TUGAS AKHIR – KI141502

APLIKASI ANDROID 'BEAI' TENTANG PERHITUNGAN BIAYA PUNGUTAN UNTUK BARANG YANG DIBAWA PENUMPANG DARI LUAR NEGERI

ANDRE EXAUDI JEREMY RUMAPEA
NRP 05111440000031

Dosen Pembimbing I
Abdul Munif S.Kom, M.Sc.Eng

Dosen Pembimbing II
Nurul Fajrin Ariyani, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KI141502

APLIKASI ANDROID 'BEAI' TENTANG PERHITUNGAN BIAYA PUNGUTAN UNTUK BARANG YANG DIBAWA PENUMPANG DARI LUAR NEGERI

ANDRE EXAUDI JEREMY RUMAPEA
NRP 05111440000031

Dosen Pembimbing I
Abdul Munif S.Kom, M.Sc.Eng

Dosen Pembimbing II
Nurul Fajrin Ariyani, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

ANDROID APPLICATION 'BEAI' ABOUT TAXATION CALCULATION FOR GOODS BROUGHT BY PASSENGERS FROM FOREIGN COUNTRIES

ANDRE EXAUDI JEREMY RUMAPEA
NRP 051111440000042

Supervisor I
Abdul Munif S.Kom, M.Sc.Eng.

Supervisor II
Nurul Fajrin Ariyani, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

APLIKASI ANDROID 'BEAI' TENTANG PERHITUNGAN BIAYA PUNGUTAN UNTUK BARANG YANG DIBAWA PENUMPANG DARI LUAR NEGERI

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh
ANDRE EXAUDI JEREMY RUMAPEA
NRP : 05111440000031

Disetujui oleh Dosen Pembimbing Tugas Akhir :

1. Abdul Munif S.Kom, M.Sc.Eng. .....
   NIP: 198608232015041004 (Pembimbing 1)

2. Nurul Fajrin Ariyani, S.Kom., M.Sc. .....
   NIP: 198607222015042003 (Pembimbing 2)

SURABAYA
JUNI, 2018

[Halaman ini sengaja dikosongkan]

APLIKASI ANDROID 'BEAI' TENTANG PERHITUNGAN BIAYA PUNGUTAN UNTUK BARANG YANG DIBAWA DARI LUAR NEGERI

Nama Mahasiswa	**:**	**ANDRE EXAUDI JEREMY RUMAPEA**
NRP	**:**	**05111440000031**
Jurusan	**:**	**Informatika FTIK-ITS**
Dosen Pembimbing 1	**:**	**Abdul Munif S.Kom, M.Sc.Eng**
Dosen Pembimbing 2	**:**	**Nurul Fajrin Ariyani, S.Kom., M.Sc.**

Abstrak

BEAI merupakan aplikasi Android untuk membantu para wisatawan dalam memperkirakan jumlah pajak yang perlu mereka bayar saat membawa pulang barang yang mereka beli di luar negeri. Terdapat aplikasi yang mirip dengan BEAI yang dikeluarkan oleh Direktorat Jenderal Bea dan Cukai, yaitu CEISA Mobile. Namun aplikasi tersebut hanya dapat menghitung pajak yang harganya sudah diketahui oleh pengguna dan tidak menyimpan input apapun dari pengguna.

Pada aplikasi BEAI, pengguna dapat memilih opsi kalkulasi yang dilakukan, apakah harga barang yang diinput sudah diketahui oleh penumpang atau belum. Pengguna dapat memilih untuk menginput sendiri harga barang atau pengguna dapat memilih untuk menggunakan harga yang didapat dari hasil webscraping situs e-commerce seperti Lazada dan Amazon, supaya mendapatkan perkiraan harga barang yang akan dibeli dengan biaya pajak yang perlu dibayar. Pengguna juga dapat melihat historis dari input-input yang dilakukan sebelumnya, sehingga tidak perlu mengingat input yang dilakukan sebelumnya maupun mencari harga barang dari yang sama.

Pada tugas akhir ini, permasalahan tersebut akan ditangani dengan membuat aplikasi BEAI. Aplikasi tersebut memiliki fitur-fitur seperti kalkulator pajak untuk wisatawan yang membawa barang yang dibeli dari luar negeri, baik menggunakan harga dari penumpang maupun online. Fitur lainnya adalah Currency yang menyediakan kurs pajak yang berlaku pada saat ini, dan History yang menyimpan historis penginputan data dari penumpang dan harga yang diterima dari server. Diharapkan dengan adanya aplikasi ini, wisatawan yang membawa pulang barang yang dibeli di luar negeri dapat memperkirakan biaya pajak yang perlu dibayar saat kembali pulang ke Indonesia.

Berdasarkan hasil pengujian aplikasi dengan metode blackbox, dapat disimpulkan fitur yang dibangun berfungsi dengan baik serta kegunaan dari fitur tercapai.

Kata kunci: Aplikasi Android, Webscraping, Perhitungan Pajak

ANDROID APPLICATION 'BEAI' ABOUT TAXATION CALCULATION FOR GOODS BROUGHT BY PASSENGERS FROM FOREIGN COUNTRIES

Student's Name	**: ANDRE EXAUDI JEREMY RUMAPEA**
Student's ID	**: 05111440000031**
Department	**: Informatics FTIK-ITS**
First Advisor	**: Abdul Munif S.Kom, M.Sc.Eng**
Second Advisor	**: Nurul Fajrin Ariyani, S.Kom., M.Sc.**

Abstract

BEAI is an Android app to assist travelers in estimating the amount of taxes they need to pay when bringing home the goods they bought abroad. Applications similar to BEAI issued by the Direktorat Jenderal Bea dan Cukai, namely CEISA Mobile. However the app can only calculate the tax which price is already known to the user and does not store any input from the user.

In BEAI application, the user can choose the calculation option done, whether the price of the inputed goods is known by the passenger or not. Users can choose to input their own price of goods or users can choose to use the price obtained from webscraping e-commerce sites such as Lazada and Amazon, in order to get an estimate of the price of goods to be purchased with tax costs to pay. Users can also see the history of the inputs made before, so they do not need to remember the input made before and look again for the price of same goods.

On this undergraduated thesis, those problem will be handled by application of BEAI. The app has features such as a tax calculator for travelers carrying goods purchased from

overseas, using either the price of passengers or online. Another feature is Currency that provides the current tax rate, and History that stores the history of inputting data from passengers and the price received from the server. It is expected that with this application, tourists who bring home goods purchased abroad can estimate the tax costs to be paid when returning home to Indonesia.

Based on the results of testing the application with blackbox method, it can be concluded that the built features work well and the usefulness of the features achieved.

Keywords: Android Application, Webscraping, Tax Calculation

KATA PENGANTAR

Puji syukur saya ucapkan kepada Tuhan Yang Maha Esa, yang atas izin dan karunianya penulis dapat menyelesaikan Tugas Akhir yang berjudul **“APLIKASI ANDROID 'BEAI' TENTANG PERHITUNGAN BIAYA PUNGUTAN UNTUK BARANG YANG DIBAWA DARI LUAR NEGERI”.**

Dalam pengerjaan tugas akhir ini, penulis mendapatkan banyak sekali ilmu baru dan memperdalam ilmu-ilmu yang sebelumnya telah diajarkan selama masa perkuliahan di Departemen Informatika ITS.

Terselesaikannya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Tuhan Yang Maha Esa yang selalu memberikan berkah sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Keluarga penulis yang senantiasa menyemangati dan menceriakan penulis selama pengerjaan tugas akhir.
3. Bapak Abdul Munif S.Kom, M.Sc.Eng selaku pembimbing I yang telah membantu, membimbing, dan memberikan ilmunya kepada penulis dalam menyelesaikan tugas akhir ini.
4. Nurul Fajrin Ariyani, S.Kom., M.Sc**.** selaku pembimbing II yang juga telah membantu, membimbing , dan memberikan ilmunya kepada penulis dalam menyelesaikan tugas akhir ini.
5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.

6. Teman-teman yang membantu baik penulisan dan pencarian bahan maupun moral dalam pengerjaan tugas akhir: Syauki, Fandy, Dzaky.
7. Adik-adik jurusan Departemen Informatika dan UKM IFLS yang telah menemani penulis selama pengerjaan tugas akhir.
8. Serta pihak lain yang telah membantu penulis menyelesaikan tugas akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2018

Andre Exaudi Jeremy Rumapea

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada era globalisasi ini pertumbuhan penumpang yang keluar-masuk Indonesia sangat pesat sekali. Penumpang yang masuk Indonesia ini pasti membawa barang-barang, baik dari yang diperoleh di luar negeri maupun yang dibawa saat keluar Indonesia. Namun, seringkali penumpang tidak mengetahui peraturan bea masuk yang diberlakukan, sehingga terjadi kesalahpahaman antara penumpang dengan petugas bea cukai. Hal ini disebabkan karena beberapa faktor; kurangnya sosialisasi peraturan yang berlaku, jumlah peraturan yang cukup banyak sehingga perlu dibaca secara cermat, dan kesulitan dalam mencari tahu peraturan yang berlaku.

Aplikasi 'BEAI' adalah aplikasi Android yang dapat membantu penumpang yang membawa barang dari luar negeri ini untuk mengetahui total pungutan yang perlu dibayarkan saat membawa barang yang dibeli di luar negeri. Untuk menentukan jumlah bea impor yang perlu dibayarkan, aplikasi ini akan mengirimkan request ke server, dimana server akan mencari dari *database* yang dibuat dari hasil web scraping harga barang yang bersangkutan di situs-situs e-commerce yang menjual produk tersebut. Harga barang yang dianggap paling cocok akan dikirim kembali ke aplikasi untuk diolah dan memberikan output estimasi biaya pungutan.

Tujuan dari pengerjaan Tugas Akhir ini adalah mampu menghasilkan aplikasi yang dapat mempermudah penggunanya untuk memperkirakan biaya pungutan yang perlu dibayar, tanpa harus mengecek terlebih dahulu ke situs-situs e-commerce yang menjual barang tersebut.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana merancang *spider* untuk melakukan *web scraping* untuk mendapatkan harga pada situs *e-commerce* serta menyimpannya ke dalam basis data?
2. Bagaimana merancang *spider* untuk melakukan *web scraping* untuk mendapatkan nilai kurs yang berlaku pada saat perhitungan biaya pungutan dilakukan?
3. Bagaimana merancang aplikasi BEAI supaya dapat mengambil harga barang yang akan dihitung biaya pungutannya dan kurs yang berlaku dari basis data dan dapat menggunakan harga dari nota yang diinput oleh pengguna?

1.3 Batasan Masalah

Permasalan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

1. Jumlah situs *e-commerce* yang akan di-*scrape* adalah dua, yaitu Lazada dan Amazon. Lazada merupakan situs *e-commerce* yang berbasis di negara-negara Asia Tenggara, sehingga akan digunakan sebagai sumber harga untuk barang-barang yang dibeli di negara-negara Asia Tenggara. Amazon merupakan salah satu situs *e-commerce* terbesar, dan akan digunakan sebagai sumber harga barang yang dibeli dari negara-negara di luar Asia Tenggara.
2. Pada situs Lazada, barang yang akan di-*scrape* hanya halaman pertama per kategori. Sementara, pada situs Amazon, barang yang akan di-*scrape* lima halaman pertama per kategori dan sub kategori.
3. Harga barang serta barang yang berada di basis data server akan diperbarui dalam periode tertentu untuk mencegah terkena IP *ban* dari situs-situs *e-commerce* yang akan di-*scrape*.
4. Apabila barang yang dicari tidak ada, sistem akan melakukan *crawling* harga terlebih dahulu sesuai kata kunci yang diminta pengguna pada situs yang sesuai dengan negara yang

dikunjungi. Sistem kemudian akan melakukan *push notification* kepada aplikasi pengguna bahwa harga telah ditemukan setelah *crawling* selesai dilakukan. C*rawling* pada situs Amazon akan dilakukan hanya halaman pertama sementara untuk situs Lazada akan dilakukan pada dua halaman pertama.

5. Informasi kurs yang berlaku pada hari itu akan diambil dari situs resmi Badan Kebijakan Fiskal Kementrian Keuangan RI, yaitu http://bcsemarang.beacukai.go.id. Informasi kurs akan diperbarui setiap pukul 8.05 WIB.

1.4 Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah sebagai berikut:

1. Merancang aplikasi Android yang dapat memperkirakan estimasi biaya pungutan saat membawa barang ke Indonesia yang dibeli di luar negeri.
2. Merancang *web scraper* untuk mendapatkan harga dari situs *e-commerce* dan kurs yang berlaku yang kemudian disimpan hasilnya di basis data server.

1.5 Manfaat

Manfaat yang diharapkan dari pembuatan Tugas Akhir ini adalah terciptanya suatu aplikasi yang dapat mempermudah masyarakat yang membawa pulang barang yang dibeli saat berada di luar negeri ke Indonesia untuk mengetahui estimasi jumlah biaya yang dibayar saat mengurus perbeacukaian di Indonesia.

1.6 Metodologi

1.6.1 Penyusunan proposal tugas akhir

Proposal tugas akhir ini berisi deskripsi pendahuluan tugas akhir yang akan dikerjakan. Bagian pendahuluan terdiri dari latar belakang tugas akhir, rumusan masalah, batasan masalah, tujuan pengerjaan tugas akhir, dan manfaat dari hasil pengerjaan tugas akhir. Selain itu dalam proposal tugas akhir juga dijabarkan tinjauan pustaka yang digunakan sebagai referensi pendukung pengerjaan tugas akhir. Dalam proposal ini juga terdapat penjelasan mengenai metodologi yang dipakai, mulai dari tahap penyusunan proposal hingga penyusunan buku Tugas Akhir. Terdapat juga sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu perundangan PMK Nomor: 203/PMK.04/2017, bahasa pemrograman Python dan Java, PHP, JSON, Scrapy sebagai kerangka dalam *web scraping*, RESTful Web Service, dan Volley sebagai pustaka Android untuk mentransmisikan data.

1.6.3 Analisis dan desain perangkat lunak

Pada tahap ini akan dilakukan analisis dan perancangan desain aplikasi BEAI, serta perancangan *web scraping* yang akan dilakukan terhadap situs *e-commerce* yang akan ditentukan. Analisis dilakukan dengan menentukan kebutuhan fungsional sistem. Perancangan desain aplikasi serta *web scraper* dengan membuat *database* untuk menyimpan hasil *scraping*, membuat tampilan aplikasi, diagram-diagram yang dibutuhkan, serta menghubungkan server dengan aplikasi:

1.6.4 Implementasi perangkat lunak

Aplikasi ini akan dibangun dengan menggunakan kakas bantu antara lain bahasa pemrograman Java, bahasa pemrograman Python, bahasa pemrograman PHP, JSON, kerangka Scrapy, kerangka CodeIgniter, pustaka Volley, dan aplikasi Anroid Studio.

1.6.5 Pengujian dan evaluasi

Pada tahapan ini akan dilakukan uji coba terhadap fungsionalitas aplikasi BEAI, yaitu dengan sebagai berikut:

- Perhitungan cukai, melalui pengambilan data harga dan perhitungan biaya pungutan secara manual dari situs *e-commerce*, kemudian mengecek apakah harga tersebut berada dalam perhitungan otomatis biaya pungutan yang perlu dibayarkan yang didapat dari estimasi ataupun nota.
- Pengujian black box terhadap masukan dan keluaran yang dihasilkan berdasarkan skenario yang telah ditentukan

1.6.6 Penyusunan buku tugas akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir
    d. Tujuan
    e. Metodologi
    f. Sistematika Penulisan

2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi tentang desain sistem yang disajikan dalam bentuk *pseudocode*.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *code* yang digunakan untuk proses implementasi.

Bab V Uji Coba dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

2.1 PMK Nomor: 203/PMK.04/2017

PMK Nomor: 203/PMK.04/2017 mengatur tentang Ketentuan Ekspor dan Impor Barang yang Dibawa oleh Penumpang dan Awak Sarana Pengangkut [1]. Menurut PMK ini, barang impor yang dibawa penumpang wajib diberitahukan kepada pejabat bea dan cukai di kantor pabean, yang dapat dilakukan secara lisan atau secara tertulis pada tempat tertentu yang ditetapkan oleh Direktur Jenderal, sesuai dengan Pasal 4 ayat (2) PMK yang berbunyi:

Pemberitahuan sebagaimana dimaksud pada ayat (1), disampaikan kepada Pejabat Bea dan Cukai yang ditunjuk untuk mengawasi barang yang dibawa oleh Penumpang di terminal keberangkatan internasional dalam bentuk: a. data elektronik; atau b. tulisan di atas formulir.

Barang pribadi penumpang dengan nilai pabean paling banyak FOB atau *Free On Board* lima ratus dolar AS per orang untuk setiap kedatangan akan diberikan pembebasan bea masuk. Barang pribadi penumpang didefinisikan pada Pasal 7 ayat (1) dan (3) yang berbunyi:

(1) Barang impor bawaan Penumpang atau barang 1mpor bawaan Awak Sarana Pengangkut terdiri atas: a. barang pribadi Penumpang atau barang pribadi Awak Sarana Pengangkut yang dipergunakan/ dipakai untuk keperluan pribadi termasuk sisa perbekalan (*personal use*); dan/atau b. barang impor yang dibawa oleh Penumpang atau barang impor yang dibawa oleh Awak Sarana Pengangkut selain barang pribadi sebagaimana dimaksud pada huruf a (*non-personal use*).

(3) Barang impor sebagaimana dimaksud pada ayat (1) huruf a terdiri atas: a. barang yang diperoleh dari luar

Daerah Pabean dan tidak akan dibawa kembali ke luar Daerah Pabean; b. barang yang diperoleh dari dalam Daerah Pabean; dan/atau c. barang yang diperoleh dari luar Daerah Pabean, yang akan digunakan selama berada di Daerah Pabean dan akan dibawa kembali pada saat Penumpang atau Awak Sarana Pengangkut meninggalkan Daerah Pabean.

Barang pribadi penumpang dapat diartikan sebagai semua jenis barang yang dibawa oleh penumpang baik itu keperluan pribadi maupun selain keperluan pribadi yang diperoleh dari luar negeri. Kelebihan dari batasan yang ditetapkan pada Pasal 12 ayat (1) dan (2) akan dipungut bea masuk dan pajak dalam rangka impor, yang berbunyi:

(1) Terhadap barang pribadi Penumpang sebagaimana dimaksud dalam Pasal 7 ayat (1) huruf a yang diperoleh dari luar Daerah Pabean sebagaimana dimaksud dalam Pasal 7 ayat (3) huruf a, dengan nilai pabean paling banyak FOB USD500.00 (lima ratus United States Dollar) per orang untuk setiap kedatangan, diberikan pembebasan bea masuk.

(2) Dalam hal nilai pabean barang pribadi Penumpang sebagaimana dimaksud dalam Pasal 7 ayat (1) huruf a yang diperoleh dari luar Daerah Pabean sebagaimana dimaksud dalam Pasal 7 ayat (3) huruf a melebihi batas nilai pabean sebagaimana dimaksud pada ayat (1), atas kelebihan tersebut dipungut bea masuk dan pajak dalam rangka impor.

Dari Pasal 12 ayat (1) dan (2), nilai pabean barang bawaan penumpang mendapatkan pembebasan bea masuk sebesar lima ratus dollar AS, sementara yang memiliki nilai pabean melebihi FOB lima ratus dolar AS berlaku ketentuan sebagai berikut sesuai dengan Pasal 24 ayat (1) yang berbunyi:

Terhadap barang 1mpor bawaan Penumpang sebagaimana dimaksud dalam Pasal 7 ayat (1) huruf a yang memiliki nilai pabean melebihi FOB USD500.00 (lima ratus United States

Dollar), berlaku ketentuan sebagai berikut: a. tarif bea masuk ditetapkan sebesar 10% (sepuluh persen); dan b. nilai pabean ditetapkan berdasarkan keseluruhan nilai pabean barang impor bawaan Penumpang dikurangi dengan FOB USD500,00 (lima ratus United States Dollar).

2.2 Web Scraping Menggunakan Kerangka Scrapy

Web Scraping adalah *data scraping* untuk mengekstraksi data dari situs web [2]. Perangkat lunak *web scraping* dapat mengakses World Wide Web secara langsung menggunakan HTTP atau menggunakan web browser. Data yang diekstrak akan dikumpulkan dan disimpan dalam suatu *database* atau *spreadsheet* yang bisa diambil untuk ke depannya.

Scrapy adalah kerangka kerja aplikasi untuk *crawling* situs-situs web dan mengekstrak data (*scrap)* yang dapat digunakan untuk *data mining*, pemrosesan informasi, atau pengarsipan sejarah [3]. Scrapy menggunakan bahasa pemrograman Python. Python adalah bahasa pemrograman interpretative berorientasi obyek yang interaktif, mampu menggabungkan modul, *exception*, *dynamic typing*, tipe data dinamis tingkat tinggi, dan kelas [4].Scrapy memiliki beberapa komponen [5]:

- *Scrapy Engine*, bertanggung jawab atas aliran data antar semua komponen pada sistem dan memicu *event* saat suatu perilaku tertentu terjadi.
- *Scheduler*, menerima permintaan dari *engine* dan memberi umpan kepada mereka.
- *Downloader*, bertanggung jawab dalam mengunduh halaman web dan memberikan mereka ke *engine* dan kemudian diberikan kepada *spider*.
- *Spiders*, kelas yang ditulis oleh pengguna Scrapy untuk mengurai respon dan mengekstrak barang dari respon tersebut, tambahan URL, atau permintaan yang diikuti. Setiap *spider* mampu menangani domain yang spesifik.

- *Item Pipeline*, bertanggung jawab dalam memproses barang begitu mereka sudah berhasil diekstrak oleh *spider*, yaitu dengan menyimpan mereka ke dalam *database*, membersihkan serta memvalidasi.
- *Downloader Middleware*, memproses permintaan yang berasal dari *engine* ke *downloader*, serta respon yang berasal dari *downloader* ke *engine*.
- *Spider Middleware*, memproses permintaan yang berasal dari *engine* ke *spider*, serta respon yang berasal dari *downloader* ke *spider*.

Aliran data pada Scrapy yang diatur oleh *engine* sebagai berikut:

1. *Engine* membuka domain, menempatkan *spider* yang menangani domain tersebut, dan meminta URL pertama untuk di-*crawl*.
2. *Engine* mendapatkan URL pertama untuk di-*crawl* dan menjadwalkan mereka pada *scheduler*, sesuai permintaan.
3. *Engine* meminta *scheduler* URL selanjutnya untuk di-*crawl*.
4. *Scheduler* membalas dengan URL selanjutnya untuk di-*crawl*, *engine* mengirimkannya ke *downloader* melewati *downloader middleware*.
5. Begitu halaman selesai diunduh, *downloader* mengirimkan respon ke *engine* melewati *downloader middleware*.
6. *Engine* menerima respon dari *downloader* dan mengirimkannya ke *spider* untuk diproses melewati *spider middleware*.
7. *Spider* memproses respon dan membalas dengan barang hasil *scrap* dan permintaan baru ke *engine*.
8. *Engine* mengirim barang hasil *scrap* ke *item pipeline* dan permintaan baru ke *scheduler*.
9. Kembali ke langkah dua dan terus berulang hingga tidak ada permintaan lagi dari *scheduler* dan *engine* menutup domain.

2.3 *Volley*

Volley adalah pustaka HTTP untuk membuat perancangan jaringan untuk aplikasi Android lebih mudah dan cepat, dengan beberapa keuntungan yang ditawarkan sebagai berikut [6]:

- Penjadwalan otomatis permintaan jaringan (*network request*).
- Koneksi jaringan yang banyak secara bersamaan.
- Mendukung pemrioritasan permintaan.
- API untuk pembatalan permintaan.
- Pengurutan yang kuat sehingga mudah mengisi UI dengan data yang diambil secara asinkron dari jaringan.
- Perangkat untuk *debug* dan *tracing*.

Pada aplikasi Android yang akan dibuat, Volley akan digunakan untuk komunikasi data antara aplikasi dengan server dimana basis data harga berada. Oleh karena itu, dibutuhkan izin `android.permission.INTERNET` sehingga aplikasi dapat terhubung dengan jaringan internet. Kemudian, ada beberapa kelas pada Volley yang penting:

- RequestQueue, antrian yang berisi Network/HTTP *Request*.
- Request, kelas yang berisi informasi jaringan seperti metode HTTP yang akan digunakan, dimana
- StringRequest, HTTP *request* dimana respon akan di-*parse* dalam *string*.
- JsonObjectRequest, HTTP *request* dimana respon dalam bentuk JSONObject.

Aplikasi akan melakukan transmisi data dalam ke server dengan langkah-langkah berikut:

a. Membuat *RequestQueue*, yang memegang HTTP *Request*, dengan contoh penggunaan sebagai berikut:

```
1.  RequestQueue queue = Volley.newRequestQueue(this);
```

b. Menentukan URL yang akan kita tetapkan sebagai tempat mengirim HTTP *request*, dengan contoh penggunaan sebagai berikut

```
1.  String url = "http://BEAI.com/cekharga.json";
```

c. Menentukan metode HTTP yang digunakan (GET, POST, PUT) dalam hal ini akan menggunakan GET.
d. Menentukan parameter apabila HTTP *request* yang dilakukan berhasil dan gagal, kemudian membuat JsonObjectRequest dengan contoh penggunaan sebagai berikut:

```
1.  JsonObjectRequest jsonObjReq = new JsonObjectRequest(M
    ethod.GET,         url, null,         new Response.Lis
    tener() {
2.  @Override
    public void onResponse(JSONObject response) {
        Log.d("Response", response); //Success Callback
3.    }
4.  },  new Response.ErrorListener() {
5.  @Override
    public void onErrorResponse(VolleyError error) {
6.  Log.d("Error.Response", response); //Failure Callback
7.    }
8.  });
```

e. Menaruh JsonObjectRequest ke dalam barisan dengan contoh penggunaan sebagai berikut:

```
1.  queue.add(jsonObjReq);
```

2.4 RESTful Web Service Menggunakan PHP dan JSON

REST (Representation State Transfer) atau RESTful *Web Service* adalah salah satu layanan antar sistem komputer melalui internet, dimana layanan mengizinkan sistem yang meminta dapat mengakses dan memanipulasi representasi tekstual dari sumber daya web menggunakan operasi *stateless* yang seragam dan sudah didefinisikan sebelumnya [7]. Pada layanan web RESTful,

permintaan dibuat pada URI sumber daya akan menghasilkan respon dalam XML, HTML, JSON atau dalam bentuk format lain.

PHP (PHP: Hypertext Preprocessor) adalah bahasa scripting pada sisi server yang umumnya digunakan untuk pengembangan web dan bisa ditanamkan pada HTML [8]. File PHP dapat berisi text, HTML, CSS, Javascript, dan kode PHP. Kode PHP dieksekusi pada server dan hasilnya dikembalikan ke aplikasi dalam bentuk JSON. PHP dapat menambah, menghapus, dan memodifikasi data pada *database*, mengontrol akses pengguna, dan juga dapat mengenkripsi data [9].

JSON (JavaScript Object Notation) adalah format pertukaran data ringan yang mudah untuk dibaca dan ditulis oleh manusia, namun tetap mudah untuk diurai dan dibuat oleh mesin juga [10]. JSON dibuat dalam dua struktur yaitu koleksi dan daftar. Struktur koleksi dari pasangan nama atau nilai, dapat berupa obyek, *record*, *struct*, *dictionary*, *hash table*, *keyed list* yang akan diterapkan pada Tugas Akhir ini. Berikut adalah contoh data yang akan diterima aplikasi:

```
1.  {
2.      "Mata Uang": "SGD",
3.      "Harga": "589.00"
4.  }
```

Dalam Tugas Akhir ini, bahasa pemrograman PHP akan digunakan untuk mengimplementasikan bagaimana menangani permintaan yang dikirim oleh aplikasi dan pemberian respon dalam format JSON pada RESTful *Web Service*.

2.5 Aplikasi Sejenis

Aplikasi ‘CEISA Mobile’[1] adalah aplikasi Android yang dirilis oleh Direktorat Jenderal Bea dan Cukai dengan tujuan

[1] https://play.google.com/store/apps/details?id=id.go.beacukai.customer&hl=in

membantu masyarakat untuk menyimulasikan perhitungan pungutan atas barang yang diimpor, baik yang dibawa oleh penumpang maupun barang yang dikirim melalui kurir. Aplikasi ini memiliki tiga fitur, yaitu fitur pengecekan status barang kiriman, fitur kalkulator simulasi perhitungan bea masuk dan pajak dalam rangka impor, dan fitur informasi kurs. Fitur pelacakan barang kiriman terhubung langsung dengan pusat data bea cukai, serta membutuhkan nomor resi/AWB yang valid sebagai input. Fitur kalkulator disesuaikan dengan peraturan terbaru yang berlaku. Sementara fitur informasi kurs terhubung langsung dengan pusat data dari Badan Kebijakan Fiskal.

'CEISA Mobile' memiliki beberapa kekurangan, yang pertama harga yang diinput hanya berasal dari pengguna. Hal ini mengakibatkan pengguna setidaknya sudah membeli barangnya dan memiliki struk sebagai acuan harga yang akan dibeli. Sementara pada 'BEAI', pengguna dapat mengestimasikan biaya pajak yang perlu dibayar sebelum membeli barang yang akan dibeli tanpa mengetahui harga pasti barang tersebut. Harga barang yang akan dihitung juga akan menggunakan harga barang yang sebenarnya dari situs-situs *e-commerce*. Pengguna juga tidak perlu mengingat-ingat harga yang pernah diinput sebelumnya, cukup menggunakan fitur **History** yang berisikan nama dan harga barang yang pernah diinput oleh pengguna sebelumnya. Perbandingan antara aplikasi yang dibuat dalam Tugas Akhir ini dengan ‘CEISA Mobile’ ditunjukkan pada Tabel 2.1:

Tabel 2.1 Tabel Perbandingan CEISA Mobile dengan Tugas Akhir

Atribut Pembanding	CEISA Mobile	BEAI
Pelacakan Barang	Ada	Tidak Ada
Sumber Harga	Nota Belanja	Nota Belanja Situs e-commerce

Perhitungan Cukai	Ada	Ada
Estimasi Cukai	Satu, sesuai dengan yang diinput pengguna	Estimasi cukai pada harga barang termurah dan harga barang termahal.
History	Tidak Ada	Ada
Kurs	Ada	Ada

Pada Tabel 2.1 terdapat enam atribut pembanding antara 'CEISA Mobile' dan 'BEAI' yaitu pelacakan barang, sumber harga, perhitungan cukai, estimasi cukai, *history*, dan kurs. 'CEISA Mobile' dan 'BEAI' sama-sama memiliki kurs dan perhitungan cukai, namun 'BEAI' tidak memiliki fitur pelacakan barang. 'BEAI' memiliki dua sumber harga barang sementara 'CEISA Mobile' hanya memiliki satu sumber. Estimasi cukai dari 'CEISA Mobile' juga hanya satu saja sesuai yang diberikan pengguna, sementara 'BEAI' dapat memberikan rentang dari yang termurah hingga termahal. 'BEAI' juga menyimpan harga dan nama barang yang diinput pada fitur **History** sementara 'CEISA Mobile' tidak.

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dibahas analisis dan perancangan perangkat lunak dari aplikasi BEAI ini. Hasil dari proses ini berupa diagram yang akan digunakan sebagai acuan untuk proses implementasi perangkat lunak. Selain digunakan sebagai acuan untuk proses selanjutnya, beberapa diagram hasil dari proses perancangan digunakan sebagai dokumentasi dari implementasi perangkat lunak. Diagram yang dihasilkan pada proses ini disajikan dalam bentuk *Unified Modelling Language* (UML).

3.1 Analisis

Analisis meliputi analisis tampilan situs Lazada dan Amazon, pertimbangan jumlah halaman per kategori, per subkategori dan per pencarian yang akan di-*crawl*, *database* yang akan digunakan, pendeskripsian perangkat lunak secara umum, penggambaran dan penjelasan cara kerja sistem dalam bentuk diagram *flowchart*.

3.1.1 Analisis *Crawling* Situs Lazada dan Amazon

Situs lazada yang akan digunakan adalah situs https://www.lazada.sg/, karena Singapura merupakan salah satu tujuan wisata favorit dan cukup maju di kawasan Asia Tenggara. Oleh sebab itu, jumlah inventaris yang dimiliki akan lebih lengkap dibandingkan situs lazada yang di negara lain. Pada lazada, terdapat sebelas kategori:

- *Electronic Devices*; kategori ini berisi alat-alat elektronik, terdiri dari subkategori *mobile*, tablet, laptop dan lainnya,
- *Electronic Accessories*; kategori ini berisi aksesori-aksesori baik untuk device maupun manusia, terdiri dari subkategori *mobile accessories*, *wearable*, *audio* dan lainnya,

- *TV & Home Appliances*; kategori ini berisi peralatan-peralatan rumah seperti televisi dan peralatan dapur, terdiri dari subkategori *TV & Video devices*, *home audio*, *TV accessories*, dan lainnya,
- Health & Beauty; kategori ini berisi peralatan kecantikan, terdiri dari subkategori *bath & body*, *beauty tools*, *fragrances*, dan lainnya,
- *Babies & Toys*; kategori ini berisi peralatan dan mainan untuk bayi, terdiri dari *baby gear*, *baby personal care*, dan lainnya,
- *Groceries & Pets*; kategori ini berisi makanan, minuman, dan peralatan untuk hewan, terdiri dari subkategori *beverages*, *food staples*, *wines*, dan lainnya,
- *Home & Lifestyle*; kategori ini berisi furnitur, perkakas untuk rumah, terdiri dari subkategori *Tools*, *DIY, & Outdoor*, *Stationary Craft*, dan lainnya,
- *Women's Fashion*; kategori ini berisi fesyen untuk wanita, terdiri dari subkategori *clothing, shoes, girl's fashion*, dan lainnya,
- *Men's Fashion*; kategori ini berisi fesyen untuk pria, terdiri dari subkategori *clothing, shoes, boy's fashion*, dan lainnya,
- *Watches & Accessories*; kategori ini berisi jam tangan, kacamata, dan perhiasan, terdiri dari subkategori *men's watches, women's watches, men fashion jewellery*, dan lainnya,
- *Sports & Outdoor*; kategori ini berisi perlengkapan dan peralatan olahraga, terdiri dari sub kategori *boxing & martial arts*, *racket sports*, *water bottles*, dan lainnya,
- *Automotive & Motorbike*; kategori ini berisi peralatan untuk kendaraan bermotor, terdiri dari subkategori *auto accessories*, *auto electronics*, dan lainnya.

Pada situs lazada, kategori-kategori tidak memiliki halaman tersendiri, sehingga harus mengakses langsung halaman subkategori. Daftar barang pada halaman yang diakses juga melalui browser seperti Google Chrome berbeda dengan yang diakses melalui lightweight browser pada tahap kategori, namun

memiliki daftar barang yang sama apabila menggunakan fitur pencarian. Browser yang digunakan juga harus mengaktifkan *javascript* untuk dapat memroses situs Lazada. Oleh karena itu untuk *crawling* untuk kategori dan subkategori akan dilakukan pada halaman pertama saja, namun untuk *crawling* untuk pencarian dapat dilakukan lebih dari satu halaman.

Situs Amazon[2] merupakan salah satu situs *e-commerce* yang terbesar, memiliki inventaris barang lengkap dan besar. Situs amazon sendiri memiliki banyak kategori, namun tidak semua kategori dan subkategori dapat dimasukkan karena ada beberapa kategori yang mengarahkan ke situs lain ataupun tidak menjual, namun menyewakan saja. Bahkan ada beberapa barang yang terdapat subkategori yang isinya terdapat pada subkategori lainnya. Berikut beberapa kategori-kategori dan subkategori yang akan di-*crawl*:

- *Books & Audible*; berisi buku-buku, terdiri dari subkategori *books*, *magazines*, dan lainnya,
- *Movies*, *Music* & *Games*; berisi film, musik, dan gem dalam bentuk blu-ray maupun DVD, terdiri dari subkategori *movies & tv*, *blu-ray*, *video games*, dan lainnya,
- *Electronic, Computers, & Office;* berisi barang-barang elektronik maupun perlengkapan kantor, terdiri dari subkategori *camera, photo & video, headphones, drivers & storages*, dan lainnya,
- *Homes, Garden, Pets & Tools;* berisi peralatan dan perkakas untuk rumah, terdiri dari subkategori *furniture, kitchen & dining, power & hand tools*, dan lainnya,
- *Beauty & Health;* berisi peralatan kecantikan maupun suplemen kesehatan, terdiri dari subkategori *luxury beauty, vitamin & dietary supplements, health care,* dan lainnya,
- *Toys, Kids & Baby*; berisi peralatan dan mainan untuk anak-anak dan bayi, terdiri dari subkategori *toys & games, baby,* dan lainnya,

---

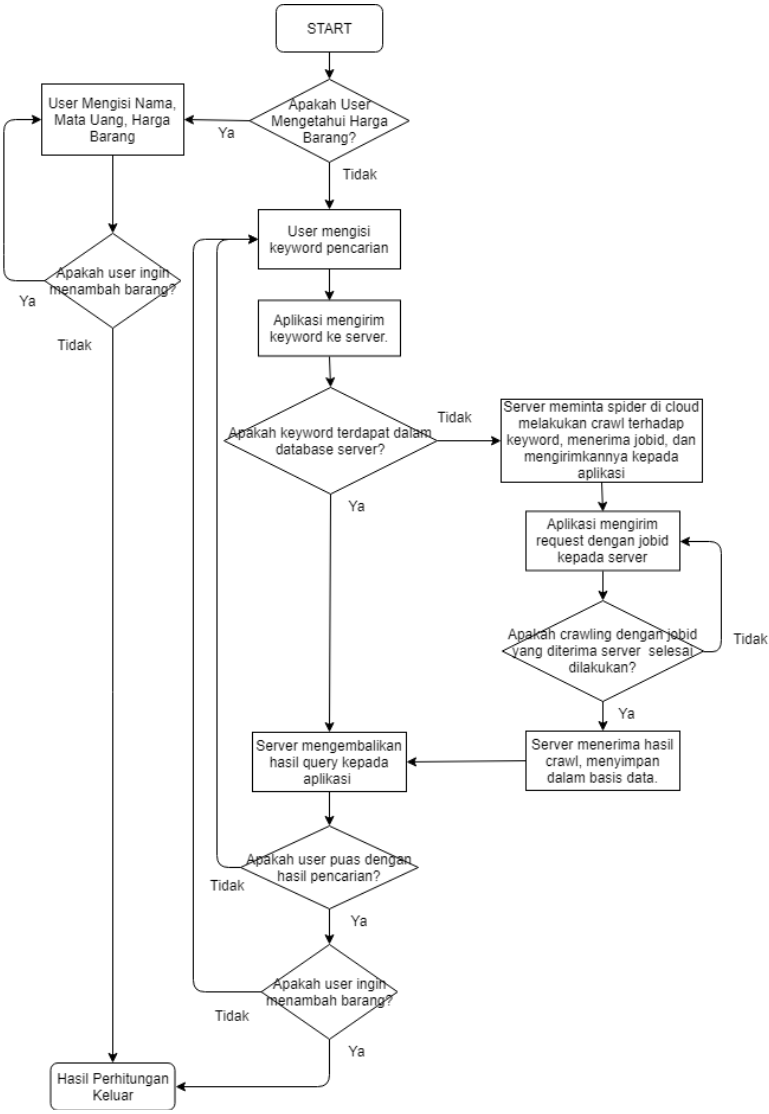
[2] https://www.amazon.com/

- *Clothing, Shoes & Jewellery*; berisi fesyen untuk semua umur dan gender, terdiri dari subkategori *men, women, luggage*, dan lainnya,
- *Handmade*; berisi barang-barang bukan hasil pabrik, terdiri dari subkategori *gifts, home & kitchen, toys & games*, dan lainnya,
- *Sports & Outdoor*; berisi perlengkapan olahraga maupun untuk ke luar rumah, terdiri dari subkategori *athletic clothing, cycling, climbing*, dan lainnya,
- *Automotive & Industrial*, berisi perlengkapan untuk otomotif, terdiri dari subkategori *automotive parts* & *accessories, automotive tools & equipment, tires & wheels*, dan lainnya.

Pada situs Amazon terdapat kategori yang memiliki daftar barang pada halamannya, namun ada juga yang hanya berisi subkategori-subkategori. Pemrosesan per halaman Amazon tidak mengharuskan penggunaan Javascript seperti situs Lazada, namun membutuhkan waktu yang lebih lama. Daftar barang yang didapat di browser biasa maupun lightweight browser juga sama, sehingga dapat di-*crawl* berdasarkan halaman. Oleh karena itu untuk *crawling* berdasarkan kategori dan subkategori dapat dilakukan lebih dari satu halaman, tetapi begitu banyak kategori dan subkategori maupun waktu yang dibutuhkan untuk memroses satu halaman lebih lama, maka akan dibatasi lima halaman per kategori maupun sub kategori. Sementara untuk *crawling* berdasarkan pencarian akan menggunakan halaman pertama saja.

3.1.2 Deskripsi Umum Sistem

Pada tugas akhir ini akan dikembangkan sebuah aplikasi android yang dapat menghitung jumlah pajak yang dikenakan saat kita membawa pulang barang dari perjalanan luar negeri. Pengguna dapat menggunakan nota untuk menginput harga atau menggunakan fitur pencarian harga yang akan digunakan dalam perhitungan pajak tersebut.



Gambar 3.1 Diagram Alir Sistem BEAI

Gambar 3.1 menjelaskan bagaimana alir dari sistem BEAI. Saat pengguna membuka aplikasi dan memilih kalkulator, pengguna akan ditanyakan terlebih dahulu apakah pengguna sudah mempersiapkan harga barang atau belum. Apabila pengguna sudah mempersiapkan harga, maka pengguna akan menginput nama, mata uang, serta harga barang tersebut. Pengguna dapat menambah sebanyak-banyaknya barang yang ingin ditambah dan menghitung total pajak yang perlu dibayar.

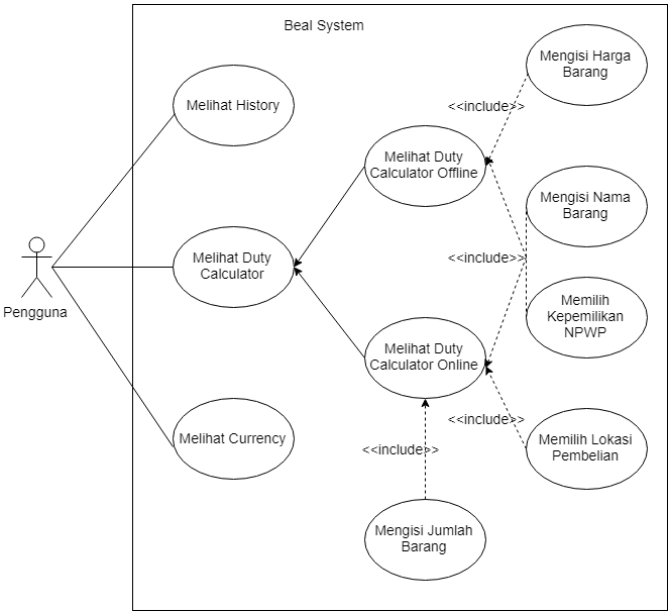
Pengguna yang tidak mempersiapkan harga barang, akan diarahkan ke dalam pencarian harga berdasarkan nama barang yang diinput, tergantung lokasi dimana mereka membeli barang tersebut. Aplikasi kemudian mengirim nama barang ke server dan server akan memroses nama barang ke dalam *database*. Apabila nama yang diinput tidak ada, maka server akan meminta spider di *cloud* untuk mencari harga barang tersebut sesuai lokasi yang diinput user. Server kemudian mendapatkan jobid dari *cloud* dan mengirimkannya kepada aplikasi pengguna. Aplikasi akan mengirim jobid lagi setelah beberapa rentang waktu kepada server dan server akan mengecek apabila spider sudah selesai melakukan *crawl*. Hal ini akan dilakukan hingga spider selesai *crawling* dan server mendapatkan hasil *crawling*.

Aplikasi menyimpan hasil *crawling* di dalam *database* dan mengirim hasil crawling kepada aplikasi pengguna. Aplikasi kemudian menotifikasi pengguna bahwa hasil sudah tersedia. Pengguna akan menentukan apakah hasil pencarian sudah sesuai atau belum. Apabila belum pengguna dapat menginput nama barang yang lebih spesifik dan mendapatkan hasil yang lebih akurat. Pengguna yang sudah puas akan hasil pencarian dapat langsung mendapatkan hasil perhitungan pajak yang perlu dibayar dengan estimasi harga yang didapat dari hasil pencarian.

3.1.3 Fitur

Sistem 'BEAI' seperti pada Gambar 3.2 memiliki beberapa fitur utama sebagai berikut:

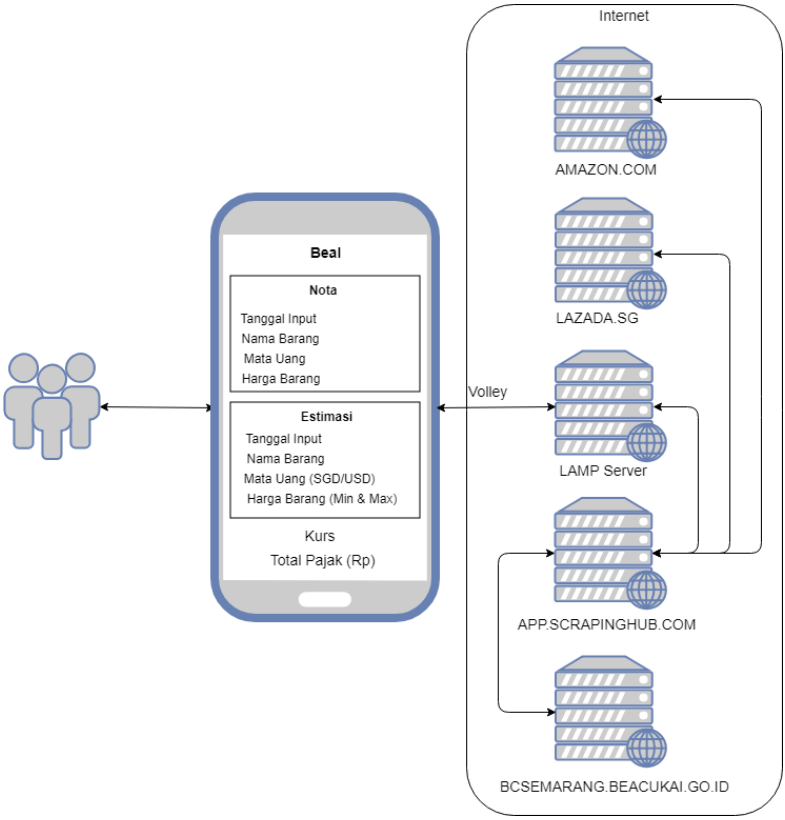
- *Duty Calculator*, fitur kalkulator dimana pengguna dapat memilih menggunakan harga yang dimasukkan oleh pengguna (*offline)* atau dari server (*online*). Hal ini ditentukan dengan menanyakan pengguna apakah mengetahui harga dari yang barang yang akan dimasukkan. Setelah memasukkan semua nama dan harga barang dan memilih kepemilikan NPWP aplikasi kemudian menampilkan rincian pembayaran pajak apa saja yang perlu dibayar.
- *Currency*, fitur dimana pengguna dapat melihat kurs yang berlaku pada hari itu.
- *History*, fitur dimana pengguna dapat melihat kembali nama dan harga barang yang telah diinput pengguna serta tanggal penginputan pada saat menggunakan fitur *Duty Calculator*.



Gambar 3.2 Use Case Diagram Sistem BEAI

3.2 Perancangan Perangkat Lunak

3.2.1 Arsitektur Perangkat Lunak



Gambar 3.3 Diagram Arsitektur Sistem BEAI

Pada Gambar 3.3 aplikasi BEAI merupakan klien dari sistem BEAI, dimana pengguna dapat menghitung jumlah pajak yang harus dibayarkan secara otomatis, baik dengan menginput harga sendiri maupun dari *database* sistem. Harga yang didapat dari sistem didapatkan dengan mengirimkan HTTP *request* ke

server. Pengguna juga dapat melihat *history* penginputan data dan kurs pajak yang berlaku.

Sistem BEAI menggunakan web server dan yang berada dalam satu mesin, yaitu LAMP server. LAMP server tersebut menggunakan kerangka CodeIgniter sebagai API untuk mengatur *request* yang diterima dan dikirim, serta pengiriman dan penerimaan *response* oleh server. *Request* yang diterima oleh server berasal dari aplikasi BEAI yang dijalankan oleh pengguna, sementara *request* yang dikirim oleh server terjadi saat server meminta token ke ScrapingHub. *Response* yang diterima oleh server adalah status *spider* dan hasil *crawl* dari *spider* tersebut, sementara yang response yang dikirim adalah token yang diterima dari ScrapingHub dan hasil pencarian. Hasil pencarian yang dari ScrapingHub akan disimpan dalam *database* server sehingga isi *database* akan terus bertambah seiring waktu.

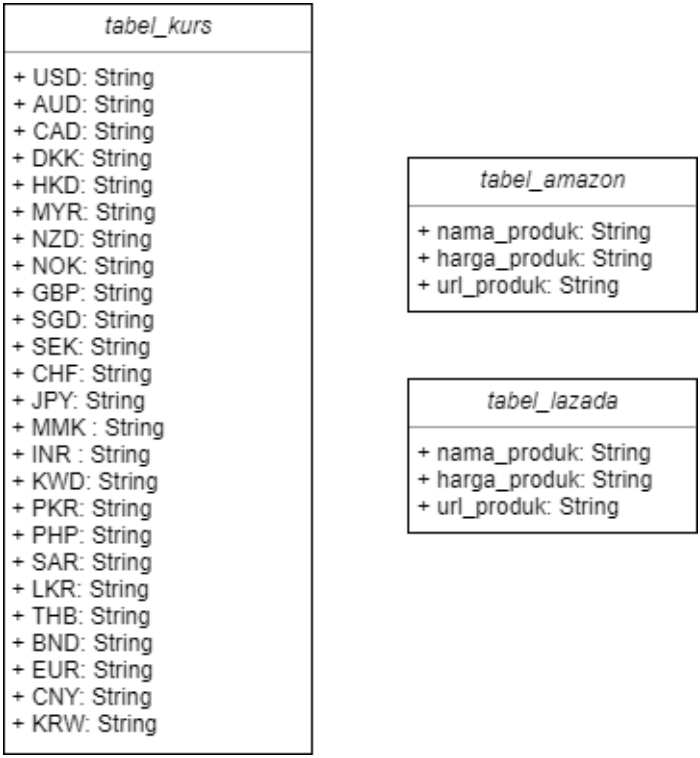
ScrapingHub merupakan tempat dimana spider disimpan dan di-*deploy*, sesuai dengan tujuan spider masing-masing. Spider yang melakukan *crawling* di lazada dan amazon bertujuan untuk menambah isi *database* yang terdapat pada LAMP server. Sementara *spider* yang melakukan crawling di beacukai akan melakukan update kurs yang tersimpan pada *database* server.

3.2.2 *Database*

3.2.2.1 Server

Database yang digunakan pada server seperti yang dilihat pada Gambar 3.4 digunakan untuk menyimpan data hasil *crawl* dari *spider* baik yang *crawling* kategori/subkategori maupun pencarian kata kunci. Dengan adanya suatu *database* yang menyimpan hasil *crawl* dari kata kunci, maka isi *database* akan bertambah banyak. Sehingga, intensitas penggunaan spider yang melakukan *crawling* terhadap pencarian keyword akan semakin berkurang karena pengguna yang terdahulu sudah menginput terlebih dahulu. *Database* pada server berisi 3 tabel: tabel amazon,

tabel lazada, dan tabel kurs, dapat dilihat contoh isinya pada Gambar 3.5. Tabel amazon dan tabel kurs cukup identik, yang membedakan adalah asal barisnya. Tabel amazon dan lazada terdiri dari 3 kolom: nama produk, harga produk, dan url tempat barang itu berada. Tabel kurs memiliki 25 kolom, dimana masing-masing kolom merupakan mata uang yang berbeda. Kolom nama_produk berisikan nama produk yang didapatkan dari hasil *crawling*, harga_produk berisikan harga dari nama produk, sementara url_produk adalah alamat dimana nama dan harga barang tersebut diambil.



Gambar 3.4 Diagram CDM pada server BEAI

nama_produk	harga_produk	url_produk
iHome iAV5	59.97	https://www.amazon.com/b/ref=s9_acss_bw_cg_AAHACAT
1 Mii 230ft Long Range	34.99	https://www.amazon.com/b/ref=s9_acss_bw_cg_AAHACAT
Ultimate Ears Wonderboom	77.66	https://www.amazon.com/b/ref=s9_acss_bw_cg_AAHACAT
All-new Sonos Beam	399.00	https://www.amazon.com/b/ref=s9_acss_bw_cg_AAHACAT

nama_produk	harga_produk	url_produk
Samsung 860 EVO SATA	199.00	https://www.lazada.sg/shop-computers-laptops-storage/?spm
2TB Pen drive USB 2.0	12.88	https://www.lazada.sg/shop-computers-laptops-storage/?spm
Seagate Backup Plus 1TB	75.00	https://www.lazada.sg/shop-computers-laptops-storage/?spm
Kingston A400	47.90	https://www.lazada.sg/shop-computers-laptops-storage/?spm

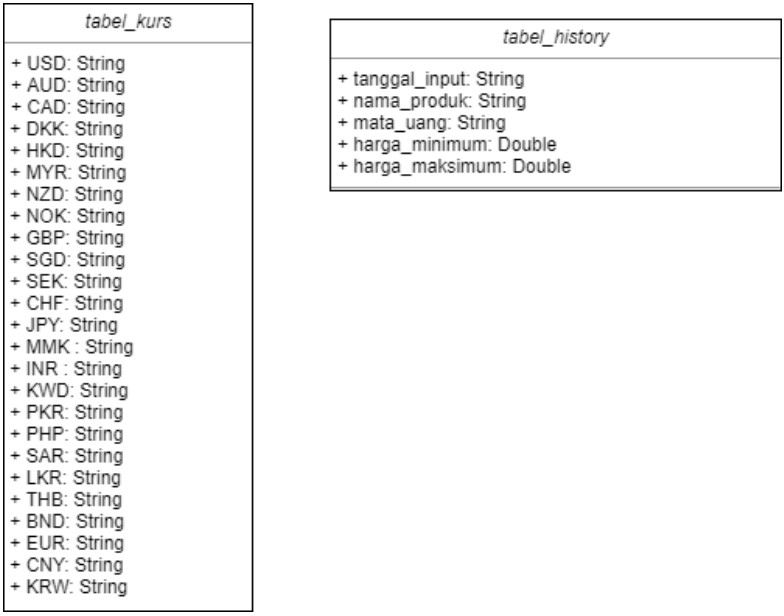
USD	AUD	DKK	CAD	HKD	MYR
14366.00	10672.70	2259.22	10953.82	1830.70	3559.31

Gambar 3.5 Contoh isi tabel pada server BEAI

3.2.2.2 Aplikasi

Tabel yang digunakan pada aplikasi seperti yang terlihat pada Gambar 3.6 digunakan untuk menyimpan hasil input pengguna sehingga pengguna dapat melihat *history* penginputan barang dan harga barang yang dilakukan sebelumnya dan menyimpan kurs yang diunduh dari server, serta menggunakannya dalam perhitungan pajak. *Database* pada aplikasi memiliki dua tabel: tabel kurs, dan tabel *history* dengan contoh isinya seperti pada Gambar 3.7. Tabel kurs memiliki 25 kolom, dimana masing masing kolom merupakan mata uang yang berbeda, sementara tabel history memiliki lima kolom: tanggal input data, nama produk, mata uang, harga minimum, dan harga maksimum. Tanggal input data merupakan kapan nama barang diinput oleh pengguna, nama produk merupakan nama barang yang diinput, mata uang merupakan mata uang dari harga yang dimasukkan pengguna, harga minimum merupakan harga minimum yang didapat dari

sistem, harga maksimum merupakan harga maksimum yang didapat dari sistem. Harga minimum dan maksimum akan sama apabila harga tersebut merupakan input pengguna atau hasil pencarian hanya satu barang saja.



Gambar 3.6 Diagram CDM pada aplikasi BEAI

tanggal_input	nama_produk	mata_uang	harga_minimum	harga_maksimum
2018/5/19	iHome iAV5	USD	45.69	59.97
2018/5/18	Samsung 860	SGD	189.99	189.99

USD	AUD	DKK	CAD	HKD	MYR
14366.00	10672.70	2259.22	10953.82	1830.70	3559.31

Gambar 3.7 Contoh isi tabel pada aplikasi BEAI

3.2.3 Scrapy Crawler

3.2.3.1 Spider Amazon untuk Kategori dan Sub Kategori

Spider amazon akan mengambil nama barang dari nilai data-attribute dari tag <h2>, dan harga barang dari isi tag yang memiliki nilai dari atribut class adalah a-offscreen seperti Kode Sumber 3.1:

```
1. < h2 data - attribute = "Tom Clancy's Jack Ryan"  data - max - rows = "0"  class = "a-
   size-medium s-inline s-access-title a-text-normal" > Tom Clancy 's Jack Ryan</h2>
2. < span class = "a-offscreen" > $5.99 - $7.99 < /span>
```

Kode Sumber 3.1 Contoh tag yang akan di-*scrape* pada situs Amazon

Kedua array hasil dari seleksi tersebut akan digabung, sehingga terselang-seling antara nama dan harga. Kemudian untuk membedakan yang mana dan harga akan mengecek karakter pertama apakah merupakan “$” atau tidak, dimana yang karakter pertama “$” dianggap sebagai harga dan yang bukan sebagai nama barang. Apabila mendapatkan nama barang, maka diset nama produk sebagai nama barang tersebut, dan dicek array selanjutnya apakah harga atau tidak. Apabila harga maka didapatkan nama dan harga suatu produk dan disimpan dalam satu obyek. Namun terkadang akan ditemukan rentang harga, maka dalam kasus ini akan diambil yang terbesar.

Untuk mendapatkan halaman selanjutnya akan dicek apakah terdapat tag <a> dengan atribut "title" yang berisi “Next Page” seperti pada Kode Sumber 3.2

```
1. < a title = "Next Page"
   id = "pagnNextLink" class = "pagnNext" href = "/s/ref=lp_2650363011_pg_2?rh=n%3
   A2625373011%2Cp_n_theme_browse-
   bin%3A2650363011&amp;page=2&amp;ie=UTF8&amp;qid=1529575381" >
```

Kode Sumber 3.2 Contoh tag halaman selanjutnya pada situs Amazon

Apabila ada akan diambil nilai atribut "href" dan dilakukan *request* baru ke url tersebut. Kemudian untuk mengecek

apakah kita sudah di halaman kelima atau belum, akan dicek url tersebut mengandung string '&page=5', dimana Spider akan mengirim *request* baru hingga mencapai halaman 5.

3.2.3.2 Spider Lazada untuk Kategori dan Sub Kategori

Untuk melakukan *parsing* nama produk dan harga produk pada situs lazada, nama serta harga disimpan dalam bentuk JSON dalam tag <script> urutan kedua. Oleh sebab itu, akan diambil teks dari tag <script> yang kedua. Teks yang diambil tersebut akan dihapus karakter spasi yang terdapat pada karakter pertama dan terakhirnya. Teks yang dibutuhkan berada setelah karakter "=" yang pertama, oleh sebab itu akan displit menjadi dua bagian yaitu yang sebelum karakter "=" dan setelah karakter "=". Akan dilakukan pembersihan escape characters, hingga akhirnya kita mendapatkan suatu teks yang memiliki format JSON. Teks ini akan di-*decode* , dan diambil bagian "listItems", yang isinya merupakan array dari nama barang dan harganya seperti pada Kode Sumber 3.3. Kemudian kita akan menyimpan masing-masing nama dan harga tersebut dalam satu obyek.

```
1. "listItems": [{ "name": "JBL Flip 3 Splashproof portable Bluetooth speaker with powe
   rful sound and speakerphone technology", "nid": "215809779", "icons": [], "product
   Url": "//www.lazada.sg/products/jbl-flip-3-splashproof-portable-bluetooth-speaker-
   with-powerful-sound-and-speakerphone-technology-i215809779-
   s327897990.html?search=1",
```

Kode Sumber 3.3 Contoh bagian HTML berbentuk JSON

Untuk penentuan url yang akan dicrawl, pertama-tama akan di-*load* terlebih dahulu basis url dari *setting*, yang kemudian akan ditambah dengan string "?spm=a2o42.home.", kunci dari tiap-tiap sub kategori, serta karakter yang dirandom sehingga seakan-akan berasal dari pengguna asli.

3.2.3.3 Spider Amazon untuk Search

Spider amazon akan melakukan *load custom setting* untuk mendapatkan kata kunci yang akan dicari. Kemudian kata kunci tersebut akan diolah hingga menjadi url pencarian di situs amazon yang semestinya. Seperti spider Amazon untuk kategori dan subkategori, spider Amazon untuk Search akan melakukan *crawl* ke url pencarian buatan tersebut dan mengambil nama barang dari nilai data-attribute dari tag <h2>, dan harga barang dari isi tag yang memiliki nilai dari atribut class adalah a-offscreen seperti pada Kode Sumber 3.1. Kedua array hasil dari seleksi tersebut akan digabung, sehingga terselang-seling antara nama dan harga. Kemudian untuk membedakan yang mana dan harga akan mengecek karakter pertama apakah merupakan "$" atau tidak, dimana yang karakter pertama "$" dianggap sebagai harga dan yang bukan sebagai nama barang. Apabila mendapatkan nama barang, maka diset nama produk sebagai nama barang tersebut, dan dicek array selanjutnya apakah harga atau tidak. Apabila harga maka didapatkan nama dan harga suatu produk dan disimpan dalam satu obyek. Namun terkadang akan ditemukan rentang harga, maka dalam kasus ini akan diambil yang terbesar.

3.2.3.4 Spider Lazada untuk Search

Keyword yang dikirim pengguna akan di-*load* terlebih dahulu dari *custom setting*. Keyword tersebut akan diganti karakter spasi " " menjadi "+". Kemudian akan dibuat sebuah URL dengan menggabungkan string "https://www.lazada.sg/catalog/?q=", *keyword* yang sudah diolah, string "&_keyori=ss&from=input&spm=a2o42.searchlist.search.go.", dan string yang karakternya sudah dirandom. Spider akan melakukan *crawling* ke halaman tersebut.

Untuk melakukan *parsing* nama produk dan harga produk pada situs lazada, nama serta harga disimpan dalam bentuk JSON dalam tag <script> urutan kedua. Oleh sebab itu, akan diambil teks

dari tag <script> yang kedua. Teks yang diambil tersebut akan dihapus karakter spasi yang terdapat pada karakter pertama dan terakhirnya. Teks yang dibutuhkan berada setelah karakter "=" yang pertama, oleh sebab itu akan displit menjadi dua bagian yaitu yang sebelum karakter "=" dan setelah karakter "=". Akan dilakukan pembersihan escape characters, hingga akhirnya kita mendapatkan suatu teks yang memiliki format JSON. Teks ini akan di-*decode* , dan diambil bagian "listItems", yang isinya merupakan array dari nama barang dan harganya. Kemudian kita akan menyimpan masing-masing nama dan harga tersebut dalam satu obyek.

```
1. < link rel = "next"  href = "https://www.lazada.sg/portable-speakers-for-tv/?page=2" >
```

Kode Sumber 3.4 Contoh tag link yang akan digunakan untuk menuju halaman selanjutnya

Kemudian untuk mengecek apakah ada halaman selanjutnya, akan dicari apakah terdapat tag <link> dengan nilai atribut "rel" adalah "next", seperti pada Kode Sumber 3.4. Apabila ada, maka akan diambil nilai atribut "href"nya dan akan dijadwalkan *request* baru. *Keyword* yang dikirim pengguna akan di-*load* terlebih dahulu dari *custom setting*. *Keyword* tersebut akan diganti karakter spasi " " menjadi "%20". Nilai atribut "href" akan dipisah menjadi dua bagian dahulu berdasarkan karakter "?". Bagian pertama akan digabung dengan string "?_keyori=ss&from=input&", yang kemudian digabung kembali dengan bagian kedua. Kemudian menambahkan string "&q=", *keyword* yang sudah diolah, string "&spm=a2o42.searchlistcategory.search.go.", serta string yang sudah dirandom supaya tampak seperti pengguna biasanya.

3.2.3.5 Spider Kurs

```
1. < a href = "http://bcsemarang.beacukai.go.id/kurs/kurs-pajak-minggu-ini-30-mei-2018-s-d-5-juni-2018/"  rel = "next" >
```

Kode Sumber 3.5 Contoh tag a dengan nilai atribut rel adalah next

Pertama-tama akan dilakukan *crawling* ke alamat http://bcsemarang.beacukai.go.id/kurs/kurs-pajak-minggu-ini-16-mei-2018-s-d-22-mei-2018/. Kemudian dicek apakah halaman ini merupakan halaman kurs paling ter-*update*, yaitu dengan mengecek apakah terdapat tag dengan atribut "class" dengan nilai "next-post" seperti pada Kode Sumber 3.5. Apabila ada, maka akan dilakukan crawling ke halaman yang didapat dari atribut"href" dari tag <a> yang memiliki nilai atribut "rel" adalah "next". Hal ini akan dilakukan terus menerus hingga tidak terdapat tag dengan atribut "clas" dengan nilai "next-post" lagi.

```
1.  < td style = "text-align: right;" > 14, 134.00 < /td>
2.  < td style = "text-align: right;" > 10, 658.45 < /td>
```

Kode Sumber 3.6 Contoh tag td untuk nilai daftar kurs

Apabila sudah tidak ditemukan lagi, kemudian akan diambil nilai dari tag <td> dengan atribut "style" bernilai "text-align: right;" seperti pada Kode Sumber 3.6 Semua nilai dari array urutan pertama hingga dua puluh lima akan disimpan dalam satu obyek.

3.2.4 Server API

3.2.4.1 Search Request API

Aplikasi mengirim HTTP *request* POST ke suatu halaman webserver, dengan *body* yang berisi *key* "keyword" dan *value* kata yang diinput oleh pengguna. Server akan menerima *key* dan *value* yang dicantumkan dalam *body*, kemudian akan mengecek di dalam *database* terlebih dahulu. Apabila *value* yang dicari ditemukan dalam *database*, maka akan langsung dikirimkan ke pengguna dengan tambahan "message" bahwa data tersedia. Hasil kueri dan "message" akan di-*encode* dalam JSON sebelum dikirimkan pengguna.

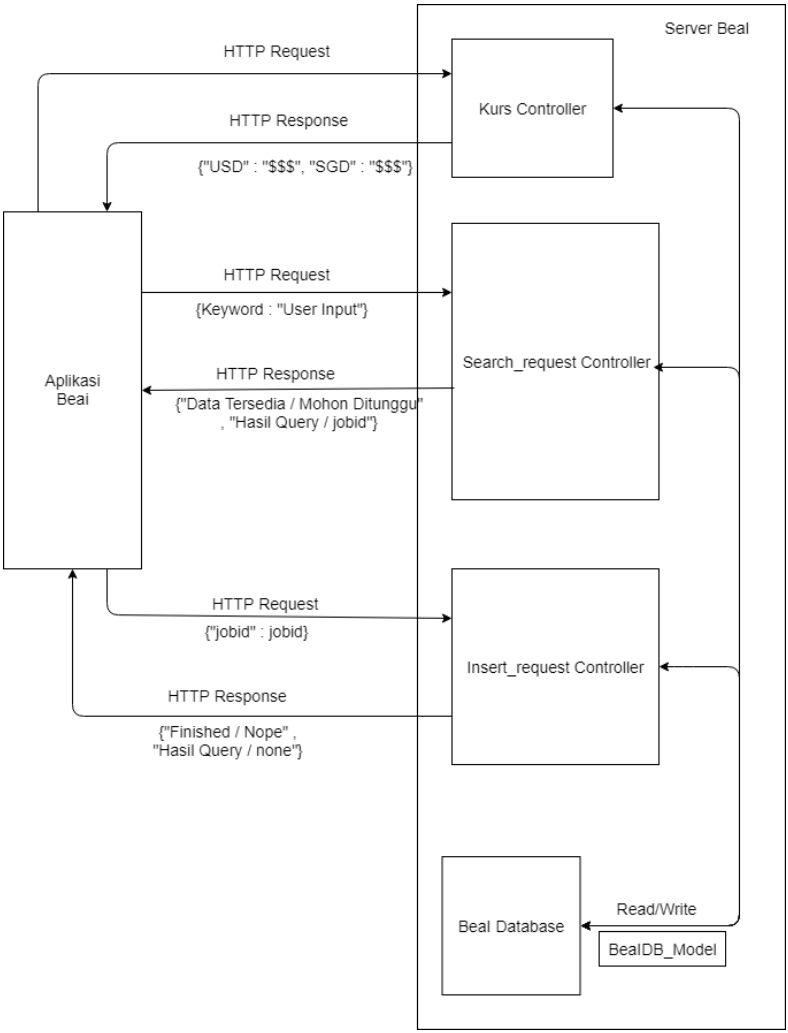
Apabila *value* yang diminta pengguna tidak ada dalam *database*, maka server akan melakukan cURL *request* ke ScrapingHub dan menerima jobid. Dibuat sebuah “message” berisi “Mohon ditunggu” dan kemudian dikirim bersama jobid dalam bentuk JSON.

Aplikasi yang menerima “message” dengan value “data telah tersedia” akan langsung menampilkan hasil kueri, sementara yang mendapatkan “Mohon ditunggu” akan menunggu dalam rentang waktu tertentu yang kemudian mengirim HTTP request POST ke suatu halaman webserver lagi dengan *body* berisi key “jobid” dan *value* jobid yang diterima aplikasi sebelumnya. Server yang menerima *request* tersebut akan mengirim cURL *request* lagi untuk mengecek apakah jobid tersebut sudah selesai atau belum. Apabila jobid sudah selesai dan mendapatkan hasil pencarian, maka server akan menyimpan data tersebut di dalam *database* dan mengirimkan “message” berisi “finished” dan hasil pencarian yang sudah di-*encode* dalam JSON ke apllikasi. Namun, jika jobid statusnya masih belum “finished” maka server akan merespon dengan “message” berisi “none” dan aplikasi akan menunggu beberapa saat lagi sebelum mengirim HTTP *request* lagi.

3.2.4.2 Kurs API

Saat pengguna memulai aplikasi, aplikasi otomatis akan mengirim HTTP *request* ke server untuk mengunduh kurs yang terakhir. Server yang menerima *request* ini akan mengambil data kurs di dalam *database*, yang kemudian akan dikirim kembali ke pengguna dalam bentuk JSON. Aplikasi kemudian melakukan *parsing* data JSON yang diterima dan menyimpan data tersebut ke dalam *database* android.

3.2.4.3 MVC Diagram



Gambar 3.8 MVC Diagram untuk Server API

Pada Gambar 3.8, Aplikasi BEAI dapat mengirimkan tiga jenis *request*:

- Meminta daftar kurs hari ini, yang dimana tersimpan pada *database* server BEAI. Server akan mengambil data daftar kurs kemudian mengirimkannya kembali ke aplikasi.
- Meminta harga barang berdasarkan kata kunci yang tersimpan pada *database* server BEAI. Server akan mengecek di *database* dan akan mengembalikan satu dari dua respons. Respon pertama apabila harga barang sudah tersedia di *database* dan mengirimkannya harga barang tersebut ke aplikasi. Respon kedua apabila harga barang belum tersedia sehingga perlu dilakukan *crawling* terlebih dahulu, dimana berisikan *jobid*.
- Meminta harga barang berdasarkan *jobid*. Server BEAI akan mengecek terlebih dahulu apakah *crawling* berdasarkan *jobid* sudah selesai atau belum, apabila belum server akan mengirimkan respon "Nope" kepada aplikasi, apabila sudah server akan menyimpan harga barang di *database* kemudian mengirimkan respon "Finished" dan harga barang.

3.2.5 Aplikasi Android

3.2.5.1 Rancangan Halaman UI Menu BEAI

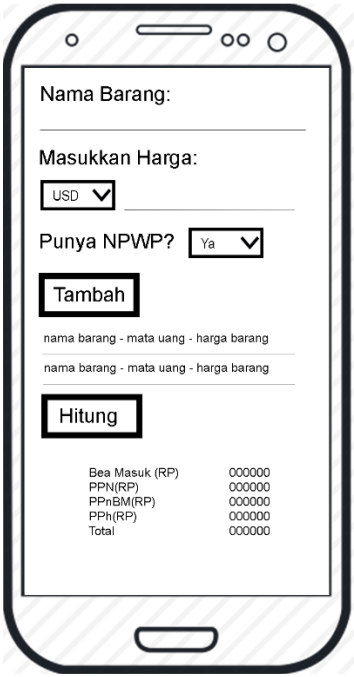
Menu BEAI pada Gambar 3.9 mempunyai tiga buah button; **Duty Calculator**, **Currency**, dan **History**. Pada saat **Duty Calculator** ditekan, akan muncul pilihan untuk memasukkan harga oleh pengguna ataupun sistem. Apabila pengguna memilih memasukkan harga sendiri, maka pengguna akan diarahkan ke Calculator Offline, sementara apabila pengguna memilih sistem yang memasukkan harga, maka pengguna akan diarahkan ke Calculator Online. Button **Currency** akan mengarahkan pengguna ke halaman Kurs untuk mengecek kurs yang berlaku. Button **History** akan mengarahkan ke halaman History untuk mengecek *history* input pengguna selama menggunakan BEAI.

Gambar 3.9 Rancangan Menu BEAI

3.2.5.2 Rancangan Halaman UI Calculator Offline

Pengguna menginput nama barang, harga, mata uang di tempatnya masing-masing pada Gambar 3.10. Kemudian pengguna menekan Tambah supaya barang tersebut dimasukkan ke dalam daftar perhitungan. Pengguna dapat menambah barang apapun sesuai keinginannya hingga akhirnya dia memencet tombol Hitung. Setelah tombol Hitung dipencet, aplikasi akan melakukan

kalkulasi pajak yang harus dibayarkan, dan menampilkan detailnya di bawah tombol Hitung.



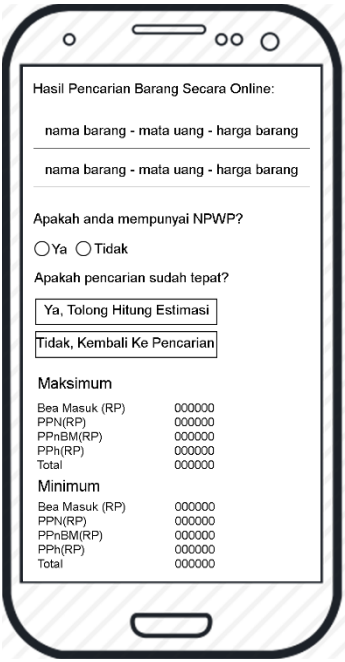
Gambar 3.10 Rancangan Calculator Offline

3.2.5.3 Rancangan Halaman UI Calculator Online

Pengguna mengisi nama barang yang ingin dihitung kalkulasi pajaknya pada Gambar 3.11, serta memilih tombol sesuai lokasi tempat dia membeli nama barang yang diisi.

Gambar 3.11 Rancangan Calculator Online

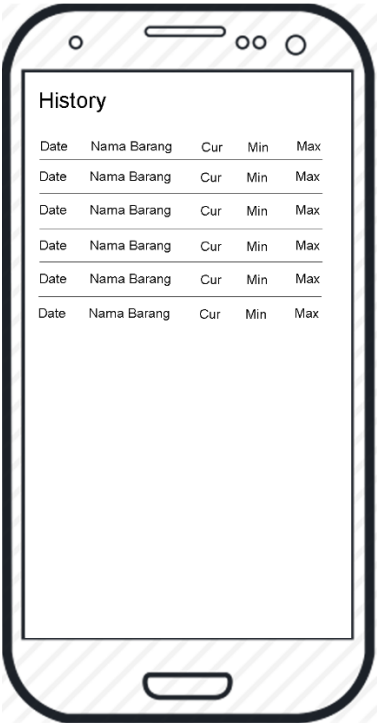
Pengguna menunggu hingga hasil pencarian keluar. Ketika hasil pencarian telah ditampilkan seperti pada Gambar 3.12, pengguna dapat memilih apakah memiliki NPWP atau tidak. Apabila pengguna tidak memilih kepemilikan NPWP, tombol perhitungan tidak dapat ditekan. Ketika pengguna sudah memilih kepemilikan NPWP, pengguna dapat memilih untuk menghitung atau tidak puas dengan hasil pencarian. Apabila pengguna memilih untuk menghitung, aplikasi akan mengeluarkan hasilnya, namun jika pemilih tidak puas maka pengguna akan kembali ke Calculator Online.



Gambar 3.12 Rancangan Calculator Online – Hasil Pencarian

3.2.5.4 Rancangan Halaman UI History

Pengguna yang memilih button History akan dialihkan ke halaman History pada Gambar 3.13, dimana pengguna dapat melihat input-input apa saja yang pernah dilakukan, baik Offline maupun Online.



Gambar 3.13 Rancangan History

3.2.5.5 Rancangan Halaman UI Kurs

Pengguna yang memencet tombol Currency akan diarahkan ke halaman seperti Gambar 3.14, dimana pengguna dapat melihat daftar kurs yang berlaku di bea cukai pajak pada saat ini.

Gambar 3.14 Rancangan Kurs

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan tahap-tahap implementasi dari perangkat lunak utama berdasarkan perancangan yang telah dibahas pada bab sebelumnya. Bab ini terdiri dari 5 sub-bab. Sub-bab pertama membahas tentang lingkungan implementasi dari aplikasi BEAI. Sub-bab kedua membahas implementasi *database* untuk aplikasi sistem BEAI, baik di android tempat aplikasi BEAI berada maupun server. Sub-bab ketiga membahas pengimplementasian spider Scrapy dan *custom setting* yang diterapkan. Sub-bab keempat membahas pengimplementasian API pada server, serta sub-bab kelima membahas pengimplementasian antar muka dari aplikasi BEAI.

4.1 Lingkungan Pengembangan

Spesifikasi perangkat keras serta perangkat lunak yang digunakan dalam tahap implementasi tugas akhir:

1. Komputer dengan processor Intel® Core™ i7-7700HQ CPU @2.80 GHz RAM 16 GB, Windows 10
2. Handphone Android versi 6.0, RAM 4 GB, ROM 64 GB
3. Android Studio 3.0
4. Draw.io
5. Sublime 3.1.1
6. PuTTY 0.70
7. FileZilla 3.33.0
8. Digitalocean droplet dengan LAMP stack dan perkakas CodeIgniter sebagai web server.
9. Anaconda Prompt
10. Perkakas Scrapy sebagai *web crawler*.

4.1.1 Persiapan lingkungan Server menggunakan LAMP dan CodeIgniter

Persiapan yang dilakukan dimulai setelah memiliki akun DigitalOcean. Kita akan membuat droplet dengan memilih image yang akan digunakan, yang dalam hal ini menggunakan Ubuntu 16.04. Setelah itu untuk menginstalasi LAMP stack, dapat memilih One-click apps untuk memilih instalasi LAMP. Kemudian memilih ukuran dari droplet yang ingin digunakan. Karena penggunaaan yang tidak cukup besar dan rumit, ukuran paling kecil (RAM 1GB, SSD 25 GB, Transfer 1 TB) cukup untuk penggunaaan. Memilih *datacenter region* supaya data transfer yang cukup aman, dalam hal ini Singapore. Yang terakhir adalah finalisasi pembuatan droplet, serta memberi nama pada droplet, dalam hal ini *BEAI*.

Kemudian untuk login ke dalam droplet membutuhkan PuTTY karena komputer yang digunakan menggunakan sistem operasi Windows 10. Isi Hostname dengan alamat IP droplet, Port 22, dan mencentang SSH pada *connection type*. Klik save untuk menyimpan setting sehingga tidak perlu konfigurasi lagi ke depannya. Setelah berhasil *connect*, masukkan password yang dikirimkan ke email.

Setelah berhasil login, untuk menginstalasi CodeIgniter jalankan *command* berikut:

```
1. wget https: //github.com/bcit-ci/CodeIgniter/archive/3.0.6.zip
2. unzip 3.0.6. zip
3. mv CodeIgniter-3.0.6 codeigniter
4. cp -R codeigniter /var/www/html/codeigniter
5. service apache2 restart
```

Kode Sumber 4.1 Persiapan Lingkungan Server

4.2 Implementasi *Database*

4.2.1 *Database* pada Android

Proses pembuatan *basis data* pada android menggunakan SQLite. Beberapa fungsi dibuat untuk memudahkan proses menampilkan data, pengisian, dan pembaruan isi *database*, antara lain. Berikut adalah tabel-tabel dan fungsi-fungsi yang dibuat, sesuai dengan Gambar 3.3:

```
1.  public void onCreate(SQLiteDatabase db) {
2.      db.execSQL("create table if not exists tambah_history(" + "tanggal TEXT," + "nam
    a_barang TEXT," + "mata_uang TEXT," + "harga_min TEXT," + "harga_max TEXT);");
3.      db.execSQL("create table if not exists kurs_tabel(" + "USD TEXT," + "GBP TEXT," +
     "LKR TEXT," + "DKK TEXT," + "CAD TEXT," + "PKR TEXT," + "KWD TEXT," + "MYR TEXT,
    " + "SAR TEXT," + "EUR TEXT," + "SEK TEXT," + "SGD TEXT," + "HKD TEXT," + "AUD TEX
    T," + "CHF TEXT," + "KRW TEXT," + "CNY TEXT," + "NZD TEXT," + "THB TEXT," + "BND T
    EXT," + "NOK TEXT," + "INR TEXT," + "JPY TEXT," + "MMK TEXT," + "PHP TEXT);");
```

Kode Sumber 4.2 Pembuatan Tabel tambah_history dan kurs_tabel pada Android

Kode Sumber 4.2 merupakan kode untuk membuat tabel tambah_history dan kurs_tabel pada Android apabila tabel yang bersangkutan belum ada.

```
1.  public List < HistoryModel > tampil_history() {
2.     List < HistoryModel > history = new ArrayList < > ();
3.     String query = "SELECT * FROM tambah_history";
4.     SQLiteDatabase db = this.getReadableDatabase();
5.     Cursor cursor = db.rawQuery(query, null);
6.     if (cursor.getCount() == 0) {
7.        HistoryModel historymodel = new HistoryModel();
8.        historymodel.tanggal = cursor.getString(cursor.getColumnIndex("tanggal"));
9.        historymodel.nama_barang = cursor.getString(cursor.getColumnIndex("nama_ba
    rang"));
10.       historymodel.mata_uang = cursor.getString(cursor.getColumnIndex("mata_uang
    "));
11.       historymodel.harga_min = cursor.getString(cursor.getColumnIndex("harga_min")
    );
```

```
12.         historymodel.harga_max = cursor.getString(cursor.getColumnIndex("harga_max"
    ));
13.         history.add(historymodel);
14.     } else if (cursor.moveToLast()) {
15.         do {
16.             HistoryModel historymodel = new HistoryModel();
17.             historymodel.tanggal = cursor.getString(cursor.getColumnIndex("tanggal"));
18.             historymodel.nama_barang = cursor.getString(cursor.getColumnIndex("nama_
    barang"));
19.             historymodel.mata_uang= cursor.getString(cursor.getColumnIndex("mata_uan
    g");
20.             historymodel.harga_min = cursor.getString(cursor.getColumnIndex("harga_mi
    n"));
21.             historymodel.harga_max= cursor.getString(cursor.getColumnIndex("harga_ma
    x"));
22.             history.add(historymodel);
23.         } while (cursor.moveToPrevious());
24.     }
25.     return history;
26. }
```

Kode Sumber 4.3 Menampilkan isi tabel tambah_history

Kode Sumber 4.3 merupakan kode sumber yang digunakan untuk menampilkan *history* pengguna, urut dari paling terakhir yang diinput hingga yang paling lama.

```
1.  public String select_kurs(String mata_uang) {
2.      String nilai_mata_uang;
3.      String query = "SELECT " + mata_uang + " FROM kurs_tabel";
4.      SQLiteDatabase db = this.getReadableDatabase();
5.      Cursor cursor = db.rawQuery(query, null);
6.      cursor.moveToFirst();
7.      nilai_mata_uang = cursor.getString(cursor.getColumnIndex(mata_uang));
8.      return nilai_mata_uang;  }
```

Kode Sumber 4.4 Mengambil nilai mata uang yang dipilih

Kode Sumber 4.4 merupakan kode sumber yang digunakan untuk menampilkan mata uang yg dipilih. Misal pada saat menampilkan seluruh kurs yang ada, masing-masing kurs akan mengambil nilai yang sesuai dengan nama kurs tersebut.

```
1.  public void update_kurs(List < KursModel > kursmodellist) {
2.     SQLiteDatabase db = this.getWritableDatabase();
3.     db.execSQL("drop table if exists kurs_tabel;");
4.     db.execSQL("create table if not exists kurs_tabel(" + "USD TEXT," + "GBP TEXT," + "
    LKR TEXT," + "DKK TEXT," + "CAD TEXT," + "PKR TEXT," + "KWD TEXT," + "MYR TEXT,"
    + "SAR TEXT," + "EUR TEXT," + "SEK TEXT," + "SGD TEXT," + "HKD TEXT," + "AUD TEXT,
    " + "CHF TEXT," + "KRW TEXT," + "CNY TEXT," + "NZD TEXT," + "THB TEXT," + "BND TE
    XT," + "NOK TEXT," + "INR TEXT," + "JPY TEXT," + "MMK TEXT," + "PHP TEXT);");
5.     ContentValues values = new ContentValues();
6.     for (int i = 0; i < kursmodellist.size(); i++) {
7.        values.put(kursmodellist.get(i).nama_mata_uang, kursmodellist.get(i).nilai_mata
    _uang);
8.     }
9.     db.insert("kurs_tabel", null, values);
10.        }
```

Kode Sumber 4.5 Mengupdate nilai mata uang

Kode Sumber 4.5 merupakan kode sumber yang digunakan untuk memperbarui masing-masing nilai mata uang.

```
1.  public void add_history(String tanggal, String name, String mata_uang, Double harga
    _min, Double harga_max) {
2.     SQLiteDatabase db = this.getWritableDatabase();
3.     ContentValues values = new ContentValues();
4.     values.put("tanggal", tanggal);
5.     values.put("nama_barang", name);
6.     values.put("mata_uang", mata_uang);
7.     values.put("harga_min", harga_min);
8.     values.put("harga_max", harga_max);
9.     db.insert("tambah_history", null, values);
10.        }
```

Kode Sumber 4.6 Menambah Entry tambah_history

Kode Sumber 4.6 merupakan kode sumber yang digunakan untuk menambah entry nama barang, harga, tanggal, dan mata uang yang diinput oleh pengguna ke dalam tabel tambah_history.

4.2.2 *Database* pada Server

Database pada server terdiri dari tiga tabel: tabel amazon_DB, tabel lazada_DB, dan tabel kurs_tabel. Kode Sumber 4.7 merupakan kode untuk membentuk ketiga tabel tersebut:

```
1. CREATE TABLE amazon_DB(product_name text, product_sale_price varchar(10), page
   _url text);
2. CREATE TABLE lazada_DB(product_name text, product_sale_price varchar(10), list_p
   age_url text);
3. CREATE TABLE kurs_tabel(USD varchar(9), GBP varchar(9), LKR varchar(9), DKK varcha
   r(9), CAD varchar(9), PKR varchar(9), KWD varchar(9), MYR varchar(9), SAR varchar(9)
   , EUR varchar(9), SEK varchar(9), SGD varchar(9), HKD varchar(9), AUD varchar(9), CH
   F varchar(9), KRW varchar(9), CNY varchar(9), NZD varchar(9), THB varchar(9), BND va
   rchar(9), NOK varchar(9), INR varchar(9), JPY varchar(9), MMK varchar(9), PHP varcha
   r(9));
```

Kode Sumber 4.7 Membuat Tabel amazon_DB, lazada_DB, kurs_tabel

4.3 Implementasi Scrapy Crawler

Jalankan Anaconda Prompt, kemudian masukkan Kode Sumber 4.8 untuk membuat project BEAI dan membuat lima spider:

```
1. scrapy startproject BEAI
2. scrapy genspider lazada_search_crawler http://www.lazada.sg/
3. scrapy genspider lazada_crawler http://www.lazada.sg/
4. scrapy genspider amazon_search_crawler http://www.amazon.com/
5. scrapy genspider amazon_crawler http://www.amazon.com/
6. scrapy genspider kurscukai http://bcsemarang.beacukai.go.id/
```

Kode Sumber 4.8 Membuat project BEAI, spider lazada_crawler, spider lazada_search_crawler, spider amazon_search_crawler, spider amazon_crawler, dan spider kurscukai

4.3.1 Spider Amazon untuk Kategori dan Sub Kategori

```
1.  def start_requests(self):
2.      urls = settings.getlist('AMAZON_START_URLS', default = None)
```

```
3.      for url in urls:
4.
    yield SplashRequest(url = url, endpoint = 'render.html', args = {'wait': 7.0,  'images':
    0  })
```

Kode Sumber 4.9 Mengirim *request* ke URL yang didapat dari custom setting AMAZON_START_URLS.

Pada Kode Sumber 4.9 spider akan melakukan SplashRequest() ke alamat yang didapat dari custom setting AMAZON_START_URLS, dengan argument wait = 7.0 dan images = 0, dimana artinya setelah selesai mengunduh alamat yang diberikan, spider akan menunggu selama 7 detik sebelum mengolah isi halaman yang diunduh, serta untuk tidak mengunduh gambar-gambar supaya tidak semakin lama proses *loading* halamannya.

```
1.  product_names_prices = response.selector.xpath('(//h2/@data-
    attribute | //span[@cla Kode Sumber 4.9 Funggsi untuk membedakan harga dan
    barang ss="a-offscreen"]//text())').extract()
```

Kode Sumber 4.10 Membuat array yang terdiri dari harga dan barang.

Pada Kode Sumber 4.10 spider amazon akan mengambil nama barang dari nilai data-attribute dari tag <h2>, dan harga barang dari isi tag yang memiliki nilai dari atribut class adalah a-offscreen. Kedua array hasil dari seleksi tersebut akan digabung, sehingga terselang-seling antara nama dan harga.

```
1.  def RepresentsString(self, s):
2.  is_it_dollar = s[0]
3.  if is_it_dollar == '$':
4.    return False
5.  else:
6.    return True
```

Kode Sumber 4.11 Fungsi untuk membedakan harga dan barang

Kemudian pada Kode Sumber 4.11 untuk membedakan yang mana dan harga akan mengecek karakter pertama apakah merupakan “$” atau tidak, dimana yang karakter pertama “$” dianggap sebagai harga dan yang bukan sebagai nama barang.

```
1.  what_is_product_name = ''
2.  for product_attributes in product_names_prices:
3.     if self.RepresentsString(product_attributes):
4.           what_is_product_name = product_attributes
5.           x = x+1
6.     else:
7.           if (is_it_price_again or product_attributes=='$0.00'):
8.              is_it_price_again = False
9.           else:
10.             if ' - ' in product_attributes:
11.                        first_price, second_price = product_attributes.split(' - ')
12.                        item = BEAIItem()
13.                        item['product_name'] = what_is_product_name
14.                        item['product_sale_price'] = second_price[1:]
15.
16.                        item['page_url'] = response.url
17.                        is_it_price_again = True
18.                        yield item
19.             else:
20.                        item = BEAIItem()
21.                        item['product_name'] = what_is_product_name
22.                        item['product_sale_price']=product_attributes[1:]
23.                        item['page_url'] = response.url
24.                        is_it_price_again = True
25.                        yield item
```

Kode Sumber 4.12 Mendapatkan pasangan nama dan harga barang

Pada Kode Sumber 4.12 , apabila mendapatkan nama barang, maka diset nama produk sebagai nama barang tersebut, dan dicek array selanjutnya apakah harga atau tidak. Apabila harga maka didapatkan nama dan harga suatu produk dan disimpan dalam satu obyek. Namun terkadang akan ditemukan rentang harga, maka dalam kasus ini akan diambil yang terbesar.

```
1.  if nextPage:
2.           if '&page=5' in response.url:
3.              a = 2
4.           else:
5.              next_Page = ''.join(nextPage)
6.              next_page_URL = 'https://www.amazon.com' + next_Page
```

```
7.            yield SplashRequest(url=next_page_URL, endpoint='render.html',
    args={'wait': 7.0, 'images': 0}) })
```

Kode Sumber 4.13 Melanjutkan ke halaman selanjutnya

Pada

Kode Sumber **4.13**, untuk mendapatkan halaman selanjutnya akan dicek apakah terdapat tag <a> dengan atribut title yang berisi "Next Page". Apabila ada akan diambil nilai atribut href dan dilakukan *request* baru ke url tersebut. Kemudian untuk mengecek apakah kita sudah di halaman kelima atau belum, akan dicek url tersebut mengandung string '&page=5', dimana spider akan mengirim *request* baru hingga mencapai halaman 5 menggunakan SplashRequest().

4.3.2 Spider Lazada untuk Kategori dan Sub Kategori

```
1. def start_requests(self):
2.     urls = settings.getlist('START_URLS',  def start_requests(self):
3.     random_character = self.generate_random_character()
4.     cate = self.reference_for_categories(url)
5.     get_url = url + '?spm=a2o42.home.' + cate + '.654346' + random_character
6.     yield SplashRequest(url=get_url, endpoint='render.html', args={'wait': 5.0})
```

Kode Sumber 4.14 Mendapatkan URL dari custom setting START_URL, menghasilkan URL baru dan melakukan SplashRequest() ke URL tersebut

Pada Kode Sumber 4.14, spider mengolah terlebih dahulu URL yang didapat dari custom setting START_URLS menjadi url yang mirip dengan URL dihasilkan saat browsingUntuk penentuan url yang akan dicrawl, pertama-tama akan di-*load* terlebih dahulu basis url dari *setting*, yang kemudian akan ditambah dengan string "?spm=a2o42.home.", kunci dari tiap-tiap sub kategori, serta karakter yang dirandom sehingga seakan-akan berasal dari pengguna asli.

```
1. products = response.selector.xpath('(//script)/text()').extract()
2. product = products[1].encode('utf-8').strip()
3. a, b = product.split("=", 1)
```

```
4.  d = b.replace('\\"', '')
5.  e = d.replace('\\', '')
6.  g = e.replace("""", "")
7.  t = g.replace('"{$0}"""', '')
```

Kode Sumber 4.15 Pemrosesan halaman dari URL yang diunduh

Kode Sumber 4.15 melakukan parsing nama produk dan harga produk pada situs lazada, nama serta harga disimpan dalam bentuk JSON dalam tag <script> urutan kedua. Oleh sebab itu, akan diambil teks dari tag <script> yang kedua. Teks yang diambil tersebut akan dihapus karakter spasi yang terdapat pada karakter pertama dan terakhirnya. Teks yang dibutuhkan berada setelah karakter "=" yang pertama, oleh sebab itu akan displit menjadi dua bagian yaitu yang sebelum karakter "=" dan setelah karakter "=". Akan dilakukan pembersihan escape characters, hingga akhirnya kita mendapatkan suatu teks yang memiliki format JSON.

```
1.  jsonresponse = json.loads(t)
2.  jsonresponseMods = jsonresponse['mods']
3.  jsonresponseInfo = jsonresponse['mainInfo']
4.  jsonresponseResults = jsonresponseInfo['totalResults']
5.  jsonresponsePage = jsonresponseInfo['page']
6.  jsonresponseList = jsonresponseMods['listItems']
7.  for i in jsonresponseList:
8.      item = BEAIItem()
9.      item['product_name'] = i['name']
10.     item['product_sale_price'] = i['price']
11.     item['page_url'] = response.url
12.         yield item
```

Kode Sumber 4.16 Penyimpanan nama dan harga barang

Teks ini akan di-*decode* , dan diambil bagian "listItems", yang isinya merupakan array dari nama barang dan harganya. Kemudian kita akan menyimpan masing-masing nama dan harga tersebut dalam satu obyek.

4.3.3 Spider Amazon Berdasarkan Keyword

```
1. def start_requests(self):
2.    get_url = 'https://www.amazon.com/s/ref=nb_sb_noss_2?url=search-
   alias%3Daps&field-keywords='
3.    keywords = settings.get('AMAZON_SEARCH_URL',  default = None)
4.    new_keywords = keywords.replace(' ', '+')
5.    get_url = get_url + new_keywords
6.    yield SplashRequest(url = get_url, endpoint = 'render.html', args = {  'wait': 10.0  })
```

Kode Sumber 4.17 Mengirim *request* ke URL hasil penggabungan kata kunci yang didapat dari custom setting AMAZON_SEARCH_URL dan template URL

Pada Kode Sumber 4.17 spider akan melakukan mengambil kata kunci dari *custom setting* AMAZON_SEARCH_URL, kemudian menggabungkannya dengan template URL. Spider kemudian melakukan SplashRequest() ke hasil gabungan tersebut dengan argument wait = 10.0 dimana artinya setelah selesai mengunduh alamat yang diberikan, spider akan menunggu selama 10 detik sebelum mengolah isi halaman yang diunduh.

```
1. product_names_prices = response.selector.xpath('(//h2/@data-
   attribute | //span[@cla Kode Sumber 4.9 Funggsi untuk membedakan harga dan
   barang ss="a-offscreen"]//text())').extract()
```

Kode Sumber 4.18 Membuat array yang terdiri dari harga dan barang.

Pada Kode Sumber 4.18 spider amazon akan mengambil nama barang dari nilai data-attribute dari tag <h2>, dan harga barang dari isi tag yang memiliki nilai dari atribut class adalah a-offscreen. Kedua array hasil dari seleksi tersebut akan digabung, sehingga terselang-seling antara nama dan harga.

```
1. def RepresentsString(self, s):
2. is_it_dollar = s[0]
3. if is_it_dollar == '$':
```

```
4.    return False
5.  else:
6.    return True
```

Kode Sumber 4.19 Fungsi untuk membedakan harga dan barang

Kemudian pada Kode Sumber 4.19 untuk membedakan yang mana dan harga akan mengecek karakter pertama apakah merupakan “$” atau tidak, dimana yang karakter pertama “$” dianggap sebagai harga dan yang bukan sebagai nama barang.

```
1.  what_is_product_name = ''
2.  for product_attributes in product_names_prices:
3.    if self.RepresentsString(product_attributes):
4.          what_is_product_name = product_attributes
5.          x = x+1
6.    else:
7.          if (is_it_price_again or product_attributes=='$0.00'):
8.            is_it_price_again = False
9.          else:
10.           if ' - ' in product_attributes:
11.                   first_price, second_price = product_attributes.split(' - ')
12.                   item = BEAIItem()
13.                   item['product_name'] = what_is_product_name
14.                   item['product_sale_price'] = second_price[1:]
15.
16.                   item['page_url'] = response.url
17.                   is_it_price_again = True
18.                   yield item
19.           else:
20.                   item = BEAIItem()
21.                   item['product_name'] = what_is_product_name
22.                   item['product_sale_price']=product_attributes[1:]
23.                   item['page_url'] = response.url
24.                   is_it_price_again = True
25.                   yield item
```

Kode Sumber 4.20 Mendapatkan pasangan nama dan harga barang

Pada Kode Sumber 4.20, apabila mendapatkan nama barang, maka diset nama produk sebagai nama barang tersebut, dan dicek array selanjutnya apakah harga atau tidak. Apabila harga

maka didapatkan nama dan harga suatu produk dan disimpan dalam satu obyek. Namun terkadang akan ditemukan rentang harga, maka dalam kasus ini akan diambil yang terbesar.

4.3.4 Spider Lazada Berdasarkan Keyword

```
1. def generate_random_character(self):
2.     allchar = string.ascii_letters + string.digits
3.     rand_chara = "".join(choice(allchar) for x in range(0, 14))
4.     return rand_chara
```

Kode Sumber 4.21 Mendapatkan 14 karakter random

Pada Kode Sumber 4.21, spider mengimitasi empat belas karakter random pada URL pencarian dengan melakukan randomisasi *string* yang terdiri dari empat belas karakter.

```
1. def start_requests(self):
2.     get_url = 'https://www.lazada.sg/catalog/?q='
3.     keywords = settings.get('SEARCH_URL',  default = None)
4.     new_keywords = keywords.replace(' ', '+')
5.     random_character = self.generate_random_character()
6.
   get_url = get_url + new_keywords + '&_keyori=ss&from=input&spm=a2o42.searchlis
   t.search.go.' + random_character
7.     yield SplashRequest(url = get_url, endpoint = 'render.html', args = { 'wait': 5.0 })
```

Kode Sumber 4.22 Mengirim *request* ke URL hasil penggabungan kata kunci yang didapat dari custom setting SEARCH_URL, karakter random dan template URL

Pada Kode Sumber 4.22**Kode Sumber 4.17** spider akan melakukan mengambil kata kunci dari *custom setting* SEARCH_URL, kemudian menggabungkannya dengan template URL dan hasil fungsi generate_random_character. Spider kemudian melakukan SplashRequest() ke hasil gabungan tersebut dengan argument wait = 5.0.

```
8.  products = response.selector.xpath('(//script)/text()').extract()
9.  product = products[1].encode('utf-8').strip()
10. a, b = product.split("=", 1)
```

```
11. d = b.replace('\\"', '')
12. e = d.replace('\\', '')
13. g = e.replace("""", "")
14. t = g.replace('"{$0}"""', '')
```

Kode Sumber 4.23 Pemrosesan halaman dari URL yang diunduh

Kode Sumber 4.23 melakukan parsing nama produk dan harga produk pada situs lazada, nama serta harga disimpan dalam bentuk JSON dalam tag <script> urutan kedua. Oleh sebab itu, akan diambil teks dari tag <script> yang kedua. Teks yang diambil tersebut akan dihapus karakter spasi yang terdapat pada karakter pertama dan terakhirnya. Teks yang dibutuhkan berada setelah karakter "=" yang pertama, oleh sebab itu akan displit menjadi dua bagian yaitu yang sebelum karakter "=" dan setelah karakter "=". Akan dilakukan pembersihan escape characters, hingga akhirnya kita mendapatkan suatu teks yang memiliki format JSON.

```
13. jsonresponse = json.loads(t)
14. jsonresponseMods = jsonresponse['mods']
15. jsonresponseInfo = jsonresponse['mainInfo']
16. jsonresponseResults = jsonresponseInfo['totalResults']
17. jsonresponsePage = jsonresponseInfo['page']
18. jsonresponseList = jsonresponseMods['listItems']
19. for i in jsonresponseList:
20.     item = BEAIItem()
21.     item['product_name'] = i['name']
22.     item['product_sale_price'] = i['price']
23.     item['page_url'] = response.url
24.         yield item
```

Kode Sumber 4.24 Penyimpanan nama dan harga barang

Pada Kode Sumber 4.24 teks ini akan di-*decode* , dan diambil bagian "listItems", yang isinya merupakan array dari nama barang dan harganya. Kemudian kita akan menyimpan masing-masing nama dan harga tersebut dalam satu obyek.

```
1.  if nextPage:
2.     if '&page=2' in response.url: a = 2
3.    else :
```

```
4.        next_page = ''.join(nextPage)
5.        next_page_base, next_page_page = next_page.split('?')
6.        rand_chara = self.generate_random_character()
7.        keywords = settings.get('SEARCH_URL',  default = None)
8.        new_keywords = keywords.replace(' ', '%20')
9.
    gotoPage = next_page_base + '?_keyori=ss&from=input&' + next_page_page + '&q='
    + new_keywords + '&spm=a2o42.searchlistcategory.search.go.' + rand_chara
10.
    request = SplashRequest(url = gotoPage, endpoint = 'render.html', args = { 'wait': 5.0 }
    )
11.       yield request
```

Kode Sumber 4.25 Pembuatan URL halaman selanjutnya dan mengakses URL tersebut

Pada Kode Sumber 4.25 Spider akan mengecek apakah spider akan menuju halaman selanjutnya dengan mengecek apakah halaman berikutnya ada atau tidak sesuai dengan nilai variabel "nextPage" dan apakah sudah di halaman kedua atau belum. Kalau halaman selanjutnya ada dan belum berada di halaman kedua, maka akan membuat URL yang akan dituju, dengan menggabungkan elemen-elemen sesuai dengan baris kesembilan. Spider kemudian mengirim *request* baru terhadap URL tersebut menggunakan SplashRequest() dengan argument "wait" 5.0.

4.3.5 Spider Kurs Cukai

```
1.  is_it_not_latest_page = response.selector.xpath('//div[@class="next-
    post"]').extract()
```

Kode Sumber 4.26 Pengecekan halaman terbaru atau belum

Pada Kode Sumber 4.26, Spider akan mengecek apakah halaman yang diakses berisi kurs terbaru atau belum dengan mengecek apakah ada *tag* "div" dengan atribut "class" bernilai "next-post".

```
1.  if is_it_not_latest_page:
2.     go_to_next_page = response.selector.xpath('//a[@rel="next"]/@href').extract()
3.     goToNext = ''.join(go_to_next_page)
```

```
4.      yield scrapy.Request(url = goToNext, callback = self.parse)
5.  else :
6.      selectTD = []
7.      selectTD = response.selector.xpath('//td[@style="text-
    align: right;"]/text()').extract()
8.      kurs = KursCukai()
9.      kurs['USD'] = selectTD[0]
10.     kurs['AUD'] = selectTD[1]
11.     kurs['CAD'] = selectTD[2]
12.     kurs['DKK'] = selectTD[3]
13.     kurs['HKD'] = selectTD[4]
14.     kurs['MYR'] = selectTD[5]
15.     kurs['NZD'] = selectTD[6]
16.     kurs['NOK'] = selectTD[7]
17.     kurs['GBP'] = selectTD[8]
18.     kurs['SGD'] = selectTD[9]
19.     kurs['SEK'] = selectTD[10]
20.     kurs['CHF'] = selectTD[11]
21.     kurs['JPY'] = selectTD[12]
22.     kurs['MMK'] = selectTD[13]
23.     kurs['INR'] = selectTD[14]
24.     kurs['KWD'] = selectTD[15]
25.     kurs['PKR'] = selectTD[16]
26.     kurs['PHP'] = selectTD[17]
27.     kurs['SAR'] = selectTD[18]
28.     kurs['LKR'] = selectTD[19]
29.     kurs['THB'] = selectTD[20]
30.     kurs['BND'] = selectTD[21]
31.     kurs['EUR'] = selectTD[22]
32.     kurs['CNY'] = selectTD[23]
33.     kurs['KRW'] = selectTD[24]
34.     yield kurs
```

Kode Sumber 4.27 Penyimpanan daftar kurs

Pada Kode Sumber 4.27 spider akan menuju halaman selanjutnya apabila halaman tersebut bukan halaman yang berisi daftar kurs terbaru, sementara akan menyimpan daftar kurs pada halaman tersebut bila halaman tersebut berisi daftar kurs yang terbaru.

4.4 Implementasi Server API

4.4.1 Controller

4.4.1.1 Fungsi first_search_lazada

```
1.  $keyword = $this - > input - > post("keyword");
2.  $keyword = str_replace("'", '', $keyword);
3.  $result_lazada = $this - > BEAIDB_model - > first_try_search_lazada($keyword);
4.  if (count($result_lazada) == 0) {
5.     $ch = curl_init();
6.     curl_setopt($ch, CURLOPT_URL, "https://app.scrapinghub.com/api/run.json");
7.     curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
8.     $job_settings = array('SEARCH_URL' => $keyword);
9.     $job_settings_encoded = json_encode($job_settings);
10.    $toPost = "project=305820&spider=lazada_search_crawler&units=1&job_settings=
   $job_settings_encoded";
11.    curl_setopt($ch, CURLOPT_POSTFIELDS, $toPost);
12.    curl_setopt($ch, CURLOPT_POST, 1);
13.    curl_setopt($ch, CURLOPT_USERPWD, "ce1064bb139a4c9abf2e983c4e623345".
14.       ":".
15.       "");
16.    $headers = array();
17.    $headers[] = "Content-Type: application/x-www-form-urlencoded";
18.    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
19.    $result = curl_exec($ch);
20.    if (curl_errno($ch)) {
21.       echo 'Error:'.curl_error($ch);
22.    }
23.    curl_close($ch);
24.    $final_result = json_decode($result);
25.    $job_id = $final_result - > jobid;
26.    $data = array('data_exist' => $error, 'message' => "Mohon ditunggu", 'jobid' => $jo
   b_id);
27.    echo json_encode($data);
28. } else {
29.    $data = array('data_exist' => true, 'message' => "Data tersedia", 'result' => $result_l
   azada);
30.    echo json_encode($data);
```

```
31. }
```

Kode Sumber 4.28 Pengecekan harga pada *database* untuk pembelian barang di Asia Tenggara untuk pertama kali

Pada Kode Sumber 4.28 server menerima HTTP *request* POST dari aplikasi, dengan *body* yang berisi *key* “keyword” dan *value* kata yang diinput oleh pengguna. Server akan menerima *key* dan *value* yang dicantumkan dalam *body*, kemudian akan mengecek di dalam *database* terlebih dahulu. Apabila *value* yang dicari ditemukan dalam *database*, maka akan langsung dikirimkan ke pengguna dengan tambahan “message” bahwa data tersedia. Hasil kueri dan “message” akan di-*encode* dalam JSON sebelum dikirimkan pengguna.

Apabila *value* yang diminta pengguna tidak ada dalam *database*, maka server akan melakukan cURL *request* ke ScrapingHub dan menerima jobid. Dibuat sebuah “message” berisi “Mohon ditunggu” dan kemudian dikirim bersama jobid dalam bentuk JSON.

4.4.1.2 Fungsi first_search_amazon

```
1.  $keyword = $this - > input - > post("keyword");
2.  $keyword = str_replace("'", '', $keyword);
3.  $result_amazon = $this - > BEAIDB_model - > first_try_search_amazon($keyword);
4.  if (count($result_amazon) == 0) {
5.     $ch = curl_init();
6.     curl_setopt($ch, CURLOPT_URL, "https://app.scrapinghub.com/api/run.json");
7.     curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
8.     $job_settings = array('AMAZON_SEARCH_URL' => $keyword);
9.     $job_settings_encoded = json_encode($job_settings);
10.    $toPost = "project=305820&spider=amazon_search_crawler&units=1&job_settings
    =$job_settings_encoded";
11.    curl_setopt($ch, CURLOPT_POSTFIELDS, $toPost);
12.    curl_setopt($ch, CURLOPT_POST, 1);
13.    curl_setopt($ch, CURLOPT_USERPWD, "ce1064bb139a4c9abf2e983c4e623345".
14.       ":".
15.       "");
16.    $headers = array();
17.    $headers[] = "Content-Type: application/x-www-form-urlencoded";
```

```
18.    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
19.    $result = curl_exec($ch);
20.    if (curl_errno($ch)) {
21.       echo 'Error:'.curl_error($ch);
22.    }
23.    curl_close($ch);
24.    $final_result = json_decode($result);
25.    $job_id = $final_result - > jobid;
26.    $data = array('data_exist' => $error, 'message' => "Mohon ditunggu", 'jobid' => $jo
    b_id);
27.    echo json_encode($data);
28. } else {
29.    $data = array('data_exist' => true, message ' => "Data tersedia",
30.       'result' => $result_amazon);
31.    echo json_encode($data);
32. }
33. }
```

Kode Sumber 4.29 Pengecekan harga pada *database* untuk pembelian barang di luar Asia Tenggara untuk pertama kali

Pada Kode Sumber 4.29 server menerima HTTP *request* POST dari aplikasi, dengan *body* yang berisi *key* “keyword” dan *value* kata yang diinput oleh pengguna. Server akan menerima *key* dan *value* yang dicantumkan dalam *body*, kemudian akan mengecek di dalam *database* terlebih dahulu. Apabila *value* yang dicari ditemukan dalam *database*, maka akan langsung dikirimkan ke pengguna dengan tambahan “message” bahwa data tersedia. Hasil kueri dan “message” akan di-*encode* dalam JSON sebelum dikirimkan pengguna.

Apabila *value* yang diminta pengguna tidak ada dalam *database*, maka server akan melakukan cURL *request* ke ScrapingHub dan menerima jobid. Dibuat sebuah “message” berisi “Mohon ditunggu” dan kemudian dikirim bersama jobid dalam bentuk JSON.

4.4.1.3 Fungsi nth_search_lazada

```
1.  $keyword = $this - > input - > post("jobid");
2.  $ch = curl_init();
```

```
3.  curl_setopt($ch, CURLOPT_URL, "https://app.scrapinghub.com/api/jobs/list.json?pro
    ject=305820&job=$keyword");
4.  curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
5.  curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
6.  curl_setopt($ch, CURLOPT_USERPWD, "ce1064bb139a4c9abf2e983c4e623345".
7.    ":".
8.    "");
9.  $result = curl_exec($ch);
10. if (curl_errno($ch)) {
11.   echo 'Error:'.curl_error($ch);
12. }
13. curl_close($ch);
14. $semifinal_result = json_decode($result, true);
15. if ($semifinal_result['jobs'][0]['state'] == 'finished') {
16.   $ch = curl_init();
17.   $url_to_get = "https://storage.scrapinghub.com/items/".$keyword.
18.   "?format=json";
19.   curl_setopt($ch, CURLOPT_URL, $url_to_get);
20.   curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
21.   curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
22.   curl_setopt($ch, CURLOPT_USERPWD, "ce1064bb139a4c9abf2e983c4e623345".
23.     ":".
24.     "");
25.   $result = curl_exec($ch);
26.   if (curl_errno($ch)) {
27.     echo 'Error:'.curl_error($ch);
28.   }
29.   $items = json_decode($result, true);
30.   foreach($items as $item) {
31.     $item['product_name'] = str_replace("'", '', $item['product_name']);
32.     $succeed = $this - > BEAIDB_model - > insert_data_lazada($item['product_name'
    ], $item['product_sale_price'], $item['page_url']);
33.   }
34.   $data = array('message' => 'finished', 'result' => $items);
35.   curl_close($ch);
36.   echo json_encode($data);
37. } else {
38.   $data = array('message' => 'nope', 'result' => 'none');
39.   echo json_encode($data);
40. }
```

Kode Sumber 4.30 Pengecekan apakah *crawling* pada Lazada sudah selesai atau belum dan respon yang dikirim

Pada Kode Sumber 4.30 server menerima HTTP request POST dari aplikasi lagi dengan *body* berisi key “jobid” dan *value* jobid. Server yang menerima *request* tersebut akan mengirim cURL *request* lagi untuk mengecek apakah jobid tersebut sudah selesai atau belum. Apabila *crawling* dengan jobid yang diterima sudah selesai dan mendapatkan hasil pencarian, maka server akan menyimpan data tersebut di dalam *database* dengan memanggil "insert_data_lazada" dan mengirimkan “message” berisi “finished” dan hasil pencarian yang sudah di-*encode* dalam JSON ke aplikasi. Namun, jika *crawling* statusnya masih belum “finished” maka server akan merespon dengan “message” berisi “none”.

4.4.1.4 Fungsi nth_search_amazon

```
41. $keyword = $this - > input - > post("jobid");
42. $ch = curl_init();
43. curl_setopt($ch, CURLOPT_URL, "https://app.scrapinghub.com/api/jobs/list.json?pro
    ject=305820&job=$keyword");
44. curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
45. curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
46. curl_setopt($ch, CURLOPT_USERPWD, "ce1064bb139a4c9abf2e983c4e623345".
47.     ":".
48.     "");
49. $result = curl_exec($ch);
50. if (curl_errno($ch)) {
51.     echo 'Error:'.curl_error($ch);
52. }
53. curl_close($ch);
54. $semifinal_result = json_decode($result, true);
55. if ($semifinal_result['jobs'][0]['state'] == 'finished') {
56.     $ch = curl_init();
57.     $url_to_get = "https://storage.scrapinghub.com/items/".$keyword.
58.     "?format=json";
59.     curl_setopt($ch, CURLOPT_URL, $url_to_get);
60.     curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
61.     curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
62.     curl_setopt($ch, CURLOPT_USERPWD, "ce1064bb139a4c9abf2e983c4e623345".
63.         ":".
64.         "");
65.     $result = curl_exec($ch);
```

```
66.     if (curl_errno($ch)) {
67.         echo 'Error:'.curl_error($ch);
68.     }
69.     $items = json_decode($result, true);
70.     foreach($items as $item) {
71.         $item['product_name'] = str_replace("'", '', $item['product_name']);
72.         $succeed = $this - > BEAIDB_model - > insert_data_amazon($item['product_nam
    e'], $item['product_sale_price'], $item['page_url']);
73.     }
74.     $data = array('message' => 'finished', 'result' => $items);
75.     curl_close($ch);
76.     echo json_encode($data);
77. } else {
78.     $data = array('message' => 'nope', 'result' => 'none');
79.     echo json_encode($data);
80. }
```

Kode Sumber 4.31 Pengecekan apakah *crawling* pada Amazon sudah selesai atau belum dan respon yang dikirim

Pada Kode Sumber 4.31 server menerima HTTP request POST dari aplikasi lagi dengan *body* berisi key “jobid” dan *value* jobid. Server yang menerima *request* tersebut akan mengirim cURL *request* lagi untuk mengecek apakah jobid tersebut sudah selesai atau belum. Apabila *crawling* dengan jobid yang diterima sudah selesai dan mendapatkan hasil pencarian, maka server akan menyimpan data tersebut di dalam *database* dengan memanggil "insert_data_amazon" dan mengirimkan “message” berisi “finished” dan hasil pencarian yang sudah di-*encode* dalam JSON ke aplikasi. Namun, jika *crawling* statusnya masih belum “finished” maka server akan merespon dengan “message” berisi “none”.

4.4.1.5 Fungsi kurs_hari_ini

```
1.  $url = "https://app.scrapinghub.com/api/items.json?project=305820&spider=kurscuk
    ai&apikey=ce1064bb139a4c9abf2e983c4e623345";
2.  $result = file_get_contents($url);
3.  $result_decoded = json_decode($result);
4.  $kurs = [];
5.  $kurs = $result_decoded[0];
6.  foreach($kurs as $kurs_ke_rupiah) {
```

```
7.      $kurs_ke_rupiah = str_replace(",", '', $kurs_ke_rupiah);
8.      $succeed = $this - > BEAIDB_model - > insert_kurs(key($kurs), $kurs_ke_rupiah);
9.      next($kurs);
10. }
```

Kode Sumber 4.32 Pembaruan daftar kurs pada *database* server

Pada Kode Sumber 4.32 server akan mengunduh daftar kurs dari ScrapingHub dalam bentuk JSOn dan melakukan *parsing*, yang kemudian akan disimpan ke dalam *database* dengan memanggil insert_kurs. Fungsi ini akan di-*cronjob* untuk dilakukan sekali setiap harinya.

4.4.1.6 Fungsi update_kurs

```
1.  $result_kurs = $this - > BEAIDB_model - > update_kurs();
2.  echo json_encode($result_kurs);
```

Kode Sumber 4.33 Pembaruan daftar kurs pada *database* aplikasi

Pada Kode Sumber 4.33 server akan menerima *request* dari aplikasi untuk mengambil daftar kurs dari *database* server. Server akan memanggil 'update_kurs' dan akan men-*encode* daftar kurs dalam JSON dan mengirimkannya ke aplikasi.

4.4.1.7 Fungsi update_amazon_lazada

```
1.  $url = "https://app.scrapinghub.com/api/items.json?project=305820&spider=amazo
    n_crawler&apikey=ce1064bb139a4c9abf2e983c4e623345";
2.  $result = file_get_contents($url);
3.  $items = json_decode($result, true);
4.  if (is_array($items)) {
5.      foreach($items as $item) {
6.          $item['product_name'] = str_replace("'", '', $item['product_name']);
7.          $succeed = $this - > BEAIDB_model - > update_data_amazon($item['product_na
    me'], $item['product_sale_price'], $item['page_url']);
8.      }
9.  }
10. $url = "https://app.scrapinghub.com/api/items.json?project=305820&spider=lazada_
    crawler&apikey=ce1064bb139a4c9abf2e983c4e623345";
11. $result = file_get_contents($url);
```

```
12. $items = json_decode($result, true);
13. if (is_array($items)) {
14.     foreach($items as $item) {
15.         $item['product_name'] = str_replace("'", '', $item['product_name']);
16.         $succeed = $this - > BEAIDB_model - > update_data_lazada($item['product_nam
    e'], $item['product_sale_price'], $item['page_url']);
17.     }
18. }
```

Kode Sumber 4.34 Pembaruan isi tabel amazon_DB dan lazada_DB

Pada Kode Sumber 4.34 server akan mengunduh dua *file* JSON, yang pertama adalah hasil dari *crawling* kategori dan subkategori pada Amazon dan yang kedua pada Lazada. Server kemudian akan melakukan *parsing* kedua *file* tersebut dan memperbarui nilai harga sesuai dengan nama produk pada tabel amazon_DB dan lazada_DB dengan memanggil 'update_data_amazon' dan 'update_data_lazada'.

4.4.2 Model

```
1.  public function first_try_search_lazada($keywords) {
2.      $query = "SELECT product_name, product_sale_price FROM lazada_DB WHERE
    product_name LIKE '%$keywords%'";
3.      $result = $this - > db - > query($query);
4.      return $result - > result();
5.  }
6.  public function first_try_search_amazon($keywords) {
7.      $query = "SELECT product_name, product_sale_price FROM amazon_DB WHER
    E product_name LIKE '%$keywords%'";
8.      $result = $this - > db - > query($query);
9.      return $result - > result();
10. }
```

Kode Sumber 4.35 Pencarian harga barang berdasarkan kata kunci pada tabel lazada_DB dan amazon_DB

Pada Kode Sumber 4.35, server melakukan pencarian harga barang berdasarkan kata kunci yang diberikan pengguna menggunakan fungsi 'SELECT', dimana harga yang dikembalikan memiliki nama yang mengandung kata kunci di dalamnya.

```
1.  public function insert_data_lazada($name, $price, $page_url) {
2.     $query = "INSERT INTO lazada_DB VALUES ('$price','$name','$page_url')";
3.     $result = $this - > db - > query($query);
4.     return $result;
5.  }
6.  public function insert_data_amazon($name, $price, $page_url) {
7.     $query = "INSERT INTO amazon_DB VALUES ('$name','$price','$page_url')";
8.     $result = $this - > db - > query($query);
9.     return $result;
10. }
```

Kode Sumber 4.36 Pengisian barang baru pada tabel lazada_DB dan amazon_DB

Pada Kode Sumber 4.36, server melakukan pengisian nama, harga, dan alamat produk di kedua tabel lazada_DB dan amazon_DB.

```
1.  public function update_data_amazon($name, $price, $page_url) {
2.     $query = "UPDATE amazon_DB SET product_sale_price='$price' WHERE product_n
    ame = '$name'";
3.     $result = $this - > db - > query($query);
4.     return $result;
5.  }
6.  public function update_data_lazada($name, $price, $page_url) {
7.     $query = "UPDATE lazada_DB SET product_sale_price='$price' WHERE product_na
    me = '$name'";
8.     $result = $this - > db - > query($query);
9.     return $result;
10. }
```

Kode Sumber 4.37 Pembaruan harga dan alamat produk pada tabel lazada_DB dan amazon_DB

Pada Kode Sumber 4.37, server akan memperbarui harga-harga yang ada pada server apabila barang tersebut sudah ada di *database*.

```
1.  public function insert_kurs($kurs, $kurs_value) {
2.     $query = "UPDATE kurs_tabel SET $kurs='$kurs_value'";
3.     $result = $this - > db - > query($query);
4.     return $result;
5.  }
```

Kode Sumber 4.38 Pembaruan daftar kurs pada tabel kurs_tabel

Pada Kode Sumber 4.38, server akan memperbarui daftar kurs pada tabel kurs_tabel.

```
1.  public function update_kurs() {
2.     $query = "SELECT * FROM kurs_tabel";
3.     $result = $this - > db - > query($query);
4.     return $result - > result();
5.  }
```

Kode Sumber 4.39 Pengambilan daftar kurs dari tabel kurs_tabel

Pada Kode Sumber 4.39, server akan *fetching* daftar kurs pada kurs_tabel dan mengembalikannya kepada *controller*.

4.4.3 Pertukaran Data menggunakan JSON

4.4.3.1 Percobaan pertama pencarian harga barang

Server akan menerima berbagai *request* dengan isi *body* yang berbeda. Kode Sumber 4.40 adalah contoh isi POST *request* yang server terima pada percobaan pertama kali dari aplikasi.

```
1.  {
2.     "keyword": "dragonball"
3.  }
```

Kode Sumber 4.40 Contoh isi *request* aplikasi pada percobaan pertama dari aplikasi ke server

Server memiliki 2 opsi dalam merespon *request* ini yang pertama adalah Kode Sumber 4.41 dimana server memiliki harga pada *database*. Sementara Kode Sumber 4.42 dimana server tidak memiliki harga pada *database*. Server akan mengirim jobid yang akan aplikasi gunakan untuk *request* selanjutnya.

```
1.  {
2.     "dataexist": true,
3.     "message": "Data tersedia",
```

```
4.    "result": [{
5.      "product_name": "dragonball",
6.      "product_price": "500"
7.    }, {
8.      "product_name": "dragonball x",
9.      "product_price": "45"
10.   }]
11. }
```

Kode Sumber 4.41 Contoh isi *response* server kepada aplikasi apabila barang terdapat di *database*

```
1. {
2.   "dataexist": false,
3.   "message": "Mohon ditunggu",
4.   "jobid": "4/50"
5. }
```

Kode Sumber 4.42 Contoh isi *response* server kepada aplikasi apabila barang tidak terdapat di *database*

4.4.3.2 Pencarian harga barang dengan pengecekan status *crawling*

Apabila harga barang tidak tersedia pada *database*, server akan mengirim jobid kepada aplikasi. Aplikasi kemudian akan mengirim kembali jobid tersebut ke server dengan menggunakan metode POST dengan *body* pada Kode Sumber 4.43.

```
1. {
2.   "jobid": "4/50"
3. }
```

Kode Sumber 4.43 Contoh isi *request* aplikasi mengecek harga barang dengan jobid

Server kemudian akan mengecek *crawling* sudah selesai atau belum. Apabila sudah akan dikembalikan dengan contoh Kode Sumber 4.44 Sementara apabila belum, server akan mengirim dengan isi *body* Kode Sumber 4.45

```
1. {
2.   "message": "finished",
```

```
3.      "result": [{
4.        "product_name": "dragonball",
5.        "product_price": "500"
6.          }, {
7.        "product_name": "dragonball x",
8.        "product_price": "45"
9.      }]
10. }
```

Kode Sumber 4.44 Contoh isi *response* server apabila *crawling* selesai

```
1. {
2.    "message": "nope",
3.    "result": "none"
4. }
```

Kode Sumber 4.45 Contoh isi *response* server apabila masih *crawling*

4.4.3.3 Permintaan daftar kurs hari ini oleh aplikasi

Untuk mendapatkan kurs hari ini, server akan menerima *request* GET dari aplikasi, yang kemudian server akan mengirimkan respon dengan isi *body* dengan contoh Kode Sumber 4.46.

```
1. {
2.     "USD": "13950.00",
3.     "SGD": "10512.00"
4. }
```

Kode Sumber 4.46 Contoh isi *response* server mengenai daftar kurs

4.4.4 Penjadwalan pembaruan nilai data menggunakan cron

Ada beberapa fungsi yang perlu dipanggil secara otomatis oleh server, karena tidak dapat diakses langsung menggunakan aplikasi. Fungsi-fungsi seperti pembaruan daftar kurs pada *database* server, dan pembaruan harga barang pada *database* server perlu menggunakan cron untuk pemanggilan berkala pada periode tertentu. Kode Sumber 4.47 merupakan implementasi dari *cron* yang dilakukan.

```
5 1 * * * /usr/bin/wget 128.199.186.191/codeigniter/Kurs_hari ini
0 5 1, 15 * * /usr/bin/wget 128.199.186.191/codeigniter/Insert_request/updata_amazon_lazada
```

Kode Sumber 4.47 Cron pada server

4.5 Implementasi Android

4.5.1 Menu Utama

```
offlineCalculator.setOnClickListener(new View.OnClickListener() {@
    Override public void onClick(View view) {
        AlertDialog.Builder pilih_online_offline = new AlertDialog.Builder(Menu_BEAI.this);
        pilih_online_offline.setMessage("Apakah anda mengetahui harga barang yang ingin dimasukkan?").setPositiveButton("Ya", new DialogInterface.OnClickListener() {@
            Override public void onClick(DialogInterface dialog, int which) {
                Intent intent_offline = new Intent(getBaseContext(), Duty_Calculator_Offline.class);
                startActivity(intent_offline);
            }
        }).setNegativeButton("Tidak", new DialogInterface.OnClickListener() {@
            Override public void onClick(DialogInterface dialog, int which) {
                Intent intent_online = new Intent(getBaseContext(), Duty_Calculator_Online.class);
                startActivity(intent_online);
            }
        }).create().show();
            } });
```

Kode Sumber 4.48 Implementasi Pop Up

Pada menu utama, terdapat tiga tombol : Duty Calculator, Currency, dan History. Pada tombol Duty Calculator ditekan, maka akan muncul *pop up* pilihan untuk menggunakan harga yang diketahui oleh pengguna atau harga dari server. Kode Sumber 4.48 merupakan implementasi dari fungsi *pop up* yang terdapat pada tombol tersebut.

```
private void getKurs() {
    String url;
    url = "http://128.199.186.191/codeigniter/Search_request/update_kurs";
```

```
4.    offlineCalculator.setEnabled(false);
5.    onlineCalculator.setEnabled(false);
6.    StringRequest stringRequest = new StringRequest(Request.Method.GET, url, new R
    esponse.Listener < String > () {@
7.      Override public void onResponse(String response) {
8.        kursmodellist.addAll(ParseJson.parseKurs(response));
9.        database.update_kurs(kursmodellist);
10.     }
11.   }, new Response.ErrorListener() {@
12.     Override public void onErrorResponse(VolleyError error) {
13.       Log.d("check_err", error.toString());
14.     }
15.   });
16.   RequestQueue requestQueue = Volley.newRequestQueue(this);
17.   requestQueue.add(stringRequest);
18.   offlineCalculator.setEnabled(true);
19.   onlineCalculator.setEnabled(true);
20. }
```

Kode Sumber 4.49 Implementasi pembaruan daftar kurs

Pada saat aplikasi dibuka, menu merupakan *activity* yang pertama kali akan dilakukan. Oleh sebab itu, kita akan menjalankan fungsi untuk memperbarui daftar kurs pada aplikasi saat aplikasi dibuka. Pada Kode Sumber 4.49 aplikasi mengirim POST *request* ke server untuk mendapatkan daftar kurs terbaru. Respon server akan di-*parse* dan tabel kurs pada aplikasi akan diperbarui.

4.5.2 Duty Calculator Offline

```
1.  tambahBarang.setOnClickListener(new View.OnClickListener() {@
2.    Override public void onClick(View view) {
3.      BarangModel barangModel = new BarangModel();
4.      barangModel.nama = namaBarang.getText().toString();
5.      barangModel.mataUang = valuta.getSelectedItem().toString();
6.      barangModel.harga = Double.parseDouble(hargaI.getText().toString());
7.      hitungBEAI.setVisibility(View.VISIBLE);
8.      daftar_text.setVisibility(View.VISIBLE);
9.      barangModelList.add(barangModel);
10.     barangAdapter.notifyDataSetChanged();
11.     double nilaiItem = barangModel.harga * nilai_tukar / 14136.00;
12.     hitung = hitung + nilaiItem;
```

```
13.         Date c = Calendar.getInstance().getTime();
14.         SimpleDateFormat df = new SimpleDateFormat("dd-MMM-yyyy");
15.         String formattedDate = df.format(c);
16.         database.add_history(formattedDate, namaBarang.getText().toString(), listBaran
    g[urutan][2], barangModel.harga, barangModel.harga);
17.         Context context = getApplicationContext();
18.         CharSequence text = "Barang telah ditambah.";
19.         int duration = Toast.LENGTH_SHORT;
20.         Toast toast = Toast.makeText(context, text, duration);
21.         toast.show();
22.     }
23. });
```

Kode Sumber 4.50 Implementasi menambah barang pada offline duty calculator

```
1.  hitungBEAI.setOnClickListener(new View.OnClickListener() {@
2.      Override public void onClick(View view) {
3.          tambahBarang.setEnabled(false);
4.          double nilaiPabean, nilaiPPN, nilaiPPnBM, nilaiPPh, total;
5.          if (hitung > 500.0) {
6.              totalBEAI.setVisibility(View.VISIBLE);
7.              hitung = (hitung - 500.0) * 14136.00;
8.              nilaiPabean = Math.floor(hitung * 0.1 * 100) / 100;
9.              nilaiPPN = Math.floor((hitung + nilaiPabean) * 0.1 * 100) / 100;
10.             nilaiPPnBM = Math.floor((hitung + nilaiPabean) * 0);
11.             nilaiPPh = Math.floor((hitung + nilaiPabean) * ratePPh * 100) / 100;
12.             total = nilaiPabean + nilaiPPN + nilaiPPnBM + nilaiPPh;
13.         } else {
14.             totalBEAI.setVisibility(View.VISIBLE);
15.             hitung = 0;
16.             nilaiPabean = 0;
17.             nilaiPPN = 0;
18.             nilaiPPnBM = 0;
19.             nilaiPPh = 0;
20.             total = nilaiPabean + nilaiPPN + nilaiPPnBM + nilaiPPh;
21.         }
22.         Bea.setText(String.valueOf(nilaiPabean));
23.         PPN.setText(String.valueOf(nilaiPPN));
24.         PPnBM.setText(String.valueOf(nilaiPPnBM));
25.         PPh.setText(String.valueOf(nilaiPPh));
26.         Total.setText(String.valueOf(total)); //TextView valuta = (TextView) findViewById
    (R.id.valuta);
27.     }
```

```
28. });
```

Kode Sumber 4.51 Implementasi menghitung pajak pada offline duty calculator

```
1.  public class BarangModel {
2.     public String nama;
3.     public String mataUang;
4.     public Double harga;
5.  }
```

Kode Sumber 4.52 kelas barangModel

Pada menu Offline Duty Calculator, ada dua fungsi yaitu fungsi menambah barang ke dalam tabel yang terdiri dari nama barang, mata uang, dan harga barang, yang akan dihitung totalnya menggunakan fungsi menghitung pajak dari total semua barang yang sudah ditambahkan ke dalam tabel. Fungsi pertama dapat dilihat pada Kode Sumber 4.50, dimana di-*trigger* saat pengguna memencet tombol TAMBAH. Nama, harga dan mata uang yang diinput akan ditambahkan ke dalam barangModel pada Kode Sumber 4.52. Kumpulan barangModel akan dibuat menjadi sebuah list yang bernama barangModelList. Barang yang ditambahkan akan langsung diubah menjadi USD dan ditambahkan ke dalam variabel 'hitung', dan membuat tombol HITUNG dan tabel daftar barang yang sudah diinput menjadi *visible*.

Sementara Kode Sumber 4.51 terdapat pada tombol HITUNG yang muncul setelah tombol TAMBAH ditekan. Ketika tombol HITUNG ditekan pengguna, aplikasi cukup mengecek variabel 'hitung' dan melakukan kalkulasi pajak terhadap variabel 'hitung', dan mengganti nilai pada masing-masing variabel Pabean, PPN, PPnBM, PPh, dan total pajak yang akan ditampilkan pada layar.

4.5.3 Duty Calculator Online

```
1.  search_amazon.setOnClickListener(new View.OnClickListener() {@
2.     Override public void onClick(View view) {
3.        barangModelList.clear();
4.        valuta = "USD";
```

```
5.        Intent intent_search = new Intent(view.getContext(), search_result.class);
6.        String message = namaBarang.getText().toString();
7.        intent_search.putExtra("keyword", message);
8.        intent_search.putExtra("valuta", valuta);
9.        intent_search.putExtra("sites", "amazon");
10.       startActivity(intent_search);
11.       String sites = "amazon"; ;
12.     }
13. });
```

Kode Sumber 4.53 Implementasi mengirim Intent untuk melakukan pencarian harga pada barang yang dibeli di luar Asia Tenggara

```
1.  search_lazada.setOnClickListener(new View.OnClickListener() {@
2.     Override public void onClick(View view) {
3.        barangModelList.clear();
4.        valuta = "SGD";
5.        Intent intent_search = new Intent(view.getContext(), search_result.class);
6.        String message = namaBarang.getText().toString();
7.        intent_search.putExtra("keyword", message);
8.        intent_search.putExtra("valuta", valuta);
9.        intent_search.putExtra("sites", "lazada");
10.       startActivity(intent_search);
11.       String sites = "lazada";
12.     }
13. });
```

Kode Sumber 4.54 Implementasi mengirim Intent untuk melakukan pencarian harga pada barang yang dibeli di Asia Tenggara

Pengguna dapat memilih lokasi pembelian barang, dimana akan men-*trigger* Kode Sumber 4.53 apabila memilih di luar Asia Tenggara atau Kode Sumber 4.54 apabila memilih di Asia Tenggara. Ketika pengguna mendapatkan harga dari luar Asia Tenggara, maka mata uang yang digunakan adalah USD dan situs tempat harga acuan adalah Amazon, sementara dari Asia Tenggara mata uang yang digunakan adalah SGD dan situs acuannya adalah Lazada.sg. Oleh sebab itu, terdapat 3 extra yang akan kita taruh dalam Intent; kata kunci, mata uang, serta situs yang akan digunakan.

```
1.  String url;
```

```
2.  if (sites.equals("amazon")) {
3.     url = "http://128.199.186.191/codeigniter/Search_request/first_search_amazon";
4.  } else {
5.     url = "http://128.199.186.191/codeigniter/Search_request/first_search_lazada";
6.  }
7.  StringRequest stringRequest = new StringRequest(Request.Method.POST, url, new Re
    sponse.Listener < String > () {@
8.     Override public void onResponse(String response) {
9.        String dataExist = ParseJson.parseError(response);
10.       if (dataExist.equals("true")) {
11.          loadthis.setVisibility(View.GONE);
12.          barangModelList.addAll(ParseJson.parseBArang(response, sites));
13.          ya.setEnabled(true);
14.          tidak.setEnabled(true);
15.          barangAdapter.notifyDataSetChanged();
16.       } else {
17.          Intent intent = new Intent(getBaseContext(), CobaLagiService.class);
18.          String jobId = ParseJson.parseJobId(response);
19.          String jobId_cleared = jobId.replaceAll("\\\\", "");
20.          intent.putExtra("job_id", jobId_cleared);
21.          intent.putExtra("site", sites);
22.          startService(intent);
23.       }
24.    }
25. }, new Response.ErrorListener() {@
26.    Override public void onErrorResponse(VolleyError error) {
27.       Log.d("check_keyword", error.toString());
28.    }
29. }) {@
30.    Override protected Map < String, String > getParams() throws AuthFailureError {
31.       Map < String, String > param = new HashMap < > ();
32.       param.put("keyword", keyword);
33.       return param;
34.    }
35. };
36. RequestQueue requestQueue = Volley.newRequestQueue(this);
37. requestQueue.add(stringRequest);
```

Kode Sumber 4.55 Implementasi pencarian harga pada kata kunci yang diberikan

Pada Kode Sumber 4.55 aplikasi akan menentukan alamat URL yang akan menjadi tujuan pengiriman POST *request* dengan

extra situs yang didapatkan dari kode sebelumnya. Aplikasi kemudian akan melakukan *parsing* JSON yang diterima, dan menentukan langkah selanjutnya berdasarkan variabel dataExist, apabila *true* akan langsung ditambahkan ke barangModel sementara akan menyimpan jobid dan memulai *service* CobaLagiService bila variabel dataExist *false*.

```
1.  @Override public int onStartCommand(Intent intent, int flags, int startId) {
2.    final String jobId = intent.getStringExtra("job_id");
3.    final String sites = intent.getStringExtra("site");
4.    timer = new Timer();
5.    timerTask = new TimerTask() {@
6.      Override public void run() {
7.        requestAgain(jobId, sites);
8.      }
9.    };
10.   timer.schedule(timerTask, 45 * 1000, 45 * 1000);
11.   return START_REDELIVER_INTENT;
12. }
```

Kode Sumber 4.56 Implementasi *service* CobaLagiService

```
1.  private void requestAgain(final String jobId, final String sites) {
2.    String url;
3.    Log.d("check_keyword", sites);
4.    if (sites.equals("amazon")) {
5.      url = "http://128.199.186.191/codeigniter/Insert_request/nth_search_amazon";

6.    } else {
7.      url = "http://128.199.186.191/codeigniter/Insert_request/nth_search_lazada";
8.    }
9.    StringRequest stringRequest = new StringRequest(Request.Method.POST, url, new
    Response.Listener < String > () {@
10.     Override public void onResponse(String response) {
11.       String message = ParseJson.parseMessage(response);
12.       if (message.equals("finished")) {
13.         ArrayList < BarangModel > barangModels = new ArrayList < > ();
14.         barangModels.addAll(ParseJson.parseBArang(response, sites));
15.         sendDataBack(barangModels);
16.         timer.cancel();
17.         timerTask.cancel();
18.         stopSelf(); //Send Notif
```

```
19.         }
20.     }
21.   }, new Response.ErrorListener() {@
22.     Override public void onErrorResponse(VolleyError error) {}
23.   }) {@
24.     Override protected Map < String, String > getParams() throws AuthFailureError {

25.         Map < String, String > param = new HashMap < > ();
26.         param.put("jobid", jobId);
27.         return param;
28.     }
29.   };
30.   RequestQueue requestQueue = Volley.newRequestQueue(this);
31.   requestQueue.add(stringRequest);
32. }
```

Kode Sumber 4.57 Implementasi fungsi requestAgain

Pada Kode Sumber 4.56, CobaLagiService akan memberi waktu 45 detik sebelum melakukan fungsi requestAgain pada Kode Sumber 4.57. Fungsi requestAgain akan mengirim POST *request* lagi ke server sesuai dengan variabel 'sites'. Respon server akan di-*parse* dan mengecek apakah variabel 'message' bernilai 'finished'. Apabila iya, semua nama barang dan harga barang akan dimasukkan ke barangModels, dikirimkan kembali menggunakan fungsi sendDataBack, dan timer pada CobaLagiService akan diberhentikan.

4.5.4 Currency

```
1.  init() {
2.    USD = (TextView) findViewById(R.id.USD);
3.    USD.setText(database.select_kurs("USD"));
4.    GBP = (TextView) findViewById(R.id.GBP);
5.    GBP.setText(database.select_kurs("GBP"));
6.    LKR = (TextView) findViewById(R.id.LKR);
7.    LKR.setText(database.select_kurs("LKR"));
8.    DKK = (TextView) findViewById(R.id.DKK);
9.    DKK.setText(database.select_kurs("DKK"));
10.   CAD = (TextView) findViewById(R.id.CAD);
11.   CAD.setText(database.select_kurs("CAD"));
```

```
12.     PKR = (TextView) findViewById(R.id.PKR);
13.     PKR.setText(database.select_kurs("PKR"));
14.     KWD = (TextView) findViewById(R.id.KWD);
15.     KWD.setText(database.select_kurs("KWD"));
16.     MYR = (TextView) findViewById(R.id.MYR);
17.     ...
18.     ...
19. }
```

Kode Sumber 4.58 Implementasi kurs terbaru dari *database* aplikasi

Pada saat tombol Currency ditekan di menu utama, aplikasi cukup mengambil daftar kurs yang sudah tersimpan di *database* aplikasi menggunakan select_kurs, seperti yang terdapat pada Kode Sumber 4.58.

4.5.5 History

```
1.  public class HistoryModel {
2.      public String tanggal;
3.      public String nama_barang;
4.      public String mata_uang;
5.      public String harga_min;
6.      public String harga_max;
7.  }
```

Kode Sumber 4.59 Implementasi History Model

Fitur History memiliki model tersendiri, yang terdiri dari tanggal kapan input dilakukan, nama barang yang diinput pengguna, mata uang yang digunakan, harga minimal dan maksimal suatu barang. Harga minimal dan harga maksimal akan sama apabila merupakan input pengguna, sementara dapat berbeda apabila terdapat lebih dari satu harga yang didapat dari server. Model ini dapat dilihat dari implementasi pada Kode Sumber 4.59.

```
1.  private void init() {
2.      daftarhistory = (RecyclerView) findViewById(R.id.daftarHistory);
3.      daftarhistory.setLayoutManager(new LinearLayoutManager(this));
4.      historyAdapter = new HistoryAdapter(historymodelList, this);
5.      daftarhistory.setAdapter(historyAdapter);
6.      historymodelList.addAll(database.tampil_history());
```

```
7.      historyAdapter.notifyDataSetChanged();
8.  }
```

Kode Sumber 4.60 Implementasi init view history

Untuk view History, akan menggunakan RecyclerView seperti pada Kode Sumber 4.60. Aplikasi akan membuat HistoryAdapter yang datanya terdiri dari historymodelList, yang kemudian akan dibuat sebagai adapter menggunakan fungsi setAdapter. historymodelList sendiri mengambil data dari *database* aplikasi tampil_history. historyAdapter kemudian memanggil fungsi notifyDataSetChanged untuk mengubah tampilan view.

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan pengujian dan evaluasi dari perangkat lunak ini yang meliputi rincian lingkungan uji coba dan pengujian pada semua fungsionalitas administratif dan semua menu pada modul yang terintegrasi. Aspek yang diperhatikan dalam pengujian perangkat lunak ini adalah terpenuhinya fungsionalitas administratif dan integrasi modul ke dalam perangkat lunak.

5.1 Lingkungan Uji Coba

Lingkungan uji coba merupakan perangkat keras dan perangkat lunak yang digunakan selama melakukan pengujian. Pengujian dilakukan dengan menggunakan satu lingkungan pengujian yaitu aplikasi Android BEAI. Rincian dari lingkungan pengujian ditunjukkan pada Tabel 5.1

Tabel 5.1 Tabel Lingkungan Uji Coba Klien

Spesifikasi	Deskripsi
Tipe	Oppo F3
Versi Android	6.0
Prosesor	Octacore
RAM	4 GB

5.2 Uji Coba

Pada bagian ini akan dijelaskan pengujian yang dilakukan pada aplikasi BEAI. Pengujian yang akan dilakukan adalah pengujian fungsionalitas sistem dan pengujian pada pengujian pengguna. Uji coba fungsionalitas aplikasi dilakukan untuk mengetahui apakah fungsionalitas fitur sudah bekerja dengan baik

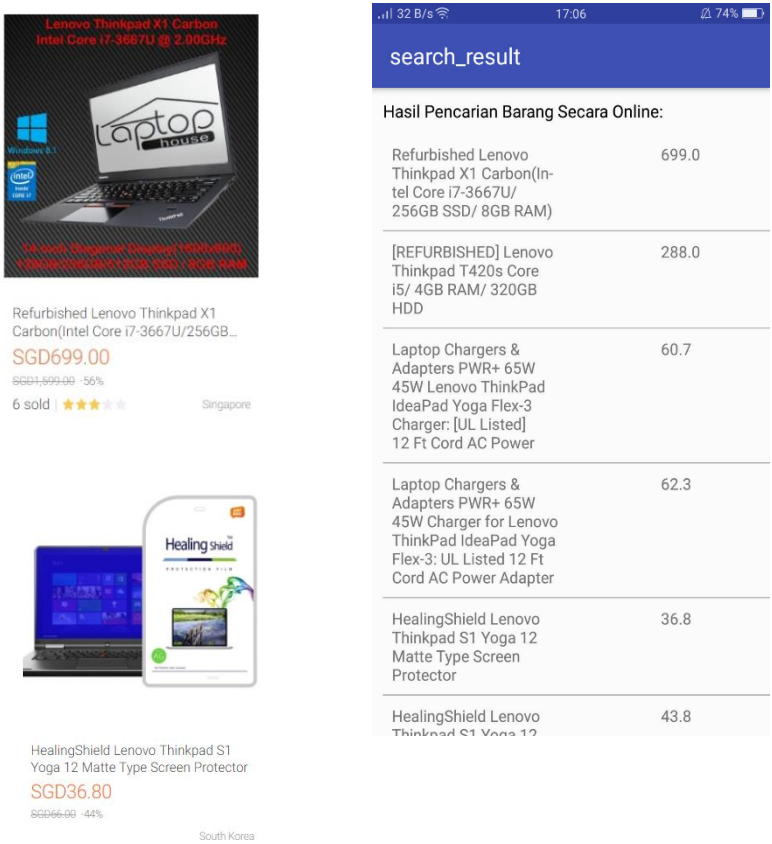
atau belum. Pengujian pengguna akan melibatkan pengguna untuk mencoba aplikasi BEAI. Tujuannya adalah mendapatkan *feedback* dari pengguna serta mengetahui apakah fitur memenuhi kegunaan yang dimaksudkan.

5.2.1 Pengujian Fungsionalitas Sistem

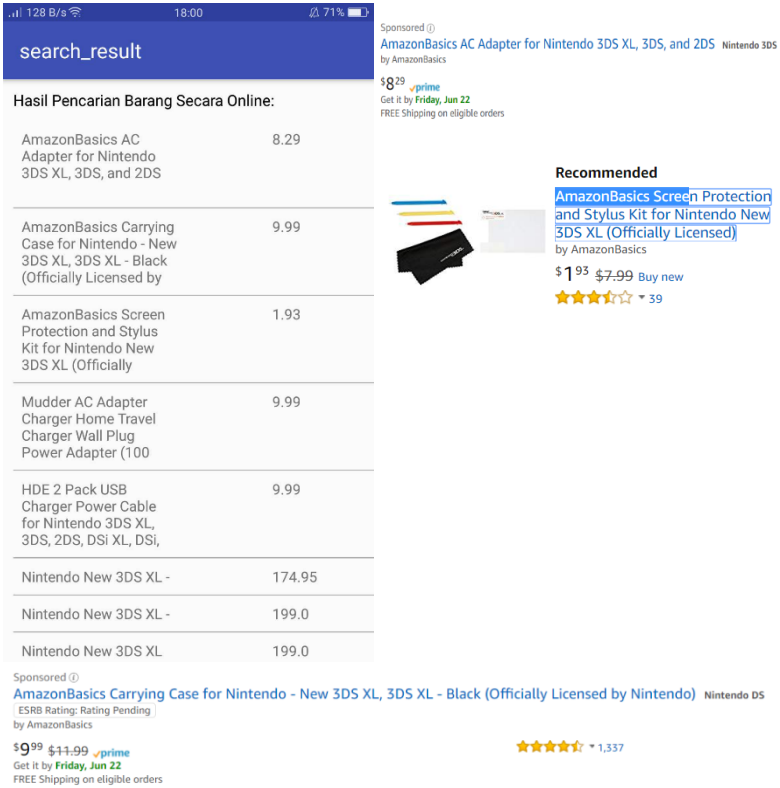
Pada uji coba fungsionalitas ini hasil perhitungan pajak pada aplikasi BEAI akan dibandingkan dengan aplikasi CEISA Mobile yang dikeluarkan oleh Direktorat Jenderal Bea dan Cukai. Pengujian akan dilakukan dengan menyiapkan beberapa skenario yang menjadi tolak ukur keberhasilan.

5.2.1.1 Uji Coba Pengecekan Harga yang Diterima dari Server

Pada uji coba ini, pengguna melakukan pencarian harga berdasarkan nama barang. Tabel 5.2 menjelaskan secara detail uji coba pengecekan harga yang didapat dari server. Pada Gambar 5.1 menampilkan perbandingan dari pencarian aplikasi dengan Lazada.sg, sementara pada Gambar 5.2 menampilkan perbandingan harga dari hasil pencarian aplikasi dengan Amazon.com



Gambar 5.1 Hasil Pengecekan Lazada



Gambar 5.2 Hasil Pengecekan Amazon

Tabel 5.2 Tabel Uji Coba Pengecekan Nama Barang

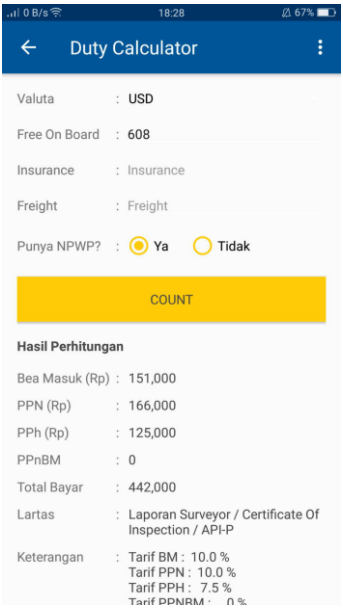
ID	**TC-001**
Kasus Penggunaan	Melakukan Pencarian Nama Barang.
Tujuan Pengujian	Menguji kebenaran harga yang diterima dari server

Skenario 1	*Pengguna menginput nama barang dan memilih lokasi di Asia Tenggara*
Kondisi Awal	Nama dan harga barang belum tampil
Langkah Pengujian	1. Pengguna menginput nama barang. 2. Pengguna memilih lokasi pembelian di Asia Tenggara 3. Pengguna memilih beberapa nama yang muncul di daftar hasil pencarian dan mencari di situs Lazada.sg
Hasil yang Diharapkan	Harga dan barang yang ditampilkan sesuai.
Kondisi Akhir	Aplikasi BEAI berhasil menampilkan harga dan barang yang sesuai dan berada di lazada.sg
Skenario 2	*Pengguna menginput nama barang dan memilih lokasi di luar Asia Tenggara*
Kondisi Awal	Nama dan harga barang belum tampil
Langkah Pengujian	1. Pengguna menginput nama barang. 2. Pengguna memilih lokasi pembelian di luar Asia Tenggara 3. Pengguna memilih beberapa nama yang muncul di daftar hasil pencarian dan mencari di situs Amazon.com
Hasil yang diharapkan	Harga dan barang yang ditampilkan sesuai.
Kondisi Akhir	Aplikasi BEAI berhasil menampilkan harga dan barang yang sesuai dan berada di Amazon.com

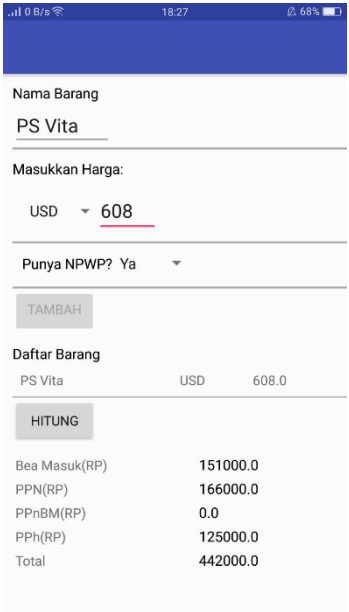
5.2.1.2 Uji Coba Perhitungan Pajak Menggunakan Pengisian Harga dari Pengguna

Pada uji coba ini, pengguna melakukan perhitungan pajak dimana harga yang diinput bersumber dari pengguna. Pengguna kemudian dapat membandingkan hasil perhitungan dengan aplikasi perhitungan pajak lainnya seperti CEISA Mobile yang

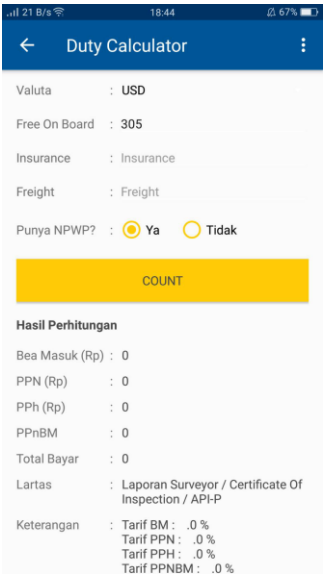
dikeluarkan Direktorat Jenderal Bea dan Cukai. Tabel 5.3 menjelaskan secara detail langkah-langkah uji coba perhitungan pajak menggunakan harga dari pengguna. Hasil perbandingan dapat dilihat dengan membandingkan Gambar 5.3 dengan Gambar 5.4 serta Gambar 5.5 dengan Gambar 5.6. Pada Gambar 5.3 biaya yang tercantum pada Bea Masuk (Rp), PPN(Rp), PPh(Rp), PPnBM, Total Bayar sama dengan Gambar 5.4. Demikian pula biaya Bea Masuk (Rp), PPN(Rp), PPh(Rp), PPnBM, Total Bayar yang tercantum pada Gambar 5.5 sama dengan Gambar 5.6.



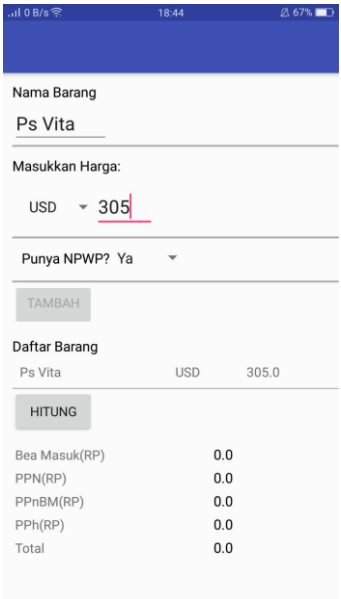
Gambar 5.3 CEISA Mobile > $500



Gambar 5.4 BEAI > $500



Gambar 5.5 CEISA Mobile < $500



Gambar 5.6 BEAI < $500

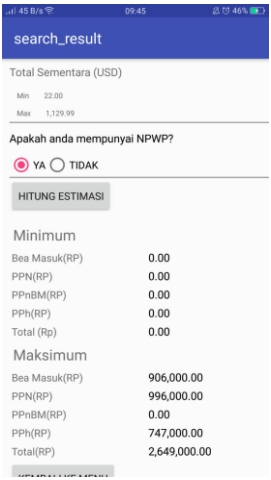
Tabel 5.3 Tabel Uji Coba Perhitungan Pajak Menggunakan Pengisian Harga dari Pengguna

ID	**TC-002**
Kasus Penggunaan	Melakukan Perhitungan Pajak Berdasarkan Harga yang Diinput Pengguna
Tujuan Pengujian	Menguji kebenaran perhitungan pajak pada Aplikasi BEAI
Skenario 1	*Pengguna menginput nama barang, harga di atas USD 500, mata uang, dan kepemilikan NPWP*
Kondisi Awal	Hasil perhitungan pajak belum muncul.
Langkah Pengujian	1. Pengguna menginput nama barang 2. Pengguna mengisi harga di atas USD 500 3. Pengguna memilih memiliki NPWP

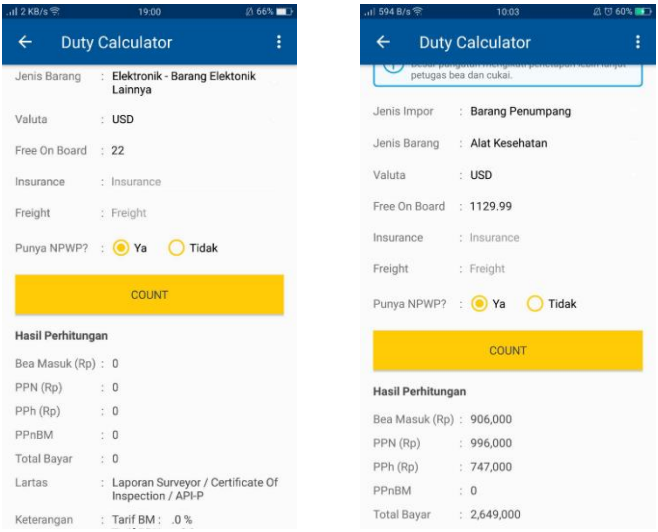
	4. Pengguna membuka aplikasi CEISA Mobile dan membuka Duty Calculator 5. Pengguna mengganti Jenis Impor menjadi Barang Penumpang 6. Pengguna mengisi Valuta USD 7. Pengguna mengisi Free On Board sesuai dengan harga yang diinput di BEAI
Hasil yang Diharapkan	Perincian pajak yang ditampilkan aplikasi BEAI sesuai dengan CEISA Mobile.
Kondisi Akhir	Aplikasi BEAI berhasil menampilkan perincian pajak yang sesuai dengan CEISA Mobile
Skenario 2	*Pengguna menginput nama barang, harga di bawah USD 500, mata uang, dan kepemilikan NPWP*
Kondisi Awal	Hasil perhitungan pajak belum muncul.
Langkah Pengujian	1. Pengguna menginput nama barang. 2. Pengguna mengisi harga di bawah USD 500 3. Pengguna memilih memiliki NPWP 4. Pengguna membuka aplikasi CEISA Mobile dan membuka Duty Calculator 5. Pengguna mengganti Jenis Impor menjadi Barang Penumpang 6. Pengguna mengisi Valuta USD 7. Pengguna mengisi Free On Board sesuai dengan harga yang diinput di BEAI
Hasil yang diharapkan	Perincian pajak yang ditampilkan aplikasi BEAI sesuai dengan CEISA Mobile.
Kondisi Akhir	Aplikasi BEAI berhasil menampilkan perincian pajak yang sesuai dengan CEISA Mobile

5.2.1.3 Uji Coba Perhitungan Pajak Menggunakan Harga yang Didapat dari Server

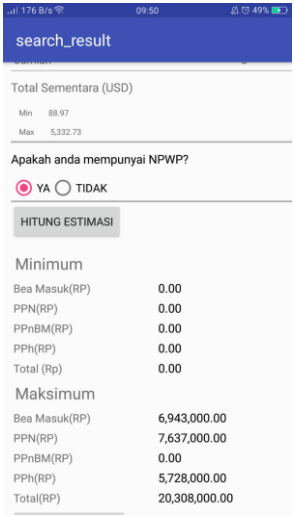
Pada uji coba ini, pengguna melakukan perhitungan pajak dimana harga yang diinput bersumber dari server. Pengguna kemudian dapat membandingkan hasil perhitungan dengan CEISA Mobile. Tabel 5.4 menjelaskan secara detail langkah-langkah uji coba perhitungan pajak menggunakan harga dari server. Hasil perbandingan dapat dilihat dengan melihat perbandingan antara Gambar 5.7 dengan Gambar 5.8 serta Gambar 5.9 dengan 5.10. Pada Gambar 5.7, total biaya pajak minimum bernilai nol, sama dengan perhitungan biaya pajak pada Gambar 5.8 bagian kiri. Total biaya pajak maksimum pada Gambar 5.7 bernilai 2.649.000 sama dengan perhitungan pajak pada Gambar 5.8 bagian kanan. Sementara pada Gambar 5.9 total nilai pajak minimum bernilai nol sama dengan Gambar 5.10 bagian kiri, dan total nilai pajak maksimum bernilai 20.308.000 sama dengan perhitungan pajak pada Gambar 5.10 bagian kanan.



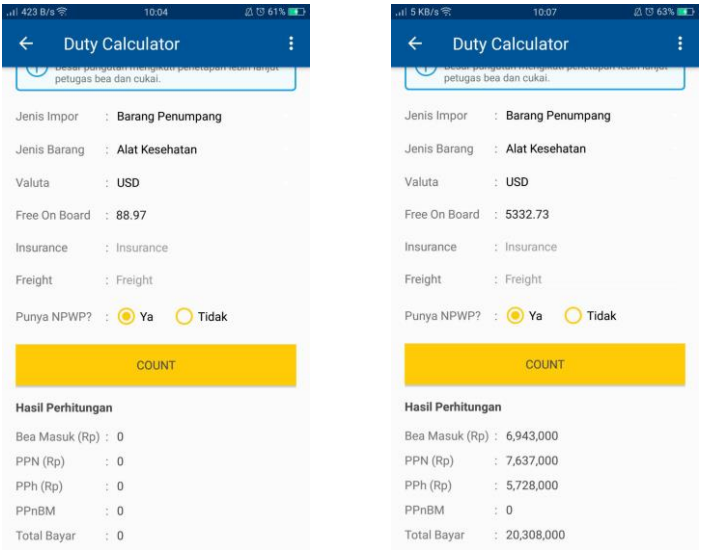
Gambar 5.7 Hasil Perhitungan Pajak BEAI terhadap satu Barang Menggunakan Harga yang Didapat dari Server



Gambar 5.8 Hasil Perhitungan Pajak CEISA Mobile



Gambar 5.9 Hasil Perhitungan Pajak BEAI terhadap Beberapa Barang Menggunakan Harga yang Didapat dari Server



Gambar 5.10 Hasil Perhitungan Pajak CEISA Mobile

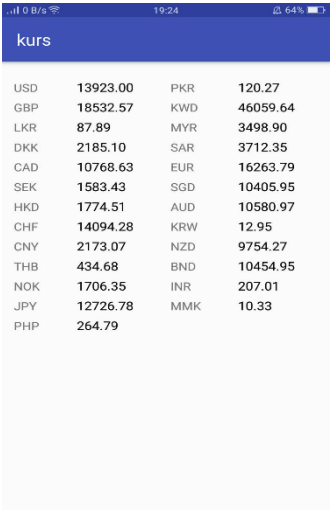
Tabel 5.4 Tabel Uji Coba Perhitungan Pajak Menggunakan Pengisian Harga

ID	**TC-003**
Kasus Penggunaan	Melakukan Perhitungan Pajak Berdasarkan Harga yang Didapat dari Server
Tujuan Pengujian	Menguji kebenaran perhitungan pajak pada Aplikasi BEAI
Skenario 1	*Pengguna menginput satu nama barang, lokasi pembelian barang dan kepemilikan NPWP*
Kondisi Awal	Hasil perhitungan pajak belum muncul.
Langkah Pengujian	1. Pengguna menginput nama barang. 2. Pengguna memilih luar Asia Tenggara untuk lokasi pembelian 3. Pengguna menginput jumlah barang 4. Pengguna menekan tombol TAMBAH

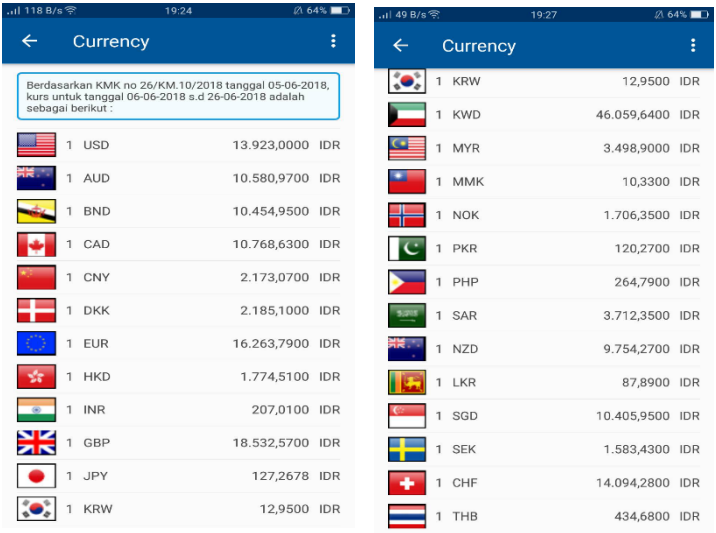
	5. Pengguna memilih memiliki NPWP 6. Pengguna menekan tombol HITUNG 7. Pengguna membuka aplikasi CEISA Mobile dan membuka Duty Calculator 8. Pengguna mengganti Jenis Impor menjadi Barang Penumpang 9. Pengguna mengisi Valuta USD 10. Pengguna mengisi Free On Board sesuai dengan harga yang diinput di BEAI
Hasil yang Diharapkan	Perincian pajak yang ditampilkan aplikasi BEAI sesuai dengan CEISA Mobile.
Kondisi Akhir	Aplikasi BEAI berhasil menampilkan perincian pajak yang sesuai dengan CEISA Mobile
Skenario 2	*Pengguna menginput beberapa nama barang dari satu lokasi pembelian barang dan kepemilikan NPWP*
Kondisi Awal	Hasil perhitungan pajak belum muncul
Langkah Pengujian	1. Pengguna menginput nama barang 2. Pengguna memilih luar Asia Tenggara untuk lokasi pembelian 3. Pengguna menginput jumlah barang 4. Pengguna menekan tombol TAMBAH 5. Pengguna menginput nama barang kedua 6. Pengguna menginput jumlah barang 7. Pengguna menekan tombol TAMBAH 8. Pengguna memilih memiliki NPWP 9. Pengguna menekan tombol HITUNG 10. Pengguna membuka aplikasi CEISA Mobile dan membuka Duty Calculator 11. Pengguna mengganti Jenis Impor menjadi Barang Penumpang 12. Pengguna mengisi Valuta USD 13. Pengguna mengisi Free On Board sesuai dengan harga yang diinput di BEAI

5.2.1.4 Uji Coba Pengecekan Kurs

Pada uji coba ini, pengguna melakukan pengecekan kurs yang tersedia pada aplikasi BEAI. Pengguna kemudian dapat membandingkan kurs yang ditampilkan BEAI dengan CEISA Mobile. Tabel 5.5 menjelaskan secara detail langkah-langkah uji coba pengecekan kurs pada aplikasi BEAI. Hasil perbandingan dapat dilihat dengan melihat perbandingan antara Gambar 5.11 dengan Gambar 5.12. Daftar kurs pada Gambar 5.11 memiliki nilai yang sesuai dengan daftar kurs pada Gambar 5.12



Gambar 5.11 Kurs BEAI



Gambar 5.12 Kurs CEISA Mobile

Tabel 5.5 Tabel Uji Coba Pengecekan Harga Kurs

ID	**TC-004**
Kasus Penggunaan	Melakukan Pengecekan Harga Kurs
Tujuan Pengujian	Menguji kebenaran kurs pada Aplikasi BEAI
Skenario 1	*Pengguna memilih Currency pada menu utama untuk mengecek kurs saat ini*
Kondisi Awal	Daftar kurs untuk perpajakan belum muncul.
Langkah Pengujian	1. Pengguna memilih Currency pada menu Utama. 2. Pengguna membuka CEISA Mobile dan memilih Currency

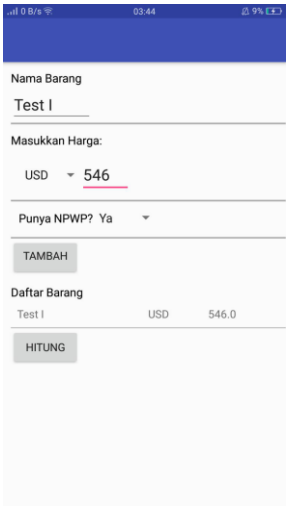
	3. Pengguna membandingkan kedua hasil output dari kedua aplikasi.
Hasil yang Diharapkan	Daftar kurs yang dikeluarkan aplikasi BEAI sesuai dengan kurs yang dikeluarkan CEISA Mobile
Kondisi Akhir	Aplikasi BEAI berhasil menampilkan kurs perpajakan yang sesuai dengan CEISA Mobile

5.2.1.5 Uji Coba Pengecekan History

Pada uji coba ini, pengguna melakukan pengecekan fitur *history* dengan membandingkan input yang dimasukkan ke dalam aplikasi dan mengecek fitur *History*. Tabel 5.6 menjelaskan secara detail langkah-langkah uji coba fitur *history* pada aplikasi BEAI. Langkah-langkah pengecekan dapat dilihat pada Gambar 5.13, Gambar 5.14, Gambar 5.15 untuk harga dari pengguna, sementara untuk harga dari server dapat dilihat dari Gambar 5.16, Gambar 5.17, Gambar 5.18.

Gambar 5.13 merupakan *screenshot* sesaat sebelum pengguna mengisi data seperti pada Gambar 5.14. Karena pengguna belum menginput nama barang dan harga barang seperti Gambar 5.14, maka menu History tidak terdapat *entry* data tersebut. Kemudian pada Gambar 5.15 adalah *screenshot* setelah pengguna memencet tombol TAMBAH, dimana *entry* data yang diinput muncul pada barisan paling atas. Gambar 5.16 merupakan *screenshot* dari aplikasi sebelum pengguna menginput nama barang seperti pada Gambar 5.17, dimana pada baris paling atas merupakan *entry* data sebelumnya. Gambar 5.18 adalah *screenshot* dari aplikasi setelah pengguna membuka menu History setelah memencet tombol HITUNG ESTIMASI, dimana *entry* data yang pengguna baru input muncul pada baris paling atas.

Gambar 5.13 History Sebelum Input Harga dari Pengguna



Gambar 5.14 Pengisian *Entry* dengan Harga dari Pengguna

Gambar 5.15 Pengecekan History Setelah Pengisian Harga dari Pengguna

Gambar 5.16 History Sebelum Input Harga dari Server

Gambar 5.17 Pengisian *Entry* dengan Harga dari Server

Gambar 5.18 Pengecekan History Setelah Pengisian Harga dari Server

Tabel 5.6 Tabel Uji Coba Pengecekan Fitur History

ID	**TC-005**
Kasus Penggunaan	Melakukan Pengecekan Fitur History dengan membandingkan input pengguna dengan *entry* History yang muncul
Tujuan Pengujian	Menguji apakah penginputan oleh pengguna menambah *entry* dari History
Skenario 1	*Pengguna menginput nama barang, harga barang, mata uang*
Kondisi Awal	Kondisi History yang belum memiliki *entry* yang akan diinput pengguna.
Langkah Pengujian	1. Pengguna membuka History sebelum menginput barang dan harga. 2. Pengguna menginput nama barang 3. Pengguna mengisi harga 4. Pengguna mengisi mata uang. 5. Pengguna memencet button Tambah 6. Pengguna kembali ke menu utama membuka History
Hasil yang Diharapkan	Input dari pengguna muncul pada fitur History
Kondisi Akhir	Aplikasi BEAI berhasil menampilkan input pengguna sebelumnya pada fitur History.
Skenario 2	*Pengguna menginput nama barang dan memilih lokasi pembelian di Asia Tenggara*
Kondisi Awal	Kondisi History yang belum memiliki *entry* yang akan diinput pengguna.
Langkah Pengujian	1. Pengguna membuka History sebelum menginput barang dan harga. 2. Pengguna menginput nama barang. 3. Pengguna memilih lokasi pembelian di Asia Tenggara

	4. Pengguna memilih button Ya, Tolong Hitung Estimasi 5. Pengguna kembali ke menu utama dan membuka History
Hasil yang diharapkan	Input dari pengguna dan harga dari server muncul pada fitur History.
Kondisi Akhir	Aplikasi BEAI berhasil menampilkan input pengguna dan harga dari server sebelumnya pada fitur History.

5.2.1.6 Uji Coba Pengecekan Pembaruan *Database* pada Server

```
mysql> select * from amazon_DB where product_name like "%girl, wash your face%";

Empty set (0.03 sec)
```

Gambar 5.19 Pengecekan Awal Pada Basis Data Server

```
mysql> select * from amazon_DB where product_name like "%girl, wash your face%";

+-------------------------------------------------------------------------------
-----------------------------+--------------------+-----------------------------
-----------------------------------------------------------------------------+
| product_name
                              | product_sale_price | page_url
                                                                              |
+-------------------------------------------------------------------------------
-----------------------------+--------------------+-----------------------------
-----------------------------------------------------------------------------+
| Girl, Wash Your Face: Stop Believing the Lies About Who You Are so You Can Bec
ome Who You Were Meant to Be | 13.79              | https://www.amazon.com/s/ref
=nb_sb_noss_2?url=search-alias%3Daps&field-keywords=girl,+wash+your+face |
+-------------------------------------------------------------------------------
-----------------------------+--------------------+-----------------------------
-----------------------------------------------------------------------------+
1 row in set (0.02 sec)
```

Gambar 5.20 Pengecekan Pada Basis Data Server Setelah Pencarian oleh Pengguna

Pada uji coba ini, akan dilakukan pengecekan apakah terdapat pembaruan *database* pada server apabila barang yang dicari pengguna tidak ada. Tabel 5.7 menjelaskan secara detail langkah-langkah bagaimana pengecekan pembaruan pada *database* server dilakukan. Pada Gambar 5.19 menampilkan barang tidak terdapat pada *database*. Kemudian pada Gambar 5.20, setelah pengguna mencari barang tersebut menggunakan aplikasi dan dilakukan *crawling*, maka dapat dicek apakah barang yang tersebut sudah berhasil ditambahkan ke dalam *database* pada server.

Tabel 5.7 Tabel Uji Coba Pengecekan Pembaruan *Database* pada Server

ID	**TC-006**
Kasus Penggunaan	Melakukan Pengecekan Pembaruan *Database* pada Server
Tujuan Pengujian	Menguji apakah penginputan nama barang oleh pengguna yang tidak ada pada *database* server menambah *record* pada *database* server.
Skenario 1	*Pengguna menginput nama barang yang belum ada pada* database *server.*
Kondisi Awal	Kondisi *database* server belum memiliki harga dari nama barang yang diinput pengguna.
Langkah Pengujian	1. Mengecek pada *database* server mencari barang yang akan diinput pengguna menggunakan PuTTY 2. Pengguna menginput nama barang 3. Pengguna memencet tombol Luar Asia Tenggara 4. Mengecek pada *database* server setelah hasil pencarian pada barang yang diinput pengguna keluar menggunakan *query* yang sama pada langkah satu menggunakan PuTTY.

Hasil yang Diharapkan	Kondisi *database* server memiliki *record* baru yang berisi harga dan nama barang.
Kondisi Akhir	Pembaruan *database* pada server berhasil dilakukan.

5.2.2 Pengujian Pengguna

Pengujian pengguna melibatkan sepuluh orang. Pengujian dilakukan dengan meminta pengguna untuk melakukan beberapa skenario yang harus dilakukan. Tabel 5.8 menampilkan daftar partisipan yang terlibat dalam pengujian pengguna ini. Pada sub-bab akan dijelaskan skenario pengguna, hasil pengujian serta kritik dan saran dari pengguna.

Tabel 5.8 Tabel Daftar Partisipan

No	Nama Pengguna	Usia	Profesi	Pernah ke Luar Negeri?
1	Dimas Dwiki	22	Mahasiswa	Ya
2	Adelia Wijayanti P.	19	Mahasiswi	Tidak
3	M. Habibur Rahman	22	Mahasiswa	Ya
4	Nadya Rosita Asri	20	Mahasiswi	Tidak
5	Arya Putra Kurniawan	22	Mahasiswa	Tidak
6	Christopher	20	Mahasiswa	Tidak
7	Kunti Utari	53	Pengusaha	Ya
8	Larasadi Harya Nugraha	27	Desainer Produk	Tidak
9	Bejo H.S.	50	Pemilik Kos	Ya
10	Yayuk Sukarni	46	Pengusaha	Ya

5.2.2.1 Skenario Pengujian Pengguna

Pengguna akan diberikan empat skenario. Tabel 5.9 menampilkan secara detail skenario pengujian pengguna. Kemudian Tabel 5.10 menampilkan daftar pernyataan pada formulir penilaian yang telah diisi oleh pengguna.

Tabel 5.9 Tabel Skenario Pengujian Pengguna

Skenario	Langkah-langkah
1	1. Pengguna sudah bersiap-siap pulang ke Indonesia. Pengguna mengumpulkan semua struk barang belanja yang dibeli di luar negeri untuk dibawa pulang ke Indonesia. 2. Pengguna ingin mengetahui jumlah pajak yang perlu dibayar pada saat kembali ke Indonesia.
2	1. Pengguna bersiap-siap pulang ke Indonesia. Namun, struk barang belanja hilang. 2. Pengguna mengecek estimasi harga barang sebelum menambahkan ke dalam keranjang belanja. 3. Setelah barang ditambahkan ke dalam keranjang belanja, pengguna ingin mengecek dan mengedit keranjang belanja terlebih dahulu jumlah atau barang apabila salah input. 4. Pengguna ingin mengetahui estimasi pajak yang perlu dibayarkan.
3	Pengguna ingin mengecek kurs yang digunakan dalam perhitungan pajak.
4	Pengguna ingin mengecek harga barang yang telah diinput sebelumnya.

Tabel 5.10 Tabel Pertanyaan Setelah Pengujian Pengguna

No	Pertanyaan
1	Kemudahan mengisi nama dan harga barang apabila mengetahui harga barang.
2	Kemudahan menambah barang baru.
3	Kemudahan melihat daftar barang yang sudah ditambahkan.
4	Kemudahan mendapatkan hasil perhitungan biaya pajak.
5	Kecepatan mendapatkan harga dari fitur pencarian apabila tidak mengetahui harga barang.
6	Kemudahan menambahkan hasil pencarian harga suatu barang ke dalam keranjang belanja.
7	Kemudahan mengisi dan mengganti jumlah barang dalam keranjang belanja.
8	Kemudahan mengecek barang yang berada dalam keranjang belanja
9	Kemudahan mendapatkan rentang biaya pajak
10	Kemudahan mengecek daftar kurs yang berlaku.
11	Kemudahan membaca daftar kurs yang berlaku.
12	Kemudahan mengecek historis dari detil barang-barang yang sudah dimasukkan sebelumnya.
13	Kemudahan membaca detil barang-barang pada daftar historis.
14	Fitur Duty Calculator saat mengetahui harga barang membantu saya menghitung pajak yang perlu saya bayar.
15	Fitur Duty Calculator saat tidak mengetahui harga barang membantu saya menghitung estimasi pajak yang perlu saya bayar.
16	Fitur Currency membantu saya mengetahui daftar kurs hari ini.
17	Fitur History membantu saya mengecek input saya sebelumnya.

5.2.2.2 Hasil Pengujian Pengguna

Uji coba yang dilakukan kepada pengguna memiliki dua aspek penilaian, yang pertama adalah penilaian kemudahan penggunaan fitur, sementara yang kedua adalah penilaian kegunaan fitur. Pada aspek penilaian kemudahan penggunaan fitur, ada beberapa poin penilaian yang diberikan untuk satu fitur. Untuk aspek penilaian kegunaan fitur, cukup satu poin penilaian saja. Tabel 5.11 menampilkan hasil pengujian dari setiap poin. Tabel 5.12 menampilkan hasil pengujian akhir pengguna, dimana poin yang merepresentasikan telah disatukan.

Tabel 5.11 Tabel Hasil Pengujian Pengguna

No	Pertanyaan	Penilaian					Rata-rata
		1	2	3	4	5	
Pertanyaan Kemudahan Penggunaan Fitur							
1	Kemudahan mengisi nama dan harga barang apabila mengetahui harga barang.	0	0	6	2	2	3.6
2	Kemudahan menambah barang baru.	0	0	6	3	1	3.5
3	Kemudahan melihat daftar barang yang sudah ditambahkan.	0	1	4	2	3	3.7
4	Kemudahan mendapatkan hasil perhitungan biaya pajak.	0	0	3	3	4	4.1
5	Kecepatan mendapatkan harga dari fitur pencarian apabila tidak mengetahui harga barang.	0	1	5	3	1	3.4

6	Kemudahan menambahkan hasil pencarian harga suatu barang ke dalam keranjang belanja.	0	1	4	4	1	3.5
7	Kemudahan mengisi dan mengganti jumlah barang dalam keranjang belanja.	0	3	2	2	3	3.5
8	Kemudahan mengecek barang yang berada dalam keranjang belanja	0	1	3	2	4	3.9
9	Kemudahan mendapatkan rentang biaya pajak	0	1	0	3	6	4.4
10	Kemudahan mengecek daftar kurs yang berlaku.	0	0	1	4	5	4.3
11	Kemudahan membaca daftar kurs yang berlaku.	0	1	2	1	6	4.1
12	Kemudahan mengecek historis dari detil barang-barang yang sudah dimasukkan sebelumnya.	1	0	0	5	4	4.1
13	Kemudahan membaca detil barang-barang pada daftar historis.	0	1	3	3	3	3.8
Penilaian Kegunaan Fitur							
14	Fitur Duty Calculator saat mengetahui harga barang membantu saya menghitung pajak yang perlu saya bayar.	0	1	0	5	4	4.2
15	Fitur Duty Calculator saat tidak mengetahui	0	0	1	7	2	4.1

	harga barang membantu saya menghitung estimasi pajak yang perlu saya bayar.						
16	Fitur Currency membantu saya mengetahui daftar kurs hari ini.	0	0	3	2	5	3.7
17	Fitur History membantu saya mengecek input saya sebelumnya.	0	0	2	5	3	4.1

Tabel 5.12 Tabel Hasil Pengujian Akhir Pengguna

No	Pertanyaan	Rata-rata	Total	Total (%)
Penilaian Kemudahan Penggunaan Fitur				
1	Kemudahan mengisi nama dan harga barang apabila mengetahui harga barang.	3.6	3.725	74.5%
2	Kemudahan menambah barang baru.	3.5		
3	Kemudahan melihat daftar barang yang sudah ditambahkan.	3.7		
4	Kemudahan mendapatkan hasil perhitungan biaya pajak.	4.1		
5	Kecepatan mendapatkan harga dari fitur pencarian	3.4	3.74	74.8%

	apabila tidak mengetahui harga barang.			
6	Kemudahan menambahkan hasil pencarian harga suatu barang ke dalam keranjang belanja.	3.5		
7	Kemudahan mengisi dan mengganti jumlah barang dalam keranjang belanja.	3.5		
8	Kemudahan mengecek barang yang berada dalam keranjang belanja	3.9		
9	Kemudahan mendapatkan rentang biaya pajak	4.4		
10	Kemudahan mengecek daftar kurs yang berlaku.	4.3	4.2	84%
11	Kemudahan membaca daftar kurs yang berlaku.	4.1		
12	Kemudahan mengecek historis dari detil barang-barang yang sudah dimasukkan sebelumnya.	4.1	3.95	79%
13	Kemudahan membaca detil	3.8		

	barang-barang pada daftar historis.			
Penilaian Kegunaan Fitur				
14	Fitur Duty Calculator saat mengetahui harga barang membantu saya menghitung pajak yang perlu saya bayar.	4.2	4.2	84%
15	Fitur Duty Calculator saat tidak mengetahui harga barang membantu saya menghitung estimasi pajak yang perlu saya bayar.	4.1	4.1	82%
16	Fitur Currency membantu saya mengetahui daftar kurs hari ini.	3.7	3.7	74%
17	Fitur History membantu saya mengecek input saya sebelumnya.	4.1	4.1	82%

5.2.2.3 Kritik dan Saran Pengguna

Pada formulir penilaian, pengguna juga diberikan untuk memberikan kritik dan saran yang membangun untuk aplikasi BEAI. Tabel 5.13 menampilkan kritik dan saran dari masing-masing pengguna.

Tabel 5.13 Tabel Kritik dan Saran Pengguna

No	Nama Pengguna	Kritik dan Saran
1	Dimas Dwiki	Rincian fitur history dan list barang kurang lengkap
2	Adelia Wijayanti P.	Mungkin dari segi desain dan layoutnya ditingkatkan lagi agar lebih menarik. Dan mungkin diberi keterangan web yang tertera saat harganya muncul.
3	M. Habibur Rahman	Aplikasi sudah bagus, keranjang belanja mungkin dipisah
4	Nadya Rosita Asri	Fitur duty calculator ini jika tidak diketahui jenis barangnya secara detail agak kesulitan saat mencari harga.
5	Arya Putra Kurniawan	Keranjang belanja bisa dibuat terpisah dan waktu responnya aga lama
6	Christopher	Tampilan kurang dibagusi. Judul setelah duty calculator masih bagus.
7	Kunti Utari	Kurangnya dipisah
8	Larasadi Harya Nugraha	Button untuk penambahan harusnya jangan dsbelah memasukkan jumlah barang, bisa *misleading*.
9	Bejo H.S.	Tombolnya ada yang sulit dipencet dan hurufnya kekecilan.
10	Yayuk Sukarni	Huruf dan angka beberapa terlalu kecil untuk dibaca

5.3 Evaluasi

Pada pengujian fungsionalitas, hasil secara keseluruhan dapat dilihat pada Tabel 5.14. Berdasarkan data pada tabel tersebut dapat dilihat bahwa semua skenario berhasil terlaksana. Jadi dapat disimpulkan bahwa fungsionalitas aplikasi bekerja dengan baik.

Tabel 5.14 Tabel Evaluasi Kasus Uji Fungsionalitas Aplikasi

No.	Kode Kasus Pengujian	Skenario	Terpenuhi
1	TC-001	1	√
		2	√
2	TC-002	1	√
		2	√
3	TC-003	1	√
		2	√
4	TC-004	1	√
5	TC-005	1	√
		2	√
6	TC-006	1	√

Pada pengujian hasil pengguna, dengan hasil pengujian akhir pada Tabel 5.12, nilai masing-masing poin berkisar antara 70% - 80%. Apabila dirata-ratakan berdasarkan aspek penilaian, penilaian kemudahan fitur memiliki nilai 78.075%, sementara untuk penilaian kegunaan fitur memiliki nilai 80.5%.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas kesimpulan yang dapat diambil dari hasil uji coba dan perancangan perangkat lunak sebagai jawaban dari rumusan masalah yang telah dikemukakan dan saran yang berisi pengembangan yang dapat dilakukan lebih lanjut untuk menyempurnakan perangkat lunak.

6.1 Kesimpulan

Berikut merupakan kesimpulan yang dapat diambil dari proses pengembangan dan hasil uji coba.

1. *Spider* berhasil diimplementasikan menggunakan kerangka Scrapy serta menyimpannya ke dalam basis data di dalam server. Hal ini dapat disimpulkan dari sesuainya harga yang disimpan di *database* server BEAI dengan harga pada Amazon dan Lazada seperti yang dapat dilihat pada Kasus Pengujian TC-001 dan TC-006.
2. *Spider* untuk mendapatkan kurs yang berlaku pada saat perhitungan biaya pungutan berhasil diiimplementasikan dengan melakukan *scraping* pada beacukai.go.id, dimana kurs yang berada pada *database* BEAI sama dengan kurs pada situs beacukai.go.id seperti yang dapat dilihat pada Kasus Pengujian TC-004.
3. Aplikasi BEAI berhasil diimplementasikan dengan cara memberikan opsi kepada pengguna apakah pengguna mengetahui harga barang yang akan diinput ke dalam aplikasi atau tidak. Hasil dari perhitungan pajak dari kedua opsi tersebut sesuai dengan hasil perhitungan pada aplikasi CEISA Mobile seperti yang dapat dilihat pada Kasus Pengujian TC-002 dan TC-003.

6.2 Saran

Berikut ini merupakan pengembangan lebih lanjut yang dapat dilakukan untuk menyempurnakan perangkat lunak.

1. Harga suatu barang kemungkinan dapat berubah-ubah dalam kurun waktu tertentu, sehingga diperlukan pembaruan harga secara berkala dan lebih intensif. Saat ini baru dilakukan rotasi user-agent, namun untuk mencegah *spider* tidak mendapatkan apa-apa karena terkena IP ban, disarankan melakukan rotasi IP dan rotasi user-agent.
2. User Interface dapat diperbaiki dengan menggunakan *template* sehingga lebih menarik untuk dilihat.
3. Menambah *history* untuk hasil perhitungan pajak dengan kurs yang digunakan pada saat perhitungan. Pada saat ini aplikasi baru melakukan *record* pada barang-barang yang diinput saja.

DAFTAR PUSTAKA

[1] K. K. R. Indonesia, "PERATURAN MENTERI KEUANGAN REPUBLIK INDONESIA NOMOR 203/PMK.04/2017 TENTANG KETENTUAN YANG DIBAWA OLEH PENUMPANG DAN AWAK SARANA PENGANGKUT,," 27 December 2017. [Online]. Available: http://www.sjdih.depkeu.go.id/fullText/2017/203~PMK.04~2017Per.pdf. [Accessed 8 Januari 2018].

[2] Wikipedia, "Web Scraping," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Web_scraping. [Accessed 5 January 2018].

[3] S. Developers, "Scrapy at a glance," Scrapy, 28 October 2017. [Online]. Available: https://doc.scrapy.org/en/latest/intro/overview.html. [Accessed 5 January 2018].

[4] P. S. Foundation, "General Python FAQ," 18 January 2018. [Online]. Available: https://docs.python.org/3/faq/general.html#what-is-python. [Accessed 18 January 2018].

[5] Scrapy, "Architechture overview," Scrapy, 10 January 2017. [Online]. Available: https://doc.scrapy.org/en/latest/topics/architecture.html. [Accessed 5 January 2018].

[6] A. Studio, "Transmitting Network Data Using Volley," Android Studio, [Online]. Available: https://developer.android.com/training/volley/index.html. [Accessed 9 January 2018].

[7] Wikipedia, "Representational state transfer," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer. [Accessed 10 January 2018].

[8] T. P. Group, "What is PHP?," The PHP Group, [Online]. Available: http://php.net/manual/en/intro-whatis.php. [Accessed 5 January 2018].

[9] Wikipedia, "PHP," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/PHP. [Accessed 5 January 2018].

[10] JSON, "Introducing JSON," JSON, [Online]. Available: https://www.json.org/. [Accessed 5 January 2018].

BIODATA PENULIS

Andre Exaudi Jeremy Rumapea, lahir di Medan, pada tanggal 19 Mei 1996. Penulis merupakan anak pertama dari dua bersaudara dari pasangan Jason Rumapea dan Clara Simangunsong. Penulis menempuh pendidikan dasar di SD Buddhaya II (2002-2008), pendidikan menengah pertama di SMP Kristen 5 Penabur Jakarta (2008-2011), pendidikan menengah atas di SMA Kristen 7 Penabur Jakarta (2011-2014) dan pendidikan tinggi di S1 Teknik Informatika ITS (2014-2018).

Selama kuliah di Departemen Informatika ITS, penulis mendalami bidang minat Manajemen Informasi (MI). Penulis memiliki ketertarikan dalam bidang pengembangan aplikasi mobile. Penulis dapat dihubungi melalui surel: **andre.rumapea@gmail.com**..