p-ISSN 2087-1716
e-ISSN 2548-7779
ILKOM Jurnal Ilmiah Volume 10 Nomor 2 Agustus 2018

MEMBANGUN *WEB CRAWLER* BERBASIS *WEB SERVICE* UNTUK *DATA CRAWLING* PADA WEBSITE GOOGLE PLAY STORE

Lutfi Budi Ilmawan
lutfibudi.ilmawan@umi.ac.id
Universitas Muslim Indonesia

Abstrak

Saat ini toko aplikasi milik Google, Google Play Store tidak menyediakan API khusus untuk mengakses data-data tentang aplikasi-aplikasi yang terdapat pada Google Play Store. Jumlah data yang besar pada toko aplikasi tersebut sangat menarik untuk dijadikan sebagai objek penelitian dalam bidang *data mining*. Pada penelitian ini, dibangun sistem untuk dapat mengambil data-data tersebut. Kemudian agar bisa digunakan secara *cross platform*, maka sistem yang dibangun ini adalah sistem yang berbasis *web service*. Sistem yang dibangun berhasil mengambil data pada website Google Play Store dengan baik dan benar sesuai dengan kebutuhan pada analisis sistem dan dapat diintegrasikan dengan *web service* berbasis REST untuk mendukung penggunaan sistem secara *cross platform*.
Kata Kunci: *Data mining*, *web crawler*, *web service*

Abstract

At this time, Google Play Store is not providing API that can be used for accessing datas from applications on it's application store. With that plenty application's data, it could be used to make it a good research object, specially on data mining field. In this research, the system that is built is the system that can retrieve that applications' data. For multiplatform's purpose, web services are used for being an interface between client and server. Finally, the built system is working as expected. The system can retrive data from Google Play Store and it is suitable from requirements of data analysis stage. It can also integrated with REST web service to provide multiplatform access.
Keywords: Data mining, web crawler, web service

1. Pendahuluan

Google Play Store (dulunya bernama Android Market) merupakan toko aplikasi resmi milik Google untuk perangkat yang menggunakan sistem operasi Android. Saat ini dalam toko aplikasinya, Google Play Store memiliki aplikasi (*update* per tanggal 10 Desember 2017), sejumlah 3.532.448 aplikasi. Banyaknya aplikasi yang terdapat pada Google Play Store membuat toko aplikasi ini sangat menarik untuk dijadikan sebagai objek penelitian, khususnya dalam bidang *data mining*. Salah satu contohnya yaitu analisis sentimen yang merupakan cabang ilmu dari *data mining*, sangat cocok jika diaplikasikan pada Google Play Store. Analisis sentimen digunakan untuk menentukan sentimen para pengguna dari setiap aplikasi yang terdapat pada toko aplikasi tersebut dengan mengklasifikasikan ratusan, bahkan ribuan *text review* dari pengguna secara otomatis. Namun kendalanya saat ini adalah pihak Google sendiri tidak menyediakan API (*Application Programming Interface*) agar data pada Google Play Store dapat diintegrasikan dengan perangkat lunak yang sedang dikembangkan oleh *software developer* atau untuk keperluan sebagai data penelitian. Saat ini, pihak Google menyediakan API hanya untuk *developer* android yang aplikasinya terdaftar pada Google Play Store, API tersebut juga sangat terbatas, developer hanya bisa memanipulasi data-data tertentu dari aplikasi miliknya sendiri.

Salah satu cara untuk mengatasi permasalahan tersebut adalah dengan memanfaatkan teknologi *web crawler*. Web crawler merupakan sebuah aplikasi yang secara otomatis melintasi web dengan mengunduh dokumen dan mengikuti link dari halaman ke halaman [1], sehingga *web crawler* dapat dijadikan sebuah alat untuk mengambil konten-konten yang ditampilkan oleh halaman website, kemudian konten-konten tersebut dikelompokkan dalam satu atau lebih atribut.

Terdapat banyak cara untuk membuat *web crawler*. Namun dengan alasan agar aplikasi ini nantinya dapat diintegrasikan dengan aplikasi lainnya dan penggunaannya mendukung *cross platform*, maka

penulis memilih untuk membuat *web crawler* yang berbasis *web service*. *Web Service* dapat dibuat menggunakan Flask dari Python dan *web crawler*-nya sendiri akan dibuat menggunakan Scrapy dari Python. Adapun alasan penulis menggunakan Bahasa pemrograman python untuk pembuatan *web crawler* pada Google Play Store sebab bahasa Python adalah bahasa pemrograman yang sederhana namun sangat unggul dengan fungsionalitas yang sangat baik untuk memproses data linguistik [2]. Jadi dengan digunakannya bahasa Python, maka diharapkan dapat mempermudah para peneliti lain yang ingin menggunakan aplikasi ini pada bidang *data mining*.

Web crawler telah digunakan dalam berbagai penelitian [3][4][5] untuk proses ekstraksi dan analisis data. Penelitian yang dilakukan [3] dan [4] melakukan crawling terhadap website media sosial. *Web crawler* memiliki beberapa jenis sesuai yang disebutkan dalam [6]. Pada penelitian [3], *web crawler* yang digunakan merupakan *web crawler* dengan tipe *focused crawler* yang dikembangkan oleh Chakrabarti, S [7]. Pada penilitian [4], menggunakan dua jenis web crawler, yaitu *breadth first crawler* dan *uniform crawler* namun *uniform crawler* tersebut memiliki arsitektur yang sama dengan *breadth fisrt crawler*, perbedaannya hanya berada pada proses generasi dan manajemen antriannya [4]. Pada penelitian ini web crawler yang digunakan adalah *web crawler* dengan jenis *breadth first crawler*.

2. Metode

Berikut adalah uraian dari tahapan dan metode yang digunakan dalam penelitian :

2.1 Studi Kepustakaan

Pengumpulan bahan referensi, seperti jurnal penelitian, prosiding, tesis, buku-buku teori dan sumber-sumber lain termasuk informasi yang diperoleh melalui internet.

2.2 Analisis Sistem

Metode pengembangan sistem yang digunakan adalah metode SDLC (*Software Development Life Cycle*), yang memiliki tahapan secara berurutan dimulai dari perencanaan, analisis, perancangan, implementasi, dan pengujian [8]. Sistem yang dibangun pada penelitian ini merupakan aplikasi *web crawler* berbasis *web service* agar aplikasi ini dapat diintegrasikan dengan aplikasi lainnya untuk keperluan pengumpulan data pada sub proses dari *data mining*. Adapun proses *crawling*-nya akan mengambil data-data penting tentang aplikasi dari Google Play Store. Crawling data pada Google Play Store tidak semudah melakukan *crawling* pada halaman web biasa. Sebab pada halaman *website*-nya, Google Play Store menggunakan AJAX[1] untuk proses pengambilan data pada servernya. Karena penggunaan AJAX maka data yang akan diambil tidak tampak di halaman web jika event AJAX-nya tidak di-*trigger*, seperti pada saat menekan tombol tertentu atau *event-event* lainnya. Jadi untuk mengambil data-data yang menggunakan AJAX, maka *crawler* atau *spider* harus melakukan simulasi *request* yang sama persis dengan proses AJAX aslinya berdasarkan XHR-nya. XHR atau XMLHttpRequest merupakan objek yang menyediakan kemampuan kepada *client* untuk berkomunikasi secara asinkron dengan *server* [9]. Setelah proses *request* dari objek XHR disimulasikan pada web crawler, maka data-data dari halaman *web* Google Play Store dapat diambil oleh *web crawler* yang dibuat. Adapun aplikasi yang dibuat memiliki karakteristik, antara lain:

a. Sistem yang dibangun merupakan sistem yang berbasis REST[2] *web service* yang bertugas untuk menyediakan layanan pengambilan data dari halaman website Google Play Store.
b. Pengambilan data dilakukan oleh *web crawler* yang dieksekusi melalui *web service*.

[1] AJAX (Asynchronous JavaScript And XML) dapat mengambil data pada *server* tanpa proses *reloading page* dan dapat mengirimkan data pada server melalui *background* prosesnya. AJAX merupakan kombinasi dari pemanfaatan XML Http Request dan JavaScript serta HTML DOM.

[2] REST (*REpresentational State Transfer*) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP (*Hypertext Transfer Protocol*) sebagai protocol untuk komunikasi data. Pada arsitektur REST, REST server menyediakan resources (sumber daya/data) dan REST client mengakses dan menampilkan resource tersebut untuk penggunaan selanjutnya. Setiap resource diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID. Resource tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML.

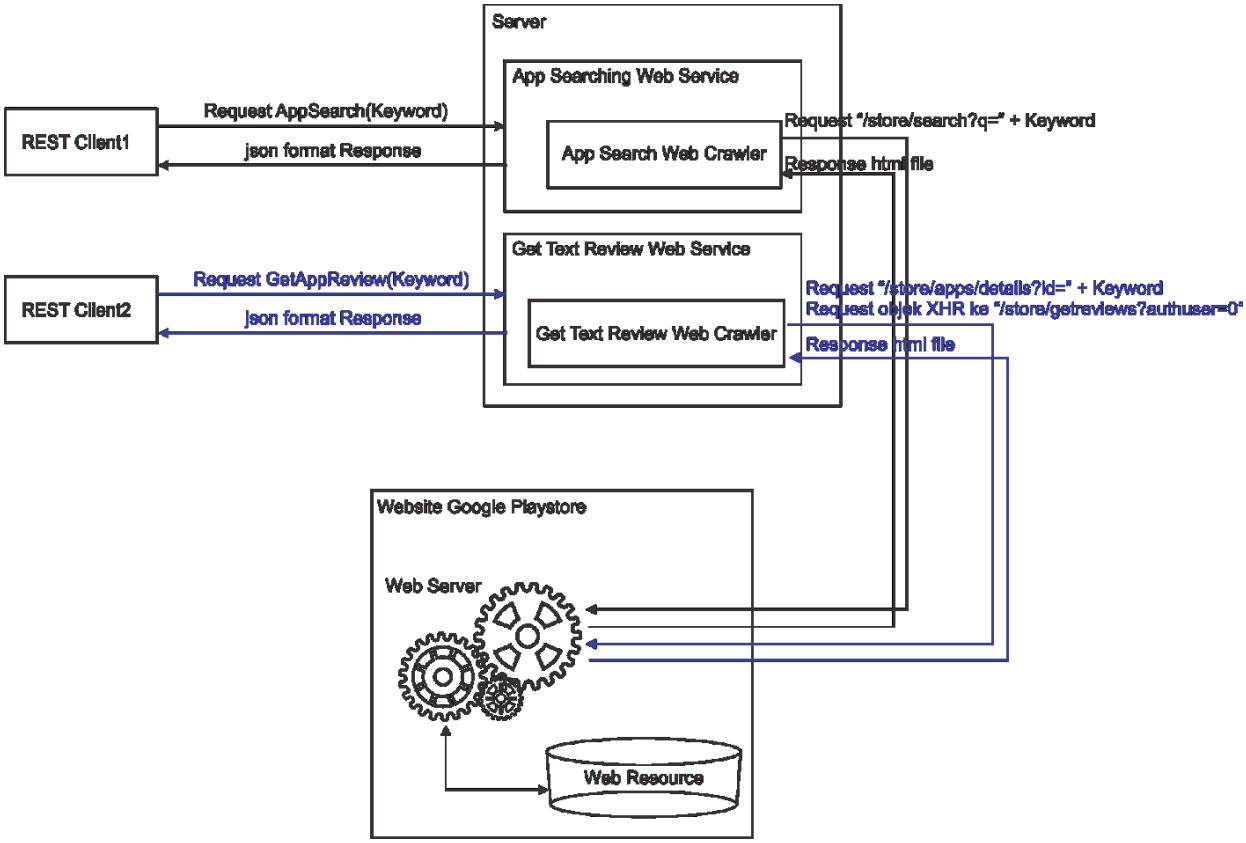
c. Terdapat dua buah layanan pada *web service* yang dibuat. Layanan pertama yaitu ***App Searching Service*** untuk pencarian aplikasi berdasarkan kata kunci yang berupa nama aplikasi, *response* yang dikembalikan oleh *web service* terdiri dari atribut: nama aplikasi, nama paket (*package name*), nama *developer*, dan URL *icon*-nya. Aplikasi kedua yaitu ***Get Text Review Service*** untuk pengambilan komentar (*text review*) sebuah aplikasi berdasarkan package name-nya, *response* yang dikembalikan terdiri dari atribut: nama user, tanggal komentar, dan isi komentar.

2.3 Perancangan Sistem

Sistem yang dibangun merupakan *web service* yang memiliki layanan untuk pengambilan data-data pada website dari Google Play Store, pengambilan data tersebut menggunakan web crawler. Pada *web service* ini, terdapat dua komponen, antara lain: *App Searching Service* dan *Get Text Review Service*. *Web service* yang disediakan akan melakukan eksekusi terhadap *web crawler* sesuai dengan *service* yang diinginkan oleh *user*. Adapun perancangan sistem terdiri dari:

Perancangan Arsitektur Sistem

Gambaran umum dari arsitektur sistem yang dirancang dapat dilihat pada Gambar 1. Terdapat dua komponen pada *web service server*, komponen tersebut terdiri dari *App Searching Service* dan *Get Text Review Service*. *App Searching Service* merupakan layanan untuk pencarian aplikasi dan *Get Text Review Service* adalah layanan untuk pengambilan komentar *review* dari aplikasi. Kedua komponen ini memanfaatkan *web crawler* untuk pegambilan data pada website Google Play Store.



Gambar 1. Arsitektur Sistem secara Umum

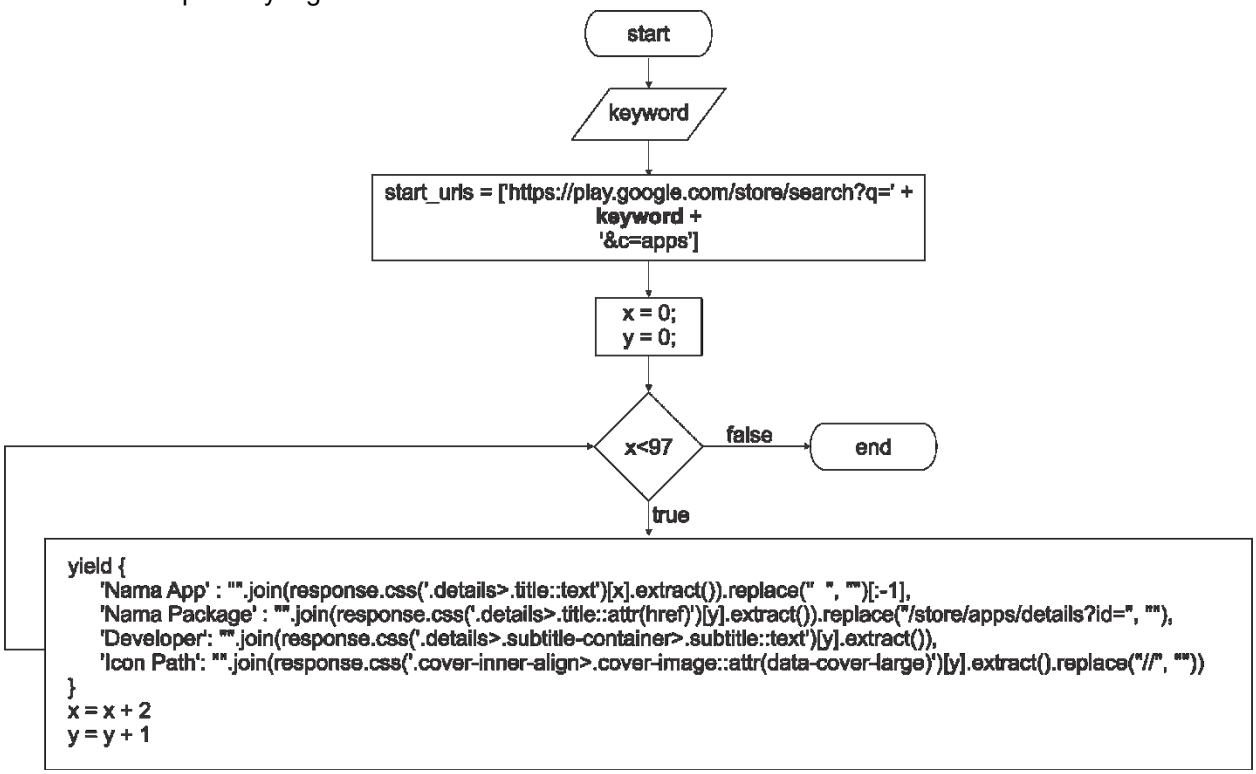
Komponen dan Proses pada Perangkat Lunak

Pada bagian ini akan dijelaskan tentang komponen dan proses dari *App Searching Service* dan *Get Text Review Service*.

a. *App Searching Service*

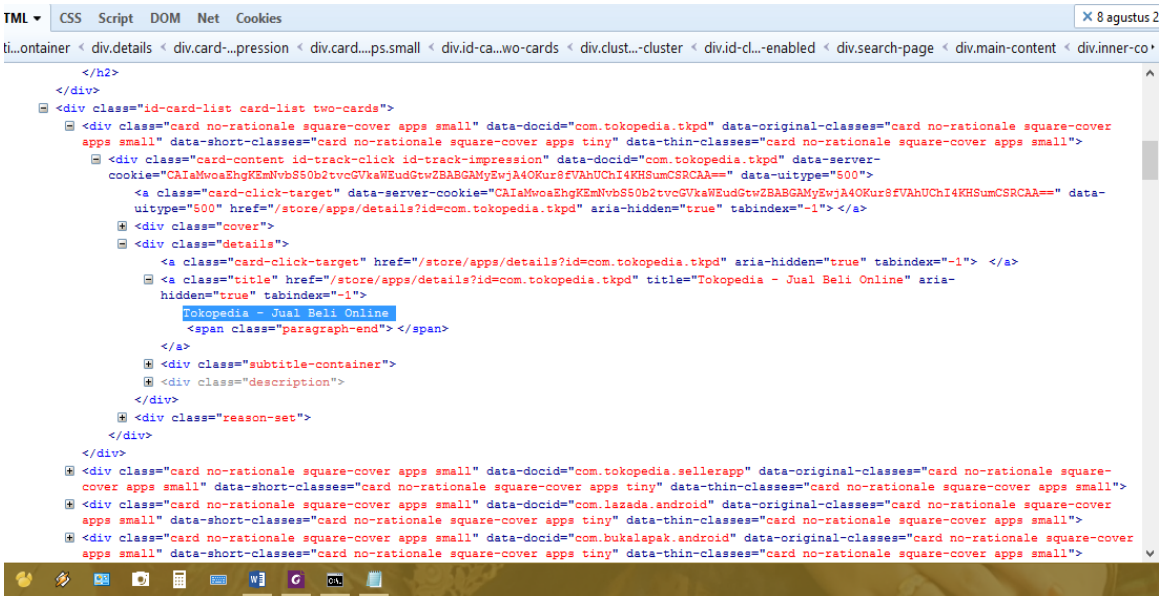
Gambaran umum dari *service*/layanan ini dapat dilihat pada Gambar 2. Layanan ini merupakan layanan yang disediakan untuk proses pencarian aplikasi berdasarkan nama aplikasi yang terdaftar

di Google Play Store, *response* dikembalikan dalam bentuk format JSON yang terdiri dari atribut: nama aplikasi, nama paket (*package name*, yang nantinya dijadikan parameter untuk pengambilan *text review*), nama *developer*, dan URL *icon* aplikasinya. Adapun jumlah *record* yang dapat diambil maksimal 48 *record*. Proses pengambilan *record* untuk *service* ini tidak terlalu rumit seperti pengambilan *record* pada ***Get Text Review Service*** yang *record*-nya hanya dapat diambil dengan mensimulasikan AJAX melalui XHR pada *page*-nya. Adapun URL dari halaman yang di-*crawling* adalah "https://play.google.com/store/search?q=" + Keyword. Keyword pada URL tersebut adalah nama dari aplikasi yang akan dicari.



Gambar 2. *Flowchart* dari AppSearch *Web Service*

Record-record yang diambil oleh *web crawler* adalah *content* tententu dari sebuah *selector*. *Selector*-nya dengan mudah dapat diketahui dengan menggunakan *Developer Tools atau Web Developer* yang merupakan add-on bawaan dari sebuah *web browser*. Misalkan atribut yang ingin diambil adalah nama aplikasi, maka yang harus dicari adalah tag HTML yang memiliki *content* nama aplikasi tadi. Gambar 2 memperlihatkan atribut nama aplikasi merupakan content dari *tag* <a> dan *tag* tersebut memiliki atribut *class* dengan *value* "title" jadi *selector* berdasarkan *tag* tersebut adalah ".value". Namun tidak menjamin bahwa semua content dari *selector* ".value" adalah nama aplikasi, maka untuk mengantisipasi permasalahan tersebut makan parent tag atau *class* nya harus juga diikutsertakan pada *selector*. Adapun parent tag dari *tag* <a> tadi adalah *tag* yang memiliki atribut *class* dengan *value* "*details*", jadi *selector* dari *content* nama aplikasi adalah ".details>.value". Selector tersebut akan memilih tag yang memiliki class dengan nama "details" yang memiliki *child* dengan nama *class "value"*, *class* ditandai dengan tanda titik (.) kemudian diikuti dengan nama *class*-nya. Begitu pula dengan atribut yang lain, cara mendapatkan *record*-nya adalah dengan mengambil *content* halaman berdasarkan selector yang telah didefinisikan. Seluruh atribut yang diambil berserta *selector*-nya dapat dilihat pada Tabel 1.



Gambar 3. *Add-on Web Developer* pada Mozilla Firefox

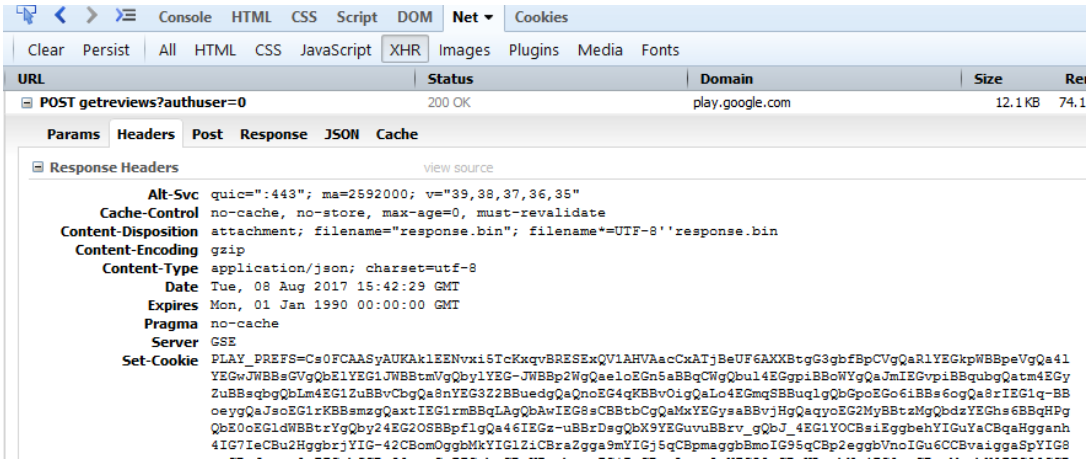
Tabel 1 Atribut dan *Selector* dari *web crawler* App Searching Service

No.	Nama Atribut	Selector
1.	Nama Aplikasi	.details>.title
2.	Nama Package	.details>.title::attr(href)
3.	Developer	.details>.subtitle-container>.subtitle
4.	Icon Path	.cover-inner-align>.cover-image::attr(data-cover-large)

Web service ini nantinya akan diakses dengan memasukkan alamat http://[ip-address]/AppSearch pada REST *client* dengan method "POST" dan sebuah parameter "NamaApp", value-nya merupakan kata kunci berupa nama dari aplikasi yang akan dicari.

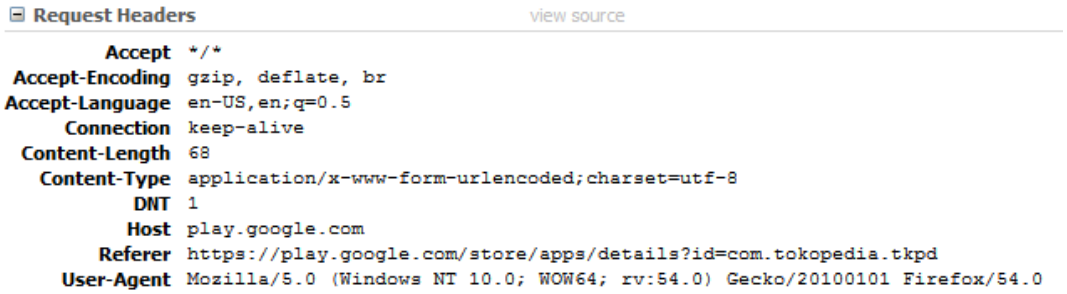
b. *Get Text Review Service*

Layanan ini merupakan layanan yang disediakan untuk proses pengambilan komentar dan atribut lainnya yang terkait. *Response* dikembalikan dalam bentuk format JSON yang terdiri dari atribut: nama *user*, komentar, dan tanggal. Adapun maksimal *record* yang bisa diambil tidak terbatas, tergantung parameter jumlah iterasi yang diberikan pada *web crawler*-nya. Satu kali iterasi dapat mengambil maksimal 40 *record*, hal ini disebabkan oleh *response* yang diperoleh dari website Google Play Store memang hanya 40 *record* dalam satu kali *request* untuk pengambilan data komentar. Proses pengambilan *record* pada *service* ini cukup rumit, karena data yang akan diambil berasal dari halaman yang menggunakan AJAX. Cara yang dapat dilakukan oleh *web crawler* agar data pada halaman web yang menggunakan AJAX dapat diambil adalah dengan melakukan simulasi *request* yang sama persis yang dilakukan oleh AJAX-nya. Hal ini dapat diketahui dengan melihat *request* dari objek XHR pada halaman web tersebut, ketika AJAX *event*-nya tereksekusi, seperti terlihat pada Gambar 3.

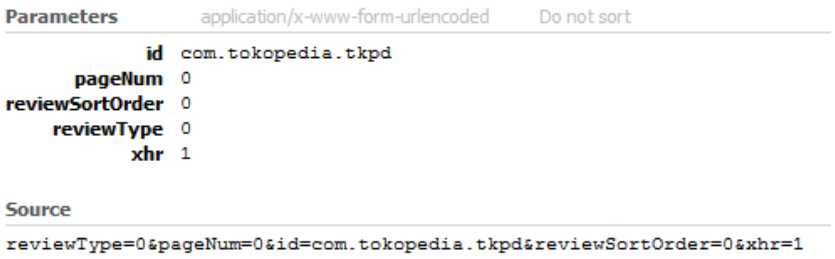


Gambar 4 Tampilan Objek XHR pada Add-on Web Developer

Hal yang perlu diperhatikan pada objek XHR ini adalah *Request Header* dan *Form Data*-nya yang nantinya akan dijadikan parameter untuk proses simulasi *AJAX Request* pada *web crawler*. Pada Gambar 4, dapat dilihat *request header* dan *form data* secara lengkap untuk pengambilan komentar *review* berdasarkan komentar yang paling baru.



Gambar 5.a Request Header



Gambar 5.b Form Data

Semua atribut pada *request header* harus sama persis dengan atribut *request header* pada *web crawler*, kecuali untuk atribut *referer*. Pada atribut *referer*, *value* yang dimasukkan adalah alamat URL dari aplikasi yang komentar reviewnya akan diambil. Alamat tersebut dapat diketahui dari nama *package* dari aplikasi yang akan diambil komentarnya yang diawali dengan teks "https://play.google.com/store/ apps/details?id=". Misalkan aplikasi Tokopedia, nama *package*-nya adalah "com.tokopedia.tkpd" jadi *referer value* untuk *request header*-nya adalah "https://play.google.com/store/apps/details?id= com.tokopedia.tkpd". Adapun atribut pada form data untuk pengambilan komentar terdiri dari lima, dua diantaranya memiliki *value* yang paten. Atribut yang nilainya dapat dibuah sesuai kebutuhan adalah:

- id : atribut id aplikasi, *value*-nya adalah nama package
- pageNum : atribut halaman dari komentar, *value*-nya bilangan bulat dimulai 0, satu halaman terdiri dari 40 komentar

- reviewSortOrder : atribut urutan komentar, *value*-nya bilangan bulat mulai dari 0 sampai 2. Value 0 untuk mengurutkan komentar berdasarkan yang paling baru, value 1 berdasarkan nilai rating komentar paling tinggi, dan value 2 berdasarkan kegunaan.

Setelah request AJAX disimulasikan pada web crawler, maka data yang diinginkan dapat diambil. Adapun seluruh atribut yang diambil berserta selector-nya dapat dilihat pada Tabel 1.

Tabel 2. Atribut dan *Selector* dari *web crawler* Get Text Review Service

No.	Nama Atribut	Selector
1.	Text Review	.review-body
2.	Tanggal	.review-date
3.	Nama User	.review-info>.author-name
4.	Rating	.review-info-star-rating>.tiny-star::attr(aria-label)

Web service dapat diakses dengan memasukkan alamat http://[ip-address]/GetReview pada REST *client* dengan method "POST" dan membutuhkan dua parameter. Parameter pertama adalah "package", value-nya merupakan kata kunci berupa nama package dari aplikasi yang akan diambil komentar *review*-nya dan parameter "jmlIterasi", *value*-nya berupa bilangan bulat, satu iterasi dapat mengambil maksimal 40 komentar.

2.3 Implementasi dan Pengujian

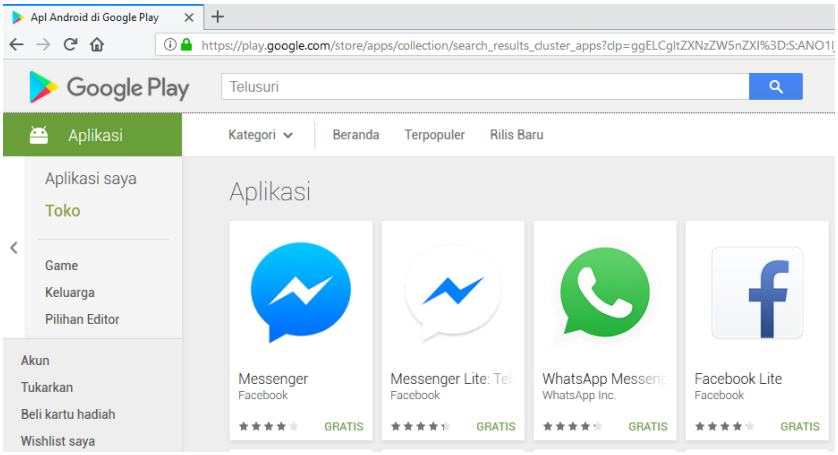
Pada penelitian ini, pembuatan *Web Crawler* berbasis *Web Service* pada Website Google Play Store diimplementasikan dengan bahasa pemrograman Python dengan memanfaatkan library Scrapy dan Flask. *Library* Scrapy digunakan untuk pembuatan *web crawler*, sedangkan library Flask untuk pembuatan *web service*-nya. Proses pengujian menggunakan metode *white box testing* dengan membandingkan *website* asli dengan hasil *crawling*/*scraping* dari sistem yang dibangun, lebih lanjut tentang pengujian dibahas pada sub bab selanjutnya.

3. Hasil dan Pembahasan

Tahap ini merupakan tahap pengujian sistem untuk mengetahui apakah sistem yang telah dibuat bekerja sesuai *requirement* yang terdapat pada tahap analisis sistem. *Web crawler* yang dibangun ditujukan untuk kebutuhan para *software developer* atau para peneliti di bidang *data mining*, maka untuk pengujian sistemnya menggunakan *white box testing*. Sebab *white box testing* merupakan pengujian yang diambil dari sudut pandang *developer*, berbeda dengan *black box testing* yang hanya melakukan pengujian berdasarkan sudut pandang *end user*-nya.

Tahapan pengujian terhadap aplikasi menggunakan *white box testing*, merupakan pengujian yang biasanya dilakukan oleh developer aplikasi. Pengujian white box digunakan untuk mendeteksi kesalahan logis dalam kode program. Hal ini digunakan untuk debugging kode dan menemukan asumsi pemrograman yang salah [10]. Pengujian white box untuk aplikasi dilakukan dengan melakukan *test case* [11], yaitu mencocokkan data yang ditampilkan pada *website* Google Play Store dengan hasil yang didapatkan oleh aplikasi.

Pengujian pertama dilakukan dengan menggunakan kata kunci "messenger", hasil yang ditampilkan oleh website Google Play Store dapat dilihat pada Gambar 6.

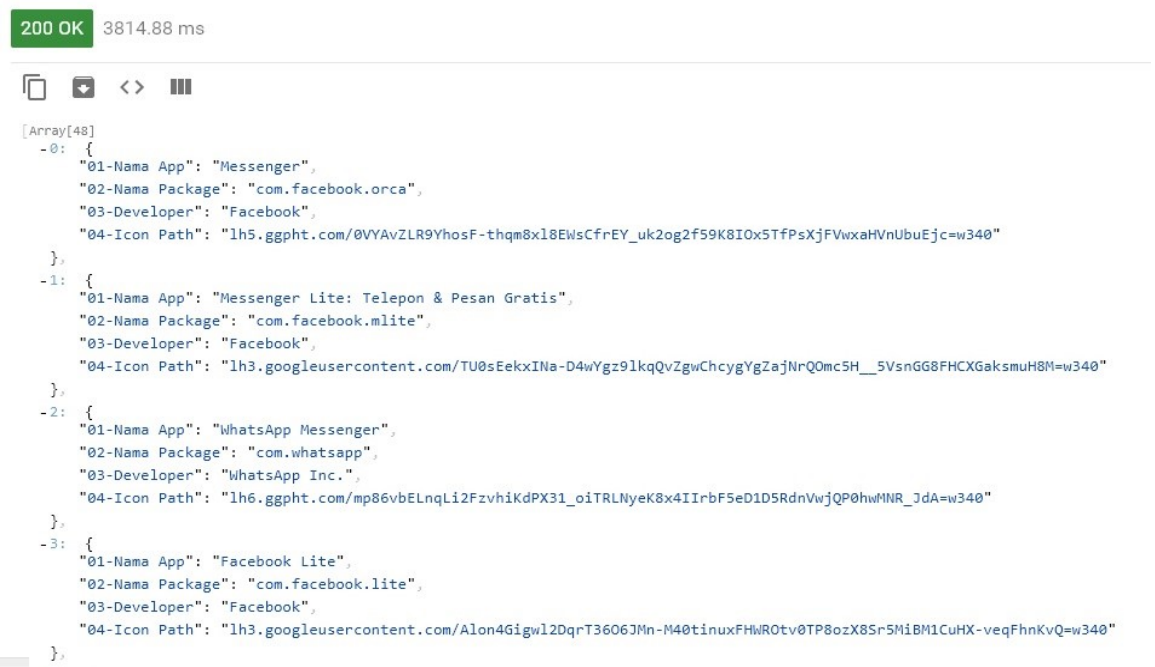


Gambar 6 Hasil Pencarian pada Website Resmi Google Play Store

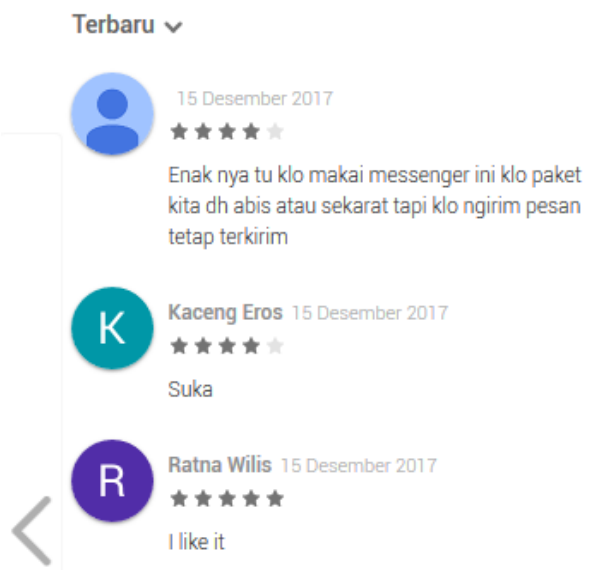
Pada Gambar 6 terlihat bahwa, hasil pencarian aplikasi dengan kata kunci “messenger”, menghasilkan beberapa *record*. Adapun hasil dari pencarian melalui aplikasi yang dibagun dapat dilihat pada Gambar 7.

Aplikasi yang dibangun, diuji menggunakan REST *client* yang tersedia sebagai *Adds-On* pada *web browser* Google Chrome. Ketika hasilnya dibandingkan dengan website asli dari Google Play Store, maka hasilnya adalah sama.

Pengujian kedua dilakukan untuk menguji, apakah *text review* dari aplikasi yang terdapat pada *website* Google Play Store sama dengan hasil dari aplikasi yang dibagun. Pengujian kedua ini mengambil *text review* dari sebuah aplikasi, misal: aplikasi Messenger dari facebook yang *package name*-nya adalah com.facebook.orca. Lima komentar terbaru dari aplikasi tersebut dapat dilihat pada Gambar 8.



Gambar 7 Hasil *Crawling* dari *App Search Service*



Gambar 8 *Text Review* pada *Website* Google Play Store


```
[Array[40]
 -0:  {
      "Nama User": "",
      "Rating": "4",
      "Tanggal": "15 Desember 2017",
      "Text Review": "Enak nya tu klo makai messenger ini klo paket kita dh abis atau sekarat tapi klo ngirim pesan tetap terkirim"
  },
 -1:  {
      "Nama User": "Kaceng Eros",
      "Rating": "4",
      "Tanggal": "15 Desember 2017",
      "Text Review": "Suka"
  },
 -2:  {
      "Nama User": "Ratna Wilis",
      "Rating": "5",
      "Tanggal": "15 Desember 2017",
      "Text Review": "I like it"
  },
```


Gambar 9 Hasil *Crawling* dari *Get Text Review Service*

Kemudian hasil yang didapatkan oleh aplikasi dapat dilihat pada Gambar 9. Terlihat bahwa hasil dari *website* Google Play Store dan dari aplikasi yang dibangun, memiliki hasil sama. Dengan demikian, pengujian ini dapat dijadikan sebagai tolak ukur bahwa aplikasi yang dibangun bekerja dengan baik dan benar.

4. Kesimpulan dan Saran

Berdasarkan dari hasil penelitian dan pembahasan yang dilakukan maka diperoleh kesimpulan bahwa *Web crawler* yang dibangun berhasil mengambil data pada *website* Google Play Store dengan baik dan benar sesuai dengan kebutuhan pada analisis sistem dan *Web Crawler* yang dibangun dapat diintegrasikan dengan *web service* berbasis REST untuk mendukung penggunaan sistem secara *cross platform*. Dari Hasil yang didapatkan, masih terdapat banyak kekurangan, seperti library flask yang digunakan tidak mampu untuk menangani *request* secara konkuren. Diharapkan pada penelitian selanjutnya masalah ini dapat diatasi dengan *deploy* aplikasi pada WSGI *server* yang berbeda agar dapat menerima *request* dari *client* yang berbeda pada saat yang sama. Kemudian diharapkan aplikasi ini nantinya dapat dijadikan sebagai *library* untuk python yang menyediakan API untuk mengakses data-data yang terdapat pada Google Play Store.

Daftar Pustaka

[1] S. S. Dhenakaran and K. T. Sambanthan, “Web Crawler - an Overview,” *Int. J. Comput. Sci. Commun.*, vol. 2, no. 1, pp. 265–267, 2011.

[2] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st Editio., vol. 43. Sebastopol: O’Reilly, 2009.

[3] C. Wong, K. Wong, K. Ng, W. Fan, and K. Yeung, “Design of a Crawler for Online Social Networks Analysis,” *WSEAS Trans. Commun.*, vol. 13, pp. 263–274, 2014.

[4] S. A. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, “Crawling Facebook for Social Network Analysis Purposes,” pp. 0–7, 2011.

[5] Amudha and M. Phil, “Web Crawler for Mining Web Data,” *Int. Res. J. Eng. Technol.*, vol. 4, no. 2, pp. 128–136, 2017.

[6] M. S. Ahuja, J. S. Bal, and Varnica, “Web Crawler : Extracting The Web Data,” *Int. J. Comput. Trends Technol.*, vol. 13, no. 3, pp. 132–137, 2014.

[7] S. Chakrabarti, “Mining the Web Discovering Knowledge from Hypertext Data,” 2002.

[8] R. Kneuper, “Sixty years of software development life cycle models,” *IEEE Ann. Hist. Comput.*, vol. 39, no. 3, pp. 41–54, 2017.

[9] G. Song, “The Reconstruction Pattern of MVC,” *Int. J. u- e- Serv. Sci. Technol.*, vol. 7, no. 2, pp. 147–156, 2014.

[10] S. Nidhra, “Black Box and White Box Testing Techniques - A Literature Review,” *Int. J. Embed. Syst. Appl.*, vol. 2, no. 2, pp. 29–50, 2012.

[11] M. Kumar, S. K. Singh, and R. . Dwivedi, “A Comparative Study of Black Box Testing and White Box Testing Techniques,” *Int. J. Adv. Res. Comput. Sci. Manag. Stud.*, vol. 3, no. 10, pp. 32–44, 2015.

