca. Format ini juga mudah untuk diurai dan dihasilkan oleh mesin. JSON merupakan bagian dari Bahasa pemrograman JavaScript. Secara umum, JSON merupakan format teks yang bebas yang dibuat dengan bentuk yang familiar bagi programmer Bahasa C, Java, Perl, Python, dan sebagainya. Hal ini membuat JSON merupakan format pertukaran data yang ideal. JSON dibuat atas 2 struktur data, yakni; 1. kumpulan *object*, atau *dictionary*, atau *associative array*. 2. Nilai data yang berurutan, atau yang biasa lebih dikenal dengan *array*, atau vector, atau *list* [5].

2.4 cURL

cURL merupakan *tool* command-line yang memanfaatkan sintaks URL untuk mengirim dan menerima data dan memiliki fitur yang beraneka ragam. Hampir semua jenis perangkat yang terhubung dengan internet dan melakukan pertukaran data melalui internet menggunakan cURL sebagai alat untuk saling mengirim dan menerima data. Beberapa fitur yang didukung cURL antara lain adalah, FTP/FTPS, HTTP/HTTPS, IMAP, LDAP, POP3, dan lain sebagainya [2].

2.5 DOM

Document *Object* Model berfungsi untuk menghubungkan halaman suatu *web* dengan sebuah script atau bahasa pemrograman dengan cara mewakili struktur sebuah dokumen, seperti HTML mewakili sebuah halaman *web* didalam *memory*. DOM mewakili sebuah dokumen dalam bentuk logical tree. Setiap cabang pada tree diakhiri dengan sebuah node, dan setiap node mengandung beberapa objek. Metode DOM mengijinkan akses untuk mengubah struktur dokumen, *style*, dan konten dari tree secara terprogram [6].

2.6 XPath

XPath adalah sebuah bahasa yang mendeskripsikan lokasi dari sebuah item pada dokumen XML dengan memetakan alamat item berdasarkan struktur logika atau hirarki dari sebuah dokumen XML. hal ini mempermudah dalam penulisan ekspresi pemrograman dari pada membuat tiap ekspresi harus memahami XML markup tertentu dan urutan XML dalam dokumen. XML membantu programmer untuk menangani dokumen dengan level abstraksi yang lebih tinggi [11].

2.7 Penelitian Sebelumnya

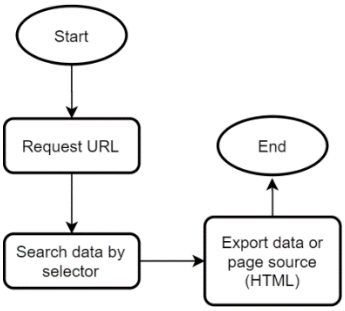
Aplikasi Aticket merupakan aplikasi yang membandingkan harga tiket pesawat dari beberapa situs *travel* yang ada di Indonesia [9]. Tiap situs *travel* memiliki daya tarik masing – masing, salah satunya adalah segi harga. Harga yang ditawarkan berbeda – beda. Banyaknya variasi harga ini menyebabkan pengguna kesulitan mencari harga yang sesuai dengan *budget*. Pengguna harus membuka *website travel* satu per satu untuk membandingkan harga tiket pesawat yang dicari. Aplikasi yang dibuat melakukan perbandingan harga tiket pesawat dari beberapa situs *travel*. Pertama, scanning data dari *server* menggunakan cURL dan mencari data yang diinginkan seperti harga tiket, nama maskapai, dan waktu keberangkatan. Setelah proses *scraping* selesai, data yang didapat disimpan kedalam *database* untuk ditampilkan dalam aplikasi Android. Pengujian yang dilakukan terhadap sistem adalah pengujian *scraping* harga ke masing – masing situs *travel*. Kedua, pengujian terhadap harga yang didapat dari hasil *scraping*, up-to-date atau tidak. Ketiga, pengujian terhadap tampilan aplikasi yang dibuat. Pengujian lainnya adalah pengujian terhadap data promo yang di-*scraping* dari masing – masing situs *travel* yang menyediakan [4].

3. ANALISA dan DESAIN SISTEM

3.1 Scraping dengan Selenium

Selenium *Web Driver* merupakan sebuah *tools* untuk menjalankan *browser* secara otomatis sesuai dengan perintah *user*. Selenium dapat diterapkan pada mayoritas *browser* seperti Firefox, Safari, Edge, Chrome, Internet Explorer, dan sebagainya [10]. Selenium dapat melakukan interaksi seperti yang dilakukan oleh *user* ketika menelusuri *web* seperti melakukan klik pada tombol, mengisi

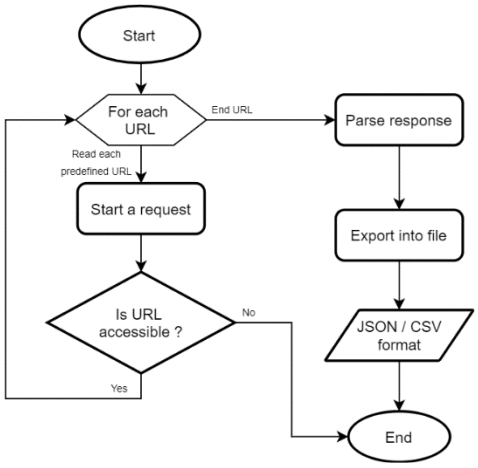
form, membuka tab baru, membuka halaman *web*, dan lain-lain. Karena Selenium merupakan sebuah *web driver* atau biasa disebut headless *browser* yakni, *Browser* tanpa GUI yang dapat diprogram, Selenium dapat mengeksekusi Javascript pada *website* yang menggunakan Javascript untuk memuat data atau *website* dinamis. Proses *scraping* menggunakan Selenium dapat dilihat pada Gambar 1.



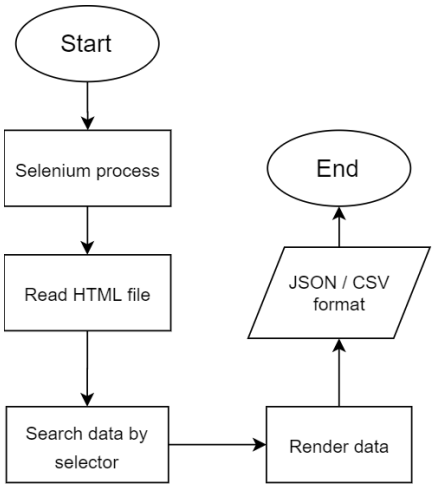
Gambar 1. *Flowchart* alur *scraping* menggunakan Selenium

3.2 Scraping dengan Scrapy

Penelitian dilakukan dengan menjalankan Scrapy untuk mengekstrak data dengan dua metode, yaitu melalui *Request* API dan mendapatkan data *response* berupa JSON, dan melalui DOM dan mendapatkan data berupa HTML dari halaman yang di-*scraping* kemudian di*filter* menggunakan XPath dari elemen yang ingin didapat. Terdapat beberapa proses yang dijalankan ketika melakukan *scraping* menggunakan Scrapy. Pertama, Scrapy akan membuat *crawler* bot lebih dari satu untuk melakukan *request* ke tiap URL yang telah dipersiapkan sebelumnya. Selanjutnya, *response* dari tiap – tiap URL akan di-*parsing* untuk mengambil data yang diinginkan, di dalam proses *parsing* juga terdapat proses *filter*ing data hasil *response*. Setelah proses *parsing* selesai, data hasil *parsing* akan di-*export* kedalam *file* dengan format JSON atau CSV untuk penelitian lebih lanjut. Alur kerja *scraping* menggunakan Scrapy melalui API dapat dilihat pada Gambar 2, dan Alur kerja *scraping* menggunakan Scrapy melalui DOM dapat dilihat pada Gambar 3.

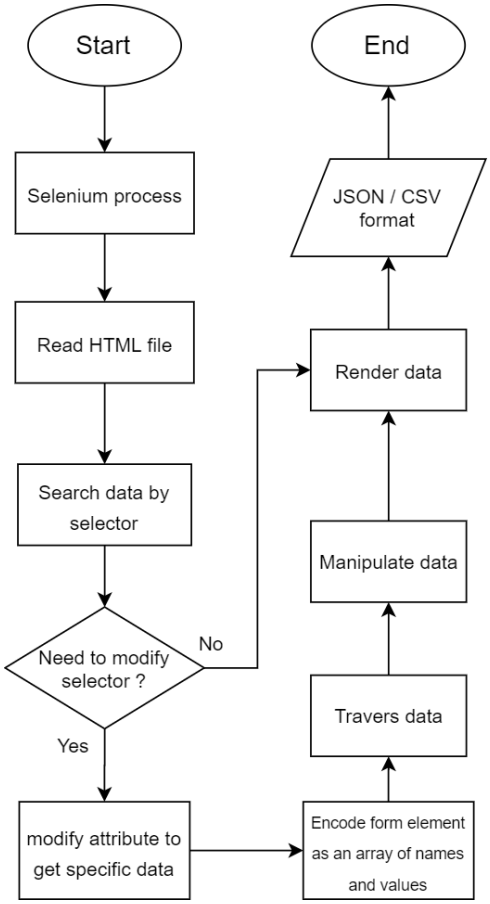


Gambar 2. *Flowchart* alur *scraping* menggunakan Scrapy melalui API



Gambar 3. *Flowchart* alur *parsing* menggunakan Scrapy melalui DOM

3.3 Scraping dengan Cheerio

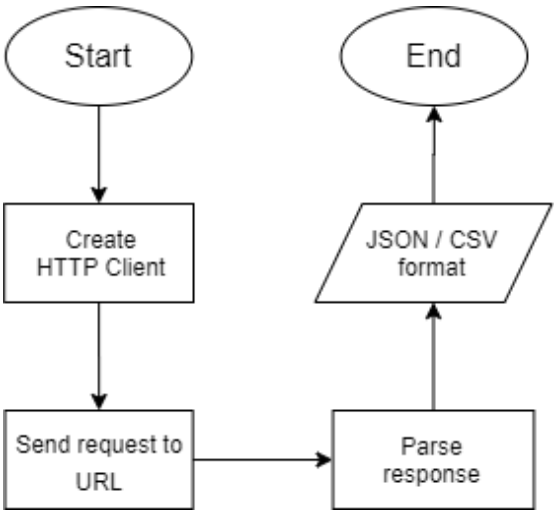


Gambar 4. *Flowchart* alur *parsing* dari Cheerio

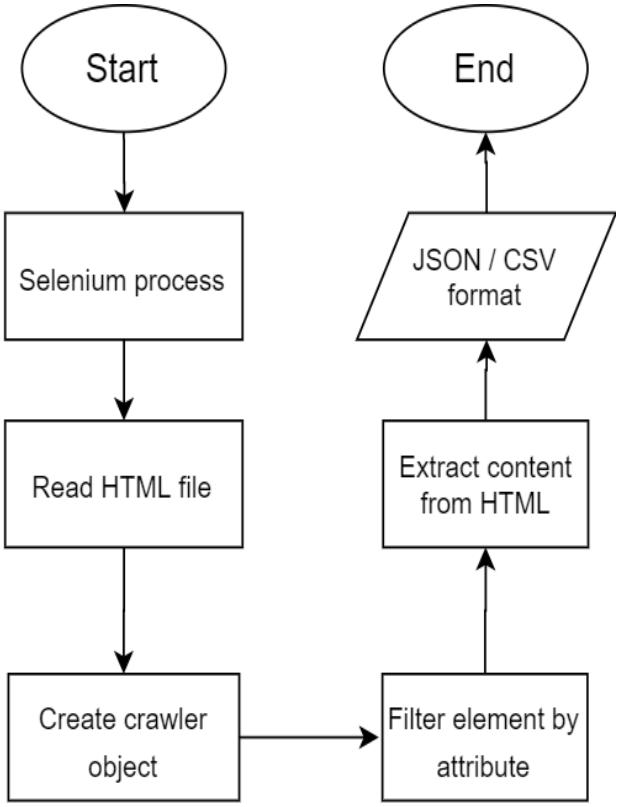
Pertama proses *request* pada *website* yang dituju dilakukan oleh Selenium yang dapat dilihat pada subbab 3.1, agar dapat memuat data yang menggunakan Javascript. Selanjutnya Selenium akan menghasilkan *file* HTML yang berisi *page source* dari tiap halaman yang dikunjungi. Selanjutnya Cheerio membaca isi *file* HTML. Kemudian, Cheerio melakukan *search* data menggunakan *selector*. *Selector* dan *context* dapat berupa sebuah *string expression*, elemen DOM, *array* elemen DOM, atau Cheerio *object*. Setelah proses *search* data selesai, terdapat pilihan untuk memanipulasi dokumen atau langsung mengekspor data yang didapat tanpa perlu manipulasi dokumen DOM. Apabila diperlukan modifikasi dokumen DOM lebih lanjut maka dilanjutkan dengan proses mendapatkan dan memodifikasi atribut untuk mencari data yang diinginkan secara spesifik. Selanjutnya proses *encoding* elemen *form* menjadi URL *query string* atau *array of names* dan *values*, sehingga lebih mudah untuk dicari. Selanjutnya proses *Traversing* atau menelusuri data hingga elemen *child* dari tiap elemen yang ada, setelah proses *traversing* selesai dilanjutkan dengan proses memanipulasi data atau mengubah struktur dari DOM. Berikutnya proses *rendering* data dilakukan untuk menghasilkan output hasil modifikasi kedalam bentuk format JSON atau CSV.

3.4 Scraping dengan Goutte

Alur *scraping* dari Goutte memiliki beberapa tahapan yaitu, pertama membuat HTTP *Client* untuk melakukan *request* ke setiap URL yang akan di-*scraping*. Setelah membuat *client*, Goutte akan melakukan *request* ke setiap URL yang sudah dipersiapkan sebelumnya. Selama proses *scraping*, dilakukan juga proses *filter*ing elemen HTML berdasarkan atribut menggunakan CSS *Selector*. Terakhir, mengekstrak konten dari halaman *web* yang sudah di*filter* sebelumnya dan disimpan kedalam *file* dalam format JSON atau CSV. Gambar 5 menjelaskan langkah – langkah *scraping* menggunakan Goutte melalui API. Gambar 6 menjelaskan langkah – langkah *scraping* menggunakan Goutte melalui DOM.



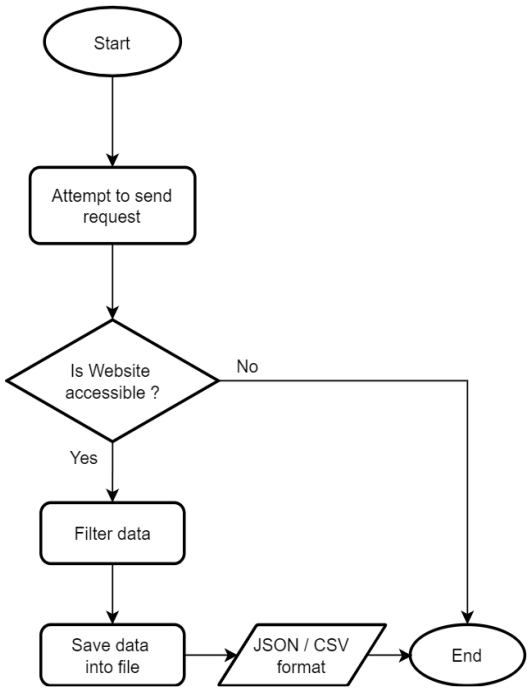
Gambar 5. *Flowchart* alur *scraping* dari Goutte melalui API



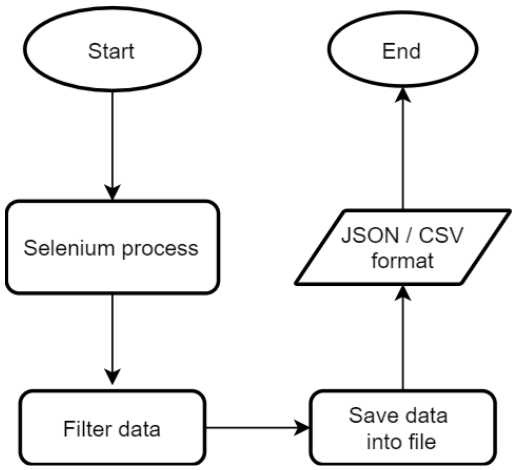
Gambar 6. *Flowchart* alur *parsing* menggunakan Goutte melalui DOM

3.5 Scraping dengan cURL

Untuk penggunaan dalam hal *web scraping*, protokol yang digunakan adalah HTTP, dimana *user* membuat *client* untuk mengirimkan *request* ke *website* yang dituju dan mendapatkan *response* dari *website* tersebut berupa JSON atau HTML. Semua pertukaran data berlangsung dengan menggunakan protokol HTTP. Setelah data didapat, dilakukan proses *filter*ing data untuk mendapatkan nilai data yang diinginkan. *Filter*ing data dapat dilakukan dengan menggunakan CSS *selector* yaitu mencari elemen dalam dokumen HTML yang memiliki atribut CSS yang sesuai dengan yang dicari. Selain menggunakan CSS *selector*, juga dapat menggunakan XPath, dimana XPath mem*filter* isi halaman *web* berdasarkan node – node yang ada pada dokumen XML. Cara yang terakhir dalam melakukan *filter*ing data dan bias disebut juga cara universal yang bisa diterapkan diberbagai *tools web scraping* yakni menggunakan Regular expression, yaitu *user* secara manual membaca dan menentukan pola dokumen dari hasil *response* yang didapat. Setelah proses *filter*ing data selesai, data yang telah di-*filter* akan disimpan ke dalam *file* dengan format JSON atau CSV untuk analisa lebih lanjut. Gambar 7 menjelaskan alur *scraping* menggunakan cURL melalui API. Gambar 8 menjelaskan alur *scraping* menggunakan cURL melalui DOM.



Gambar 7. *Flowchart* alur *scraping* dengan cURL melalui API



Gambar 8. *Flowchart* alur *scraping* dengan cURL melalui API

4. PENGUJIAN SISTEM

Proses pengujian terhadap cpu *usage*, *memory*, dan *time*, dilakukan dengan menggunakan *library* "psutil". *Library* ini memiliki fungsi untuk mengambil data cpu *usage* secara general maupun tiap *core*, dan juga memiliki fungsi lain seperti mengukur penggunaan *memory*, mengukur *disk usage*, dan sebagainya. Fungsi yang digunakan dalam pengujian adalah fungsi untuk mengambil data persentase cpu, dan fungsi untuk mengambil data persentase *memory* selama proses *scraping* berlangsung. Untuk mengukur waktu yang dibutuhkan selama proses *scraping*, menggunakan *library* "time" yang akan mencatat waktu ketika program dijalankan. Pengujian dilakukan dengan mengukur *resource* komputer ketika dalam kondisi *idle* selama 1 menit kemudian diambil rata – rata. Setelah itu dilakukan pengukuran *resource* komputer ketika proses *scraping* dijalankan hingga selesai, kemudian diambil rata – rata. Kemudian data *resource* komputer ketika proses *scraping* dijalankan, dikurangkan dengan data *resource* komputer ketika *idle* sehingga didapatkan total kenaikan *resource* komputer selama proses *scraping* berlangsung.

Tabel 1. Perbandingan efisiensi *scraping* melalui JSON API

		Scrapy	Goutte	cURL
		API	API	API
Tokopedia	CPU (%) [stdev]	13.56 [4.54]	9.34 [2.21]	10.44 [3.75]
	Memory (%)[stdev]	0.35 [0.15]	0.19 [0.36]	0.12 [0.04]
	Time (s)	12.74	19.99	34.89
	Speed (item/s) [stdev]	14.01 [9.06]	6.31 [1.41]	4.08 [1.55]
OVO	CPU (%)[stdev]	8.46 [2.89]	5.92 [0.89]	4.69 [1.38]
	Memory (%)[stdev]	0.8 [0.32]	0.19 [0.20]	0.22 [0.11]
	Time (s)	65.62	127.69	193.81
	Speed (item/s) [stdev]	3.77 [1.45]	1.77 [0.59]	1.12 [0.29]
Airpaz	CPU (%)[stdev]	0.76 [1.26]	3.80 [4.31]	0.12 [0.38]
	Memory (%)[stdev]	0.23 [0.14]	-0.03 [0.28]	-0.03 [0.11]
	Time (s)	32.04	35.65	50.12
	Speed (item/s) [stdev]	1.08 [0.39]	5.36 [9.15]	0.87 [0.4]

Tabel 1 menampilkan rata – rata data perbandingan penggunaan CPU, *memory* dan waktu yang diperlukan selama proses *scraping* menggunakan JSON API. Pengujian tiap *tools* dilakukan sebanyak 5 kali dan diambil rata - rata. Pada situs Tokopedia dan OVO, *tools* Scrapy lebih unggul dalam segi kecepatan mengambil data, namun menggunakan CPU paling banyak daripada *tools* yang lain. Data yang didapat tiap *request* dapat berbeda dengan *request* sebelumnya, sehingga dilakukan komparasi kecepatan mengambil data tiap *tools*. Untuk aspek *memory* tidak mengalami perubahan yang signifikan dari kondisi *idle* dan ketika *scraping*. Pada situs Airpaz, penghitungan CPU *usage* tidak menentu, dikarenakan ketika proses *scraping*, tiap *tools* perlu menunggu situs Airpaz selesai *request* ke tiap maskapai, dan ketika menunggu, CPU *usage* dihitung seperti kondisi *idle*, sehingga ketika dihitung rata – rata, selisih CPU *usage* pada saat *scraping* dapat lebih rendah daripada ketika pengujian kondisi *idle*. waktu yang diperlukan selama proses *scraping* dengan *tools* yang sama dapat berbeda antara *request* pertama dan *request* kedua. Hal ini dikarenakan *server* dari situs Airpaz ketika melakukan *request* ke *server* tiap maskapai terkadang mengalami gangguan.

Tabel 2. Perbandingan efisiensi *scraping* melalui DOM

		Scrapy	Cheerio	Goutte	cURL	Selenium
		DOM	DOM	DOM	DOM	DOM
Toko-pedia	CPU (%) [stdev]	18.78 [3.3]	16.17 [2.09]	15.51 [2.81]	15.43 [1.84]	18.2 [3]
	MemOry (%) [stdev]	3.15 [0.73]	4.15 [0.31]	3.93 [0.35]	3.87 [0.64]	3.63 [0.33]
	Time (s)	2167	2196	2266	2301	1910
	Speed (item/s) [stdev]	0.06 [0]	0.05 [0]	0.05 [0.01]	0.05 [0.01]	0.06 [0]
OVO	CPU (%) [stdev]	11.69 [3.65]	13.18 [1.64]	13.51 [0.62]	11.94 [2.63]	12.1 [1.72]
	Memory (%) [stdev]	2.73 [0.19]	2.3 [0.33]	2.22 [0.24]	2.14 [0.3]	3.06 [0.82]
	Time (s)	672	653	648	663	756
	Speed (item/s) [stdev]	0.3 [0.03]	0.3 [0.03]	0.29 [0.03]	0.28 [0.08]	0.16 [0.01]
Airpaz	CPU (%) [stdev]	42.71 [4.94]	42.76 [7.42]	42.4 [4.27]	42.77 [4.4]	46.55 [4.26]
	Memory (%) [stdev]	2.38 [0.1]	2.58 [0.15]	2.53 [0.17]	2.46 [0.05]	3.77 [0.57]
	Time (s)	57	54	56	55	66
	Speed (item/s) [stdev]	0.56 [0.1]	0.59 [0.1]	0.55 [0.1]	0.56 [0.08]	0.44 [0.07]

Tabel 2 menampilkan data perbandingan penggunaan CPU *memory* dan waktu yang diperlukan selama proses *scraping* menggunakan DOM. Situs Tokopedia, OVO, dan Airpaz menggunakan Javascript untuk memuat data, sehingga dapat digolongkan sebagai *website* dinamis. Sedangkan *Tools* Scrapy, Cheerio, Goutte, dan cURL tidak dapat mengeksekusi Javascript ketika melakukan *scraping*, sehingga tidak memungkinkan untuk melakukan *scraping* melalui DOM. Berbeda dengan Selenium, Selenium merupakan *web driver* yang dapat melakukan interaksi seperti user seperti membuka alamat URL, membuka *browser tab*, meng-klik *link*, dan lainnya, dan juga Selenium dapat mengeksekusi Javascript ketika proses *scraping*, sehingga memungkinkan untuk mengekstrak data dari DOM. Dalam pengujian pada Tabel 5.2, *tools* Scrapy, Cheerio, Goutte, dan cURL, digabungkan dengan proses Selenium untuk menghasilkan *file* HTML dari tiap halaman yang akan di-*scraping*, yang kemudian akan di-*parsing* oleh masing – masing *tools*. Data CPU *usage*, *memory*, dan waktu diukur dari Proses Selenium berjalan hingga proses *scraping* menghasilkan *file* data hasil *scraping* dengan format CSV. Proses *scraping* melalui DOM untuk tiap *tools* memiliki kecepatan *scraping* yang sama dalam tiap *scraping*, hal ini dibuktikan dengan standar deviasi untuk kecepatan yang mendekati 0. Penggunaan CPU pada situs Airpaz sangat tinggi dibandingkan *tools* yang lain, karena ketika *scraping* situs Airpaz, *selector* dari selenium yang mencari elemen "*loading bar*" dan melakukan pengecekan secara terus – menerus hingga elemen "*loading bar*" hilang / ajax berhasil dimuat.

5. KESIMPULAN

Dari hasil pengujian sistem yang telah dilakukan, dapat diambil beberapa kesimpulan, diantara lain:

- Penggunaan metode *scraping* melalui JSON API sangat efisien dalam segi waktu. Contoh pada situs Tokopedia dengan mengambil 120 data, *scraping* menggunakan tools cURL melalui API membutuhkan waktu rata-rata 34 detik, sedangkan *scraping* melalui DOM membutuhkan waktu rata-rata 38 menit 20 detik.
- Scrapy terkadang membutuhkan waktu beberapa saat sebelum melakukan *scraping*. Hal ini dikarenakan ekstensi *"AutoThrottle"* dari Scrapy yang menyesuaikan kecepatan *scraping* dengan *server* Scrapy dan *server website* yang akan di-*scraping*.
- Perlu menggunakan *Network Log* sebagai acuan untuk mendapatkan data dari *server* situs, apabila data tidak tersedia pada *page* HTML atau DOM.
- *Scraping* melalui DOM dapat mengambil data yang tidak terdapat pada API namun kurang efisien dalam segi waktu karena diperlukan waktu untuk menunggu tiap halaman memuat Javascipt. Contoh pada situs Tokopedia, untuk men-*scraping* 1 produk diperlukan waktu sekitar 15 detik.
- Terdapat perbedaan data yang diperoleh dari *scraping* melalui API dan *scraping* melalui DOM, hal ini dikarenakan ada beberapa data yang tidak terdapat pada API situs tersebut seperti "Deskripsi produk" pada situs Tokopedia, dan ada data yang hanya ditampilkan di API situs seperti "*Redeemed* Promo" pada situs OVO karena hanya untuk keperluan sistem *backend*.
- Situs Airpaz kurang stabil, ketika melakukan *scraping* data yang didapat dari request pertama dapat berubah pada request kedua. Hal ini dikarenakan request yang dilakukan situs Airpaz ke *server* tiap maskapai yang terkadang mengalami gangguan.
- Terdapat kemungkinan bahwa data yang didapat dari hasil *scraping* yang telah dibuat tidak sama dengan situs aslinya terutama pada situs Airpaz. Hal ini karena perubahan data

pada situs Airpaz sangat cepat hanya dalam hitungan beberapa jam.

- *Tools* Cheerio merupakan HTML *parser* dan tidak dapat melakukan *request*. Untuk melakukan *request*, Cheerio membutuhkan *library* lain. Cheerio juga tidak dapat menerima data *response* berupa JSON, sehingga pada penelitian ini, *tools* Cheerio hanya diuji pada *scraping* melalui DOM.

6. DAFTAR PUSTAKA

[1] Ambre, A., Gaikwad, P., Pawar, K., & Patil, V. 2019. Web and Android Application for Comparison of E-Commerce Products. *International Journal of Advanced Engineering, Management and Science (IJAEMS) [Vol-5, Issue-4, Apr-2019]*, 266-268. URI= http://d.researchbib.com/f/3jnJcuMJ1mYzAioF91pTkiLJEsnJ1uM2ImY2ymp3IyK2McoTImYmHgFHcOEH1GYHSDHv0lZQR5YGVgI2IvLJ5xYaOxMt.pdf.

[2] curl. *command line tool and library*. URI= https://curl.haxx.se/.

[3] Draxl, V. 2018. *BACHELOR PAPER Web Scraping Data Extraction from websites*URI= https://www.academia.edu/35901535/BACHELOR_PAPER_Web_Scraping_Data_Extraction_from_websites.

[4] Irawan, B., Palit, H. N., & Andjarwirawan, J. 2018. Aplikasi Android untuk Mencari Harga Tiket Pesawat Termurah dari Beberapa Situs Travel di Indonesia. *Jurnal Infra VOL 7, NO 2 (2019)*, 49-54. URI= http://publication.petra.ac.id/index.php/teknik-informatika/article/view/8752/7900.

[5] json. *Introducing JSON*. URI= https://www.json.org/json-en.html.

[6] MDN Web Docs. 2020. *Document Object Model (DOM)*. URI= https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model.

[7] Mitchell, R. 2015. *Web Scraping with Python.* Sebastopol: O'Reilly Media, Inc.

[8] MuleSoft. *What is an API? (Application Programming Interface)*. URI= https://www.mulesoft.com/resources/api/what-is-an-api.

[9] Saurkar, A. V., Pathare, K. G., & Gode, S. A. 2018. An Overview on Web Scraping Techniques and Tools. *International Journal on Future Revolution in Computer Science & Communication Engineering Volume: 4 Issue: 4*, 363-367. URI= http://www.ijfrcsce.org/download/browse/Volume_4/April_18_Volume_4_Issue_4/1524638955_25-04-2018.pdf.

[10] Selenium. *Selenium Projects*.URI= https://www.selenium.dev/projects/.

[11] TechTarget. 2005. *XPath*. URI= https://whatis.techtarget.com/definition/XPath.

[12] *what-is-web-scraping*. 2019. URI= https://hirinfotech.com/what-is-web-scraping/.