UNIVERSITAS KRISTEN MARANATHA

Volume 5 | Nomor 3 | Desember 2019

JuTISI

Jurnal Teknik Informatika dan Sistem Informasi

9 772443 221007 9 772443 222004

p-ISSN: 2443-2210
e-ISSN: 2443-2229
Halaman 281-437

Ekstraksi dan Analisis Produk di *Marketplace* Secara Otomatis dengan Memanfaatkan Teknologi *Web Crawling*

http://dx.doi.org/10.28932/jutisi.v5i3.1977

Edward Hanafi Fernando [#1], Hetthroh Sagala [*2], Ariel Elbert Budiman [#3],
Ivan Nathaniel Husada[#4], Hapnes Toba [✉#5]
[#]*Program Studi Magister Ilmu Komputer, Fakultas Teknologi Informasi*
Universitas Kristen Maranatha
Jl. Prof. drg. Surya Sumantri No.65 Bandung
[1]mi1879004@student.it.maranatha.edu
[2]mi1879014@student.it.maranatha.edu
[3]mi1879009@student.it.maranatha.edu
[4]mi1879008@student.it.maranatha.edu
[5]hapnestoba@it.maranatha.edu

Abstract — **Along with the advancement in technology, today's community begins to abandon conventional shopping methods where buyers must come to the seller's shop. Nowadays the community mostly doing online shopping because the process is considered more convenience. Because of this, there are more and more online marketplace users. Much more data can be retrieved with the increasing number of online marketplace users. Because of the large amount of data, the process for extracting the data so that it can be seen and utilized becomes possible. The purpose of this study is to show data and extraction methods from an online marketplace system so that the results can be visualized and users can analyze the data. The data extraction methods that will be used are web crawling and web scraping where after the data is successfully extracted and cleaned it will be visualized with the Power BI application. The experiments show that the methods are useful to conduct analysis.**

Keywords— ***data visualize; online marketplace; web crawling; web scraping*.**

I. PENDAHULUAN

Online Marketplace dewasa ini mengalami perkembangan yang sangat pesat. Dengan dukungan dari semakin pesatnya perkembangan dari *mobile website* diperkirakan *online marketplace* akan bertumbuh dengan cepat pada tahun-tahun mendatang [14]. *Marketplace* yang awalnya ada di forum forum *online* kini sudah memiliki sebuah sistem terpusat dimana para pengguna internet dapat memasukan data barang jualannya untuk nantinya akan dilihat oleh calon konsumen. Tidak hanya sebagai penyedia fitur untuk menemukan barang, *online marketplace* telah memiliki fitur pembayaran yang dapat melayani transaksi antara calon pembeli dengan para penjual. Perkembangan yang pesat ini menjadikan *online marketplace* salah satu alternatif yang menjadi favorit dalam melakukan transaksi jual beli baik bagi para penjual dan pembeli.

Selain informasi mengenai barang pada *online marketplace* sekarang ini menampilkan beberapa informasi seperti jumlah barang terjual, info mengenai kapan seorang penjual bergabung, ranking dari penjual level dari seorang penjual, rating, testimoni seorang penjual dan banyak lagi. Selain mengenai penjual pada *online marketplace* pun menyediakan informasi barang yang dijual oleh setiap penjual.

Melalui halaman-halaman pada *online marketplace* tersebut dapat mengumpulkan data - data untuk membentuk suatu informasi yang berguna. Namun untuk mengumpulkan variasi data tersebut bukanlah hal yang mudah. Banyaknya jumlah dari penjual dan ragam barang yang terdapat pada web tersebut menyebabkan kesulitan untuk mengumpulkan data yang diperlukan. Diperlukan sebuah cara atau sistem yang secara otomatis dapat mengumpulkan data dari setiap halaman yang menampilkan atau data yang disediakan pada halaman web tersebut.

Selain pengumpulan data, hal lain yang harus diperhatikan adalah makna data. Pada sebuah halaman data terdapat data yang cukup banyak. Agar data data tersebut dapat berguna, perlu dirancang sebuah tempat pengelolaan data dimana pada tempat tersebut variasi data akan disimpan. Data harus dipilih dan disusun dengan baik agar ketika penggunaan dapat memberi arti yang baik dan dapat menunjang di setiap kegiatan yang diperlukan.

II. KAJIAN LITERATUR

Berdasarkan pada pernyataan diatas, maka pada riset ini membutuhkan beberapa metode dan *tools* dalam pengerjaannya. Metode yang digunakan adalah *web scrapping*, dan *web crawling* sementara *tools* yang digunakan adalah *Scrappy*, *Splash*, dan *Power BI*.

A. Web Scraping

Web Scraping disebut juga *web harvesting*, *web data extraction* atau *web mining*, yaitu kegiatan mengkonstruksi agen untuk mengunduh, mengurai, dan mengatur data dari halaman *web* secara otomatis. Hal ini dimaksudkan kegiatan pengguna dalam mengambil data dari layar *web* dilakukan oleh *web scraper* dengan tujuan agar pengambilan data dapat dilakukan dengan lebih cepat dan lebih tepat.

Web crawler melintasi halaman-halaman *web* untuk diekstraksi oleh *Web scrapin*g. Saat pengambilan data dari halaman *web*, cukup sulit untuk menentukan halaman *web* yang cukup relevan. Dengan *web crawler*, pengunjungan halaman *web* dapat menggunakan *local search algorithm* untuk membatasi URL antara halaman *web* [3].

Dengan *web scraping*, pengguna dapat mengambil data - data mentah yang ada di *web page* seperti tabel pada halaman Wikipedia, atau *review* sebuah film untuk melakukan *text mining*, daftar *real - estate* untuk *geolocation*, membangun *data set* dari penjualan atau *forecasting*, analisa jejaring sosial, atau bahkan memantau berita atau topik terbaru di internet.

Lalu bagaimana dengan API yang disediakan oleh *web* tertentu untuk memuat data yang ada pada *website* tersebut? Twitter dan Amazon adalah salah satu contoh *web* yang menyediakan layanan API dimana dapat mengakses API tersebut untuk mendapatkan data - data yang diperlukan. Namun pemanfaatan API memiliki beberapa kekurangan seperti [2] :

1. Tidak semua *website* memberikan fasilitas API.
2. Fasilitas API yang diberikan umumnya tidak gratis.
3. Penggunaan API tersebut dibatas umumnya seberapa sering pengguna dapat mengakses API tersebut.
4. API tidak memberikan data secara keseluruhan.

Sedangkan dengan menggunakan *web scrapin*g, apapun yang pengguna dapat lihat pada *web browser* dapat mengambil data tersebut kemudian menyimpan dan membersihkannya dengan baik.

B. Web Crawling

Web Crawling memiliki peran yang berbeda dengan *web scraping* kendati sama - sama digunakan pada halaman *web*. Sebuah *web crawler* bertugas untuk berpindah-pindah di antara halaman *web* secara otomatis selama *web scraping* bekerja. *Crawling* adalah aktifitas dalam membuat salinan dalam porsi yang relevan yang diambil dari sebuah *World Wide Web*[10] dan *web crawling* adalah tahap terpenting dalam pencarian informasi *web*[13]. Dalam membangun sebuah *web crawler* perlu memperhatikan beberapa hal yang dapat mempengaruhi cakupannya seperti *Web crawler* terbatas pada sebuah halaman yang sudah terstruktur dengan baik, dimana pengguna bisa tahu apa yang ada pada halaman tersebut [8].

Karena pengguna dan konten dari *World Wide Web* semakin meningkat dengan pesat sehingga dibutuhkan metode untuk mencari dan mengambil data dengan cepat [11]. Karena itu *web crawling* adalah metode yang penting untuk mengumpulkan data [12]. Terkadang diperlukan membuat tiruan dari halaman *web* supaya dapat mengambil data pada halaman tersebut. Kasus ini biasanya terjadi ketika berhadapan dengan halaman yang menggunakan *javascript*.

Membangun sebuah *web crawler* memerlukan perhatian khusus karena tidak menginginkan *web crawler* menemui jalan buntu atau proses yang sangat lama. Beberapa contoh hal yang perlu diperhatikan adalah harus mengetahui halaman - halaman *web* apa saja yang sudah dikunjungi sebelumnya, data apa saja yang ingin diambil, lalu bagaimana cara penyimpanan data tersebut. Berikut beberapa hal yang perlu diperhatikan dalam pembuatan *web crawler*:

1. Pastikan data yang akan diambil, apakah halaman tersebut memiliki data - data yang diperlukan.
2. Gunakan *dataset*, hal ini dapat dilakukan dengan menggunakan media penyimpanan data seperti *database*, jangan lupa memberikan penanda waktu pada data tersebut sehingga tahu apabila ada data baru yang masuk.
3. Pisahkan *crawling* dan *scraping*.
4. Apabila *web crawler* tidak menemukan data yang diinginkan setelah beberapa saat, ada baiknya untuk menghentikan proses.
5. Ketika terjadi kesalahan pada saat proses, perlu memperhatikan apa perlu untuk mencoba ulang proses atau mengakhirinya.
6. Perhatikan antrian saat melakukan proses *crawling*, jika melakukan proses terlalu cepat dapat menemukan kegagalan karena proses yang terlalu cepat. Taruh jeda waktu di antara antrian.
7. Perhatikan aspek legal ketika melakukan *scraping*. Beberapa situs tidak menginginkan halaman *web*nya dibombardir dengan banyaknya HTTP request.

C. Scrapy

Scrapy adalah sebuah *web framework* yang kuat untuk mengambil data dari berbagai sumber [9]. Sumber data yang dimaksud berupa situs *web* dan *scrapy* akan mengekstraksi data terstruktur yang dapat digunakan untuk berbagai aplikasi yang bermanfaat, seperti data mining, pemrosesan informasi atau arsip sejarah. *Scrapy* pada awalnya dirancang untuk melakukan *web crawling* tapi pada saat ini *Scrapy*

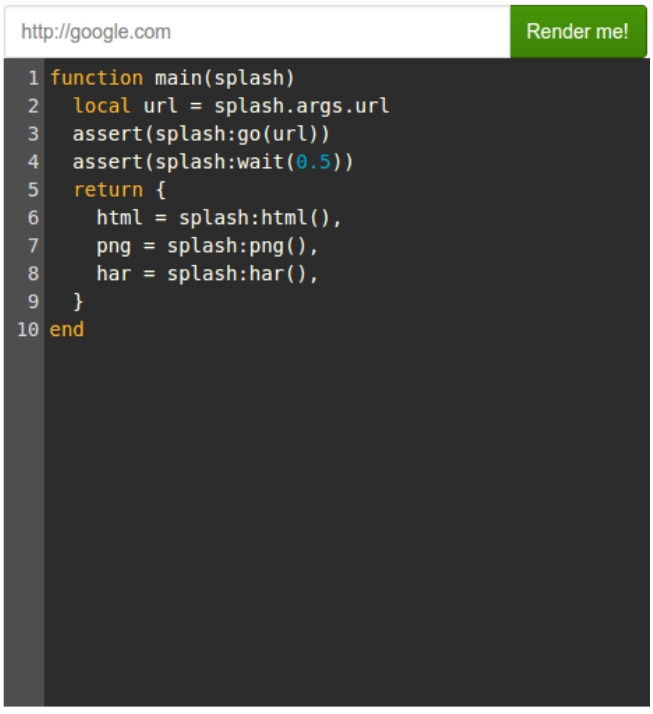
p-ISSN : 2443-2210
e-ISSN : 2443-2229

juga dapat digunakan untuk mengekstraksi data menggunakan API (seperti *Amazon Associates Web Services*) atau sebagai *web crawler* dengan tujuan lainnya. Gambar 1 merupakan tampilan terminal dari penggunaan *scrapy*[5].

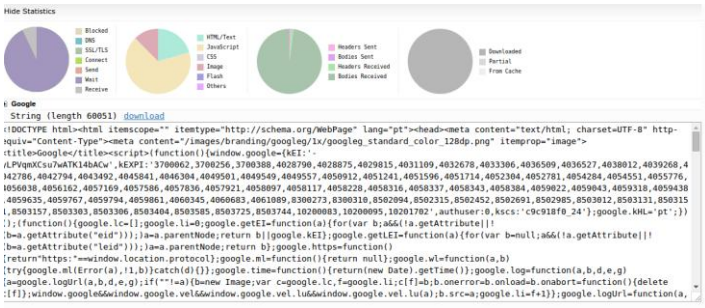
Gambar 1. *Terminal* dari *Scrapy*

D. *Splash*

Splash adalah layanan untuk melakukan *rendering* pada *javascript website* yang diimplementasikan dalam *Python* 3. *Scrapy* sebagai *web crawler* memiliki kelemahan ketika akan melakukan ekstraksi pada halaman *web* yang berbasis *javascript*. Hal ini disebabkan karena halaman pada *web* berbasis *javascript* atau *ajax* tidak mengunggah data secara keseluruhan sehingga menyebabkan *scrapy* mengekstrak data secara premature. Gambar 2 merupakan tampilan terminal dari *Splash*. Dan gambar 3 merupakan contoh hasil response dari penggunaan *Splash* [6].



Gambar 2. *Terminal* dari *Splash*



Gambar 3. Contoh hasil *response* dari *splash*

Untuk mengatasi masalah ini *splash* akan digunakan sebagai perantara dalam proses memuat halaman *web*. Dalam prosesnya *splash* bertindak sebagai aplikasi yang memuat halaman *web* terlebih dahulu secara keseluruhan, kemudian *web* yang sudah dimuat tersebut akan di ekstrak oleh *scrapy*. Oleh karena itu, *scrapy* bisa mendapatkan data halam secara keseluruhan walaupun halaman *web* tersebut merupakan halaman *web* berbasis *javascript.*

E. *Power BI*

Power BI adalah kumpulan *tools* analisis bisnis dari *Microsoft* untuk menganalisis data dan membuat suatu *insight* dari data tersebut dalam bentuk suatu laporan dan *dashboards*. Data pengguna dalam berbagai bentuk seperti *spreadsheet*, *file* teks, *database*, dll adalah *input* untuk *Power BI*, lalu data ditransformasikan sesuai kebutuhan bisnis menjadi *dataset*. Setelah *dataset* siap, laporan bisa dibuat dengan cara memvisualisasikan dengan berbagai elemen yang disediakan oleh *Power BI*. *Power BI* juga menyediakan fitur *Quick Insights* dimana fitur ini menggunakan berbagai algoritma untuk menganalisis data dan membuatkan laporannya secara otomatis. Setelah laporan-laporan tersebut sudah selesai, dapat ditampilkan dalam suatu *dashboards* yang berisi berbagai informasi-informasi penting bagi penggunanya [1]. Gambar 4 merupakan contoh visualisasi data dengan *Power BI* [7].

Gambar 4. Contoh Visualisasi Data dengan PowerBI

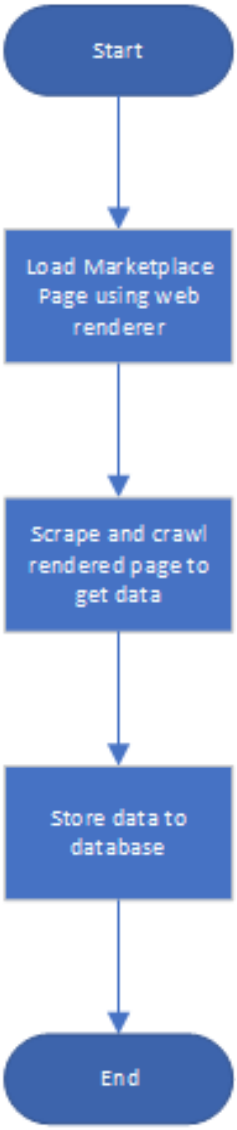
F. *Data Analytics*

Data Analytics adalah ilmu tentang mengekstraksi nilai atau insight dari sebuah sumber data dimana data harus terlebih dahulu diproses sedemikian rupa sebelum menjadi *insight* yang berharga. *Data Analytics* sebagai dalam organisasi seringkali digambarkan sebagai alur kerja yang menggambarkan langkah-langkah yang harus dilakukan dalam proyek analisis data, baik itu pembangunan suatu model prediksi, seperti pelanggan mana yang bereaksi positif terhadap suatu kampanye pemasaran, segmentasi pelanggan, atau sekadar pembuatan otomatis laporan berkala yang mencantumkan beberapa statistik deskriptif [1]. Dalam proses analisis data, *web scraping* mempunyai peran dalam mengidentifikasi dan menyeleksi sumber data untuk digunakan dalam membuat suatu model. Dalam kasus tertentu, *web scraping* dapat menjadi komponen penting dalam proyek *data analytics* dimana saat diperlukan statistik dasar dan visualisasi dari hasil data *scraping*. Dalam *web scraping* perlu memperhatikan *data quality*, khususnya jika berurusan dengan *World Wide Web*, dimana banyak sekali data yang tidak terstruktur dan berantakan, sehingga diperlukan banyak sekali *data cleaning* dan *fail-safe* dalam program *scraper* yang dibuat.

III. SUMBER DATA DAN METODOLOGI

Metodologi Pengumpulan data akan dilakukan dengan membangun sebuah aplikasi yang mengimplementasikan metode *web scraping* dan *web crawling*. Dengan data dari *marketplace* "T" yang merupakan salah satu *marketplace* terbesar dan juga mempunyai user aktif terbanyak dan memakai input produk *smartphone* "g" sebagai input pencariannya diharapkan hasil data yang dapat diambil akan dapat divisualisasi sehingga hasil dari visualisasi tersebut akan dapat digunakan sebagai bahan penelitian untuk menentukan kota mana di Indonesia yang merupakan pasar terbesar dari produk *smartphone* "g" tersebut.

Gambar 5 menunjukan *flowchart* langkah kerja dimana proses penarikan data diawali dengan memuat *web page marketplace* dengan menggunakan *splash*. Setelah *web page* berhasil dimuat kemudian *script python* akan menarik data hasil render untuk disimpan ke dalam *database*. Sumber data untuk proyek kali ini akan diambil dari web *online marketplace* yang ada di Indonesia dengan kategori item *smartphone* merk tertentu.



Gambar 5. Flowchart Langkah Kerja

A. *Marketplace "t"*

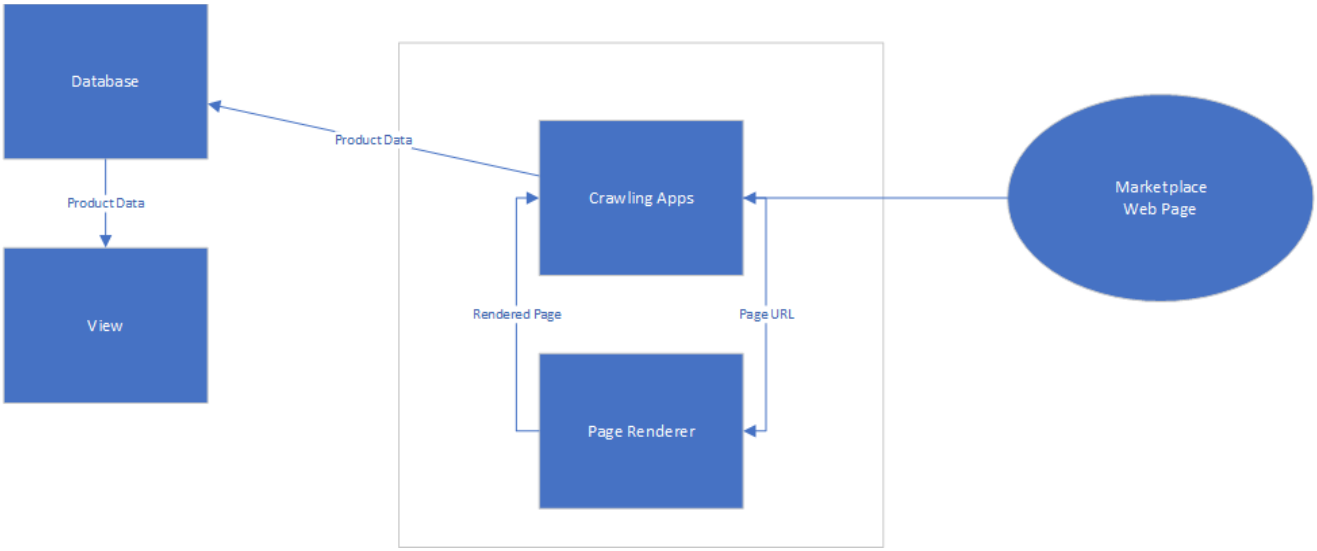
Marketplace "t" adalah salah satu *marketplace* terbesar di Indonesia. Selain itu *marketplace* tersebut juga merupakan salah satu *marketplace* dimana masyarakat Indonesia banyak menggunakannya untuk transaksi barang dengan kategori *Gadget*. Diharapkan dengan dipilihnya *marketplace* tersebut sebagai sumber data, data yang diambil akan lebih *valid* dan lebih dapat dipercaya. Selain itu penulisan-penulisan dalam metode pencarian yang terdapat pada halaman *marketplace* tersebut juga dianggap lebih dapat dimengerti sehingga akan memberikan kenyaman dalam proses pengambilan data.

B. *Produk Smartphone "g"*

Smartphone "g" adalah nama produk *smartphone* dengan banyak peminat. Produk-produk dari *smartphone* "g"

p-ISSN : 2443-2210
e-ISSN : 2443-2229

memiliki variasi harga yang cukup luas dan variasi jenis



Gambar 6. Arsitektur Sistem

yang cukup banyak. Karena peminatnya cukup banyak lokasi penjual juga menjadi sangat variatif.

Gambar 6 menunjukan proses ekstrasi data dari *web* hingga menjadi sebuah grafik. Aplikasi akan melakukan *render* pada *URL* yang diberikan. Setelah *URL* berhasil di render kemudian data akan diproses dan diambil sesuai dengan kriteria yang sudah diatur sebelumnya. Setelah seluruh data pada *URL* selesai dilakukan data tersebut kemudian disimpan ke dalam penyimpanan basis data. Data yang sudah tersimpan ke dalam basis data kemudian akan ditampilkan melalui grafik *chart* sesuai dengan kriteria yang sudah ditentukan.

C. Manipulasi IP

Pada saat proses *web crawling*, *user agent* yang digunakan untuk *web crawling* ada kemungkinan diblokir oleh *marketplace*[4]. Berikut merupakan beberapa cara yang disarankan untuk mengantisipasi pemblokiran oleh *marketplace* :

1. Memanipulasi *IP*. *IP proxy* yang digunakan berbeda beda pada setiap user agent untuk menghindari pemblokiran *IP*. Terdapat beberapa *IP* Server yang menyediakan *IP*
2. Perlu memperhatikan berapa banyak request yang akan dilakukan untuk menghindari blok *IP* yang disebabkan terlalu banyaknya request pada satu waktu.

IV. ARSITEKTUR SISTEM

Pada proyek ini data yang diambil berasal dari web *marketplace online*. Data pada *marketplace online* tersebut akan diambil dengan menggunakan *python* dengan memanfaatkan *library scrapy* dan *splash.* Sebelum data pada *marketplace* diambil, halaman *web page marketplace* di render terlebih dahulu menggunakan *splash*, hal ini diperlukan apabila halaman *web marketplace* tersebut merupakan halaman *web* berbasis *javascript*. Setelah di render kemudian data halaman web *marketplace* tersebut diambil datanya menggunakan *scrapy*. Data diambil berdasarkan *tag HTML*. Data yang diambil kemudian akan disimpan ke dalam *database* untuk dapat dipakai untuk membangun sebuah diagram.

1. Database - MySQL

Sebagai penyimpanan basis data dari hasil proses pengerjaan *web crawling* dan *web scraping*. Data yang sudah diambil akan disimpan di *database* MySQL sesuai dengan data-data yang sudah ditentukan.

2. Bahasa Pemrograman : Python, SQL

Sebagai bahasa penulisan program yang akan dipakai sebagai basis *environment* ketika menuliskan *script web crawling* dan *web scraping* dan juga bertugas untuk menjalankan program *web crawling* dan *web scraping* tersebut.

3. Scrapy

Framework yang dituliskan dengan menggunakan bahasa *python*, dimana framework ini akan menjadi kerangka kerja dalam pengerjaan proses pengambilan data dalam *web crawling* dan *web scraping*.

4. Splash

Digunakan sebagai perantara untuk merender situs dari halaman *marketplace* “T” yang pada halaman *marketplace* tersebut menggunakan web dinamis dengan javascript sehingga untuk dapat melakukan pengambilan datanya sesuai dengan yang diinginkan dibutuhkan proses rendering terlebih dahulu.

5. Docker

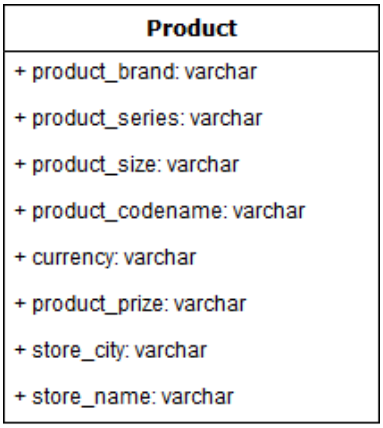
Layanan virtualisasi sebagai lokasi untuk melakukan instalasi dan menjalankan *Splash.*

6. PowerBI

Untuk melakukan visualisasi pada data yang sudah bersih sehingga data yang didapat dapat dianalisis sehingga menghasilkan suatu kesimpulan yang dapat dimanfaatkan untuk keperluan lainny

V. RANCANGAN SISTEM

Berikut merupakan rancangan database product pada *Postgree:*



Gambar 7. Database Produk

Gambar 7 berisi data data diambil dari *marketplace* yang telah ditentukan. Rincian atribut sebagai berikut :

1. *Product Brand*: Merk dari produk sesuai dengan input nama merk produk smartphone dari penjual yang menjual barangnya di *marketplace*.
2. *Product Series*: Jenis seri dari produk sesuai dengan input sesuai jenis produk smartphone yang diinputkan oleh penjual yang menjual barangnya di *marketplace*
3. *Product Size*: Ukuran besarnya *memory* dari produk *smartphone* yang dijual oleh penjual di *marketplace*.
4. *Product Codename*: Tipe dari produk *smartphone* yang dijual oleh penjual di *marketplace* sesuai dengan jenis *smartphone* yang dijual.
5. *Currency*: Mata uang yang digunakan sebagai alat pembayaran yang akan digunakan di dalam *Marketplace* untuk dipakai dalam membeli produk *smartphone* yang dijual oleh penjual di *marketplace* tersebut.
6. *Product Price*: Harga dari produk *smartphone* yang sesuai dengan *input* nama produk dari penjual yang menjual barangnya di *marketplace*.
7. *Store City*: Lokasi dari penjual *smartphone* pada *marketplace*.
8. *Store Name*: Nama toko dari penjual *smartphone* di *marketplace.*

VI. HASIL IMPLEMENTASI SISTEM

Web crawling dan *web scraping* dilakukan pada situs *online marketplace*. *Web scraper* bertugas untuk mengambil data data produk dari hasil pencarian di situs tersebut, sedangkan *web crawler* bertugas untuk berpindah antar halaman pencarian secara otomatis.

Web crawler & *web scraping* dibangun dengan menggunakan bahasa pemrograman *Python,* dan menggunakan *library scrapy* serta *splash*.

Scrapy adalah *library Python* yang memiliki fungsi utama dalam proses ekstraksi data dari halaman *web* dengan menyertakan url dari halaman *web* yang akan di ekstrak. *Scrapy* melakukan ekstrak data berdasarkan dari *tag html* yang ada di dalam halaman *web*. Untuk itu perlu menganalisa apa saja tag yang diperlukan untuk diekstrak nilainya. Tag yang digunakan harus dalam bentuk *css selector* atau dalam bentuk *xpath.*

```
for tr in response.xpath("//div[@class='vlEGRFVq']"):
p_name = tr.xpath('.//h3/text()').extract_first()
p_price = tr.xpath('.//div/span/span/text()').extract_first()
s_city =
tr.xpath(".//span[@class='_3ME2eGXf']/text()").extract_firs
t()
s_name =
tr.xpath(".//span[@class='_2rQtYSxg']/text()").extract_first(
)
```

Script 1. *Xpath*

Pada *for loop* script 1 perulangan berjalan berdasarkan pada elemen *html* dengan kelas “vlEGRFVq” didapatkan dari *inspect* halaman *HTML* dan berulang berdasarkan jumlah elemen yang terdapat pada kelas tersebut yang. Untuk setiap perulangan program akan mengambil data berdasarkan path yang dimulai dari elemen *html* yang digunakan pada *for loop*. Kode *extract.first()* dimaksudkan agar aplikasi hanya mengambil nilai pertama saja karena terkadang terdapat lebih dari satu nilai yang dikembalikan.

Kelas Untuk halaman *web* yang berbasis *javascript* perlu menggunakan *splash* agar *scrapy* dapat melakukan ekstraksi secara keseluruhan. Ada lebih dari satu metode yang disediakan oleh *splash* dalam proses memuat situs sebelum siap digunakan atau siap diekstraksi oleh *scrapy*, salah satunya adalah *splash*: *set_viewport_full()*. Metode ini memerintahkan *splash* untuk memuat dulu *web* secara keseluruhan.

```
def start_requests(self):
url = 'https://www.t.com/search?st=product&q=g'
script = """
function main(splash, args)
        assert(splash:go(args.url))
        assert(splash:wait(5))
        splash:set_viewport_full()
        return {
```

```
                html = splash:html(),
                png = splash:png(),
                har = splash:har(),
}
end
"""
```

Script 2. *Splash*

Script 2 merupakan *script* lua default untuk *splash* dengan beberapa modifikasi. Titik utama modifikasi terdapat pada "url = 'https://www.t.com/search?st=product&q=g" di mana *script* ini menunjukan alamat *web site* yang akan diambil datanya. Kemudian yang perlu diperhatikan dalam modifikasi *script* ini adalah "splash:set_viewport_full()", script ini menjadi kunci utama dalam memuat halaman web secara keseluruhan terlebih dahulu baru kemudian proses pengambilan data dilakukan. *Script* ini ditujukan untuk mencegah proses pengambilan data terjadi prematur dikarenakan data belum dimuat secara keseluruhan.

Untuk proses *crawling* dapat membuatnya secara manual. Sebagai contoh pada riset ini proses *crawler* dengan memanfaatkan perulangan. Dalam proses ini proses perulangan dilakukan pada url pencarian dimana akan memodifikasi angka halaman hasil pencarian.

```
for i in range(2, 3):
print(i)
next_url=
'https://www.xxx.com/search?page='+str(i)+'&st=product&
q=g'
script = """
function main(splash, args)
        assert(splash:go(args.url))
        assert(splash:wait(5))
        splash:set_viewport_full()
        return {
                html = splash:html(),
                png = splash:png(),
                har = splash:har(),
}
end
"""
yield SplashRequest(
        url = next_url, callback=self.parse_next,
endpoint='execute', priority=i, args={'lua_source' : script}
)
```

Script 3. *Splash*

Pada kode Script 3 perulangan untuk pengambilan data datur manual pada baris kode "for i in range(2, 3):", dimana perulangan ini bermaksud mengulan mulai dari halaman 2 hingga halaman 3. Kemudian baris kode "next_url = 'https://www.t.com/search?page='+str(i)+'&st=product&q=g'" bertujuan menyimpan alamat *web* pada *variable* url. Alamat *web* dibentuk secara dinamis berdasarkan halaman yang akan diambil datanya.

Dalam menjalankan program *scrapy* ini perlu diperhatikan beberapa pengaturan seperti pada gambar 8 agar program sesuai dengan beberapa aturan pada beberapa situs. Umumnya perlu memperhatikan aturan pada robot.txt yang ada pada sebuah situs.

```
User-agent: *
Disallow: /search?*
Disallow: /search/*
Disallow: /image-search/
Disallow: /*.pl
Disallow: /amp/api/*
Disallow: /feed?sc=*
Disallow: /graphql
Disallow: /reputationapp/*
Disallow: */tokopedia-lite-production/
Disallow: /similar-products*
Disallow: /provi/check*
Disallow: /kartu-kredit*?id=*
Disallow: /*/*/talk
Disallow: /*/*/review

User-agent: Alexabot
Disallow: /
Allow: /blog/


Sitemap: https://www.t        .com/sitemap/catalog-index.xml
Sitemap: https://www.t        .com/sitemap/category-index.xml
Sitemap: https://www.t        .com/sitemap/products-index-0.xml
Sitemap: https://www.t        .com/sitemap/products-index-1.xml
Sitemap: https://www.t        .com/sitemap/products-index-2.xml
Sitemap: https://www.t        .com/sitemap/products-index-3.xml
Sitemap: https://www.t        .com/sitemap/products-index-4.xml
Sitemap: https://www.t        .com/sitemap/products-index-5.xml
Sitemap: https://www.t        .com/sitemap/products-index-6.xml
Sitemap: https://www.t        .com/sitemap/products-index-7.xml
Sitemap: https://www.t        .com/sitemap/products-index-8.xml
Sitemap: https://www.t        .com/sitemap/products-index-9.xml
Sitemap: https://www.t        .com/sitemap/hotlist-index.xml
Sitemap: https://www.t        .com/sitemap/product-find-index.xml
Sitemap: https://www.t        .com/sitemap/recharge-index.xml
Sitemap: https://www.t        .com/sitemap/shop-index.xml
Sitemap: https://www.t        .com/sitemap/flight-index.xml
Sitemap: https://www.t        .com/sitemap/events-index.xml
Sitemap: https://www.t        .com/sitemap/deals-index.xml
```

Gambar 8. Aturan pada Robot.txt

Selain aturan pada *file robots.txt* yang disediakan oleh situs yang akan diekstrak, perlu juga memperhatikan berapa banyak halaman yang akan diproses dalam satu waktu. Hal ini untuk menghindari terlalu banyaknya *http request* yang dikirimkan pada situs yang dapat menyebabkan di *ip* di blok. Untuk mengatur hal ini pada *scrapy* dapat melakukanya dengan mengubah nilai dari CONCURRENT_REQUEST yang terdapat pada file *settings* seperti pada script 4.

```
CONCURRENT_REQUESTS = 1
```

Script 4. Pengaturan Scrappy

Concurrent_request adalah pengaturan pada *scrappy* untuk membatasi berapa banyak pemanggilan yang akan di lakukan pada *web marketplace* yang dituju. Hal ini bertujuan agar tidak membebani *web marketplace* dan mencegah agar tidak terjadi blocking pada ip.

Setelah data diekstrak, data perlu disimpan ke dalam sebuah media penyimpanan. Sebagai media penyimpanan dapat menggunakan file dengan format *csv*, *json*, atau

sebuah *database*. Untuk mengatur tempat penyimpanan data dapat melakukanya pada file *pipelines.*

```
import mysql.connector
import sqlite3

class ProjectPipeline(object):
    def __init__(self):
        self.create_connection()
        self.prepare_table()

    def process_item(self, item, spider):
        self.store_db(item)
        return item

    def create_connection(self):
        self.conn = mysql.connector.connect(
            host = '',
            user = '',
            passwd = '',
            database = ''
        )
        # self.conn = sqlite3.connect("products")
        self.curr = self.conn.cursor()

    def prepare_table(self):
        self.curr.execute("""
            create table if not exists products(
                p_name text,
                p_price text,
                s_location,
                s_name
            )
        """)

    def store_db(self, item):
        self.curr.execute("""insert into products values (%s,%s,%s,%s)""", (
            item['product_name'],
            item['product_price'],
            item['store_city'],
            item['store_name']
        )) self.conn.commit()
```

Script 5. Database

Script 5 diatas adalah kode untuk mempersiapkan *database* tempat data yang telah diambil disimpan. Kode pada "def __init__(self):" adalah kode untuk mempersiapkan koneksi *database*. "self.create_connection()" berfungsi untuk membuat koneksi ke *database*, "self.prepare_table()" berfungsi untuk mempersiapkan tabel apabila tabel belum ada di dalam database. Kode "def process_item(self, item, spider):" adalah fungsi yang dipanggil untuk menyimpan data ke dalam *database* dengan menggunakan fungsi "self.store_db(item)". Fungsi "def prepare_table(self):" adalah fungsi yang digunakan untuk membuat tabel tempat menyimpan data yang sudah diambil apabila tabel belum tersedia dengan menggunakan kode "self.curr.execute" dengan parameter query sql. Fungsi "def store_db(self, item):" digunakan untuk menyimpan data ke dalam database dengan menggunakan kode self.curr.execute dengan parameter sql.

VII. HASIL EKSPERIMEN DAN EVALUASI

A. *Ekstraksi Produk*

Script *Scrapy* dengan *class spiders* akan dijalankan menggunakan *jupyter notebook console* berdasarkan kata kunci yang ditentukan. *Splash* akan menjadi perantara antara *marketplace* dengan *website* dinamis dengan pengguna yang akan melakukan proses *crawling* dan *scraping* pada *website marketplace* tersebut sehingga data yang telah ditentukan sebelumnya akan dapat diambil dan hasilnya akan sesuai dengan data yang sudah ditentukan sebelumnya. Proses akan dijalankan sampai semua data yang ditentukan berhasil diambil sesuai dengan hasil pencarian untuk barang yang sudah ditentukan sebelumnya dari halaman pertama hingga terakhir.

```
019-06-23 15:52:32 [scrapy,core,scraper] DEBUG: Scraped from <200 https://www.t.com/search?page=3&s

{'currency': 'rp',

'product_brand': 's',

'product_codename': 'g',

'product_price': '433.000',

'product_series': 'xs',

'product_size': '120',

'store city': 'Jakarta' }

s g tab a 10 2019 sm-t515 32gb - black

[s, g, tab, a, 10, 2019, sm, t515, 32, gb, -, black]

019-06-23 15:52:32 [scrapy,core,scraper] DEBUG: Scraped from <200 https://www.t.com/search?page=3&s

{'currency': 'rp',

'product_brand': 's',

'product_codename': 'g',

'product_price': '4.750.000',

'product_series': 'xs',
```

p-ISSN : 2443-2210
e-ISSN : 2443-2229

'product_size': '32',

'store city': 'Medan'}

s g s10 s 10 ume classic flip case cover leather

[s, g, slO, s, 10, ume, classic, flip, case, cover, leather]

019-06-23 15:52:32 [scrapy,core,scraper] DEBUG: Scraped from <200 https://www.t.com/search?page=3&s

{'currency': 'rp',

'product_brand': 's',

'product_codename': 'g',

'product_price': '50.000',

'product_series': 'xs',

'product_size': '32',

'store city': 'Jakarta'}

S g s10 s 10 case soft anti crack casing jelly bening

[s, g, slO, s, 10, case, soft, anti, crack, casing, jelly, bening]

Script 6. Hasil *Web Scraping*

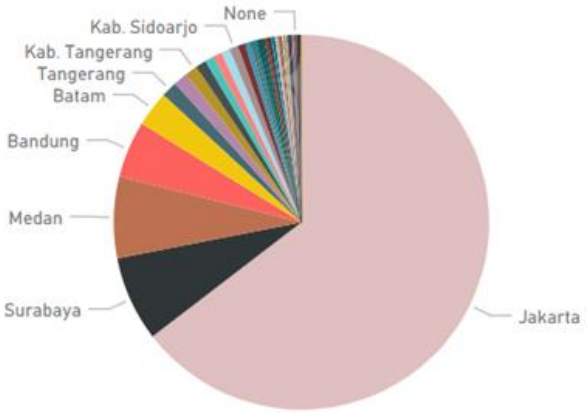
Berdasarkan hasil pada Script 6, *Web Scraping* berhasil dilakukan dan mendapatkan data yang sudah dipisahkan berdasarkan: *product_name*, *product_price*, *store_city*, *store_name*. Data tersebut langsung dimasukan ke dalam *database*.

B. *Analisis Data*

Data yang sudah dibersihkan akan divisualisasi menggunakan *power BI*. Berikut merupakan hasil visualisasi dari *power BI*.

location	Count of price
Jakarta	292
Surabaya	45
Bandung	29
Medan	23
Batam	16
Malang	6
Tangerang	6
Kab Bandung	5
Kab Tangerang	5
Jambi	4

Gambar 9. Total Penjual Berdasarkan Lokasi



Gambar 10 Pie Chart Penjualan Berdasarkan Lokasi

Berdasarkan Gambar 9 dan Gambar 10 dapat disimpulkan bahwa penjual dari kota Jakarta merupakan penjual terbanyak yang menjual produk *smartphone* "g" pada *marketplace* "T". Berdasarkan data yang berhasil diambil, jumlah penjual *smartphone* "g" yang menjual barangnya di *marketplace* "T" ada sebanyak 292 penjual. Berdasarkan data yang didapat juga dapat disimpulkan bahwa aktivitas penjualan dari *smartphone* "g" pada *marketplace* "T" di kota Jakarta merupakan yang tertinggi di Indonesia. Selain itu dari data tersebut juga dapat disimpulkan bahwa peminat paling banyak dari *smartphone* "g" di Indonesia berasal dari kota Jakarta. Sedangkan kota Surabaya merupakan penjualan kedua terbesar yaitu 45, namun perbedaannya cukup signifikan dengan jumlah penjual yang ada di kota Jakarta.

C. *Evaluasi*

Untuk proses *web crawling* dan *web scraping* sudah berhasil sesuai dengan yang diharapkan dan data yang disimpan juga sudah sesuai dengan data yang sudah ditentukan sebelumnya. Dalam pengerjaannya juga data sudah dapat divisualisasikan dengan menggunakan *Power BI* sesuai dengan rencana awal. Dan hasil dari visualisasi juga sudah dapat sesuai dengan kesimpulan dari data keseluruhan yang didapat dari proses *web crawling* dan *web scraping*. Tetapi berdasarkan eksperimen yang dilakukan, hasil yang didapat menunjukan perbedaan yang cukup signifikan antara jumlah pengamatan terbanyak dan juga jumlah pengamatan yang terbanyak berikutnya.

VIII. KESIMPULAN DAN SARAN

Proses pengambilan data dengan sumber *marketplace* yang sudah ditentukan sebelumnya sudah berhasil, dan data yang disimpan pun sudah sesuai dengan yang diinginkan. Data yang diambil juga sudah berhasil dilakukan visualisasi sesuai dengan tujuan awal dari pengambilan data.

Diharapkan untuk pengerjaan lebih lanjutnya akan dapat melakukan proses *web scraping* dengan masuk kedalam perhalamannya sehingga pemilihan dari data yang akan diambil diharapkan akan lebih komplit dibanding dengan daftar data yang dapat diambil dalam halaman perkategori item yang ditentukan sehingga hasil dari visualisasi pun akan lebih dapat bervariasi tergantung data mana yang akan dijadikan objek penelitian dan perbandingan.

DAFTAR PUSTAKA

[1] Krishnan, Vijay, Bharanidharan, S., Krishnamoorthy, G., "Research Data Analysis with Power BI," *11th International CALIBER*, 2017, p. 212.

[2] D. Glez-Pen‹a, A. Lourenco, H. Lopez-Fernandez, M. Reboiro-Jato, F. Fdez-Riverola, "Web scraping technologies in an API world," *Briefings in Bioinformatics*, pp. 1-10, Feb. 2013.

[3] P. Ayar, C. Sandip, "Efficient Focused Web Crawling Approach for Search Engine," *International Journal of Computer Science and Mobile Computing*, vol. 4, issue. 5, pp. 545–551, May. 2015.

[4] Seppe v. Broucke & B. Baesens, *Practical Web Scraping for Data Science Best Practices and Examples with Python*, New York: Springer Nature-Apress, 2018.

[5] (2019) Scrapy website. [Online]. Tersedia: http://scrapy.org/

[6] (2019) Splash website. [Online]. Tersedia: http://blog.scrapinghub.com/

[7] (2019) PowerBi website [Online]. Tersedia: http://powerbi.microsoft.com/

[8] Olston, C. and Najork, M, "Web Crawling," *Foundations and Trends® in Information Retrieval*, vol. 4(3), pp. 175-246, Feb. 2010.

[9] Kouzis-Loukas D., *Learning Scrapy*. Birmingham: Packt Publishing, 2016.

[10] Genovese, L., & Geraci, F, "Web Crawling and Processing with Limited Resources for. Journal of Software," *Journal of Software*, vol. 7, pp. 300-316, May. 2018.

[11] Vandana Shrivastava, "A Methodical Study of Web Crawler," *Journal of Engineering Research and Application*, vol. 8, pp. 01-08, Nov. 2018.

[12] Mini Singh Ahuja, Dr Jatinder Singh Bal, Varnica. "Web Crawler: Extracting the Web Data," *International Journal of Computer Trends and Technology*, vol. 13(3), pp. 132-137, Jul. 2014.

[13] Shubham Age, Tushar Indorkar, Shital Kokate and Manisha Shitole, "A Self Adaptive Semantic Focused Web Crawler," *International Journal of Research In Science & Engineering*, Vol. 1(6), pp. 74-79, Nov. 2017.

[14] Hana Choi, Carl F. Mela. "Online Marketplace Advertising," International Choice Symposium, 2016.