# SISTEM MANAJEMEN LAYANAN WEB BERBASIS PLATFORM AS A SERVICE (PAAS) DENGAN API OPENSTACK

**SKRIPSI**



**IDA BAGUS RATHU EKA SURYA WIBAWA**

**1308605045**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**JIMBARAN**

**2021**

# SURAT PERNYATAAN KEASLIAN KARYA ILMIAH

Yang bertanda tangan di bawah ini menyatakan bahwa naskah Skripsi dengan judul:

**“SISTEM MANAJEMEN LAYANAN WEB BERBASIS PLATFORM AS A SERVICE (PAAS) DENGAN API OPENSTACK”**

Nama : Ida Bagus Rathu Eka Surya Wibawa

NIM : 1308605045

Program Studi : Teknik Informatika

E-mail : guzekawibawa@gmail.com

Nomor telp/HP : 081337482800

Alamat : Jalan Pratu Made Rambug, Br. Sasih No. 49 Batubulan

Belum pernah dipublikasikan dalam dokumen skripsi, jurnal nasional maupun internasional atau dalam prosiding manapun, dan tidak sedang atau akan diajukan untuk publikasi di jurnal atau prosiding manapun. Apabila di kemudian hari terbukti terdapat pelanggaran kaidah-kaidah akademik pada karya ilmiah saya, maka saya bersedia menanggung sanksi-sanksi yang dijatuhkan karena kesalahan tersebut, sebagaimana diatur oleh Peraturan Menteri Pendidikan Nasional Nomor 17 Tahun 2010 tentang Pencegahan dan Penanggulangan Plagiat di Perguruan Tinggi.

Demikian Surat Pernyataan ini saya buat dengan sesungguhnya untuk dapat dipergunakan bilamana diperlukan.

Badung, 29 Januari 2021

Yang membuat pernyataan,

Ida Bagus Rathu Eka Surya Wibawa

NIM. 1308605045

# LEMBAR PENGESAHAN TUGAS AKHIR

Judul : Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack

Nama : Ida Bagus Rathu Eka Surya Wibawa

NIM : 1308605045

Tanggal Seminar : 29 Januari 2021

Disetujui Oleh :

|  |  |  |
| --- | --- | --- |
| Pembimbing I | Penguji I | |
|  |  | |
| I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs.  NIP. 198901272012121001 | I Gede Santi Astawa, S.T., M.Cs.  NIP. 198012062006041003 | |
| Pembimbing II | Penguji II | |
|  |  | |
| I Komang Ari Mogi, S.Kom., M.Kom.  NIP. 198409242008011007 | Gst. Ayu Vida Mastrika Giri, S.Kom., M.Cs.  NIP. 1990060620181123001 | |
|  | Penguji III | |
|  |  | |
|  | Made Agung Raharja,S.Si.,M.Cs  NIP. 1985091920130122003 | |
| Mengetahui,  Ketua Jurusan Ilmu Komputer  FMIPA Universitas Udayana  Ketua,  Dr. Ir. I Ketut Gede Suhartana, S.Kom., M.Kom  NIP. 197201102008121001 | | |

Judul : Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack

Nama : Ida Bagus Rathu Eka Surya Wibawa (NIM : 1308605045)

Pembimbing : 1. I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs.

2. I Komang Ari Mogi, S.Kom., M.Kom.

# ABSTRAK

Cloud computing merupakan teknologi yang saat ini mulai berkembang dalam banyak aktivitas teknologi informasi. Cloud computing merupakan model komputasi yang semua sumber daya yang ada dalam layanan cloud dijalankan dengan media jaringan internet. Dengan adanya cloud computing memudahkan para developer dalam melakukan komputasi tanpa harus melakukan instalasi aplikasi pada komputer, developer hanya perlu mengaksesnya melalui internet. Cloud computing memiliki beberapa fasilitas yang dapat dipilih oleh developer sesuai kebutuhan developer seperti Infrastructure as a Service(IaaS), Platform as a Service(PaaS). Serta Software as a Service(SaaS).

Pengembangan Platform as a Service sebagai salah satu teknologi cloud computing yang dapat digunakan oleh pengembang aplikasi untuk mengembangkan aplikasi yang akan dibuat tanpa perlu menyediakan infrastruktur, database, framework aplikasi dan lain sebagainya serta bersifat dinamis. Dalam pengembangan layanan cloud yang dikelola oleh seorang sistem administrator atau developer, tugas menginstalasi dan menkonfigurasi sistem pada server maupun software aplikasi dilakukan dengan otomatis dengan menggunakan platform otomatisasi sistem linux untuk efisiensi waktu dan memanagemen penggunanya dengan lebih mudah.

**Kata kunci: *Cloud Computing, Platform as a Service, automation, ansible, API, Openstack***

Judul : Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack

Nama : Ida Bagus Rathu Eka Surya Wibawa (NIM : 1308605045)

Pembimbing : 1. I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs.

2. I Komang Ari Mogi, S.Kom., M.Kom.

***ABSTRACT***

*Cloud computing is a technology that is currently starting to develop in many information technology activities. Cloud computing is a computing model in which all resources in cloud services run on the internet network media. With cloud computing, it makes it easier for users to perform computations without having to install applications on computers, users only need to access them via the internet. Cloud computing has several facilities that can be selected by users according to user needs, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS). As well as Software as a Service (SaaS).*

*Development of Platform as a Service as a cloud computing technology that can be used by application developers to develop applications that will be created without the need to provide infrastructure, databases, application frameworks and so on and are dynamic. In developing cloud services that are managed by a system administrator or sysadmin, the task of installing and configuring systems on servers and application software is carried out automatically using the Linux system automation platform for time efficiency and easier management of users.*

***Key Words : Cloud Computing, Platform as a Service, automation, ansible, API, Openstack.***

# KATA PENGANTAR

Penelitian dengan judul " Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack " ini disusun dalam rangkaian kegiatan pelaksanaan Tugas Akhir di Jurusan Ilmu Komputer Universitas Udayana.

Selama menyelesaikan penelitian tugas akhir ini, penulis telah banyak menerima bimbingan, pengarahan, petunjuk dan saran. Untuk itu penulis mengucapkan terima kasih yang sebesar-besarnya kepada yang terhormat:

1. Bapak I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs. selaku Pembimbing I dan Bapak I Komang Ari Mogi, S.Kom., M.Kom. selaku Pembimbing II yang telah banyak membimbing dan membantu menyempurnakan penelitian ini.
2. Bapak Dr. I Ketut Gede Suhartana, S.Kom., M.Kom. selaku Ketua Jurusan Ilmu Komputer Universitas Udayana, Bapak I Gusti Ngurah Anom Cahyadi Putra, S.T., M.Cs. selaku Ketua Komisi Tugas Akhir Jurusan Ilmu Komputer Universitas Udayana yang telah banyak memberikan masukan dan motivasi sehingga memperlancar dalam proses pelaksanaan penelitian ini.
3. Bapak dan ibu dosen di Jurusan Ilmu Komputer Universitas Udayana yang bersedia meluangkan waktunya untuk memberikan masukan dalam penyempurnaan penelitian ini.
4. Rekan – rekan mahasiswa Jurusan Ilmu Komputer khususnya angkatan 2013 yang telah memberi dukungan, motivasi, semangat dan kerja sama dalam penelitian ini.

Penulis menyadari tugas akhir ini masih jauh dari kesempurnaan. Untuk itu dengan segala kerendahan hati, penulis mengharapkan kritik dan saran yang bermanfaat dari pembaca demi kesempurnaan tugas akhir ini.

Jimbaran, Januari 2021

Penulis

# DAFTAR ISI

[SISTEM MANAJEMEN LAYANAN WEB BERBASIS PLATFORM AS A SERVICE (PAAS) DENGAN API OPENSTACK i](#_Toc74481435)

[SURAT PERNYATAAN KEASLIAN KARYA ILMIAH i](#_Toc74481436)

[LEMBAR PENGESAHAN TUGAS AKHIR ii](#_Toc74481437)

[ABSTRAK iii](#_Toc74481438)

[KATA PENGANTAR v](#_Toc74481439)

[DAFTAR ISI vi](#_Toc74481440)

[DAFTAR TABEL ix](#_Toc74481441)

[DAFTAR GAMBAR x](#_Toc74481442)

[DAFTAR LAMPIRAN xii](#_Toc74481443)

[BAB I PENDAHULUAN 2](#_Toc74481444)

[1.1. Latar Belakang 2](#_Toc74481445)

[1.2. Rumusan Masalah 2](#_Toc74481446)

[1.3. Batasan Masalah 3](#_Toc74481447)

[1.4. Tujuan Penelitian 3](#_Toc74481448)

[1.5. Manfaat Penelitian 3](#_Toc74481449)

[1.6. Metodelogi Penelitian 3](#_Toc74481450)

[3.1.1. Metode Pengumpulan Data 4](#_Toc74481451)

[3.1.2. Kerangka Kerja Penelitian 4](#_Toc74481452)

[3.1.3. Perancangan Sistem 5](#_Toc74481453)

[3.1.4. Metode Pengembangan Perangkat Lunak 7](#_Toc74481454)

[3.1.5. Rancangan ERD (Entity Relationship Diagram) 10](#_Toc74481455)

[3.1.6. Evaluasi dan Validasi Hasil 10](#_Toc74481456)

[BAB II TINJAUAN PUSTAKA 11](#_Toc74481457)

[2.2. Tinjauan Empiris 11](#_Toc74481458)

[2.3. *Cloud Computing* 12](#_Toc74481459)

[1. IaaS (Infrastructure as a Service) 13](#_Toc74481460)

[2. Platform as a Service (PaaS) 13](#_Toc74481461)

[3. Software as a Service (SaaS) 13](#_Toc74481462)

[2.4. Openstack 14](#_Toc74481463)

[2.5. REST API 15](#_Toc74481464)

[2.6. ANSIBLE 15](#_Toc74481465)

[BAB III ANALISIS DAN PERANCANGAN SISTEM 17](#_Toc74481466)

[3.1. Analisis Kebutuhan 17](#_Toc74481467)

[3.1.1. Kebutuhan Fungsional Sistem 17](#_Toc74481468)

[3.1.2. Kebutuhan Non Fungsional Sistem 18](#_Toc74481469)

[3.2. Penentuan Proses – proses dalam Operasi Sistem 18](#_Toc74481470)

[3.3. Perancangan Antarmuka Sistem 18](#_Toc74481471)

[3.4. Skenario Pengujian Sistem 20](#_Toc74481472)

[1. BlackBox Testing 20](#_Toc74481473)

[2. Performance Testing 21](#_Toc74481474)

[BAB IV HASIL DAN PEMBAHASAN 23](#_Toc74481475)

[4.1. Gambaran Umum Sistem 23](#_Toc74481476)

[4.2. Lingkungan Perancangan dan Implementasi Sistem 23](#_Toc74481477)

[4.3. Implementasi Basis Data 24](#_Toc74481478)

[4.4. Implementasi Antarmuka 25](#_Toc74481479)

[4.5. Pengujian Sistem 37](#_Toc74481480)

[4.5.1. Black Box Testing 37](#_Toc74481488)

[4.5.2. Performance Testing 39](#_Toc74481489)

[4.6. Analisa 42](#_Toc74481490)

[BABI V SIMPULAN DAN SARAN 43](#_Toc74481491)

[5.1. Simpulan 43](#_Toc74481492)

[5.2. Saran 43](#_Toc74481493)

[DAFTAR PUSTAKA 44](#_Toc74481494)

[LAMPIRAN 45](#_Toc74481495)

# DAFTAR TABEL

[Tabel 3.1 Tabel Kebutuhan Fungsional 17](#_Toc74469238)

[Tabel 3.2 Tabel Kebutuhan Non Fungsional 18](#_Toc74469239)

[Tabel 3.1. Tabel Pengujian Black Box 21](#_Toc74469240)

[Tabel 4.1 Kode Input File 37](#_Toc74469241)

[Tabel 4.2 Kode Result dari Input File 37](#_Toc74469242)

[Tabel 4.3 Kode Mengirim File 38](#_Toc74469243)

[Tabel 4.4 Kode Menerima File 39](#_Toc74469244)

[Tabel 4.5 Kode Ekstraksi Fitur 40](#_Toc74469245)

[Tabel 4.6 Kode Menerima Hasil Ekstraksi Fitur 44](#_Toc74469246)

[Tabel 4.7 Kode Jaccard Distance 46](#_Toc74469247)

[Tabel 4.8 Kode KNN 47](#_Toc74469248)

[Tabel 4.9 Hasil Pengujian ke-1 50](#_Toc74469249)

[Tabel 4.10 Hasil Pengujian ke-1 52](#_Toc74469250)

[Tabel 4.11 Hasil Pengujian ke-2 53](#_Toc74469251)

[Tabel 4.12 Hasil Pengujian ke-3 53](#_Toc74469252)

[Tabel 4.13 Hasil Pengujian ke-5 54](#_Toc74469253)

# DAFTAR GAMBAR

[Gambar 1.1. Kerangka kerja penelitian 4](#_Toc74479496)

[Gambar 1.2. Desain Kerja Sistem 6](#_Toc74479497)

[Gambar 1.3. *Flowchart* Konfigurasi Otimatis 8](#_Toc74479498)

[Gambar 1.4. *Flowchart Front-end* Sistem 9](#_Toc74479499)

[Gambar 1.5. Rancangan Entity Relationship Diagram 10](#_Toc74479500)

[Gambar 3.38. Tampilan *Landing Page* 20](#_Toc74479501)

[Gambar 4.9. Tabel basis data *user* 24](#_Toc74479502)

[Gambar 4.10. Tabel basis data *project* 24](#_Toc74479503)

[Gambar 4.11. Tampilan Menu Awal Openstack 25](#_Toc74479504)

[Gambar 4.12. Tampilan Dashboard Openstack 26](#_Toc74479505)

[Gambar 4.13. Tampilan Menu Detail 26](#_Toc74479506)

[Gambar 4.14. Tampilan Menu Deteksi 27](#_Toc74479507)

[Gambar 4.15. Tampilan Pilih File 27](#_Toc74479508)

[Gambar 4.16. Tampilan Server 28](#_Toc74479509)

[Gambar 4.17. Hasil Ekstraksi Fitur pada Server 29](#_Toc74479510)

[Gambar 4.18. Buat Project Baru 29](#_Toc74479511)

[Gambar 4.19. Tampilan Hasil Deteksi 30](#_Toc74479512)

[Gambar 4.20. Tampilan Hasil Deteksi 31](#_Toc74479513)

[Gambar 4.21. Tampilan Hasil Deteksi 31](#_Toc74479514)

[Gambar 4.22. Tampilan Hasil Deteksi 32](#_Toc74479515)

[Gambar 4.23. Tampilan Hasil Deteksi 33](#_Toc74479516)

[Gambar 4.24. Tampilan Hasil Deteksi 33](#_Toc74479517)

[Gambar 4.25. Tampilan Hasil Deteksi 34](#_Toc74479518)

[Gambar 4.26. Tampilan Hasil Deteksi 35](#_Toc74479519)

[Gambar 4.27. Tampilan Hasil Deteksi 35](#_Toc74479520)

[Gambar 4.28. Tampilan Hasil Deteksi 36](#_Toc74479521)

[Gambar 4.29. Tampilan Hasil Deteksi 36](#_Toc74479522)

[Gambar 4.30. Tampilan Hasil Deteksi 37](#_Toc74479523)

[Gambar 4.31. Grafik Hasil Pengujian ke-1 39](#_Toc74479524)

[Gambar 4.32. Grafik Hasil Pengujian ke-2 40](#_Toc74479525)

[Gambar 4.33. Grafik Hasil Pengujian ke-3 41](#_Toc74479526)

[Gambar 4.34 Grafik Hasil Pengujian ke-4 41](#_Toc74479527)

[Gambar 4.35 Grafik Hasil Pengujian ke-5 41](#_Toc74479528)

[Gambar 4.36 Grafik Akurasi 42](#_Toc74479529)

# DAFTAR LAMPIRAN

Data Permissions 60

Data API Calls 67

# BAB I PENDAHULUAN

## Latar Belakang

Pada era teknologi seperti sekarang ini, perkembangan teknologi sangat meningkat pesat, dimana para pengembang aplikasi saat ini dengan mudah membangun sebuah *web server* yang yang dapat dikerjakan secara bersama-sama dengan menggunakan komputasi awan ( *Cloud Computing*). Sebelum teknologi *cloud computing* ditemukan pengembang aplikasi berbasis *web* lebih sering mengembangkan aplikasi *web* secara lokal di perangkat komputer. Membuat *client* dari pengembang aplikasi *web* tidak dapat *memonitoring* atau menilai sejauh mana aplikasi web yang di buat oleh pengembang aplikasi *web* telah selesai, menyebabkan kurangnya dalam memanajemen aplikasi *web*. Serta pada proses perilisan aplikasi web pun membutuhkan waktu yang tidak sebentar, banyak hal yang butuh diperhitungkan seperti penyimpanan data yang besar, ketersediaan *domain*, *data processing* dan masih banyak lainnya.

Menurut Ricky W. Griffin menyatakan bahwa manajemen adalah sebuah proses pengorganisasian, pengkoordinasian, perencanaan, dan pengontrolan sumber daya agar dapat mencapai sasaran (*goals*) secara efisien dan efektif. Efektif sendiri sebuah tujuan mampu dicapai sesuai dengan apa yang telah direncanakan. Perlunya memanajemen aplikasi *web* yang dibuat untuk memudahkan pengembang aplikasi beserta tim pengembang lainnya dapat saling meninjau kebutuhan aplikasi *web* yang di buat secara bersama – sama walau berada di lokasi yang berbeda.

Konsep komputasi awan ini sudah banyak menarik minat industri digital maupun pendidikan. Solusi berbasis cloud sepertinya menjadi kunci bagi organisasi teknologi informasi yang memiliki permasalahan dengan keterbatasan anggaran (Teng & Magoules,2010). Komputasi awan merupakan paradigma yang baru dalam komputasi terdistribusi, menyajikan banyak konsep teknologi, ide, serta arsitektur yang disajikan secara *service-oriented.* Pada pemanfaatanya komputasi awan mengubah cara bagaimana layanan informasi disediakan maupun di sebarkan. Dengan memanfaatkan teknologi komputasi awan dalam memanajemen aplikasi web, *developer* / pengembang aplikasi web bekerja secara efektif dan efisien dari segi waktu maupun perangkat. Saat ini sudah banyak penyedia layanan *cloud* computing untuk para pengembang aplikasi seperti *Google Cloud Platform,* *Amazon Web Services, Alibaba Cloud Server*, bahkan untuk di Indonesia ada *Biznet GIO cloud.*

Pada Penelitian sebelumnya, Nishant Kumar Singh dkk, menjelaskan pada perkembangan cloud computing pada sisi infrastruktur menuntut penyebaran layanan yang cepat, pada waktu yang bersamaan kebutuhan penyediaan secara otomatis ikut berperan dalam pengembangan aplikasi melalui komputasi awan. Maka dari itu, dalam Tugas Akhir ini penelitian dilakukan dengan membangun sebuah Platform website berbasis cloud untuk pengembang aplikasi web dalam membangun sebuah website secara otomatasasi untuk meningkatkan efisiensi kinerja tim secara bersamaan mampu menajemen kebutuhan pengembangan aplikasi yang dibuat.

## Rumusan Masalah

Berdasarkan latar belakang masalah dapat dirumuskan rumusan masalah sebagai berikut :

* 1. Bagaimana otomatisasi dengan *Ansible* berkerja pada ruang lingkup komputasi awan.
  2. Bagaimana membangun sistem yang memfasilitasi developer dalam mengembangkan *website* ke dalam komputasi awan.

## Batasan Masalah

Agar penelitian lebih fokus, permasalah yang dicakup tidak akan terlalu luas dan akan sesuai dengan maksud dan tujuan yang ingin dicapai. Ruang lingkup yang dibahas dalam penelitian ini akan dibatasi hanya pada :

1. Sistem dibuat dalam bentuk purwa rupa.
2. Layanan komputasi awan menggunakan *Openstack.*
3. Sistem meyalani *virtual server* untuk membangun layanan aplikasi *web*.
4. Sistem dibangun menggunakan Bahasa permrograman *python.*
5. Sistem tidak menyediakan *domain* untuk layanan yang diterima *developer website*

## Tujuan Penelitian

Berdasarkan rumusan masalah, tujuan dari penelitian ini adalah sebagai berikut :

* + 1. Membantu *developer website* mengurangi beban resource komputer dalam mengembangkan aplikasi yang dibuat.

1. Membantu *developer website* membuat sebuah *server website* dan melakukan instalasi *web server* secara otomatis*.*
2. Membantu efektivitas *developer website* dengan membangun *virtual server* secara *online*, serta flesibelitas dalam pengembasngan website yang dibuat.

## Manfaat Penelitian

Dengan dilakukannya penelitian ini, diharapkan mampu mengurangi beban kerja komputer *developer website* dengan komputasi awan yang menyediakan *virtual server web* dan sebuah *platform digital* dalam mengembangkan *website*.

## Metodelogi Penelitian

Bagian ini akan menjelaskan mengenai langkah – langkah yang akan dilakukan dalam merancang sistem manajemen layanan web berbasis *Platform as a Service (PaaS)* dengan *API openstack.*

### Metode Pengumpulan Data

Pada tahap ini dilakukan analisis kebutuhan sistem, analisis kebutuhan sistem meliputi *data* yang digunakan, pembelajaran dari referensi yang sudah ada dan perangkat yang digunakan baik perangkat lunak maupun perangkat keras:

Tahapan pertama yang dilakukan dalam penelitian ini adalah mengidentifikasi permasalahan. Tahap ini merupakan tahap yang paling penting dalam penelitian karena jalannya penelitian didasarkan atas permasalahan yang terjadi. Setelah menentukan masalah yang terjadi, tahapan yang diperlukan selanjutnya adalah menentukan rumusan masalah dan tujuan yang ingin dicapai dalam penenlitian. Pada penelitian ini identifikasi permasalahan dilakukan dengan menggunakan teknik observasi, dari teknik ini maka akan dapat diketahui mengenai keluhan – keluhan yang ada di lapangan.

Tahap kedua yang dilakukan dalam metodelogi penelitian ini adalah studi literatur. Studi literatur dilakukan dengan mengambil literatur – literatur pendukung dari jurnal – jurnal ilmiah, baik jurnal dalam negeri ataupun jurnal luar negeri dan dari beberapa buku. Dalam studi literatur ini, penulis mencari sumber terkait permasalahan – permasalahan yang perlu menjadi perbaikan dalam penelitian selanjutnya..

### Kerangka Kerja Penelitian

Bagian ini menjelaskan tentang bagaimana penelitian ini dilakukan. Berikut adalah beberapa proses penting yang dilakukan :



Gambar 1.. Kerangka kerja penelitian

Instalasi kebutuhan konfigurasi *VPS(Virtual Private Server)*

Tahap pertama dalam penelitian ini dengan mempersiapkan sistem dan software yang dibutuhkan dalam mengkonfigurasi kebutuhan *web server* dalam menginstalasi modul-modul kebutuhan sistem seperti sistem operasi, *database, framework*, *web server .*

Menentukan proses konfigurasi kebutuhan aplikasi

Pada tahap ini mempersiapkan kebutuhan konfigurasi untuk membangun sebuah *web server*, dari *web service* sampai *database* yang diperlukan untuk menampung setiap *developer*.

Melakukaan proses otomatisasi konfigurasi modul aplikasi

Pada tahap ini melakukan otomatisasi dalam mengkonfigurasi sebuah *web server*. Proses otomatisasi menggunakan Ansible.

Merancang *web front-end* untuk *developer website.*

Pada tahap ini bila semua konfigurasi sudah berjalan dengan baik, dibuat *web* yang dapat digunakan oleh *developer* mengatur kebutuhan *web* yang dibuat serta dapat melihat informasi dari setiap *VPS* yang dimiliki.

Melakukan test pada hasil konfigurasi

Pada tahap ini menguji hasil konfigurasi dari *web* *interface*. Untuk menguji konfigurasi berjalan dengan baik.

Melakukan uji beban *server*

beban ini dilakukan untuk menguji tingkat kemampuan *server* dalam memberikan pelayanan pada developer aplikasi.

### Perancangan Sistem

Pada tahap Perancangan sistem, dilakukan berdasarkan hasil analisa kebutuhan sistem yang sudah dilakukan sebelumnya. Hal ini dilakukan agar, perancangan tidak keluar dari tujuan sistem yang dikembangkan. Berikut merupakan desain arsitektur kerja sistem :



Gambar 1.. Desain Kerja Sistem

Pada gambar 1.2 merupakan hubungan antara setiap perangkat lunak yang ada pada server utama. Berikut ini merupakan penjelasannya :

*Web dan database*

Database digunakan untuk menyimpan informasi pengguna serta informasi pada *VPS* yang dimiliki oleh setiap sysadmin.

1. *Web* dan *framework* flask *REST API*

*Flask REST API* digunakan untuk menghubungkan *web front-end* dengan back-end. Dimana *back-end* dibuat dengan menggunakan *framework* *flask* dari *python*, selanjutnya dibuat *API* tersendiri agar *web front-end* dapat mengirim dan menerima informasi serta konfigurasi yang dilakukan yang nantinya akan diproses oleh *back*-*end* *server*.

1. *Database* dan *ansible* *API*

Hubungan *database* dengan *ansible* *API* akan mengambil informasi *developer* berupa *username*, *email* dan *password* yang nantinya akan dimasukan pada konfigurasi dalam *VPS* yang selanjutnya sebagai *super amin* pada *VPS* yang dibuat.

1. *framework* *flask* *REST* *API* dan *asible* *API*

Hubungan *framework flask REST API* dengan *asible API*. Ketika user melakukan konfigurasi pada *web* utama akan dikirim melalui *REST* *API* *flask* selanjutnya konfigurasi tersebut akan digunakan oleh *API* ansible untuk mengkonfigurasi *VPS* yang dibuat.

1. *framework* *flask* dan openstack *API*

Hubungan *framework* *flask* dan *openstack* *API* adalah pada *flask* dikonfigurasi untuk dapat terhubung dengan server *devstack*, dimana *server* *openstack* yang mengelola dalam membuat sebuah *VPS*.

1. *openstack API* dan *server*

*API* *openstack* digunakan untuk terhubung dengan *server back-end* yang dibuat dengan *framework* *flask*. Untuk dapat mengembangkan aplikasi *openstack* dari bahasa pemrograman yang berbeda.

1. *Ansible API* dan *Virtual Private Server*

hubungan *ansible* *API* dengan *VPS* adalah ansible akan mengirimkan konfigurasi pada *VPS* melalui *SSH* yang yang dibuat. Selanjutnya pada *VPS* akan melakukan konfigurasi yang diperlukan untuk membangun sebuah *web* aplikasi secara otomatis. *Developer* hanya perlu memilih keperluan yang ada menu *web front-end.*

1. *server* dan *Virtual Private Server*

hubungan antara *server* dan *VPS(Virtual Private Server)* adalah *server* mengelola dan memberikan resource pada VPS serta mengatur konektivitas setiap *VPS* yang dimiliki.

### Metode Pengembangan Perangkat Lunak

Pada bagian ini akan menjelaskan proses yang dilakukan oleh perangkat lunak, bagaimana proses tersebut berjalan yang akan dijelaskan pada setiap flowchart berikut ini:

* + - 1. *Flowchart* Konfigurasi Otomatis



Gambar 1.. *Flowchart* Konfigurasi Otimatis

Pada desain *flowchart* diatas merupakan garis besar gambaran system yang akan dikerjakan, bagaimana alur *platform* yang dibuat dapat mengotomatitasi dalam instalasi serta konfigurasi sebuah layanan pada *virtual* *machine*. Mulai dari memasukkan aplikasi pendukung yang ingin digunakan seperti *database*, *web* *service,username* dan *e-mail administrator* layanan aplikasi. Kemudian dari hasil inputan tersebut diolah oleh *flask* *API* yang dibuat untuk dimasukkan ke dalam konfigurasi yang terdapat ada *Ansible API*, selanjutnya dari *ansible API* dengan menggunakan *SSH* akan melakukan instalasi konfigurasi pada *virtual server* yang didapat setiap *user*. Bila konfigurasi telah selesai user akan menerima sebuah *ip public* untuk dapat mengkakses layanan aplikasi.

* + - 1. *Flowchart Front-end Web*



Gambar 1.. *Flowchart Front-end* Sistem

Pada desain *flowchart* *Front*-*end* *Web* merupakan alur kerja web dari sisi front-side atau pada sisi user. Setelah user melakukan registrasi pada *website*, selanjutnya user akan diarahkan ke menu instalasi dan konfigurasi dalam membangun layanan *VPS*. *Menu* akan dibuat secara *user* *friendly* sehingga *user* hanya perlu memilih menu konfigurasi system. Ketika *user* sudah selesai memilih menu konfigurasi, system akan memvalidasi informasi yang dimasukkan oleh *user*, bila sudah benar system akan mengirimkan hasil instalasi dan konfigurasi dalam sebuah data berrntuk *JSON* ke dalam system yang nantinya akan dikirim ke virtual sever untuk melakukan konfigurasi dan instalasi modul kebutuhan aplikasi selanjutnya sistem akan melakukan backup berkala untuk menjaga keamanan data developer bila tejadi satu kesalahan dapat dilakukan *restore data*.

### Rancangan *ERD (Entity Relationship Diagram)*

Pada desain *Entity Relationship Diagram (ERD)* diatas menjelaskan hubungan user dengan VPS yang dimiliki user memiliki sebuah vps dengan satu ip public. Serta pada vps akan menyimpan log mulai dari kinerja dan resource yang dimiliki.



Gambar 1.. Rancangan *Entity Relationship Diagram*

### Evaluasi dan Validasi Hasil

Pada tahap ini dilakukan evaluasi terhadap desain perancangan sistem. Bila sistem telah sesuai dengan kebutuhan awal yang didefinisikan akan dilanjutkan ke tahap implementasi. Namun apabila desain sistem belum memenuhi kebutuhan awal yang didefinisikan, maka akan dilakukan perancangan ulang desain sistem

# BAB II TINJAUAN PUSTAKA

## Tinjauan Empiris

1. **Analisis Implementasi Metode Klasifikasi Bayes Untuk Deteksi *Malware* *Android* (Ridlo & Karima, 2016)**

Pada penelitian ini, penulis mendeteksi *malware* dengan menerapkan metode klasifikasi bayes. Penulis menggunakan dua buah model properti *feature*. Model yang pertama adalah model properti *permissions* dan model yang kedua adalah model properti berbasis *code*. Peneliti melakukan tiga kali pengujian klasifikasi, pertama hanya dengan model properti berbasis *permissions* dan menunjukan akurasi *classifier* sebesar 54.0%, pengujian yang kedua dilakukan dengan model properti berbasis *code* dan menunjukan akurasi *classifier* sebesar 89.5%, dan yang terakhir peneliti melakukan pengujian dengan menggabungkan kedua properti tersebut dan menunjukan akurasi *classifier* sebesar 88.0%.

1. ***Evaluation* *of* *machine* *learning* *classifiers* *for* *mobile* *malware* *detection* (Narudin, Feizollah, Anuar, & Gani, 2014)**

Pada penelitian ini, penulis melakukan perbandingan lima buah metode untuk mendeksi *malware*. Metode yang digunakan adalah *bayes* *network*, *multi-layer* *perception*, knn, j48, dan *random* *forest*. Penulis mendeteksi *malware* dengan menggunakan pendekatan berbasis anomali, dengan duah buah dataset yaitu *public* *dataset* yang diambil dari MalGenome dan *private* *dataset* berupa *malware* yang dikumpulkan sendiri oleh penulis. *Feature* yang digunakan oleh penulis berupa *feature* lalu lintas data dalam sebuah jaringan. Dari pengujian didapatkan hasil akurasi *true* *positive* untuk dataset pertama dari metode BN sebesar 99.97%, metode MLP sebesar 91.73%, metode J48 sebesar 99.98%, metode KNN sebesar 99.65%, metode random forest sebesar 99.98%. Untuk dataset kedua, didapatkan hasil akurasi *true* *positive* dari metode BN sebesar 99.98%, metode MLP sebesar 88.25%, metode J48 sebesar 99.98%, metode KNN sebesar 99.65%, metode random forest sebesar 99.99%.

1. ***Merging* *Permission* *and* *API* *Features* *for Android Malware Detection* (Qiao, Sung, & Liu, 2016)**

Pada penelitian ini, penulis menggunakan algoritma SVM, ANN, RF untuk mendeteksi *malware*. Fitur yang digunakan penulis dalam penelitian ini berupa *permission* dan API. Pada penelitian ini, didapatkan hasil yaitu tingkat ketepatan SVM, ANN, dan RF dalam mengklasifikasi *malware* menggunakan fitur *Binary* *permission* sebesar 87.72%, 92.12%, 87.84%. tingkat ketepatan SVM, ANN, dan RF dalam mengklasifikasi *malware* menggunakan fitur *Binary* API sebesar 88.40%, 93.84%, 90.10%. tingkat ketepatan SVM, ANN, dan RF dalam mengklasifikasi *malware* menggunakan fitur gabungan *Binary* *permission* dengan *Binary* API sebesar 91.48%, 96.00%, 94.08%. Tingkat ketepatan SVM, ANN, dan RF dalam mengklasifikasi *malware* menggunakan fitur *Numerical* *permission* sebesar 79.68%, 92.52%, 92.52%. tingkat ketepatan SVM, ANN, dan RF dalam mengklasifikasi *malware* menggunakan fitur *Numerical* API sebesar 78.40%, 94.18%, 94.34%. tingkat ketepatan SVM, ANN, dan RF dalam mengklasifikasi *malware* menggunakan fitur gabungan *Numerical* *permission* dengan *Numerical* API sebesar 81.68%, 94.66%, 94.98%.

1. ***Qweqweqwe***
2. **Qweqweqe**

## *Cloud Computing*

Cloud computing mengacu pada aplikasi dan service yang berjalan dalam jaringan data terdistribusi dengan menggunakan sumberdaya virtual dan internet akses protokol pada umumnya. Hal ini dibedakan pada gagasan sumberr daya virtual dan detail dari mesin fisik sistem dalam software yang berjalan secara abtraksi dari user. (Sosinsky, 2011). Cloud Computing secara sederhana adalah “layanan teknologi informasi yang bisa dimanfaatkan atau diakses oleh pelanggannya melalui jaringan internet”. Komputasi awan adalah suatu konsep umum yang mencakup SaaS, Web 2.0, dan tren teknologi terbaru lain yang dikenal luas, dengan tema umum berupa ketergantungan terhadap Internet untuk memberikan kebutuhan komputasi pengguna. Sebagai contoh, Google Apps menyediakan aplikasi bisnis umum secara sharing yang diakses melalui suatu penjelajah web dengan perangkat lunak dan data yang tersimpan di server. Jenis-jenis dari Cloud Computing dapat dijabarkan sebagai berikut:

### IaaS (Infrastructure as a Service)

Terletak satu level lebih rendah dibanding PaaS. Ini adalah sebuah layanan yang “menyewakan” sumberdaya teknologi informasi dasar, yang meliputi media penyimpanan, processing power, memory, sistem operasi, kapasitas jaringan dan lainlain, yang dapat digunakan oleh penyewa untuk menjalankan aplikasi yang dimilikinya. Model bisnisnya mirip dengan penyedia data center yang menyewakan ruangan untuk co-location, tapi ini lebih ke level mikronya. Penyewa tidak perlu tahu, dengan mesin apa dan bagaimana caranya penyedia layanan menyediakan layanan IaaS. Yang penting, permintaan mereka atas sumberdaya dasar teknologi informasi itu dapat dipenuhi.

### Platform as a Service (PaaS)

Konsepnya hampir serupa dengan IaaS. Namun Platform disini adalah penggunaan operating system dan infrastruktur pendukungnya. Yang cukup terkenal adalah layanan dari situs Force.Com serta layanan dari para vendor server.Seperti namanya, PaaS adalah layanan yang menyediakan modul-modul siap pakai yang dapat digunakan untuk mengembangkan sebuah aplikasi, yang tentu saja hanya bisa berjalan diatas platform tersebut. Seperti juga layanan SaaS, pengguna PaaS tidak memiliki kendali terhadap sumber daya komputasi dasar seperti memory, media penyimpanan, processing power dan lain-lain, yang semuanya diatur oleh provider layanan ini. Pionir di area ini adalah Google AppEngine, yang menyediakan berbagai tools untuk mengembangkan aplikasi di atas platform Google, dengan menggunakan bahasa pemrograman Phyton dan Django. Kemudian Salesforce juga menyediakan layanan PaaS melalui Force.com, menyediakan modul-modul untuk mengembangkan aplikasi diatas platform Salesforce yang menggunakan bahasa Apex. Dan mungkin yang jarang sekali kita ketahui, bahwa Facebook juga bisa dianggap menyediakan layanan PaaS, yang memungkinkan kita untuk membuat aplikasi diatasnya.Salah satu yang berhasil menangguk untung besar dari layanan PaaS

### Software as a Service (SaaS)

Berada satu tingkat diatas PaaS dan IaaS, dimana disini yang ditawarkan adalah software atau suatu aplikasi bisnis tertentu. Contoh yang paling mutakhir adalah SalesForce.Com, Service-Now.Com, Google Apps, dsb. SaaS ini merupakan layanan Cloud Computing yang paling dahulu populer.Software as a Service ini merupakan evolusi lebih lanjut dari konsep ASP (Application ServiceProvider). Sesuai namanya, SaaS memberikan kemudahan bagi pengguna untuk bisa memanfaatkan sumberdaya perangkat lunak dengan cara berlangganan. Sehingga tidak perlu mengeluarkan investasi baik untuk in house development ataupun pembelian lisensi. Dengan cara berlangganan via web, pengguna dapat langsung menggunakan berbagai fitur yang disediakan oleh penyedia layanan. Hanya saja dengan konsep SaaS ini, pelanggan tidak memiliki kendali penuh atas aplikasi yang mereka sewa.Hanya fiturfitur aplikasi yang telah disediakan oleh penyedia saja yang dapat disewa oleh pelanggan

## Openstack

OpenStack adalah sistem aplikasi cloud yang mengelola sumberdaya seperti komputasi, penyimpan dan jaringan, yg tersedia pada infrastruktur fisik seperti dalam sebuah fasilitas pusat-data (data center). Admin atau sysadmin dapat mengendalikan dan melakukan provisioning atas sumber-daya ini melalui dashboard / antar-muka web. Developer dapat mengakses sumber-daya tersebut melalui sejumlah API standar (Mulyana, 2017).

Openstack merupakan platform cloud-computing open source yang memungkinkan sysadmin untuk membangun sebuah "IAAS" Infrastruktur sebagai service cloud yang bergerak secara massal pada komoditas hardware dan skala. Openstack mengontrol kolam besar komponen komputasi awan di seluruh datacenter, semua dikelola melalui dashboard yang menyediakan administrator kontrol penuh sambil memberikan sysadmin kemampuan untuk sumber penyediaan melalui antarmuka web.

## REST API

REST (REpresentational State Transfer) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP (Hypertext Transfer Protocol) sebagai protocol untuk komunikasi data. REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000.

REST API merupakan web service yang bertujuan untuk mendukung kebutuhan server web pada suatu kebutuhan situs atau aplikasi lainnya. program client mengunakan Application Programming Interface (API) untuk berkomunikasi dengan layanan web. Secara umum, API mengekspos seperangkat data dan fungsi untuk memfasilitasi interaksi antara proram komputer dan memungkinkan merekas saling bertukar informasi (Masse, 2012).

Dalam pengaplikasiannya, REST lebih banyak digunakan untuk web service yang berorientasi pada resource. Maksud orientasi pada resource adalah orientasi yang menyediakan resource-resource sebagai layanannya dan bukan kumpulan-kumpulan dari aktifitas yang mengolah resource tersebut

## ANSIBLE

Ansible merupakan sebuah softaware yang bisa membantu seorang sistem administrator untuk melakukan otomasi pada server. ansible merupakan teknologi yang digunakan untuk melakukan otomasi, memudahkan dalam melakukan konfigurasi server, tujuan dibuat ansible membuat hal tersebut menjadi sederhana dan mudah.namun tetap fokus pada keamana dan keandalan dalam melakukan otomasi. ansible menggunakan OpenSSH untuk transportasi ( dengan mode socket yang cepat).

Dengan ansible developer dapat melakukan instalasi, deployment hingga melakukan update server. Sistem kerja yang dimiliki oleh ansible membutuhkan koneksi khusus berupa SSH. Ansible bekerja di koneksi SSH remote client yang ingin di deploy atau dilakukan otomasi. Pada ansible memerlukan inventory atau data server tujuan untuk dapat dilakukan otomasi. Pada penerapannya, ansible menggunakan *playbook* dan *roles*, dimana konfigurasi tersebut dalam format markup *YAML* dan environment variabel dapat ditulis dalam bentuk *JSON*. .

Ansible dirancang untuk memudahkan para sysadmin dan para pakar IT mengelola lingkungan server dengan mudah. ansible mengelola mesin dengan cara yang tidak biasa, tidak pernah bertanya cara melakukan upgrade daemon jarak jauh atau masalah karena tidak dapat mengelola sistem karena daemon sistem terhapus.

Ansible merupakan salah satu jenis *Configuration* *Management* *Tools* ayan dapat digunakan merubah proses infrastruktur manajemen dari program manual menjadi otomatis. Dalam zaman cloud kehadiran ansible membantu para sysadmin atau para devops dalam instalasi dan konfigurasi server dengan otomatis, oleh karena itu ansible menjadi satu platform yang digunakan untuk mengelola server – server.

# BAB III ANALISIS DAN PERANCANGAN SISTEM

## Analisis Kebutuhan

Pada tahap ini dilakukan analisis kebutuhan sistem, analisis kebutuhan sistem meliputi data yang digunakan, pembelajaran dari referensi yang sudah ada dan perangkat yang digunakan baik perangkat lunak maupun perangkat keras:

Tahapan pertama yang dilakukan dalam penelitian ini adalah mengidentifikasi permasalahan. Tahap ini merupakan tahap yang paling penting dalam penelitian karena jalannya penelitian didasarkan atas permasalahan yang terjadi. Setelah menentukan masalah yang terjadi, tahapan yang diperlukan selanjutnya adalah menentukan rumusan masalah dan tujuan yang ingin dicapai dalam penelitian. Pada penelitian ini identifikasi permasalahan dilakukan dengan menggunakan teknik observasi, dari teknik ini maka akan dapat diketahui mengenai keluhan – keluhan yang ada di lapangan.

Tahap kedua yang dilakukan dalam metodelogi penelitian ini adalah studi literatur. Studi literatur dilakukan dengan mengambil literatur – literatur pendukung dari jurnal – jurnal ilmiah, baik jurnal dalam negeri ataupun jurnal luar negeri dan dari beberapa buku. Dalam studi literatur ini, penulis mencari sumber terkait permasalahan – permasalahan yang perlu menjadi perbaikan dalam penelitian selanjutnya.

### Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem mencakup fungsi – fungsi yang mampu dilakukan oleh sistem. Kebutuhan fungsional sistem pada penelitian ini ditunjukan pada tabel 3.1.

Tabel 3.1 Tabel Kebutuhan Fungsional

|  |  |  |
| --- | --- | --- |
| No. | Kebutuhan Fungsional | User |
| 1 | Membuat virtual server dengan cloud | √ |
| 2 | Melakukan instalasi kebutuhan package dalam membangun web server | √ |

### Kebutuhan Non Fungsional Sistem

Adapun kebutuhan non fungsional yang harus dipenuh oleh sistem untuk melengkapi sistem secara keseluruhan, diantaranya sebagai berikut :

Tabel 3.2 Tabel Kebutuhan Non Fungsional

|  |  |  |
| --- | --- | --- |
| No. | Kebutuhan Non Fungsional | Keterangan |
| 1 | *Efficient* | Sistem mampu membantu kebutuhan pengembang aplikasi web secara tepat guna, |
| 2 | *User* *Friendly* | Sistem yang didukung dengan tampilan yang menarik dan mudah digunakan |

## Penentuan Proses – proses dalam Operasi Sistem

Setelah melakukan Teknik kajian pustaka pada tahap sebelumnya, secara garis besar proses-proses yang ada pada Sistem Manajemen Layanan *Web Berbasis Platform as a Service (PaaS)* dengan *API Openstack* adalah :

1. Proses pembuatan *virtual server* secara otomatis dan sesuai dengan kebutuhan pengembang aplikasi *web*
2. Proses *install package* secara otomatis dengan menggunakan ansible
3. Proses memberikan IP secara otomatis dan kunci private untuk melakukan remote pada virtual server cloud.
4. Proses pengembangan secara mandiri oleh pengembang aplikasi web, jika paket yang di butuhkan tidak ada dalam menu sistem.

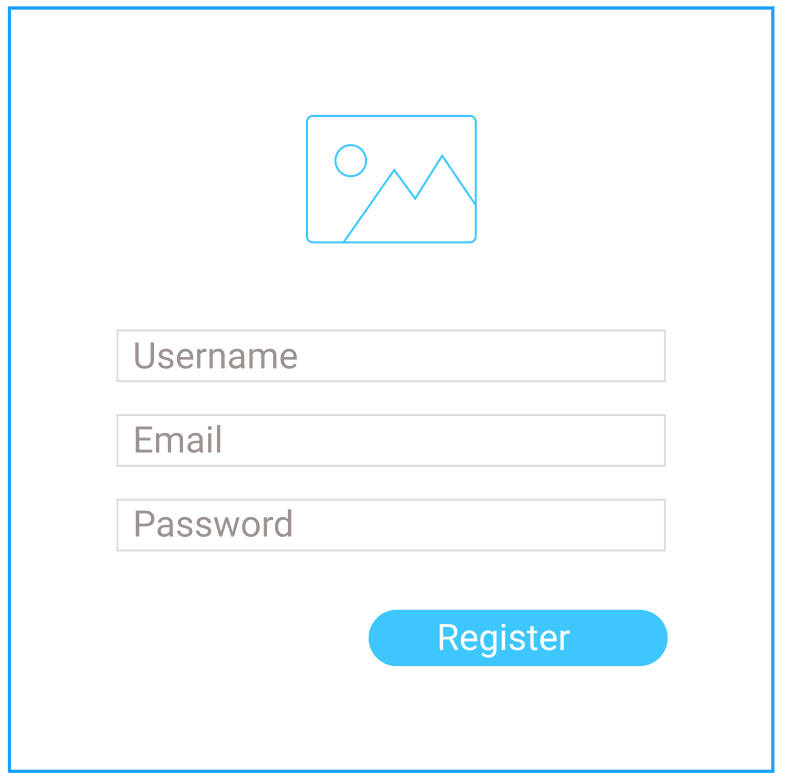
## Perancangan Antarmuka Sistem

Pada tahap ini dijelaskan perancangan antarmuka sistem dalam memanajemen layanan web berasis Platform as a Service (PaaS), pada sistem perancangan ini Sistem administrator dapat membuat dan mengembangkan websitenya pada sistem cloud computing.



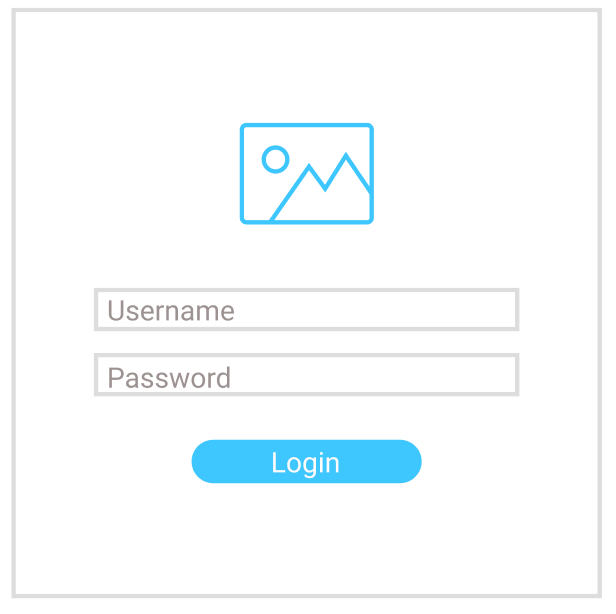
Gambar 3.1. Tampilan *Landing Page*

Gambar 3.1 merupakan rancangan landing page yang berisi informasi dari sistem cloud yang di kembangkan, serta berisi informasi layanan dan fitur yang diberikan kepada pengembang aplikasi yang akan menggunakan sistem cloud.



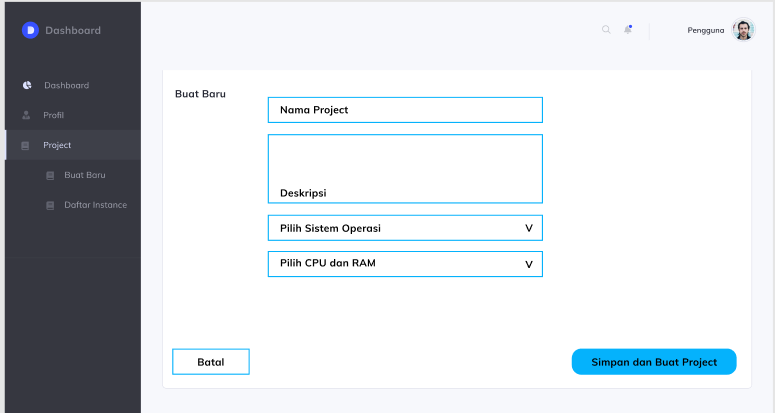
Gambar 3.2. Tampilan *Landing Page*

Pada gambar 3.2 merupakan rancangan sistem untuk dapat masuk ke dalam sistem cloud, pengembang aplikasi web harus melakukan pendaftaran, ini dimaksudkan untuk mempermudah pengelolaan sistem virtual server cloud.



Gambar 3.3. Tampilan *Landing Page*

Pada gambar 3.3 merupakan rancangan untuk login form untuk dapat masuk dalam sistem yang ingin dibuat.



Gambar 3. 4. Tampilan *Landing Page*

Pada gambar 3.4

## Skenario Pengujian Sistem

Bagian ini menjelaskan mengenai skenario pengujian sistem yang akan dilakukan pada Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack. Pengujian sistem ini dilakukan dengan menguji black box dan *Performance Testing.*

### BlackBox Testing

Black Box Testing atau dikenal sebagai “Behaviour Testing” merupakan suatu metode pengujian yang digunakan untuk menguji executable code dari suatu perangkat lunak terhadap perilakunya. Pendekatan Black Box Testing dapat dilakukan jika kita sudah memiliki executable code. Orang-orang yang terlibat dalam Black Box Testing adalah tester, end-user, dan developer.

Fokus dari pengujian ini ialah pada kebutuhan fungsional perangkat lunak, sehingga memungkinkan tester mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu untuk program. Kesalahan yang ditemukan dalam pengujian, nantinya dapat disimpulkan apakah kesalahan tersebut murni dikarenakan kesalahan dari aplikasi atau kesalahan implementasi dari tester.

Tabel .1. Tabel Pengujian Black Box

|  |  |  |
| --- | --- | --- |
| *Identifikasi* |  | |
| *Nama Kasus Uji* |  | |
| *Deskripsi* |  | |
| *Kondisi Awal* |  | |
| *Tanggal Pengujian* |  | |
| *Penguji* |  | |
| *Skenario* | | |
| *1.* |  | |
| *2.* |  | |
| *(Dst...)* |  | |
| ***Hasil Yang Diharapkan*** | ***Hasil Yang Didapatkan*** | ***Kesimpulan*** |
|  |  |  |

Untuk pengujian antarmuka pengguna atau rancangan skenario pengujian balck box dari sistem ini, dilakukan dua jenis pengujian yaitu pengujian secara happy path yaitu pengujian yang dilakukan dengan cara yang benar, serta pengujian secara alternative path yaitu mencoba segala kemungkinan yang mungkin terjadi pada sistem.

### Performance Testing

Teknik pengujian memvalidasi perilaku perangkat lunak terhadap teknik pengujian software dari sisi kecepatan. Kecepatan ini dalam konteks pengujian dalam mengukur waktu respon perangkat lunak ketika berada jumlah kerja yang berlebih yang biasa dikenal dengan beban kerja. Untuk memperlihatkan kecepatan sebenarnya sebuah perangkat lunak harus dilakukan pengujian performance testing.

Tujuan dari performance testing untuk memvalidasi kecepatan sebuah perangkat lunak terhadap kebutuhan sistem yang cepat. Secara umum harus mendefisinikan kombinasi waktu respons dan beban kerja.

# BAB IV HASIL DAN PEMBAHASAN

## Gambaran Umum Sistem

Sistem yang dibuat bertujuan membantu para pengembang aplikasi berbasis web, dengan meneriakan platform web server berbasis cloud computing, dengan menyediakan virtual machine berbasis cloud/ instance. Pengembang web dapat dengan mudah menjalankan sebuah web yang dibuat dengan mengakses alamat IP yang di berikan. Karena berbasis cloud, pengembang web dapat mengakses virtual machine yang di buat dimana saja secara online. Namun pada penetilian ini pengembang hanya dapat menggunakan jaringan Local untuk mengkases server cloud / private cloud.

Dalam implementasinya, server virtual yang diberikan pada pengembang web, secara otomatis sudah terinstall LAMP package. Namun, bila pengembang web ingin menggunakan package web server yang berbeda dapat melakukan instalasi manual dengan mengakses virtual machine menggunakan SSH.

## Lingkungan Perancangan dan Implementasi Sistem

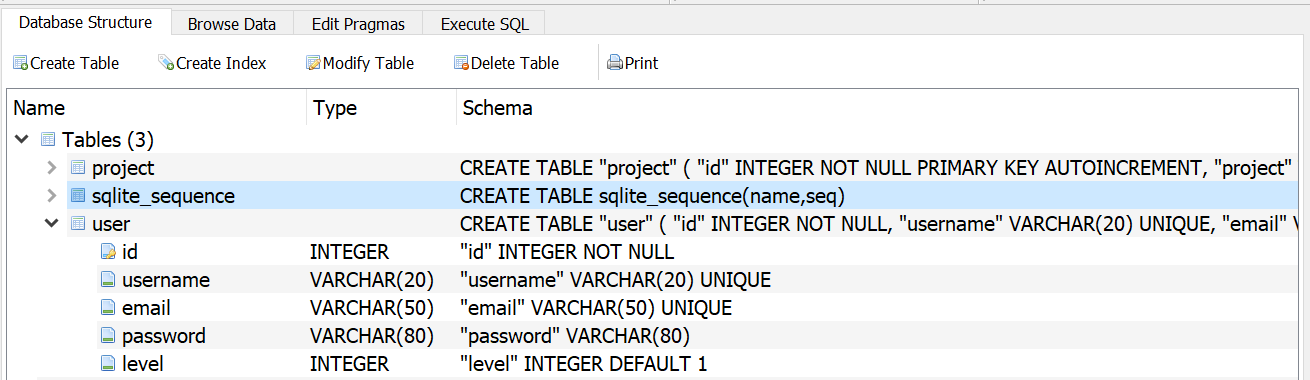
Sistem dirancang dan diimplementasikan pada sistem operasi Ubuntu dalam virtualisasi menggunakan Vmware, server dan sistem berjalan pada Laptop dengan spesifikasi i7, RAM 16 GB, dengan sistem operasi windows 10 64bit dan dengan support virtualisasi.

Sistem diimplementasikan menggunakan bahasa pemrograman python dengan *frameworks* *Flask*. Dalam perancangan dan implementasi sistem digunakan beberapa perangkat lunak untuk memenuhi semua kebutuhan sistem. Berikut adalah perangkat lunak yang digunakan :

1. *VMware*
2. *Visual Studio Codes*
3. *Putty*
4. *Jmeters*

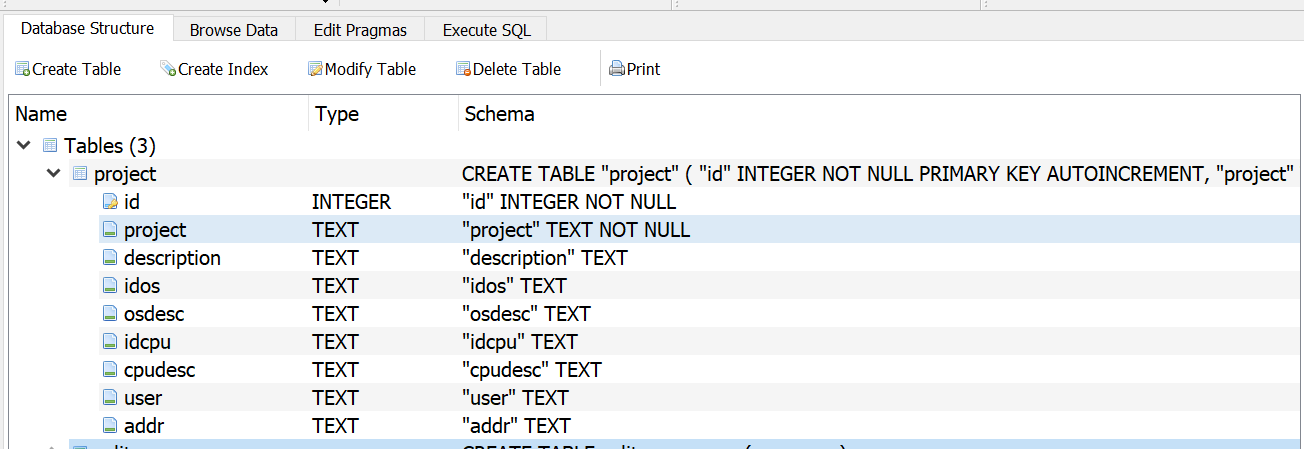
## Implementasi Basis Data

Pada sub bab ini akan dibahas mengenai tahap implementasi basis data pada sistem menginisiasi basis data dibutuhkan dua table, yaitu user atau developer aplikasi dan project.



Gambar 4.. Tabel basis data *user*

Pada gambar 4.1 tabel user berisi infromasi id yang digenerate secara otomatis, username, email, password dalam bentuk hash dan level pengguna.



Gambar 4.. Tabel basis data *project*

Selanjutnya pada gambar 4.2 tabel project berisi ID project, nama project, ID sistem operasi ID CPU, ID penyimpanan, User, dan IP Address server.

## Implementasi Antarmuka

Implementasi antarmuka sistem pada penelitian ini bangun dengan menggunakan bahasa pemrograman python dengan framework flask. Pada sub-bab ini akan menampilkan implementasi antarmuka website, diantaranya landing page, login, dashboard, menu membuat instance baru, dan daftar instance yang telah dibuat, menu profil. Berikut ini adalah pemaparannya :



Gambar 4.11. Tampilan Menu Awal Openstack

Gambar 4.9 Merupakan menu awal untuk masuk ke dalam sistem openstack diharuskan login terlebih dahulu. Untuk dapat login openstack secara otomatis menggenerate kata sandi untuk kepentingan keamanan. Namun kita dapat mengubahnya bila dirasa sangat sulit diingat. Untuk dapat mengngetahui kata sandi dari openstack dapat diakses melalui command-line



Gambar 4.12. Tampilan Dashboard Openstack

Gambar 4. Bila username dan kata sandi benar, maka akan diarahkan ke menu dashboard. Pada menu dashboard kita dapat mengetahui berapa maksimun user yang dapat dibuat oleh server openstack, berapa core CPU yang masih tersedia dan dapat digunakan dalam pembuatan instance / virtual machine.



Gambar 4.13. Tampilan Menu Detail

Gambar 4.9. pada menu Identity, peneliti membuat project baru pada sistem yang berjalan di atas server openstack. Sehingga bila terjadi sesuatu pada sistem yang dibuat tidak mempengaruhi server openstack secara keseluruhan, dan dikemudian hari ada penelitian lebih lanjut dalam pengembangan cloud computing, bisa dilakukan pada nama project yang berbeda. Jadi server openstack secara keseluruhan tidak terganggu atau mengalami masalah bila salah satu project mengalami masalah dalam pengembangannya. Administrator hanya perlu menghapus projectnya tanpa perlu menginstall ulang sistem ataupun server openstack.



Gambar 4.14. Tampilan Menu Deteksi

Untuk menu Security Group, berguna untuk memberi akses port yang bisa digunakan oleh instance. Port yang digunakan oleh instance tergantung kebutuhan yang perlukan, karena setiap instamce memungkinkan menggunakan port yang berbeda – beda. Mungkin saja ada instance yang dibuat untuk penyimpanan aplikasi, ataupun mungkin digunakan untuk jadikan aplikasinya dapat diakses secara publik.



Gambar 4.15. Tampilan Pilih File

Pada menu Network Topology, berisi informasi network IP yang yang instance peroleh, dan di jaringan mana instance terhubung. Pada server openstack secara otomatis setiap membuat project baru akan dialihkan ke jaringan private network, sehingga dalam mengembangkan sebuah project setiap user mendapat alokasi ip private masing – masing dan secara otomatis dilakukan NAT (Network Address Translation) oleh server openstack sehingga meski mendapatkan ip private dari server openstack dapat tetap mendapatkan akses ke jaringan internet, namun tidak dapat digunakan meremote instance melalui jaringan public. Untuk dapat menjadikan instance yang dibuat dapat diremote melalui jaringan public. Pada konfigurasi jaringan penelitian ini dipilih yang public atau dalam hal ini bernama int-ext.



Gambar 4.16. Tampilan Server

Gambar 4.12. merupakan tampilan dari *server* untuk menerima *file* APK yang dikirim oleh *client*. Setelah *file* diterima, *server* akan menjalankan proses ekstraksi fitur seperti yang ditunjukan pada gambar 4.13. *Server* dijalankan pada ip 0.0.0.0 dengan port 5000. Pada gambar 4.12. tampak *server* sedang *standby listening client*. Ketika *server* menerima APK dari *client*, maka akan terlihat tampilan seperti pada Gambar 4.13.



Gambar 4.17. Hasil Ekstraksi Fitur pada Server

Pada penelitian ini, pengguna baru harus melakukan registrasi untuk dapat masuk ke dalam sistem, Setelah melakukan pendaftaran, pengguna akan diarahkan ke form login, selanjutnya pengguna dapat memasukan username dan password yang sudah di buat pada form registrasi.



Gambar 4.18. Buat Project Baru

Setelah proses login, pengguna akan diarahkan langsung ke dashboard menu. Pada dashboard menu, selanjutnya membuat virtual mesin baru pada menu “project”, kemudian pilih “buat baru”, Setelah dipilih akan muncul menu seperti pada gambar , selanjutnya pengguna memasukan nama project. Nama project nantinya akan menjadi nama instance baru sebagai hostname, untuk deskripsi bersifat opsional, dapat ditambahkan keterangan ataupun tidak. Lalu pilih siste operasi yang akan digunakan sebagai server pada lingkungan aplikasi yang akan dibuat. Lalu pilih CPU, CPU disini berisi berapa core yang di butuhkan untuk lingkungan sistem yang akan dibangun oleh pengguna, seberapa besar penyimpanan dan memori RAM yang dibutuhkan untuk mengembangkan sistem yang akan dibuat. Lebih jelasnya dapat dilihat pada gambar 4.17.



Gambar 4.19. Tampilan Hasil Deteksi

Pada proses yang di tampilkan pada gambar , pada backend dibuat fungsi untuk mendapatkan list CPU,RAM dan Storage didefinisikan dengan flavor, dengan mengauthentikasikan terlebih dahulu dengan API openstack menggunakan osconn. Dilanjutkan dengan fungsi yang akan ditampilkan, compute.flavors() untuk menampilakn list cpu ram dan storage, compute.images() untuk menampilkan list sistem operasi yang disediakan.



Gambar 4.20. Tampilan Hasil Deteksi

Pada segi antarmuka terlihat pada gambar tampilan proses dalam membuat instance baru. Pada sistem operasi hanya menampilkan satu sistem operasi saja dikarenakan pada server openstack hanya ada 1 sistem operasi yang dibuat. Begitu pula dengan CPU, RAM dan penyimpanan yang tersedia



Gambar 4.21. Tampilan Hasil Deteksi

Selanjutnya ketika melakukan klik pada tombol buat dan simpan project, sistem aplikasi akan mengirimkan perintah untuk membuat sebuah instance baru pada server openstack dengan menggunakan API dan ansible playbook yang sudah dibuat.



Gambar 4.22. Tampilan Hasil Deteksi

Setelah proses ansible dijalankan, proses debug akan muncul seperti gambar dibawah ini.keluaran data yang ditampilkan oleh hasil debug pada terminal berupa json. Ketika proses pembuatan instance selesai, sistem akan otomatis mengarahkan ke list instance yang sudah kita buat, seperti pada gambar berikut ini.



Gambar 4.23. Tampilan Hasil Deteksi



Gambar 4.24. Tampilan Hasil Deteksi

Pada proses ini sistem hanya akan membuat instance baru dan menginstall sistem operasi saja, bila pengguna ingin menginstall paket XAMPP / LAMP, pengguna hanya perlu memberi centang pada menu xampp dan selanjutnya pilih menu install. Maka sistem akan melakukan instalasi XAMPP pada instance server pengguna secara otomatis dengan menggunakan ansible



Gambar 4.25. Tampilan Hasil Deteksi

Pada gambar di atas tampilan dari server openstack ketika berhasil membuat sebuah instance melalui website yang digunakan, dibutuhkan waktu beberapa menit untuk membuat instance sampai instance tersebut dapat siap digunakan oleh para pengguna untuk mengembangkan aplikasinya. Untuk dapat mengakses instance yang dibuatnya, pengguna dapat melakukannya melalui SSH ke ip address instance yang dibuat



Gambar 4.26. Tampilan Hasil Deteksi

Pada gambar ... merupakan tampilan files yaml, pada direktori ansible ditambahkan sebuah folder bernama roles, jadi folder roles ini nantinya akan di eksekusi oleh lamp.yaml yang sudah berisikan perintah tugas yang akan di install yaitu Apache2, PHP dan MySQL.



Gambar 4.27. Tampilan Hasil Deteksi

Pada gambar diatas, sistem membuat alamat ip address dibuat pada file hosts digunakan untuk mengenali server mana yang akan dieksekusi ketika file yaml dijalankan, serta menyertakan username dan password dari instance yang telah dibuat secara default melalui openstack.



Gambar 4.28. Tampilan Hasil Deteksi

Pada gambar merupakan hasil setelah menjalankan perintah untuk melakukan instalasi lamp.yaml. Bila sukses melakukan instalasi pada instance akan muncul pada gambar di atas semua hasil debug tidak ada kegagalan ketika instalasi.



Gambar 4.29. Tampilan Hasil Deteksi

Untuk menguji webserver yang telah kita buat dapat mengakses ip address yang di miliki instance melalui web browser, bila muncul seperti gambar di atas web server yang di buat dinyatakan berhasil melakukan instalasi. Selanjutnya bila pengguna ingin menambahkan web yang dibuatnya dalam kasus ini diambil menggunakan bahasa pemrograman PHP, pengguna hanya perlu menyalin web yang sudah dibuat ke /var/www/html.



Gambar 4.30. Tampilan Hasil Deteksi

Pada gambar ... merupakan tampilan dari database yang sudah terinstall pada instance, pengguna dapat melakukan pengecekan kembali dengan meremote instance server yangs sudah dibuat dengan menggunakan SSH. Serta melalukan perintah mysql –u root –p untuk mengecek ataupun untuk membuat project database di dalam instance server openstack.

## Pengujian Sistem

Pada penelitian ini pengujian sistem “Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) Dengan API Openstack”, pengujian akan di lakukan dengan metode black box testing dan performance testing. Pengujian dilakukan dengan tujuan aplikasi yang dibuat dapat berjalan sesuai kebutuhan developer.



### Black Box Testing

Black Box Testing merupakan pengujian fungsional dari sistem yang dibuat. Tujuan dari pengujian fungsional adalah untuk memvalidasi perilaku perangkat lunak yang dibuat terhadap fungsionalitas kebutuhan para sysadmin. Berikut merupakan tabel pengujian menggunakan black box :

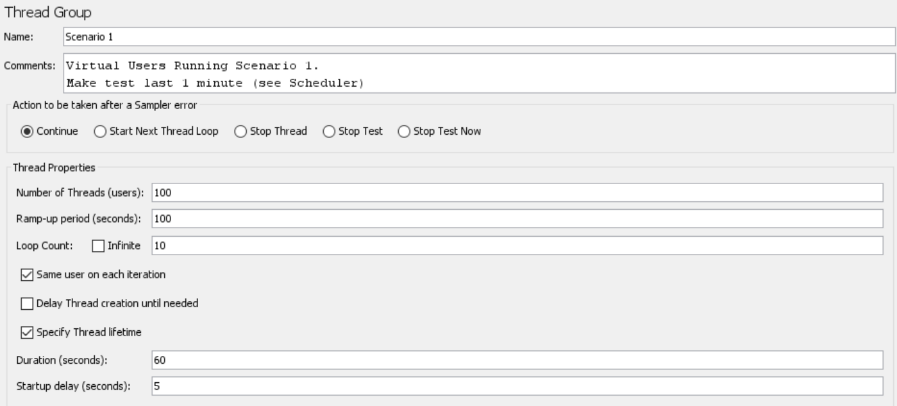
Tabel 4.1 Hasil Pengujian ke-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Kode Uji** | **Butir Uji** | **Teknik Pengujian** | **Hasil Pengujian** |
| OS1 | Developer baru melakukan pendaftaran untuk dapat masuk ke dalam sistem | Black Box | Diterima |
| OS2 | Developer melakukan login ke dalam sistem | Black Box | Diterima |
| OS3 | Admin dan developer memiliki antarmuka yang berbeda. | Black Box | Diterima |
| OS4 | Developer dapat membuat instance (server virtual) baru untuk pertama kali, | Black Box | Diterima |
| OS5 | Developer dapat membuat kembali instance (server virtual) baru. | Black Box | Diterima |
| OS6 | Developer dapat memilih sendiri kebutuhan spesifikasi prosessor, RAM, penyimpanan dan sistem Operasi yang akan di buat | Black Box | Diterima |
| OS7 | Developer dapat melihat list instance yang telah dibuat. | Black Box | Diterima |
| OS8 | Developer dapat mendownload SSH Key, untuk mengakses instance melalui SSH. | Black Box | Diterima |
| OS9 | Developer dapat mengakses instance melalui SSH dengan menggunakan SSHkey. | Black Box | Diterima |
| OS10 | Developer dapat melihat IP publik dari instance melalui web | Black Box | Diterima |
| OS11 | Developer dapat melakukan instalasi paket Apache, PHP dan MySQL (LAMPP) secara otomatis. | Black Box | Diterima |
| OS12 | Developer dapat mengganti kata sandi untuk masuk ke dalam sistem | Black Box | Diterima |
|  | Penggu tidak dapat masuk ke dashboard bila tidak melakukan login terlebih dahulu atau belum terdaftar dalam web admin | Black Box | Diterima |
| OS13 | Developer dapat keluar atau logout dari sistem pada menu yang sudah disediakan. | Black Box | Diterima |

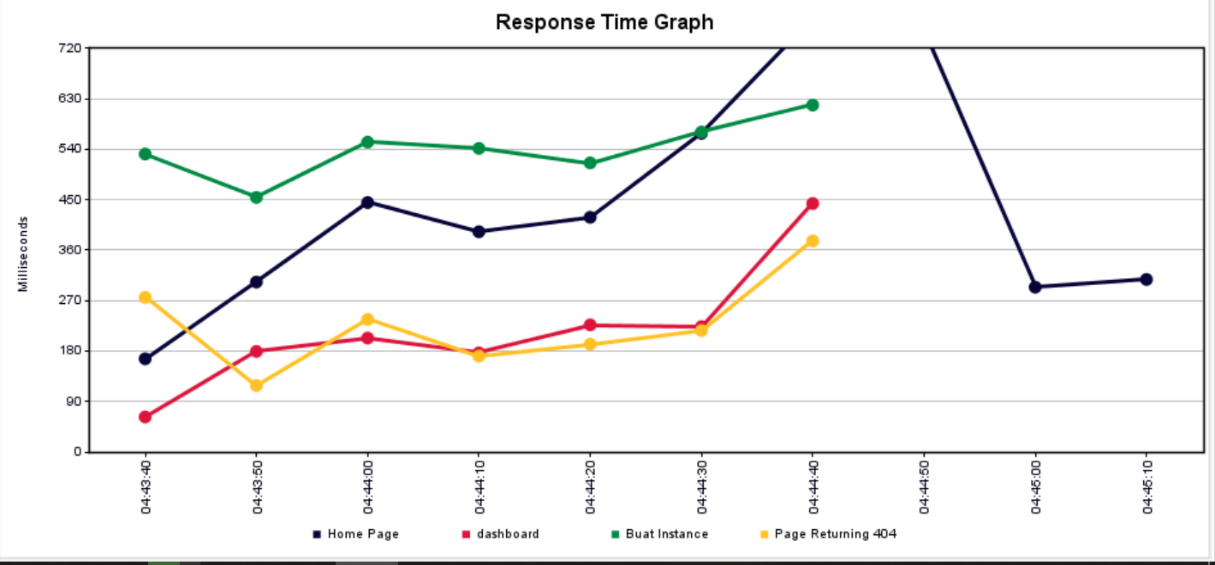
### Performance Testing

Pada pengujian *Performance Testing* merupakan lanjutan dari pengujian fungsionalitas sistem yang sudah dibuat. Performance Testing bertujuan memvalidasi “kecepatan” sistem yang dibuat. Kecepatan dalam hal ini peneliti melakukan pengujian dengan mengukur aspek waktu respon sistem. Peneliti melakukan pengujian kinerja dalam ruang lingkup yang mendekati ruang lingkup sebenarnya. Pengujian kinerja sistem ini dicapai dengan melakukan simulasi menggunakan perangkat lunak aplikasi *Jmeter*.

Pengujian dilakukan sebanyak 3x dengan menggunakan simulasi dengan menggunakan jmeter, pengujian pertama dilakukan simulasi 100 pengembang aplikasi web ( User of Threads ) dalam 1 detik ( Rump-up Period ) dan di ulang selama 10 kali ( Count ) dengan mendapatkan data uji sebanyak 1000 total sampel seperti pada gambar.



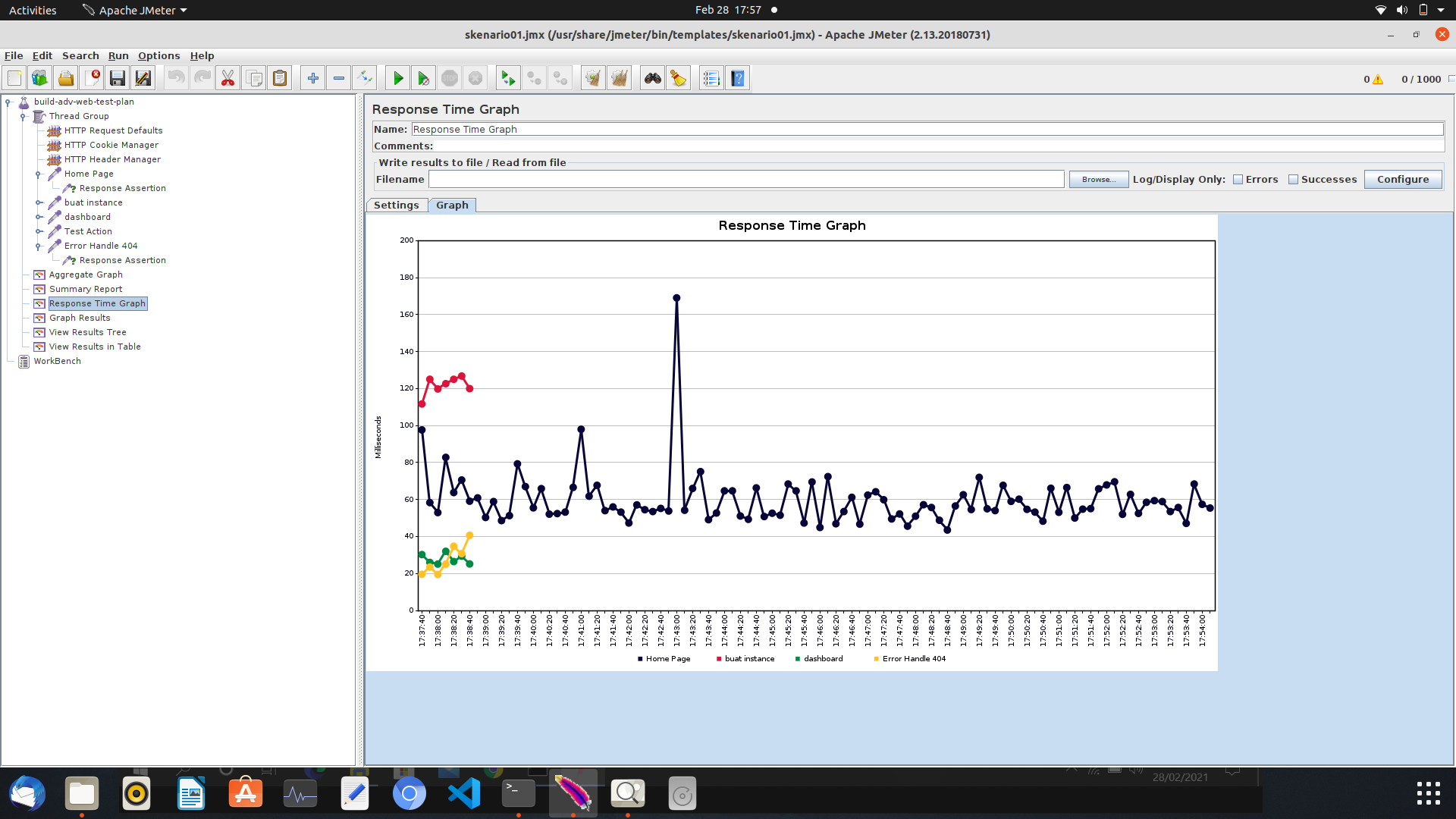
Gambar 4.31. Persiapan proses simulasi



Gambar 4.31. Grafik Hasil Pengujian ke-1

Tabel 4.10 Hasil Pengujian ke-1

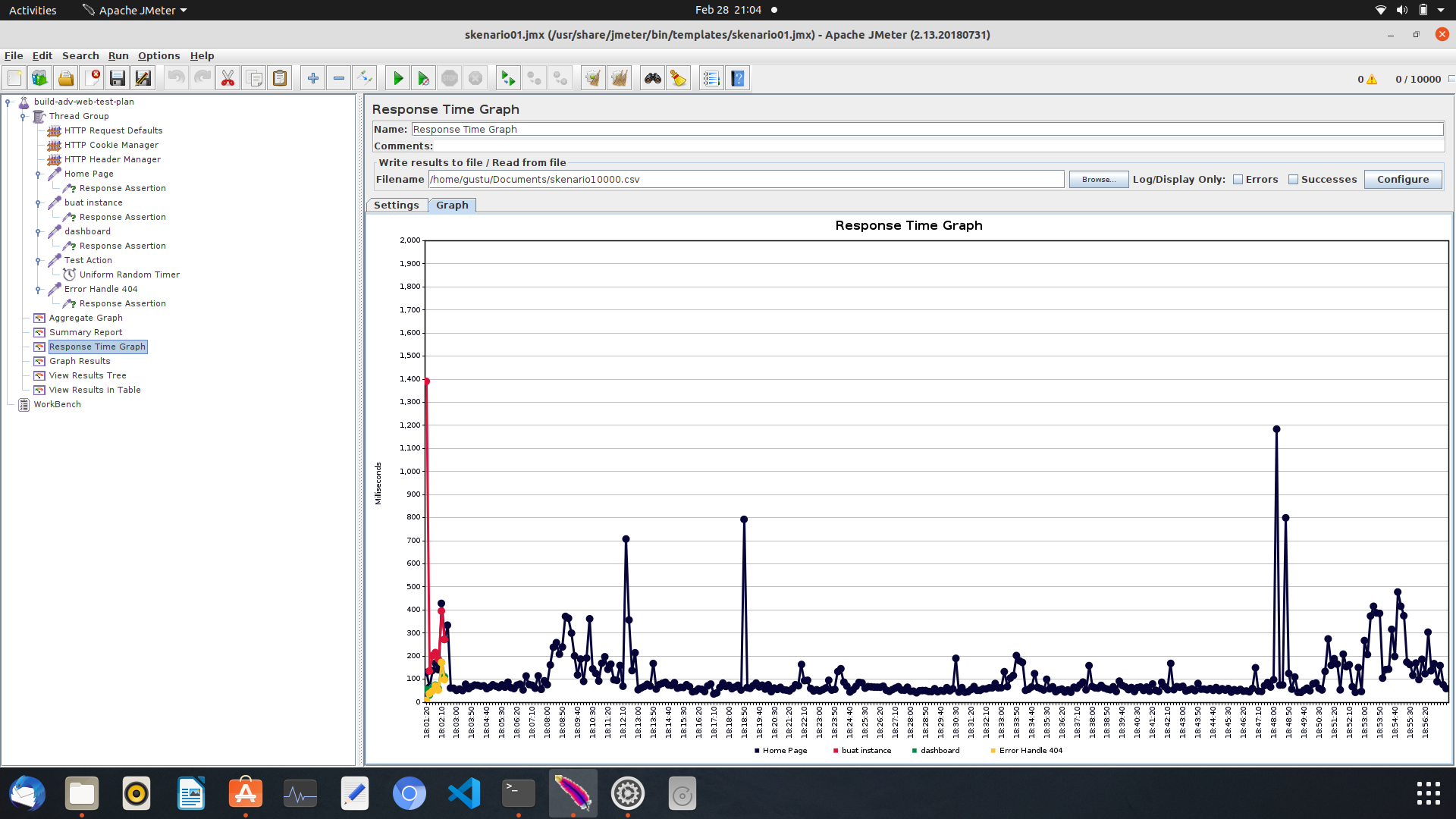
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Label | # Sample | Average | Min | Max | Std. Dev | Error % | Throughput | Received KB/sec | Send KB/sec | Avg. Bytes |
| Home Page | 314 | 463 | 49 | 2174 | 292.54 | 0.00% | 3.2/sec | 159.19 | 0.97 | 51151.0 |
| dashboard | 272 | 226 | 27 | 2971 | 262.31 | 0.00% | 4.5/sec | 54.81 | 1.42 | 12329.0 |
| Buat Instace | 268 | 539 | 241 | 1511 | 171.88 | 0.00% | 4.5/sec | 68.40 | 1.41 | 15626.0 |
| Page returning 404 | 261 | 208 | 20 | 2659 | 227.84 | 0.00% | 4.5/sec | 1.72 | 1.39 | 393.0 |
| total | 1115 | 264 | 20 | 2971 | 283.84 | 0.00% | 11.3/sec | 235.22 | 3.51 | 21284.7 |



Gambar 4.32. Grafik Hasil Pengujian ke-2

Tabel 4.11 Hasil Pengujian ke-2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Label | # Sample | Average | Min | Max | Std. Dev | Error % | Throughput | Received KB/sec | Send KB/sec | Avg. Bytes |
| Home Page | 314 | 463 | 49 | 2174 | 292.54 | 0.00% | 3.2/sec | 159.19 | 0.97 | 51151.0 |
| dashboard | 272 | 226 | 27 | 2971 | 262.31 | 0.00% | 4.5/sec | 54.81 | 1.42 | 12329.0 |
| Buat Instace | 268 | 539 | 241 | 1511 | 171.88 | 0.00% | 4.5/sec | 68.40 | 1.41 | 15626.0 |
| Page returning 404 | 261 | 208 | 20 | 2659 | 227.84 | 0.00% | 4.5/sec | 1.72 | 1.39 | 393.0 |
| total | 1115 | 264 | 20 | 2971 | 283.84 | 0.00% | 11.3/sec | 235.22 | 3.51 | 21284.7 |



Gambar 4.33. Grafik Hasil Pengujian ke-3

Tabel 4.12 Hasil Pengujian ke-3

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Label | # Sample | Average | Min | Max | Std. Dev | Error % | Throughput | Received KB/sec | Send KB/sec | Avg. Bytes |
| Home Page | 314 | 463 | 49 | 2174 | 292.54 | 0.00% | 3.2/sec | 159.19 | 0.97 | 51151.0 |
| dashboard | 272 | 226 | 27 | 2971 | 262.31 | 0.00% | 4.5/sec | 54.81 | 1.42 | 12329.0 |
| Buat Instace | 268 | 539 | 241 | 1511 | 171.88 | 0.00% | 4.5/sec | 68.40 | 1.41 | 15626.0 |
| Page returning 404 | 261 | 208 | 20 | 2659 | 227.84 | 0.00% | 4.5/sec | 1.72 | 1.39 | 393.0 |
| total | 1115 | 264 | 20 | 2971 | 283.84 | 0.00% | 11.3/sec | 235.22 | 3.51 | 21284.7 |

Gambar 4.34 Grafik Hasil Pengujian ke-4

Tabel 4.13 Hasil Pengujian ke-5

Gambar 4.35 Grafik Hasil Pengujian ke-5

## Analisa

Gambar 4.36 Grafik Akurasi

.

# BABI V SIMPULAN DAN SARAN

## Simpulan

Berdasarkan penelitian yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut :

* 1. Untuk mendeteksi *malware* dengan menggunakan K Nearest Neighbor, tahap pertama yang dilalui adalah *preprocessing* yaitu pencarian fitur berupa *Permissions* dan *API Calls* yang nantinya akan digunakan dalam proses klasifikasi. Selanjutnya proses klasifikasi dimana pertama – tama kita harus menentukan jarak dari setiap dataset menggunakan Binary Jackard Distance karena bentuk fitur yang digunakan berupa binary 1 dan 0, kemudian menentukan kelas sesuai dengan jumlah K yang *diinputkan*.
  2. Berdasarkan hasil pengujian sebanyak lima kali, didapatkan rata – rata akurasi yaitu k = 1 sebesar 82 %, k = 3 sebesar 85,8 %, k = 5 sebesar 85,8 %, k = 7 sebesar 84,4 %, k = 9 sebesar 83,2 %, k = 11 sebesar 83,4 %, k = 13 sebesar 84,2 %, k = 15 sebesar 84,4 %, k = 17 sebesar 85,8 %, k = 19 sebesar 85 %.
  3. Dari rata – rata pengujian tertinggi sebesar 85,8 % dapat disimpulkan nilai k terbaik yang dapat digunakan dalam sistem “Pendeteksian Malware Android dengan Pendekatan Permissions dan API Call menggunakan K-Nearest Neighbor (KNN)” yaitu k = 3 dikarenakan proses klasifikasi dengan k = 3 lebih cepat dibandingkan dengan k = 5 dan k = 17.

## Saran

Untuk pengembangan sistem lebih lanjut, penulis ingin memberikan beberapa saran sebagai berikut :

* + 1. Dapat menggunakan fitur lain dalam proses klasifikasi *malware* untuk mendapatkan nilai akurasi yang lebih baik.
    2. Dapat menggunakan algoritma lain atau gabungan dari beberapa algoritma untuk mendapatkan nilai akurasi yang lebih baik.

# DAFTAR PUSTAKA

Xia, Q., Lan, Y., & Xiao, L. (2015). A Heuristic Adaptive Threshold Algorithm on IaaS Clouds.

Anggeriana, H. (2011). *Cloud Computing.*

Ansible. (t.thn.). *ansible.com*. Diambil kembali dari http://docs.ansible.com/ansible/latest/index.html: http://docs.ansible.com/ansible/latest/index.html

Collings, T., & Kurt , W. (2015). *Duties of the System Administrator. In Red Hat Linux Networking System Administrator (chap. 1).*

Corporation, Exabyte. (2004). *The Basic Backup Guide.* cororado.

Doty, S. (2008). *Python Basics.*

Gerald D. Everett, R. M. (2007). *Software Testing: Testing Across the Entire Software Development Life Cycle 1st Edition.* Canada: IEEE Press.

Lei Xiaojiang, S. Y. (2013). The Design and Implementation of Resource Monitoring for Cloud Computing Service Platform.

Liyun Zuo, L. S. (2015). A Dynamic Self-adaptive Resource-Load Evaluation Method in Cloud Computing.

Masse, M. (2012). REST API. Dalam *REST API Design Rulebook* (hal. 5). America: O’Reilly Media, Inc.

Mulyana, E. (2017, 10 05). *Pengantar Openstack*. Diambil kembali dari https://eueung.gitbooks.io: https://eueung.gitbooks.io/buku-komunitas-sdn-rg/content/index.html

Nicole Ng, H. C. (2011). An Adaptive Threshold Method to Address Routing Issues in Delay-Tolerant Networks.

Saleh, Y. W. (2013). Adaptive Resource Management for Service Workflows in Cloud Environments.

Sosinsky, B. (2011). *Cloud Computing Bible.* Canada: Wiley Publishing, Inc.

# LAMPIRAN

**Data Permissions**