ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IF184802

IMPLEMENTASI HEADLESS BROWSER UNTUK LOAD TESTING BERBASIS WEB SERVICE MENGGUNAKAN INFRASTRUKTUR DOCKER

CAHYA PUTRA HIKMAWAN
NRP 05111540000119

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - IF184802

IMPLEMENTASI HEADLESS BROWSER UNTUK LOAD TESTING BERBASIS WEB SERVICE MENGGUNAKAN INFRASTRUKTUR DOCKER

CAHYA PUTRA HIKMAWAN
NRP 05111540000119

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS - IF184802

HEADLESS BROWSER IMPLEMENTATION FOR LOAD TESTING BASED ON WEB SERVICE USING DOCKER INFRASTRUCTURE

CAHYA PUTRA HIKMAWAN
NRP 05111540000119

Supervisor I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

Supervisor II
Bagus Jati Santoso, S.Kom., Ph.D

Department of INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI HEADLESS BROWSER UNTUK LOAD TESTING BERBASIS WEB SERVICE MENGGUNAKAN INFRASTRUKTUR DOCKER

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
CAHYA PUTRA HIKMAWAN
NRP: 05111540000119

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD. .....
NIP: 197708242003041001 (Pembimbing 1)

Bagus Jati Santoso, S.Kom., Ph.D .....
NIP: 198611252018031001 (Pembimbing 2)

SURABAYA
Juni 2019

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI HEADLESS BROWSER UNTUK LOAD TESTING BERBASIS WEB SERVICE MENGGUNAKAN INFRASTRUKTUR DOCKER

Nama	:	**CAHYA PUTRA HIKMAWAN**
NRP	:	**05111540000119**
Jurusan	:	**Teknik Informatika FTIf**
Pembimbing I	:	**Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.**
Pembimbing II	:	**Bagus Jati Santoso, S.Kom., Ph.D**

Abstrak

Ini abstrak

***Kata-Kunci**: headless browser, headless chrome, puppeteer, docker, docker swarm*

HEADLESS BROWSER IMPLEMENTATION FOR LOAD TESTING BASED ON WEB SERVICE USING DOCKER INFRASTRUCTURE

Name	:	**CAHYA PUTRA HIKMAWAN**
NRP	:	**05111540000119**
Major	:	**Informatics FTIf**
Supervisor I	:	**Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.**
Supervisor II	:	**Bagus Jati Santoso, S.Kom., Ph.D**

Abstract

This is abstract

***Keywords**: headless browser, headless chrome, puppeteer, docker, docker swarm*

KATA PENGANTAR

Isi kata pengantar

Surabaya, Juni 2019

Cahya Putra Hikmawan

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Saat ini pengembangan aplikasi berbasis web sangat banyak dilakukan. Dalam pengembangan web, akan diperlukan uji beban yang dilakukan pada web yang dikembangkan sebelum diluncurkan ke tahap produksi. Salah satu teknik uji beban adalah *Headless Testing*, salah satu *browser* yang menggunakan teknik ini adalah *Headless Chrome*. Teknik ini menyediakan akses kontrol seperti *browser* pada umumnya untuk mendapatkan data dokumen uji beban, hanya saja teknik ini berjalan secara *headless* atau tanpa menampilkan antarmuka pengguna. Dikarenakan berjalan secara *headless*, maka untuk melakukan pengambilan data dokumen uji beban dilakukan menggunakan baris perintah atau *CLI(Command Line Interface)*.

Pengembang aplikasi web tentu saja membutuhkan teknik untuk uji beban, namun dalam melakukan uji beban dibutuhkan *resource user* untuk mengakses web dan waktu yang cukup lama karena dilakukan secara manual. Oleh karena itu, dibutuhkan suatu sistem yang dapat melakukan automasi untuk uji beban web, membuat *load generator* yang digunakan sebagai *resource user* dan dapat mempersingkat waktu uji beban.

Pada tugas akhir ini, akan dibangun sebuah sistem uji beban menggunakan teknik secara *headless* yang akan memanfaatkan *Headless Chrome* sebagai *tester* dan membuat *load generator* yang memanfaatkan infrastruktur *Docker* sebagai *resource user*, serta automasi pengambilan data dari *tester* menggunakan pustaka *Node* yaitu *Puppeteer*[3].

1.2 Rumusan Masalah

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

1. Bagaimana cara mengimplementasikan *Headless Chrome* sebagai *tester* untuk *load generator*?
2. Bagaimana cara mengimplementasikan *Docker* bisa menjadi *load generator* untuk uji beban?
3. Bagaimana cara menghasilkan laporan uji beban dalam bentuk *dashboard*?
4. Bagaimana mengelola layanan pengujian untuk *multiuser* dalam bentuk antrian?

1.3 Batasan Masalah

Dari permasalahan yang telah dipaparkan di atas, terdapat beberapa batasan masalah pada tugas akhir ini, yaitu:

1. *Headless Browser* yang digunakan adalah *Headless Chrome*.
2. Kontainer yang digunakan adalah *Docker*.
3. Distribusi kontainer menggunakan *Docker Swarm*.
4. Aplikasi yang akan diuji berupa aplikasi web.
5. Uji coba aplikasi akan menggunakan *Node library yang menyediakan API (Application Programming Interface)* untuk mengontrol *Headless Chrome* yaitu *Puppeteer*.

1.4 Tujuan

Tujuan pembuatan tugas akhir ini antara lain:

1. Membuat sistem manajemen pengujian aplikasi secara *headless* menggunakan *Headless Chrome*.
2. Membuat sistem agar *Docker* bisa menjadi *load generator* untuk pengujian.

3. Mengimplementasikan pengujian menggunakan skenario yang sudah disiapkan.
4. Membuat sistem untuk menampilkan laporan uji beban dalam bentuk *dashboard*.
5. Mengimplementasikan sistem manajemen pengujian ini untuk aplikasi web di ITS.

1.5 Manfaat

Manfaat dari pembuatan tugas akhir ini yaitu:

1. Mempelajari penggunaan *Headless Browser* untuk pengujian suatu aplikasi yaitu *Headless Chrome*
2. Meminimalisir adanya kegagalan web saat dimuat ketika sudah diluncurkan.
3. Mengetahui performa *load* suatu aplikasi web.

1.6 Metodologi

Metodologi yang digunakan untuk pembuatan Tugas Akhir ini adalah sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri dari hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian informasi dan referensi mengenai *Headless Chrome*, *Puppeteer* dan *Docker* untuk mendukung dan memastikan setiap tahap pembuatan tugas akhir sesuai dengan standar dan konsep yang berlaku, serta dapat diimplementasikan. Sumber informasi dan referensi bisa didapatkan dari buku, jurnal dan internet.

1.6.3 Analisis dan Desain Perangkat Lunak

Pada tahap ini dilakukan analisis dan perancangan terhadap arsitektur sistem yang akan dibuat. Tahap ini merupakan tahap yang paling penting dimana segala bentuk implementasi bisa bekerja dengan baik ketika arsitektur sistem yang baik pula.

1.6.4 Implementasi Perangkat Lunak

Pada tahap ini dilakukan implementasi atau realisasi dari hasil analisis dan perancangan arsitektur yang sudah dibuat sebelumnya, sehingga menjadi sebuah infrastruktur yang sesuai dengan apa yang direncanakan.

1.6.5 Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian untuk mengukur performa web dan kegagalan saat dimuat menggunakan arsitektur sistem yang sudah dibuat menggunakan infrastruktur *Docker*. Beberapa performa yang diukur pada pengujian antara lain, *load time*, *response time*, *firstmeaningfulpain*, *css tracing*, *domcontentloadevent* dan *dominteractive* dalam satuan *ms(millisecond)* serta *error console*. Setelah dilakukan uji coba, maka dilakukan evaluasi terhadap kinerja arsitektur sistem yang telah diimplementasikan dengan harapan bisa diperbaiki ketika ada pengembangan ke depannya.

1.6.6 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan buku tugas akhir yang berisikan dokumentasi yang mencakup teori, konsep, implementasi dan hasil pengerjaan tugas akhir.

1.7 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir secara garis besar adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan dari pembuatan tugas akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan metode, algoritme, *library* dan *tools* yang digunakan dalam penyusunan tugas akhir ini. Kajian teori yang dimaksud berisi tentang penjelasan singkat mengenai *Headless Browser*, *Headless Chrome*, *Puppeteer*, *NodeJS*, *Docker*, *Docker Swarm* dan *Laravel*.
3. Bab III. Desain dan Perancangan
   Bab ini berisi mengenai analisis dan perancangan arsitektur sistem yang akan diimplementasikan pada tugas akhir ini.
4. Bab IV. Implementasi
   Bab ini berisi bahasan tentang implementasi dari arsitektur sistem yang dibuat pada bab sebelumnya. Penjelasan berupa kode program yang digunakan untuk mengimplementasikan sistem.
5. Bab V. Pengujian dan Evaluasi
   Bab ini berisi bahasan tentang tahapan uji coba terhadap performa web dan evaluasi terhadap sistem yang dibuat.
6. Bab VI. Penutup
   Bab ini merupakan bab terakhir yang memaparkan

kesimpulan dari hasil pengujian dan evaluasi yang telah dilakukan. Pada bab ini juga terdapat saran yang ditujukan bagi pembaca yang berminat untuk melakukan pengembangan lebih lanjut.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.
8. Lampiran
   Dalam lampiran terdapat kode sumber program secara keseluruhan.

BAB II

TINJAUAN PUSTAKA

2.1 *Headless Browser*

Headless Browser merupakan jenis perangkat lunak yang dapat mengakses halaman web, namun tidak menampilkan antarmuka pengguna. Seperti peramban pada umumnya, *Headless Browser* juga memiliki kemampuan yang sama, hanya saja menyisakan mesin dan lingkungan *javascript*. Oleh karena itu *Headless Browser* hanya bisa diakses menggunakan baris perintah atau *CLI(Command Line Interface)*[4]. Beberapa *Headless Browser* yaitu *Headless Chrome*, *Selenium WebDriver* dan *Firefox Headless Mode*. Salah satu *Headless Browser* yang akan digunakan pada Tugas Akhir ini adalah *Headless Chrome*, karena *Headless Chrome* memiliki fitur khusus yang bisa ditemukan pada bagian *performance*. *Headless Chrome* juga memiliki fitur umum seperti *Headless Browser* yang lain. Beberapa fitur umumnya yaitu[5]:

1. Memudahkan untuk melakukan pengujian secara otomatis(automasi).
2. Bisa dijalankan di server, mode *headless* tidak membutuhkan antarmuka pengguna.
3. Membuat dokumen atau file seperti PDF dan Screenshot.
4. *Debugging*.

Dalam tugas akhir ini, *Headless Browser* akan digunakan sebagai browser untuk menguji web yang akan diuji performanya, sedangkan jenis yang digunakan adalah *Headless Chrome*. *Headless Chrome* saat ini dikembangkan oleh *Google Developer* dan memiliki lisensi dari *Apache 2.0 License*.

2.2 *Node.js*

Node.js atau *node* adalah sebuah platform dengan lingkungan *JavaScript* sisi server. *Node.js* berbasis pada *Chrome's JavaScript Runtime* yang menggunakan teknologi V8 dan berfokus pada performa maupun konsumsi memori rendah. Tapi V8 juga mendukung proses server yang berjalan lama. Tidak seperti kebanyakan platform modern yang lain dengan mengandalkan *multithreading*. *Node.js* menggunakan penjadwalan I/O secara asinkron. Proses pada *Node.js* dibayangkan sebagai proses *single-threaded daemon*. Hal ini berbeda dengan kebanyakan sistem penjadwalan dalam bahasa pemrograman lain yang berbentuk *library*. *Node.js* seringkali digunakan pengembang sebagai web server atau layanan *API*. Selain itu *Node.js* juga mendukung *event callback* untuk setiap penggunaan fungsi, memungkinkan ketika fungsi tersebut dipanggil akan terjadi *sleep* ketika tidak ada hasil apapun.[6][7].

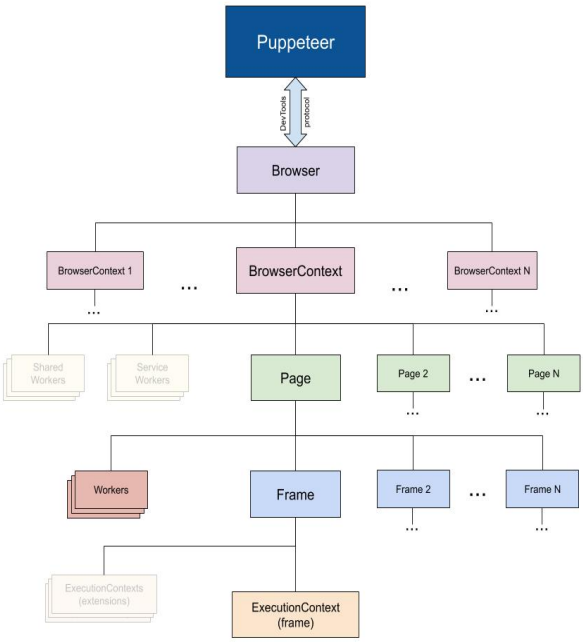
Salah satu pustaka *Node.js* yang menyediakan layanan *API* adalah *Puppeteer*. Pada tugas akhir ini, *Node.js* akan digunakan sebagai bahasa pemrograman untuk mengimplementasikan *Puppeteer*.

2.2.1 *Puppeteer*

Puppeteer adalah sebuah pustaka dari *Node* yang memiliki kemampuan yang mumpuni untuk memberikan layanan *API* yang berfungsi untuk mengontrol layanan dari *Chrome* atau *Chromium*. Selain itu, kemampuan kontrol *Puppeteer* sangat memungkinkan untuk akses pada protokol *Devtools Protocol* yang saat ini dikembangkan oleh tim *Google Developer*, dimana protokol tersebut memiliki kemampuan yang cukup berguna yaitu sebagai *tools instrument*, *inspect*, *debug*, dan *profile chrome*.[3]

Dibandingkan dengan *PhantomJS* yang sudah tidak

dikembangkan lagi, *Puppeteer* masih dikembangkan secara berkala. Begitupun fitur-fitur yang disediakan *Puppeteer* sangat mumpuni untuk melakukan beberapa pengujian terhadap web yaitu melakukan pengambilan gambar ataupun pdf, automasi, pengujian antarmuka pengguna, *keyboard input*, *timeline trace* untuk mendapatkan performa dan ekstensi pada *Chrome*. Untuk lebih jelasnya struktur diagram *Puppeteer* ditunjukkan pada Gambar 2.1.



Gambar 2.1: Struktur diagram *Puppeteer*

Pada tugas akhir ini, *Puppeteer* akan digunakan sebagai alat

untuk mengontrol *Headless Browser* yang akan diimplementasikan pada sistem perangkat lunak yang akan dibangun, karena lebih mumpuni dan memiliki beragam fitur *API* untuk mengakses *Headless Chrome* dibandingkan dengan pustaka *Node* yang lain.

2.3 *Docker*

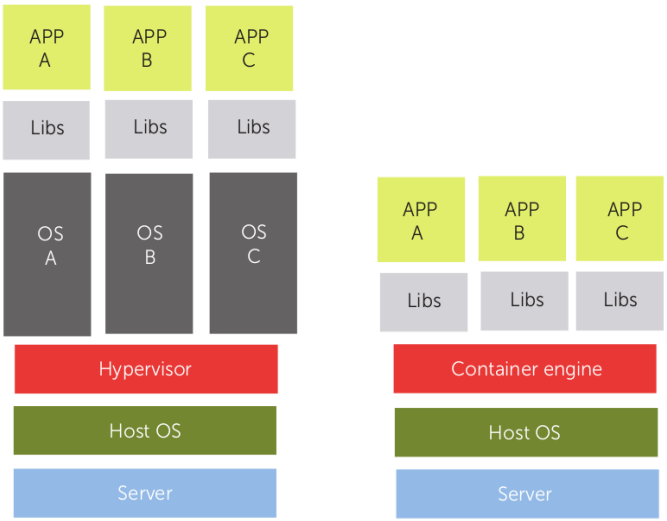
Docker adalah sebuah platform terbuka yang berfungsi sebagai wadah untuk membangun, membungkus, dan menjalankan aplikasi supaya dapat berfungsi sebagaimana mestinya. *Docker* memungkinkan untuk memisahkankan aplikasi dari infrastruktur supaya *software* dapat di jalankan dengan lebih cepat. *Docker* pada dasarnya memperluas *LXC(Linux Containers)* menggunakan kernel dan *API* pada level aplikasi yang akan dijalankan secara bersamaan pada isolasi *CPU*, memori, I/O, jaringan dan yang lainnya. *Docker* juga menggunakan *namespaces* untuk mengisolasi segala tampilan pada aplikasi yang mendasari lingkungan operasinya, termasuk *process tree*, jaringan, ID pengguna, dan file sistem.

Docker Container dibuat oleh sebuah *Docker Images*. *Docker Images* hanya mencakup dasar dari operasi sistem atau hanya memuat set dari *prebuilt* aplikasi yang sudah siap dijalankan. Ketika membuat *Docker Images*, bisa menjalankan perintah (yaitu apt-get install) membentuk lapisan baru diatas lapisan sebelumnya. Perintah tersebut bisa dijalankan manual satu-persatu atau secara otomatis menggunakan *Dockerfile*.

Setiap *Dockerfile* adalah kombinasi beberapa perintah yang dibuat menjadi menjadi satu atau *script* yang bisa dijalankan secara otomatis sebagai *Docker Images* utama atau untuk membuat *Docker Images* yang baru[8][1]. *Docker* juga menyediakan layanan untuk mengunduh dan mengupload *Docker images* melalui `https://hub.docker.com/`. Untuk

melihat perbedaan antara kontainer dan *VM(Virtual Machine)* dapat dilihat pada Gambar 2.2.

Pada tugas akhir ini, *Docker Container* akan digunakan sebagai pengguna untuk mengakses web yang akan diuji, dan bisa diumpamakan sebagai pengguna asli untuk automasi pengujiannya.



Gambar 2.2: Perbandingan kontainer dan *Virtual Machine*[1]

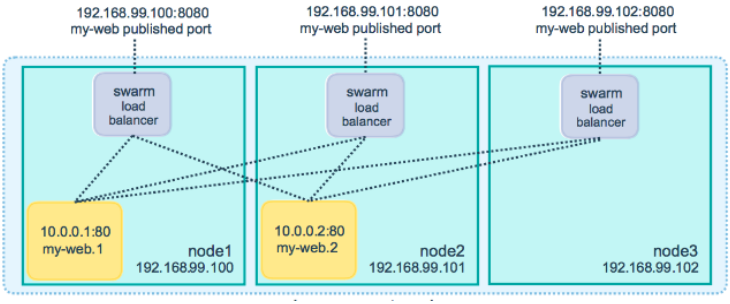
2.3.1 *Docker Swarm*

Docker Swarm - disebut juga *Swarm* adalah mode pada *Docker* yang memiliki fitur yang tertanam pada mesinnya untuk Manajemen *Cluster* atau *Orkestrasi*. Pada mode *Swarm* akan terdapat lebih dari satu *host* dimana *host* tersebut bisa berfungsi sebagai manager, worker, atau bisa juga keduanya. Konsep pada *Swarm* antara lain adalah *Nodes*, *Services*, *Tasks* dan *Load Balancing*. Mode *Swarm* juga memudahkan untuk mengatur

bagian replikasi, jaringan, penyimpanan, port dan sebagainya.

Dibandingkan dengan kontainer yang berdiri sendiri, *Swarm* lebih mudah untuk mengubah konfigurasi servis, termasuk penyimpanan dan jaringannya tanpa harus menyalakan kembali kontainer secara manual. *Docker* akan otomatis memperbarui konfigurasi dengan cara menghentikan *service task* yang memiliki konfigurasi lama, kemudian akan membuat kembali *service task* menggunakan konfigurasi yang sudah diperbarui. *Swarm* juga bisa menggunakan *Docker Compose* untuk mendefinisikan dan menjalankan kontainer, *Docker Compose* menggunakan *YAML file* sebagai konfigurasinya.[9].

Pada saat mode *Swarm*, *node manager* akan mengimplementasikan *Raft Consensus Algorithm* untuk memanajemen *cluster*. *Consensus* memungkinan manajer untuk mengatur dan menjadwal *tasks* pada setiap *cluster* dan memastikan status tetap konsisten, dimana ketika ada salah satu *nodes* yang gagal dalam mejalankan servis, manajer bisa mengembalikan servis menjadi stabil kembali[10]. Untuk melihat rute diagram ketika mode *Swarm* dapat dilihat pada Gambar 2.3.



Gambar 2.3: Rute diagram ketika mode *Swarm*[2]

Pada tugas akhir ini, *Docker Swarm* akan digunakan sebagai manager atau okestrasi yang mengatur segala servis maupun

aktivitas *Docker Container* dan sebagai *Load Balancer* untuk pembagian beban kontainer pada setiap *nodes* yang merupakan instansi dari *Docker Swarm* tersebut.

2.4 *Laravel*

Laravel adalah salah satu kerangka kerja yang berbahasa *PHP* dan dibuat untuk memudahkan pengembang untuk mengembangan dan mendesain sebuah web yang menekankan kesederhanaan dan fleksibitas. Kerangka kerja ini mendukung metode *MVC(Model-View-Controller)*. dimana *MVC* digunakan untuk mengembangkan sebuah aplikasi yang memisahkan data(*Model*) dari tampilan(*View*) dan juga dari logika dari aplikasi tersebut(*Controller*)[11].

Model digunakan untuk memanipulasi data dari basis data, *View* berhubungan dengan antarmuka web seperti *HTML*, *CSS* dan *JS* sebagai data pada pengguna. *Controller* berhubungan dengan segala urusan logika pada servis web tersebut atau juga bisa disebut otaknya. *Controller* juga berfungsi sebagai jembatan antara *View* dan *Model*[11].

Pada tugas akhir ini, kerangka kerja *Laravel* akan digunakan untuk mengimplementasikan aplikasi web yang dibangun pada tugas akhir ini, dimana kerangka kerja ini sangat banyak digunakan oleh pengembang, memiliki dokumentasi resmi yang sangat baik, serta forum yang cukup baik. *Laravel* yang akan digunakan adalah versi 5.8.

2.5 *Python*

Python adalah bahasa pemrograman tingkat tinggi yang didukung oleh struktur data *built-in* semantik dinamis, selain itu Python mendukung pemrograman *procedural*, *object-oriented* dan *functional*. *Python* merupakan bahasa pemrograman

interpreted, oleh sebab itu, *Python* tidak memakan biaya untuk kompilasi, sehingga proses pengembangan, pengujian dan *debug* menjadi lebih cepat.

Kelebihan bahasa pemrograman ini adalah memiliki modul dan *package*, serta memiliki banyak standar pustaka yang didistribusikan secara bebas dan gratis. Selain itu *Python* mudah dibaca karena memiliki sintaksis yang sederhana, sehingga dapat mengurangi biaya *maintenance*. *Debug* pada *Python* juga mudah karena tidak akan terjadi *segmentation fault*, namun akan memberi umpan balik berupa *exception* apabila terdapat kesalahan atau *error*[12].

Pada tugas akhir ini, Bahasa pemrograman *Python* akan digunakan untuk mengimplementasikan algoritme *task queue* pada sistem yang akan dibangun. *Python* yang digunakan adalah versi 3.6.8.

BAB III

DESAIN DAN PERANCANGAN

Pada bab ini dibahas mengenai analisis dan perancangan sistem.

3.1 Deskripsi Umum Sistem

Sistem yang akan dibangun pada tugas akhir ini adalah sebuah sistem yang dapat melakukan automasi uji beban terhadap suatu web. Uji beban pada sistem akan berjalan secara headless menggunakan sebuah tester yaitu Headless Chrome. Headless Chrome akan mendapatkan data uji beban ketika mengakses web yang diuji, sedangkan yang digunakan untuk mengambil data uji beban adalah sebuah pustaka Node yaitu Puppeteer. Sistem juga akan menggunakan Docker sebagai infrastruktur, sehingga Docker dapat digunakan sebagai load generator untuk melakukan uji beban yang bisa disebut kontainer.

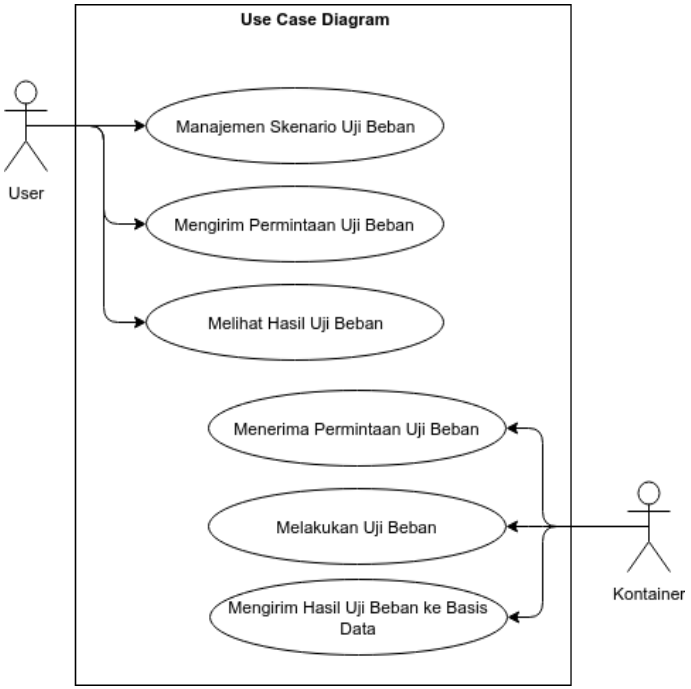
Kontainer yang akan dipasang pada sistem membutuhkan sebuah alat orkestrasi untuk memanajemen kontainer secara otomatis, alat orkestrasi yang digunakan adalah Docker Swarm. Docker Swarm akan melibatkan 3 node host yang akan dibagi menjadi 1 node host sebagai swarm manager dan 2 node host sebagai worker. Docker Swarm akan bertanggung jawab dalam mendistribusikan kontainer ke masing-masing swarm node yang tergabung pada lingkungan swawrm atau bisa disebut sebagai load balancer.

Proses uji beban akan diproses user melakukan request skenario uji beban pada web service yang disediakan sistem. Kemudian controller pada web service akan mengirimkan skenario uji pada kontainer terpilih untuk melakukan pengujian. Setiap kontainer akan terinstall Headless Chrome dan Puppeteer. Headless Chrome akan mendapatkan data uji beban dan automasi pengambilan data uji beban dilakukan oleh Puppeteer. Puppeteer

akan melakukan ekstraksi data uji beban menjadi satuan millisecond(ms).

Sistem akan menyediakan basis data untuk menyimpan data yang diperlukan sistem. Basis data akan dipasang diluar lingkungan swarm dan lingkungan web service. Sistem juga menyediakan antarmuka pengguna berupa web yang akan digunakan untuk melihat laporan hasil uji beban. Sedangkan untuk mengatasi multiuser sistem akan menggunakan Task Scheduler atau Queue(antrian).

3.2 Kasus Penggunaan



Gambar 3.1: Diagram kasus penggunaan

Terdapat dua aktor dalam sistem yang akan dibuat yaitu User dan Kontainer. User merupakan aktor yang bisa melakukan manajemen pada skenario yang ingin diuji dan melihat hasilnya, sedangkan Kontainer merupakan aktor yang akan digunakan sebagai load generator untu melakukan uji beban. Diagram kasus penggunaan digambarkan pada Gambar 3.1 dan dijelaskan masing-masing pada Table 3.1.

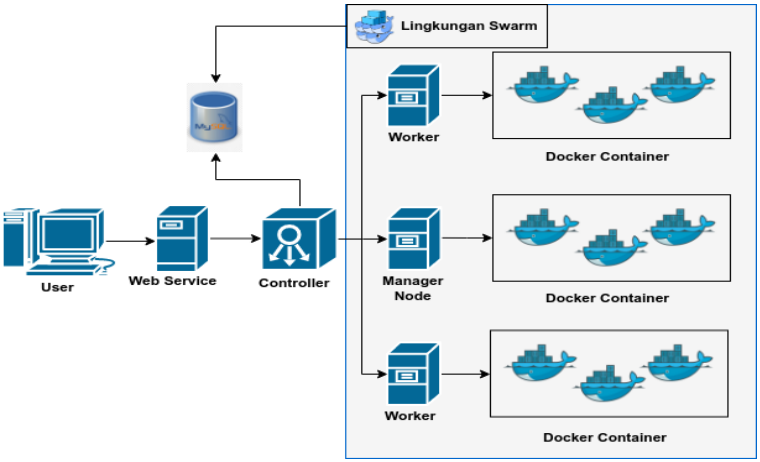
Tabel 3.1: Daftar kode kasus penggunaan

Kode Kasus Penggunaan	**Nama Kasus Penggunaan**	**Keterangan**
UC-0001	Manajemen Skenario Uji Beban	User dapat menambah, melihat dan menghapus skenario uji beban
UC-0002	Mengirim Permintaan Uji Beban	User dapat mengirimkan permintaan uji beban ke sistem melalui web service yang disediakan
UC-0003	Melihat Hasil Uji Beban	Ketika proses uji beban selesai, user dapat melihat hasilnya di antarmuka pengguna web service yang disediakan
UC-0004	Menerima Permintaan Uji Beban	Proses dimana kontainer akan menerima permintaan uji beban dari User

Tabel 3.1: Daftar kode kasus penggunaan

Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
UC-0005	Melakukan Uji Beban	Proses dimana kontainer akan melakukan uji beban sesuai skenario yang dikirim
UC-0006	Mengirim Hasil Uji Beban ke Basis Data	Ketika kontainer telah selesai melakukan pengujian, data yang didapatkan akan dikirim ke basis data MySQL

3.3 Arsitektur Sistem



Gambar 3.2: Desain arsitektur sistem

Pada sub-bab ini, akan dibahas mengenai tahap analisis arsitektur, analisis teknologi dan desain sistem yang akan dibangun. Arsitektur sistem secara umum ditunjukkan pada Gambar 3.2.

3.3.1 Desain Umum Sistem

Berdasarkan yang dijelaskan pada deskripsi umum sistem, dapat diperoleh beberapa kebutuhan sistem antara lain:

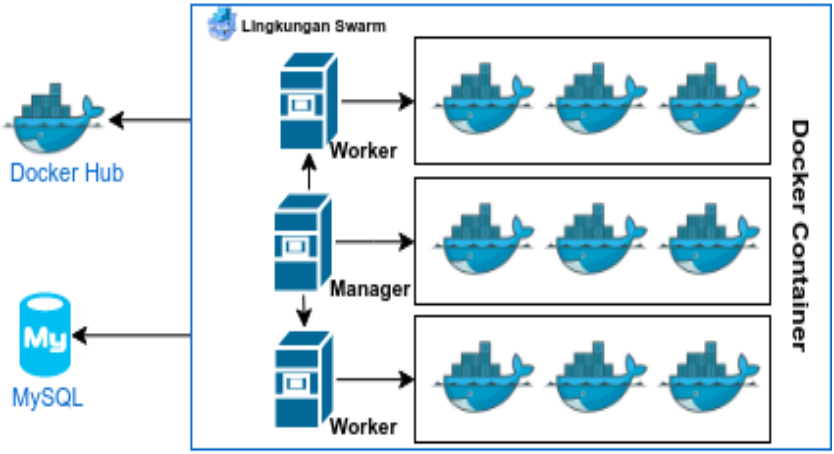
1. Load generator untuk melakukan uji beban.
2. Tester yang bisa mengambil data uji beban.
3. Web service sebagai antarmuka pengguna.
4. Basis data untuk menyimpan data sistem.
5. Task Queue untuk menangani kasus request lebih dari satu user.

Untuk memenuhi kebutuhan sistem yang dijelaskan sebelumnya, penulis membagi menjadi beberapa komponen sistem yang akan digunakan pada tugas akhir ini.

1. Load generator
   Berfungsi sebagai pengganti user yang akan melakukan akses web melalui browser.
2. Pengambil data uji beban
   Berfungsi untuk mengambil data uji beban ketika load generator mengakses web dari browser.
3. Service Controller
   Berfungsi sebagai pengatur sistem uji beban yang terdiri :
   - Web Service
     Berfungsi sebagai tampilan antarmuka pengguna untuk menggunakan sistem.
   - Basis Data
     Berfungsi untuk menyimpan data yang digunakan untuk menyimpan segala data yang dibutuhkan oleh sistem.
   - Task Queue

Berfungsi untuk membuat task scheduler atau antrian untuk menangani kasus request lebih dari satu user.

3.3.2 Perancangan Load Generator



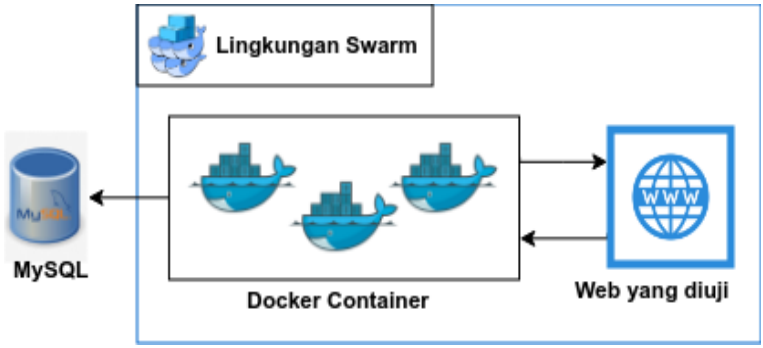
Gambar 3.3: Desain perancangan load generator

Komponen load generator akan difungsikan sebagai pengganti pengguna yang mengakses ke suatu web melalui browser. Komponen yang akan digunakan sebagai load generator adalah Docker atau bisa disebut kontainer. Ketika melakukan pemasangan kontainer sebuah Docker Image, untuk memenuhi hal tersebut, pada tugas akhir ini penulis akan membuat sebuah Docker Image yang akan diunggah ke Docker Hub. Sehingga ketika akan memasang kontainer pada node host yang baru, node host tersebut hanya perlu mengunduh Docker Image yang telah diunggah sebelumnya. Seluruh kontainer akan dibangun di dalam lingkungan swarm untuk memudahkan dalam mengatur atau memanajemen kontainer ke semua node host yang tergabung di dalam lingkungan swarm, sedangkan untuk

memudahkan akses ke setiap kontainer, maka data dari kontainer akan disimpan di dalam basis data MySQL. Load generator akan terdiri dari 3 node host, 1 sebagai manager node dan 2 lainnya sebagai worker, sedangkan basis data akan berada di luar lingkungan swarm. Desain perancangan komponen ini digambarkan pada Gambar 3.3.

3.3.3 Perancangan Pengambil Data Uji Beban

Komponen ini akan membutuhkan suatu alat yang bisa mendapatkan data uji beban terlebih dahulu. Pada tugas akhir ini, akan menggunakan Headless Chrome untuk mendapatkan data uji beban ketika mengakses web. Setelah mendapatkan data uji beban, diperlukan juga suatu alat yang bisa digunakan untuk mengambil data uji beban pada Headless Chrome, alat tersebut adalah Puppeteer. Puppeteer akan melakukan pengambilan secara otomatis ketika ada perintah yang masuk dan menyimpan data uji beban pada basis data MySQL. Alat-alat yang digunakan pada komponen ini akan dipasang pada masing-masing kontainer di setiap node host. Desain perancangan komponen ini digambarkan pada Gambar 3.4.



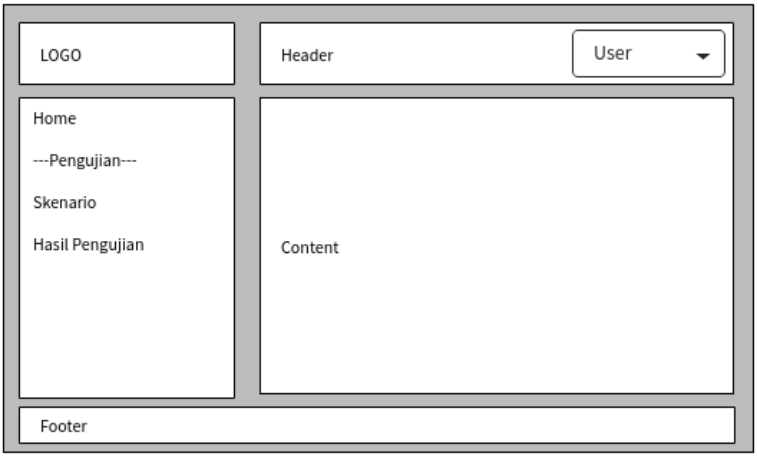
Gambar 3.4: Desain pengambil data uji beban

3.3.4 Perancangan Service Controller

Komponen ini akan digunakan untuk mengatur segala proses uji beban pada sistem. Pada komponen ini akan terdapat 3 buah sub-komponen yaitu web service, basis data dan task queue atau antrian.

3.3.4.1 Desain Web Service

Web service akan berfungsi sebagai antarmuka pengguna dan sebagai penghubung antara user dengan kontainer. Antarmuka pengguna berfungsi memudahkan user untuk membuat skenario yang akan dikirimkan ke load generator dan kemudian load generator akan melakukan uji beban sesuai dengan skenario yang dikirim user melalui web service. Sedangkan untuk mengatur segala aktivitas user dibutuhkan sebuah controller dan rute yang akan dipasang pada web service. Desain antarmuka pengguna ditunjukkan pada Gambar 3.5.



Gambar 3.5: Desain antarmuka pengguna

Selain itu, akan dirancang juga fitur-fitur pada web service yang akan digunakan user antara lain:

1. Menambah dan menghapus skenario.
2. Menentukan jumlah worker pengujian.
3. Melihat performa hasil pengujian.
4. Melihat tangkapan layar tampilan web yang diuji.
5. Melihat console error.
6. Melihat status antrian proses uji.

3.3.4.2 Desain Basis Data

Komponen basis data diperlukan untuk menyimpan data-data yang berkaitan dengan sistem. data yang disimpan adalah data node host swarm, data kontainer, data pengguna, data skenario pengujian, data antrian request, data hasil pengujian, data error console, data rata-rata hasil pengujian. Dari data-data tersebut maka dibutuhkan suatu tabel diantaranya yaitu:

- Tabel swarms
  Menyimpan data nohe host yang tergabung di dalam lingkungan swarm.
- Tabel containers
  Menyimpan data Docker Container yang telah dipersiapkan.
- Tabel users
  Menyimpan data pengguna.
- Tabel scenarios
  Menyimpan data skenario pengujian.
- Tabel queues
  Menyimpan data antrian request pengujian dari user.
- Tabel results
  Menyimpan data hasil pengujian yang dilakukan setiap kontainer.
- Tabel errors
  Menyimpan data console error yang ada di browser.

- Tabel summary results
  Menyimpan data rata-rata hasil pengujian setiap skenario.

3.3.4.3 Desain Penggunaan Task Queue

Pada service controller akan ada banyak request dari user, setiap request tentu saja akan terdapat proses yang akan berjalan dalam jangka waktu yang cukup lama. Jika proses tersebut berada di dalam fungsi yang dipanggil melalui protokol HTTP, maka akan memberikan umpan balik setelah semua proses yang ada dibaliknya selesai. Hal ini akan membuat user yang melakukan request perlu menunggu dan tidak efisien. Untuk mengatasi hal ini, akan dirancang sebuah komponen antrian atau bisa disebut task queue. Task queue akan membuat antrian untuk setiap request dibelakang layar. Antrian request tersebut akan disimpan pada basis data MySQL.

BAB IV

IMPLEMENTASI

Bab ini membahas mengenai implementasi dari sistem yang sudah di desain dan dirancang pada bab sebelumnya. Pembahasan secara rinci akan dijelaskan pada setiap komponen yang ada yaitu load generator, pengambil data uji beban dan service controller yang meliputi web service, basis data dan task queue.

4.1 Lingkungan Implementasi

Dalam mengimplementasikan sistem pada tugas akhir ini, digunakan beberapa perangkat pendukung sebagai berikut.

4.1.1 Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem adalah sebagai berikut:

1. Web service dan task queue, processor AMD FX-7600P Radeon R7, 12 Compute Cores 4C+8G dan RAM 8GB.
2. Node swarm dengan IP 167.71.194.235, processor Intel(R) Xeon(R) CPU E5-2650 v4@2.20GHz dan RAM 4GB.
3. Node swarm dengan IP 165.22.55.82, processor Intel(R) Xeon(R) CPU E5-2650 v4@2.20GHz dan RAM 4GB.
4. Node swarm dengan IP 167.71.194.233, processor Intel(R) Xeon(R) CPU E5-2650 v4@2.20GHz dan RAM 4GB.
5. Basis data MySQL dengan IP 178.128.123.143, processor Intel(R) Xeon(R) Gold 6140 CPU@2.30GHz dan RAM 1GB.

4.1.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem adalah sebagai berikut:

1. Sistem Operasi Ubuntu 18.04 LTS 64 Bit
2. Docker versi 18.09.6

3. Headless Chrome
4. Puppeteer versi 0.12.0
5. NPM versi 6.4.1
6. Node.js versi 8.15.1
7. Python versi 3.6.8
8. MySQL Ver 14.14 Distrib 5.7.26
9. Shell Script
10. PHP dan Laravel 5.8

4.2 Implementasi Load Generator

Berdasarkan perancangan dan desain, load generator merupakan aktor yang akan berfungsi menggantikan pengguna ketika mengakses ke web. Load generator yang akan digunakan adalah Docker, namun diperlukan beberapa tahap untuk bisa menggunakan Docker atau kontainer sebagai load generator, yaitu tahap pemasangan dan konfigurasi. Tahap pemasangan docker dapat dilihat di Kode Sumber A.1, sedangkan untuk konfigurasi akan dibagi menjadi beberapa tahap yaitu:

1. Pembuatan Docker Image
2. Pembuatan lingkungan kontainer
3. Pemasangan Headless Chrome dan Puppeteer

4.2.1 Implementasi Pembuatan Docker Image

Docker Image digunakan untuk menjalankan kontainer, pada tugas akhir ini Docker Image akan dibuat terlebih dahulu agar bisa digunakan untuk menjalankan Puppeteer dan Headless Chrome. Namun untuk membuat Docker Image diperlukan beberapa tahapan yaitu konfigurasi Dockerfile, pemasangan Docker Compose dapat dilihat pada Kode Sumber A.3 dan konfigurasi docker-compsoe.yml dan unggah Docker Image ke Docker Hub.

Konfigurasi Dockerfile dapat dilihat pada Kode Sumber IV.1, sedangkan konfigurasi docker-compose.yml dapat dilihat pada Kode Sumber IV.2.

```
1  FROM node:8
2  RUN apt-get update
3  # for https
4  RUN apt-get install -yyq ca-certificates
5  # install libraries
6  RUN apt-get install -yyq libappindicator1
       libasound2 libatk1.0-0 libc6 libcairo2
       libcups2 libdbus-1-3 libexpat1 libfontconfig
       1 libgcc1 libgconf-2-4 libgdk-pixbuf2.0-0
       libglib2.0-0 libgtk-3-0 libnspr4 libnss3
       libpango-1.0-0 libpangocairo-1.0-0 libstdc++
       6 libx11-6 libx11-xcb1 libxcb1 libxcomposite
       1 libxcursor1 libxdamage1 libxext6 libxfixes
       3 libxi6 libxrandr2 libxrender1 libxss1
       libxtst6
7  # tools
8  RUN apt-get install -yyq gconf-service lsb-
       release wget xdg-utils
9  RUN apt-get install -yyq fonts-liberation
10 COPY code /app/code
11 COPY output /app/output
12 WORKDIR /app/code
13 RUN yarn install
```

Kode Sumber IV.1: Konfigurasi Dockerfile

```
1  version: '3'
2  services:
3    puppeteer:
4      build: .
5      shm_size: '1gb'
```

```
6       entrypoint: ["sh", "-c", "sleep infinity"]
```

Kode Sumber IV.2: Konfigurasi docker-compose.yml

Kemudian jalankan perintah Docker Compose berikut untuk pembuatan Docker Image.

```
$ docker-compose up
```

Kode Sumber IV.3: Perintah untuk menjalankan Docker Compose

Setelah Docker Image terbuat, ubah nama Docker Image tersebut dan melakukan commit agar bisa di push. Perintah pada kode sumber IV.4 yang digunakan oleh penulis ketika mengunggah Docker Image ke Docker Hub.

```
$ docker tag puppeteer_puppeteer:latest
    cphikmawan/ta2019:newpupp

$ docker push cphikmawan/ta2019:newpupp
```

Kode Sumber IV.4: Perintah untuk mengunggah Docker Image

4.2.2 Implementasi Pembuatan Lingkungan Kontainer

Kontainer akan dipasangkan pada suatu lingkungan yang bisa mengatur segala aktivitas kontainer, lingkungan yang akan dibangun pada sistem menggunakan alat orkestrasi yaitu Docker Swarm. Untuk mengimplementasikan Docker Swarm pada sistem ini, dibutuhkan satu node host sebagai manager node dan dua lainnya sebagai worker. Tahap pertama yang dilakukan yaitu menginisiasi salah satu node host yang akan digunakan sebagai manager node. Perintah inisiasi manager node terdapat pada Kode Sumber IV.5.

```
$ docker swarm init --advertise-addr [IP NODE]
```

Kode Sumber IV.5: Perintah untuk inisiasi manager node

Setelah perintah pada kode sumber IV.5 dijalankan, maka manager node akan menghasilkan sebuah token yang digunakan oleh node host yang lain untuk bergabung sebagai worker. Perintah yang harus dijalankan pada setiap node host yang lain terdapat pada Kode Sumber IV.6.

```
$ docker swarm join --token [token] [IP MANAGER
    ]:2377
```

Kode Sumber IV.6: Perintah untuk bergabung ke Swarm

Tahap terakhir yang dilakukan yaitu memastikan semua node host sudah tergabung dengan manager node.

```
$ docker node ls
```

Kode Sumber IV.7: Perintah untuk melihat daftar Swarm Node

4.2.3 Implementasi Pemasangan Headless Chrome dan Puppeteer

Headless Chrome dan Puppeteer akan dipasang pada masing-masing kontainer menggunakan Docker Image yang telah dibuat sebelumnya, untuk pemasangannya akan dilakukan dilingkungan Docker Swarm dan dilakukan pada manager node. Namun untuk implementasinya dibutuhkan beberapa persiapan dan konfigurasi yang harus dilakukan terlebih dahulu yaitu konfigurasi unduh Docker Image, konfigurasi membuat Docker Network, konfigurasi puppeteer.yml, konfigurasi deployment.

Untuk mengunduh Docker Image dilakukan pada setiap node host menggunakan perintah pada Kode Sumber IV.8 dan membuat Docker Network pada Kode Sumber IV.9. Sedangkan konfigurasi puppeteer.yml dapat dilihat di Kode Sumber IV.10

```
$ docker image pull cphikmawan/ta2019:puppeteer
```

Kode Sumber IV.8: Perintah untuk mengunduh Docker Image

```
$ docker network create \
  --driver overlay \
  --subnet 10.0.0.0/18 \
  --attachable \
  [nama_network]
```

Kode Sumber IV.9: Perintah untuk membuat Docker Network

```
1  version: '3'
2  # konfigurasi service
3  services:
4    # nama service yang akan dibuat
5    puppeteer:
6      # docker image yang digunakan
7      image: cphikmawan/ta2019:newpupp
8      # sinkronisasi penyimpanan antara kontainer
           dengan host
9      volumes:
10       - ./output:/app/output
11       - ./code:/app/code
12     # direktori kerja didalam kontainer
13     working_dir: /app/code
14     # konfigurasi untuk jumlah kontainer dan
           handling
15     deploy:
16       replicas: 1000
17       restart_policy:
18         condition: on-failure
19     # entrypoint awal tidak akan melakukan
           apapun
20     entrypoint: ["sh", "-c", "sleep infinity"]
21 # konfigurasi jaringan
22 networks:
```

```
23      default:
24        # jaringan default akan diubah ke jaringan
              eksternal
25        external:
26          # akan terkonek ke jaringan "swarm-
                network"
27          name: swarm-network
```

Kode Sumber IV.10: Konfigurasi puppeteer.yml

Tahap selanjutnya adalah konfigurasi deployment dengan menjalankan perintah yang ditunjukkan pada Kode Sumber IV.11 di terminal manager node

```
$ docker stack deploy --compose-file=puppeteer.
    yml [nama_stack]
```

Kode Sumber IV.11: Perintah untuk pemasangan kontainer

4.3 Implementasi Pengambil Data Uji Beban

Pengambil Data Uji Beban akan dilakukan menggunakan sebuah pustaka Node yaitu Puppeteer. Pada saat dilakukan uji beban, Puppeteer akan mengambil data uji beban dari Headless Chrome menggunakan API dari Puppeteer secara otomatis. Beberapa data uji beban yang akan diambil yaitu:

1. Response End
   Atribut ini menunjukkan waktu setelah user menerima byte terakhir dari dokumen sebelum koneksi transportasi ditutup.
2. DOM Content Loaded
   Atribut ini menunjukkan waktu setelah dokumen sudah diterima oleh user.
3. Load Event End
   Atribut ini mengembalikan waktu ketika memuat dokumen selesai.

4. CSS Tracing End
   Atribut ini menunjukkan waktu dari ekstraksi akhir file CSS dimuat.
5. First Meaningfulpain
   Atribut ini adalah atribut khusus yang ada pada Chrome yang menunjukkan bahwa segala konten halaman yang dimuat sudah ditampilkan di layar.

Selain data uji beban diatas, Puppeteer juga digunakan untuk mengambil sebuah tangkapan layar sesuai skenario yang dikirimkan oleh pengguna dan mengambil data saat terjadi kegagalan saat memuat assets yang tertulis pada console browser. Adapun pseudocode untuk melakukan pengambilan data uji beban dapat dilihat pada Kode Sumber IV.12.

```
1   Variable Declaration
2   Data = Read Scenario File Configuration
3
4   TESTPAGE FUNCTION:
5     CALL HELPERS FUNCTION:
6       GET Navigation Start
7     START Trace CSS Data
8     Trying to Accessing Website
9     END Trace CSS Data
10    CALL HELPERS FUNCTION:
11      GET Extracted Performance Data
12      GET Extracted CSS Tracing
13      RETURN Extracted Data
14  END TESTPAGE FUNCTION
15
16  HELPERS FUNCTION:
17    GET Navigation Start
18      RETURN Navigation Start
19    Extracted Performance Data = Performance Data
```

```
        * 1000 - Navigation Start
      RETURN Extracted Performance Data
    Extracted CSS Tracing = CSS Tracing / 1000
      RETURN Extracted CSS Tracing
  End Helpers Function

  MAIN FUNCTION:
    START Headless Browser
    GET Error Console
      RETURN Data to Database
    TRY:
      CALL TESTPAGE FUNCTION(Data):
        Get Data From TestPage -> Save Data Uji
            Beban
      GET Page Screenshoot -> Save Screenshoot
    CATCH ERROR:
      GET Error
      GET Page Screenshoot -> Save Screenshoot
  END MAIN FUNCTION
```

Kode Sumber IV.12: Pseudocode Puppeteer

4.4 Implementasi Service Controller

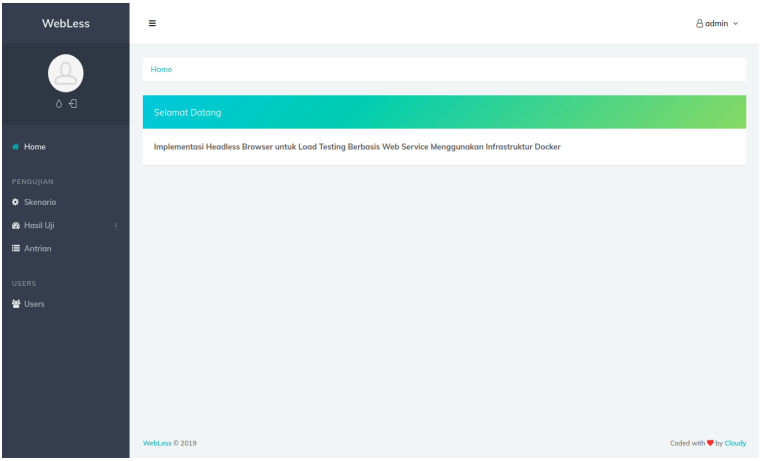
Berdasarkan desain dan perancangan, service controller terdiri dari komponen web service, basis data dan task queue. Semua komponen tersebut akan diimplementasikan pada satu komputer milik penulis yang akan digunakan untuk web service dan penggunaan task queue, serta satu buah server untuk basis data MySQL.

4.4.1 Implementasi Web Service

Pada implementasi web service dibutuhkan beberapa persiapan lingkungan yang perlu dilakukan, urutannya meliputi langkah-langkah berikut:

1. Instalasi PHP
2. Instalasi Composer
3. Instalasi Laravel versi 5.8
4. Instalasi MySQL

Web service akan menggunakan bahasa PHP dan kerangka kerja Laravel versi 5.8, sedangkan Composer berfungsi untuk memanajemen instalasi pustaka pada PHP dan untuk penyimpanan data yang digunakan pada sistem akan disimpan pada basis data MySQL. Web service berfungsi untuk memudahkan user melakukan uji beban pada suatu web. Tampilan antarmuka pengguna ditunjukkan pada Gambar 4.1.



Gambar 4.1: Tampilan web antarmuka pengguna

Web service memiliki beberapa rute HTTP yang akan digunakan oleh user ketika mengakses web sistem. Rute-rute

tersebut ditunjukkan pada Tabel 4.1.

Tabel 4.1: Rute HTTP pada web service

No	Rute	Metode	Aksi
1	/	GET	Mengakses halaman home
2	/login	GET	Mengakses halaman login
3	/login	POST	Melakukan login
4	/logout	GET	Melakukan logout
5	/skenario	GET	Melihat skenario
6	/skenario	POST	Menambahkan skenario
7	/skenario	DELETE	Menghapus skenario
8	/worker	GET	Mengakses halaman worker
9	/worker	POST	Menambahkan jumlah worker dan data antrian
10	/hasil/rata-rata	GET	Mendapatkan hasil pengujian
11	/hasil/error-console	GET	Mendapatkan error console web
12	/hasil/images	GET	Melihat tangkapan layar web
13	/antrian	GET	Melihat jumlah dan status antrian

4.4.2 Implementasi Skema Basis Data

Berdasarkan hasil perancangan basis data pada bab sebelumnya. Data yang dibutuhkan dan digunakan oleh sistem akan disimpan di dalam basis data MySQL. Data yang disimpan adalah data node host swarm, data kontainer, data pengguna, data skenario pengujian, data antrian request, data hasil pengujian, data error console, data rata-rata hasil pengujian.

4.4.2.1 Tabel Swarms

Pada tabel swarms menyimpan data-data dari node host yang tergabung dilingkungan swarm. Berikut definisi tabel swarms pada Tabel 4.2.

Tabel 4.2: Tabel swarms

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment.
2	swarm_ip	varchar(255)	Menunjukkan IP dari node host
3	swarm_username	varchar(255)	Menunjukkan username dari node host
4	swarm_password	varchar(255)	Menunjukkan password dari node host
5	is_used	smallint(6)	Menunjukkan status dari node host

4.4.2.2 Tabel Containers

Padata tabel containers menyimpan data-data dari Docker Container yang akan digunakan sebagai load generator. Berikut definisi tabel containers pada Tabel 4.3.

Tabel 4.3: Tabel containers

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment
2	task_id	varchar(100)	Menunjukkan task id dari kontainer
3	node_id	varchar(100)	Menunjukkan node id tempat kontainer dipasang
4	container_id	varchar(100)	Menunjukkan id dari kontainer
5	node_ip	varchar(100)	Menunjukkan IP tempat kontainer dipasang
6	node_host	varchar(100)	Menunjukkan hostname tempat kontainer dipasang
7	status	smallint(6)	Menunjukkan flag status dari container
8	username	varchar(100)	Menunjukkan status kontainer yang sedang digunakan user

4.4.2.3 Tabel Users

Pada tabel users menyimpan data-data pengguna web yang disediakan sistem. Berikut definisi tabel users pada Tabel 4.4. Data pengguna juga memiliki constraint dan disimpan pada Tabel role user 4.5 dan Tabel roles 4.6.

Tabel 4.4: Tabel users

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment
2	name	varchar(255)	Menunjukkan nama pengguna
3	email	varchar(255)	Menunjukkan email pengguna
4	username	varchar(255)	Menunjukkan username pengguna
5	password	varchar(255)	Menunjukkan password pengguna

Tabel 4.5: Tabel role user

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment
2	role_id	int(10)	Menunjukkan id role pengguna
3	user_id	int(10)	Menunjukkan id pengguna

Tabel 4.6: Tabel roles

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment
2	name	varchar(255)	Menunjukkan nama role
3	description	varchar(255)	Menunjukkan deskripsi hak akses dari nama role

4.4.2.4 Tabel Scenarios

Pada tabel scenarios menyimpan data-data skenario yang telah dibuat oleh pengguna. Berikut definisi tabel scenarios pada Tabel 4.7.

Tabel 4.7: Tabel scenarios

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment
2	scenario_id	varchar(255)	
3	username	varchar(255)	
4	scenario_method	varchar(255)	
5	scenario_link	varchar(255)	
6	scenario_worker	varchar(255)	
7	scenario_status	smallint(6)	

4.4.2.5 Tabel Queues

Pada tabel queues menyimpan data-data antrian request yang dilakukan oleh semua pengguna. Berikut definisi tabel queues pada Tabel 4.8.

Tabel 4.8: Tabel queues

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment
2	created_at	timestamp	Menunjukkan waktu pembuatan queue request oleh pengguna
3	username	varchar(255)	Menunjukkan keterangan pengguna pembuat request
4	worker	int(11)	Menunjukkan jumlah request yang diinginkan pengguna
5	status	smallint(6)	Menunjukkan status dari queue, nilai awal adalah 0

4.4.2.6 Tabel Results

Pada tabel results menyimpan data-data hasil uji beban yang dilakukan oleh kontainer dan puppeteer. Berikut definisi tabel results pada Tabel 4.9.

Tabel 4.9: Tabel results

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment
2	scenario_id	varchar(255)	Menunjukkan skenario id sebagai foreign key
3	link	varchar(255)	Menunjukkan link website yang diuji
4	method	varchar(100)	Menunjukkan metode uji
5	worker	varchar(255)	Menunjukkan jumlah request yang diinginkan pengguna
6	username	varchar(100)	Menunjukkan keterangan pengguna pembuat request
7	host	varchar(100)	Menunjukkan IP node host yang digunakan kontainer penguji
8	response_end	varchar(100)	Menunjukkan hasil uji response time dalam satuan detik
9	dom_content_load	varchar(100)	Menunjukkan hasil uji waktu memuat DOM web dalam satuan detik
10	load_event_end	varchar(100)	Menunjukkan hasil uji load time dalam satuan detik

Tabel 4.9: Tabel results

No	Kolom	Tipe	Keterangan
11	css_trace_end	varchar(100)	Menunjukkan hasil uji waktu memuat css time dalam satuan detik
12	first_meaningful	varchar(100)	Menunjukkan hasil uji waktu memuat konten utama dalam satuan detik
13	status	smallint(6)	Menunjukkan status apakah untuk proses rata-rata, nilai awal adalah 0

4.4.2.7 Tabel Errors

Pada tabel errors menyimpan data-data kegagalan yang terekam pada console browser ketika diakses didalam Headless Chrome. Berikut definisi tabel errors pada Tabel 4.10.

Tabel 4.10: Tabel errors

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment
2	scenario_id	varchar(255)	Menunjukkan skenario id sebagai foreign key
3	link	varchar(255)	Menunjukkan link website yang diuji

Tabel 4.10: Tabel errors

No	Kolom	Tipe	Keterangan
4	worker	varchar(255)	Menunjukkan jumlah request yang diinginkan pengguna
5	username	varchar(100)	Menunjukkan keterangan pengguna pembuat request
6	host	varchar(100)	Menunjukkan IP node host yang digunakan kontainer penguji
7	type	varchar(100)	Menunjukkan type error
8	text	varchar(255)	Menunjukkan keterangan error
9	args	varchar(255)	Menunjukkan argumen error
10	location_url	varchar(255)	Menunjukkan lokasi url error

4.4.2.8 Tabel Summary Results

Pada tabel summary_results menyimpan data rata-rata dari hasil uji beban yang ada pada tabel results. Berikut definisi tabel summary_results pada Tabel 4.11.

Tabel 4.11: Tabel summary results

No	Kolom	Tipe	Keterangan
1	id	bigint(20)	Sebagai primary key, nilai awal adalah auto_increment

Tabel 4.11: Tabel summary results

No	Kolom	Tipe	Keterangan
2	scenario_id	varchar(255)	Menunjukkan skenario id sebagai foreign key
3	link	varchar(255)	Menunjukkan link website yang diuji
4	method	varchar(100)	Menunjukkan metode uji
5	worker	varchar(255)	Menunjukkan jumlah request yang diinginkan pengguna
6	username	varchar(100)	Menunjukkan keterangan pengguna pembuat request
7	error	varchar(100)	Menunjukkan presentase kegagalan saat dilakukan uji beban
8	response_end	varchar(100)	Menunjukkan rata-rata hasil uji response time dalam satuan detik
9	dom_content_load	varchar(100)	Menunjukkan rata-rata hasil uji waktu memuat DOM web dalam satuan detik
10	load_event_end	varchar(100)	Menunjukkan rata-rata hasil uji load time dalam satuan detik
11	css_trace_end	varchar(100)	Menunjukkan rata-rata hasil uji waktu memuat css time dalam satuan detik

Tabel 4.11: Tabel summary results

No	Kolom	Tipe	Keterangan
12	first_meaningful	varchar(100)	Menunjukkan rata-rata hasil uji waktu memuat konten utama dalam satuan detik

4.4.3 Implementasi Task Queue

Task queue akan digunakan untuk mengatur antrian request dari pengguna, pada tugas akhir ini implementasi task queue akan dipasang pada komputer yang sama dengan web service. Bahasa pemrograman yang akan digunakan untuk mengimplementasikan task queue adalah bahasa pemrograman Python, sedangkan untuk basis data akan terkoneksi dengan basis data MySQL yang terpasang pada server yang berbeda. Selain itu task queue akan dijalankan setiap 1 menit sekali pada crontab. Selama task queue berjalan algoritmenya akan selalu melakukan pengecekan apakah ada antrian yang bisa dieksekusi. Pseudocode untuk task queue tertera pada Kode Sumber IV.13 dan untuk konfigurasi crontab pada Kode Sumber IV.14.

```
1  Connection
2
3  Variable Declaration
4
5  DECLARE ADDITIONAL FUNCTION
6    Get Data Task Queue from MySQL <- LIMIT 1
7    RETURN data
8
9  MAIN FUNCTION
10   Data = CALL ADDITIONAL FUNCTION
11   IF Data Not NULL
12     Do Task Queue Job
```

```
13      Else
14       RETURN Null to Output File
15    END FUNCTION
```

Kode Sumber IV.13: Konfigurasi docker-compose.yml

```
1   * * * * * /usr/bin/python3 queue.py >> output.
        log 2>&1
```

Kode Sumber IV.14: Konfigurasi docker-compose.yml

BAB V

PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas uji coba dan evaluasi dari sistem yang dibuat. Sistem akan diuji coba fungsionalitasnya dengan menjalankan skenario pengujian performa pada web. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan Uji coba sistem ini terdiri dari beberapa komponen yaitu web service, server basis data, server manager node, dua server worker. Server yang digunakan sistem menggunakan layanan Virtual Priate Server dari DigitalOcean, sedangkan web service akan dibangun di komputer penulis. Spesifikasi untuk setiap komponen ditunjukkan pada Tabel 5.1.

Tabel 5.1: Spesifikasi komponen

No	Komponen	Perangkat Keras	Perangkat Lunak
1	Web Service & Task Queue	Processor AMD FX-7600P Radeon R7, 4 Core, 8GB RAM, 250GB SSD	Ubuntu 18.04 LTS, Laravel 5.8, Python 3.6
2	Basis Data	1 Core Processor, 1GB RAM, 20GB SSD	Ubuntu 18.04 LTS, MySQL 5.7
3	Manager Node	2 Core Processor, 4GB RAM, 80GB SSD	Ubuntu 18.04 LTS, Python 3.6, Docker 18.09.6, Node.js 8.15, NPM 6.4.1, Chrome, Puppeteer 0.12.0, MySQL Client 5.7

Tabel 5.1: Spesifikasi komponen

No	Komponen	Perangkat Keras	Perangkat Lunak
4	Worker 1	2 Core Processor, 4GB RAM, 80GB SSD	Ubuntu 18.04 LTS, Python 3.6, Docker 18.09.6, Node.js 8.15, NPM 6.4.1, Chrome, Puppeteer 0.12.0, MySQL Client 5.7
5	Worker 2	2 Core Processor, 4GB RAM, 80GB SSD	Ubuntu 18.04 LTS, Python 3.6, Docker 18.09.6, Node.js 8.15, NPM 6.4.1, Chrome, Puppeteer 0.12.0, MySQL Client 5.7

Untuk akses ke setiap komponen, digunakan IP publik yang disediakan untuk masing-masing komponen. Detail ditunjukkan pada Tabel 5.2.

Tabel 5.2: IP dan hostname server

No	Komponen	IP	Hostname
1	Web Service	10.151.253.110	night
2	Basis Data	178.128.123.143	NIGHT
3	Manager Node	167.71.194.235	CLOUD
4	Worker 1	165.22.55.82	RAIN
5	Worker 2	167.71.194.233	STORM

5.2 Skenario Uji Coba

Uji coba ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasikan terhadap kebutuhan sistem benar-benar telah diimplementasikan dan bekerja seperti yang seharusnya.

5.2.1 Skenario Uji Fungsionalitas

5.3 Hasil Uji Coba dan Evaluasi

5.3.1 Uji Fungsionalitas

(Halaman ini sengaja dikosongkan)

BAB VI

PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya dengan hasil uji coba dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari proses perencangan, implementasi dan pengujian terhadap sistem, dapat diambil beberapa kesimpulan berikut:

6.2 Saran

Berikut beberapa saran yang diberikan untuk pengembangan lebih lanjut:

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," in *IEEE Internet Computing*, vol. 1, 2014, hal. 81–84.

[2] "Get started, part 4: Swarms," 2019, 24 Mei 2019. [Daring]. Tersedia pada: https://docs.docker.com/get-started/part4/. [Diakses: 24 Mei 2019].

[3] "Puppeteer," 2019, 09 April 2019. [Daring]. Tersedia pada: https://pptr.dev/. [Diakses: 09 April 2019].

[4] R. Roemer, *Backbone.js Testing*. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK: Packt Publishing Ltd, 7 2013, hal. 141–142.

[5] "Getting started with headless chrome," 10 Mei 2019. [Daring]. Tersedia pada: https://developers.google.com/web/updates/2017/04/headless-chrome. [Diakses: 10 Mei 2019].

[6] Joyent, "About node.js," 1 Juni 2019. [Daring]. Tersedia pada: https://nodejs.org/en/about/. [Diakses: 1 Juni 2019].

[7] S. Tilkov dan S. Vinoski, "Node.js: Using javascript to build high-performance network programs," in *IEEE Internet Computing*, vol. 14, 2010, hal. 80–83.

[8] "What is docker?" 2018, 01 Desember 2018. [Daring]. Tersedia pada: https://www.docker.com. [Diakses: 01 Desember 2018].

[9] "Swarm mode key concepts," 2019, 24 Mei 2019. [Daring]. Tersedia pada: https://docs.docker.com/engine/swarm/key-concepts/. [Diakses: 24 Mei 2019].

[10] "Raft consensus in swarm mode," 2019, 24 Mei 2019. [Daring]. Tersedia pada: https://docs.docker.com/engine/swarm/raft/. [Diakses: 24 Mei 2019].

[11] M. Arif, A. Dentha, dan H. W. S. Sindung, "Designing internship monitoring system web based with laravel framework," in *2017 IEEE International Conference on Communication, Networks and Satellite (Comnetsat)*, 2017.

[12] "What is python? executive summary," 2019, 20 Juni 2019. [Daring]. Tersedia pada: https://www.python.org/doc/essays/blurb/. [Diakses: 20 Juni 2019].

LAMPIRAN A

INSTALASI PERANGKAT LUNAK

Instalasi Docker

Untuk melakukan instalasi Docker, dilakukan seperti langkah-langkah berikut:

```
$ sudo apt-get -y install \
apt-transport-https \
ca-certificates \
curl

$ curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo apt-key add -

$ sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/
   linux/ubuntu \
$(lsb_release -cs) \
stable"

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli
   containerd.io
```

Kode Sumber A.1: Perintah untuk instalasi Docker

Setelah menjalankan perintah pada kode sumber A.1. Jalankan perintah berikut agar Docker bisa dijalankan sebagai Non-Root User.

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

Kode Sumber A.2: Perintah untuk mengubah hak User

Instalasi Docker Compose

Docker Compose digunakan untuk automasi dalam menjalankan Dockerfile dan berperan untuk pembuatan Docker Image. Instalasi Docker Compose dapat dilihat pada kode sumber A.3.

```
$ sudo curl -L "https://github.com/docker/compose
    /releases/download/1.24.0/docker-compose-$(
    uname -s)-$(uname -m)" -o /usr/local/bin/
    docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose

$ sudo ln -s /usr/local/bin/docker-compose /usr/
    bin/docker-compose
```

Kode Sumber A.3: Perintah instalasi Docker Compose

LAMPIRAN B

KODE SUMBER

Kode Sumber Pengambilan Data Metrics Performance

Isi berkas index.js

```
const puppeteer = require('puppeteer');
const testPage = require('./testPage');
const fs = require('fs');
const db = require('../config/databases');
const scenario_id = process.argv[2];
const counter = process.argv[3];
const worker = process.argv[4];
const host = process.argv[5];

let rawdata = fs.readFileSync('/app/code/assets/
    config_'+scenario_id+'.json');
let config = JSON.parse(rawdata);
(async () => {
  const browser = await puppeteer.launch({args: [
      '--no-sandbox']});
  const page = await browser.newPage();
  await page.on('console', msg =>
    db.query('INSERT INTO errors (scenario_id,
        link, worker, username, host, type, text,
        location_url) VALUES (?, ?, ?, ?, ?, ?, ?,
         ?)',
    [ scenario_id, config.scenario_link, worker,
        config.username, host, msg._type, msg.
        _text, msg._location.url ])
  );
  try{
    let data = await testPage(page, config,
        counter);
```

```
21      db.query('INSERT INTO tests (scenario_id,
            link, worker, username, host, response_end
            , dom_interactive, dom_content_load,
            load_event_end, css_trace_start,
            css_trace_end, first_meaningful, timestamp
            ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
             ?, ?)',
22      [ scenario_id, config.scenario_link, worker,
            config.username, host, data.Timing.
            responseEnd, data.Timing.domInteractive,
            data.Timing.domContentLoadedEventEnd, data
            .Timing.loadEventEnd, data.TraceResult.
            cssStart, data.TraceResult.cssEnd, data.
            Metrics.FirstMeaningfulPaint, data.Metrics
            .Timestamp ]);
23      db.end();
24      await page.screenshot({path: '/app/output/ss
            '+scenario_id+'.png'});
25      await browser.close();
26    } catch(error) {
27      console.error(error);
28      await page.screenshot({path: '/app/output/ss
            '+scenario_id+'.png'});
29      await browser.close();
30    }
31  })();
```

Kode Sumber B.1: Isi berkas index.js

Isi berkas testPage.js

```
1   const {
2   getTimeFromPerformanceMetrics,
```

```
3   extractDataFromPerformanceMetrics,
4   extractDataFromPerformanceTiming,
5   extractDataFromTracing,
6 } = require('./helpers');
7
8 async function testPage(page, config, counter) {
9   const client = await page.target().
        createCDPSession();
10  await client.send('Performance.enable');
11  const navigationStart =
        getTimeFromPerformanceMetrics(
12    await client.send('Performance.getMetrics'),
13    'NavigationStart'
14  );
15
16  await page.tracing.start({ path: './trace'+
        config.scenario_id+counter+'.json' });
17
18  await page.goto(config.scenario_link);
19
20  if(config.scenario_method == 'POST'){
21    if(config.incr == 'true'){
22      for(var i=0; i<config.attr.length; i++){
23        await page.type('#'+config.attr[i],
              config.val_attr[i]+counter);
24      }
25      await page.click('button[type='+config.
            scenario_button+']');
26    } else {
27      for(var i=0; i<config.attr.length; i++){
28        await page.type('#'+config.attr[i],
              config.val_attr[i]);
29      }
```

```
30        await page.click('button[type='+config.
              scenario_button+']');
31      }
32    }
33
34    const performanceTiming = JSON.parse(
35      await page.evaluate(() => JSON.stringify(
            window.performance.timing))
36    );
37
38    let firstMeaningfulPaint = 0;
39    while (firstMeaningfulPaint === 0) {
40      await page.waitFor(300);
41      performanceMetrics = await client.send('
            Performance.getMetrics');
42      firstMeaningfulPaint =
            getTimeFromPerformanceMetrics(
43        performanceMetrics, 'FirstMeaningfulPaint'
44      );
45    }
46
47    await page.tracing.stop();
48
49    const cssTracing = await extractDataFromTracing
          (
50      './trace'+config.scenario_id+counter+'.json',
51      config.scenario_link,
52    );
53
54    let TraceResult = {
55      cssEnd: cssTracing.end - navigationStart,
56    }
57
```

```
58    let Metrics = extractDataFromPerformanceMetrics
          (
59      performanceMetrics,
60      'FirstMeaningfulPaint',
61    );
62
63    let Timing = extractDataFromPerformanceTiming(
64      performanceTiming,
65      'responseEnd',
66      'domContentLoadedEventEnd',
67      'loadEventEnd',
68    );
69
70    return { TraceResult, Metrics, Timing };
71  }
72
73  module.exports = testPage;
```

Kode Sumber B.2: Isi berkas testPage.js

Isi berkas helpers.js

```
1   const fs = require('fs');
2
3   const getTimeFromPerformanceMetrics = (metrics,
        name) =>
4     metrics.metrics.find(x => x.name === name).
          value * 1000;
5
6   const extractDataFromPerformanceMetrics = (
        metrics, ...dataNames) => {
7     const navigationStart =
          getTimeFromPerformanceMetrics(
```

```
    metrics,
    'NavigationStart'
  );
  const extractedData = {};
  dataNames.forEach(name => {
    extractedData[name] =
      getTimeFromPerformanceMetrics(metrics, name
          ) - navigationStart;
  });
  return extractedData;
};

const extractDataFromPerformanceTiming = (timing,
    ...dataNames) => {
  const navStart = timing.navigationStart;

  const extractedData = {};
  dataNames.forEach(name => {
    extractedData[name] = timing[name] - navStart
        ;
  });

  return extractedData;
};

const extractDataFromTracing = (path, link) =>
    new Promise(resolve => {
  const tracing = JSON.parse(fs.readFileSync(path
      , 'utf8'));
  const resourceTracings = tracing.traceEvents.
      filter(
    x =>
      x.cat === 'devtools.timeline' &&
```

```
35        typeof x.args.data !== 'undefined' &&
36        typeof x.args.data.url !== 'undefined' &&
37        x.args.data.url.includes(link)
38    );
39    const resourceTracingSendRequest =
          resourceTracings.find(
40      x => x.name === 'ResourceSendRequest'
41    );
42    const resourceId = resourceTracingSendRequest.
          args.data.requestId;
43    const resourceTracingEnd = tracing.traceEvents.
          filter(
44      x =>
45        x.cat === 'devtools.timeline' &&
46        typeof x.args.data !== 'undefined' &&
47        typeof x.args.data.requestId !== 'undefined
              ' &&
48        x.args.data.requestId === resourceId
49    );
50    const resourceTracingStartTime =
          resourceTracingSendRequest.ts / 1000;
51    const resourceTracingEndTime =
52      resourceTracingEnd.find(x => x.name === '
            ResourceFinish').ts / 1000;
53
54    fs.unlink(path, () => {
55      resolve({
56        end: resourceTracingEndTime,
57      });
58    });
59  });
60
61  module.exports = {
```

```
62    getTimeFromPerformanceMetrics,
63    extractDataFromPerformanceMetrics,
64    extractDataFromPerformanceTiming,
65    extractDataFromTracing,
66  };
```

Kode Sumber B.3: Isi berkas helpers.js

Kode Sumber Basis Data MySQL

```
1  CREATE TABLE `swarms` (
2    `id` bigint(20) unsigned NOT NULL
       AUTO_INCREMENT,
3    `created_at` timestamp NULL DEFAULT NULL,
4    `updated_at` timestamp NULL DEFAULT NULL,
5    `swarm_ip` varchar(255) COLLATE utf8mb4
       _unicode_ci NOT NULL,
6    `swarm_username` varchar(255) COLLATE utf8mb4
       _unicode_ci NOT NULL,
7    `swarm_password` varchar(255) COLLATE utf8mb4
       _unicode_ci NOT NULL,
8    `is_used` varchar(255) COLLATE utf8mb4
       _unicode_ci NOT NULL,
9    PRIMARY KEY (`id`),
10   UNIQUE KEY `swarms_swarm_ip_unique` (`swarm_ip
       `)
11 ) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=
     utf8mb4 COLLATE=utf8mb4_unicode_ci;
12
13 CREATE TABLE `containers` (
14   `id` bigint(20) unsigned NOT NULL
       AUTO_INCREMENT,
```

```
15    `task_id` varchar(255) COLLATE utf8mb4
          _unicode_ci NOT NULL,
16    `node_id` varchar(255) COLLATE utf8mb4
          _unicode_ci NOT NULL,
17    `container_id` varchar(255) COLLATE utf8mb4
          _unicode_ci NOT NULL,
18    `node_ip` varchar(255) COLLATE utf8mb4
          _unicode_ci NOT NULL,
19    `node_host` varchar(255) COLLATE utf8mb4
          _unicode_ci NOT NULL,
20    `status` smallint(6) NOT NULL DEFAULT '0',
21    `username` varchar(100) COLLATE utf8mb4
          _unicode_ci NOT NULL DEFAULT '0',
22    PRIMARY KEY (`id`)
23  ) ENGINE=InnoDB AUTO_INCREMENT=101 DEFAULT
        CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
24
25  CREATE TABLE `users` (
26    `id` bigint(20) unsigned NOT NULL
          AUTO_INCREMENT,
27    `name` varchar(255) COLLATE utf8mb4_unicode_ci
          NOT NULL,
28    `email` varchar(255) COLLATE utf8mb4_unicode_ci
           NOT NULL,
29    `username` varchar(255) COLLATE utf8mb4
          _unicode_ci NOT NULL,
30    `email_verified_at` timestamp NULL DEFAULT NULL
          ,
31    `password` varchar(255) COLLATE utf8mb4
          _unicode_ci NOT NULL,
32    `remember_token` varchar(100) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
33    `created_at` timestamp NULL DEFAULT NULL,
```

```
34    `updated_at` timestamp NULL DEFAULT NULL,
35    `request_test` varchar(100) COLLATE utf8mb4
        _unicode_ci DEFAULT NULL,
36    PRIMARY KEY (`id`),
37    UNIQUE KEY `users_email_unique` (`email`),
38    UNIQUE KEY `users_username_unique` (`username`)
39  ) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=
      utf8mb4 COLLATE=utf8mb4_unicode_ci;
40
41  CREATE TABLE `role_user` (
42    `id` bigint(20) unsigned NOT NULL
        AUTO_INCREMENT,
43    `created_at` timestamp NULL DEFAULT NULL,
44    `updated_at` timestamp NULL DEFAULT NULL,
45    `role_id` int(10) unsigned NOT NULL,
46    `user_id` int(10) unsigned NOT NULL,
47    PRIMARY KEY (`id`)
48  ) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=
      utf8mb4 COLLATE=utf8mb4_unicode_ci;
49
50  CREATE TABLE `roles` (
51    `id` bigint(20) unsigned NOT NULL
        AUTO_INCREMENT,
52    `created_at` timestamp NULL DEFAULT NULL,
53    `updated_at` timestamp NULL DEFAULT NULL,
54    `name` varchar(255) COLLATE utf8mb4_unicode_ci
        NOT NULL,
55    `description` varchar(255) COLLATE utf8mb4
        _unicode_ci NOT NULL,
56    PRIMARY KEY (`id`)
57  ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=
      utf8mb4 COLLATE=utf8mb4_unicode_ci;
58
```

```
59 CREATE TABLE `scenarios` (
60   `id` bigint(20) unsigned NOT NULL
       AUTO_INCREMENT,
61   `created_at` timestamp NULL DEFAULT NULL,
62   `updated_at` timestamp NULL DEFAULT NULL,
63   `scenario_id` varchar(255) COLLATE utf8mb4
       _unicode_ci NOT NULL,
64   `username` varchar(255) COLLATE utf8mb4
       _unicode_ci NOT NULL,
65   `scenario_method` varchar(255) COLLATE utf8mb4
       _unicode_ci NOT NULL,
66   `scenario_link` varchar(255) COLLATE utf8mb4
       _unicode_ci NOT NULL,
67   `scenario_worker` varchar(255) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
68   `scenario_status` smallint(6) NOT NULL DEFAULT
       '0',
69   `scenario_button` varchar(255) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
70   PRIMARY KEY (`id`)
71 ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=
     utf8mb4 COLLATE=utf8mb4_unicode_ci;
72
73 CREATE TABLE `queues` (
74   `id` bigint(20) unsigned NOT NULL
       AUTO_INCREMENT,
75   `created_at` timestamp NULL DEFAULT NULL,
76   `updated_at` timestamp NULL DEFAULT NULL,
77   `username` varchar(255) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
78   `worker` int(11) DEFAULT NULL,
79   `status` smallint(6) DEFAULT NULL,
80   PRIMARY KEY (`id`)
```

```
81 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=
       utf8mb4 COLLATE=utf8mb4_unicode_ci;
82
83 CREATE TABLE `results` (
84   `id` bigint(20) unsigned NOT NULL
       AUTO_INCREMENT,
85   `scenario_id` varchar(255) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
86   `link` varchar(255) COLLATE utf8mb4_unicode_ci
       DEFAULT NULL,
87   `method` varchar(100) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
88   `worker` varchar(255) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
89   `username` varchar(100) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
90   `host` varchar(100) COLLATE utf8mb4_unicode_ci
       DEFAULT NULL,
91   `response_end` varchar(100) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
92   `dom_content_load` varchar(100) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
93   `load_event_end` varchar(100) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
94   `css_trace_end` varchar(100) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
95   `first_meaningful` varchar(100) COLLATE utf8mb4
       _unicode_ci DEFAULT NULL,
96   `status` smallint(6) DEFAULT '0',
97   PRIMARY KEY (`id`)
98 ) ENGINE=InnoDB AUTO_INCREMENT=126 DEFAULT
       CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
99
```

```
100 CREATE TABLE `errors` (
101   `id` bigint(20) unsigned NOT NULL
        AUTO_INCREMENT,
102   `scenario_id` varchar(255) COLLATE utf8mb4
        _unicode_ci DEFAULT NULL,
103   `link` varchar(255) COLLATE utf8mb4_unicode_ci
        DEFAULT NULL,
104   `worker` varchar(255) COLLATE utf8mb4
        _unicode_ci DEFAULT NULL,
105   `username` varchar(100) COLLATE utf8mb4
        _unicode_ci DEFAULT NULL,
106   `host` varchar(100) COLLATE utf8mb4_unicode_ci
        DEFAULT NULL,
107   `type` varchar(100) COLLATE utf8mb4_unicode_ci
        DEFAULT NULL,
108   `text` varchar(255) COLLATE utf8mb4_unicode_ci
        DEFAULT NULL,
109   `args` varchar(255) COLLATE utf8mb4_unicode_ci
        DEFAULT NULL,
110   `location_url` varchar(255) COLLATE utf8mb4
        _unicode_ci DEFAULT NULL,
111   PRIMARY KEY (`id`)
112 ) ENGINE=InnoDB AUTO_INCREMENT=126 DEFAULT
      CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
113
114 CREATE TABLE `summary_results` (
115   `id` bigint(20) unsigned NOT NULL
        AUTO_INCREMENT,
116   `scenario_id` varchar(255) COLLATE utf8mb4
        _unicode_ci DEFAULT NULL,
117   `link` varchar(255) COLLATE utf8mb4_unicode_ci
        DEFAULT NULL,
```

```
118   `method` varchar(100) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
119   `worker` varchar(255) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
120   `username` varchar(100) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
121   `error` varchar(100) COLLATE utf8mb4_unicode_ci
          DEFAULT NULL,
122   `response_end` varchar(100) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
123   `dom_content_load` varchar(100) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
124   `load_event_end` varchar(100) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
125   `css_trace_end` varchar(100) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
126   `first_meaningful` varchar(100) COLLATE utf8mb4
          _unicode_ci DEFAULT NULL,
127   PRIMARY KEY (`id`)
128 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=
      utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Kode Sumber B.4: Basis data MySQL

BIODATA PENULIS

Cahya Putra Hikmawan, akbrab dipanggil Awan lahir pada tanggal 09 September 1996 di Krembung, Kabupaten Sidoarjo. Penulis merupakan seorang mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember. Memiliki hobi antara lain membaca novel dan memancing. Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain Staff Departemen Pengembangan Profesi Mahasiswa Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ke-2 dan ke-3. Pernah menjadi staff dan badan pengurus harian National Logic Competition Schematics tahun 2016 dan 2017. Selain itu penulis pernah menjadi asisten dosen dan praktikum di mata kuliah Sistem Operasi dan Jaringan Komputer.