Kumpulan jurnaL Ilmu Komputer (KLIK)
Volume 10 N0.2 2023
ISSN: 2406-7857

Pengembangan *Service* Master Pada Aplikasi SiakadCloud (Studi Kasus PT. SEVIMA)

Aulia Ahmad Nabil[1], Didik Kurniawan[2], Astria Hijriani[3], Dwi Sakethi[4]
[1,2,3,4]Ilmu Komputer, Universitas Lampung
Jl. Prof. Dr. Ir. Sumantri Brojonegoro, No. 1 Kota Bandar Lampung, Lampung
email : didik.kurniawan@fmipa.unila.ac.id

Abstract

SiakadCloud adalah produk utama SEVIMA sebagai solusi manajemen akademik terintegrasi terlengkap, aman dan terbukti efektif dalam memfasilitasi manajemen pendidikan tinggi dan pelaporan PDDIKTI. SiakadCloud saat ini sedang dimigrasikan ke arsitektur microservice. Arsitektur microservice membuat aplikasi SiakadCloud harus dibagi menjadi bagian-bagian kecil. Penelitian ini bertujuan untuk mengembangkan sebuah layanan yang dapat digunakan untuk mengelola data master pada aplikasi SiakadCloud. Layanan ini disebut layanan master. Pengembangan dalam penelitian ini menggunakan metode waterfall. Pengembangan layanan dilakukan dengan menggunakan Framework Lumen. Penelitian ini menghasilkan sebuah layanan yang dapat mengelola data master untuk digunakan dan menjadi standar untuk pengembangan layanan lainnya.

Keywords*: API; Lumen; Microservice; PHPUnit; Swagger.*

Abstrak

SiakadCloud is SEVIMA's main product as an integrated academic management solution that is the most complete, secure and proven effective in facilitating higher education management and PDDIKTI reporting. SiakadCloud is currently being migrated to a microservice architecture. The microservice architecture makes the SiakadCloud application must be divided into small parts. This research is intended to develop a service that can be used to manage master data in the SiakadCloud application. The service is called the service master. The development in this research uses the waterfall method. Service development is made using the Lumen Framework. This research produces a service that can manage master data for use and become a standard for the development of other services.

Kata kunci*: API; Lumen; Microservice; PHPUnit; Swagger.*

1. PENDAHULUAN

PT Sentra Vidya Utama (SEVIMA) adalah perusahaan konsultan serta pengembang teknologi informasi yang miliki pengalaman > 15 tahun guna implementasikan sistem informasi di perguruan tinggi dan telah membantu lebih dari 400 perguruan tinggi. SiakadCloud adalah produk unggulan dari SEVIMA yang dibuat jadi solusi manajemen akademik terintegrasi yang terlengkap, aman, dan terbukti efektif guna permudah manajemen perguruan tinggi serta pelaporan PDDIKTI. Saat ini terdapat 174 universitas yang terdaftar sebagai pengguna SEVIMA SiakadCloud.

SiakadCloud dibangun menggunakan arsitektur monolitik. Arsitektur monolitik adalah sebuah arsitektur yang dalam pembuatan aplikasi semua komponennya terbentuk sebagai satu kesatuan kode yang tidak bisa terpisah satu dan yang lainnya. Dampak dari arsitektur monolitik ini yaitu penurunan performa ketika aplikasi menjadi semakin besar dan banyak yang mengakses, untuk menggunakan teknologi baru kode aplikasi harus ditulis ulang secara keseluruhan, jika terjadi error pada salah satu bagian maka dapat mempengaruhi keseluruhan aplikasi. Dampak-dampak tersebut akan sangat mempengaruhi baik di sisi end user maupun programmer di kemudian hari. Contoh kasus yang sudah terjadi pada SEVIMA SiakadCloud ini yaitu ketika waktu pengisian KRS, *traffic* menjadi sangat tinggi, sehingga membuat seluruh aplikasi menjadi down. Hal ini mempengaruhi mahasiswa lainnya yang hanya sekedar ingin melihat nilai ataupun jadwal kuliah menjadi tidak bisa.

SiakadCloud saat ini akan dilakukan migrasi ke arsitektur *microservice*. Arsitektur *microservice* membuat aplikasi SiakadCloud harus dibagi menjadi bagian-bagian kecil atau *service* yang mana setiap *service* memiliki skema database sendiri. Setiap *service* harus berhubungan satu sama lain. Komunikasi yang digunakan antar *service* memerlukan protokol yang ringan agar komunikasi antar *service* bisa berjalan dengan baik.

Berdasarkan uraian di atas, aplikasi SiakadCloud 2.0 akan dibagi menjadi beberapa *service*. SiakadCloud 2.0 belum mempunyai sebuah *service* yang dapat mengelola data-data master dalam aplikasi, maka pada studi ini akan dikembangkan sebuah *service* yang dapat digunakan untuk memanajemen data-data master yang bisa dipakai *service* lainnya dan juga menjadi standarisasi pengembangan *service* selanjutnya. *Service* tersebut dinamakan dengan *service* master. *Service* master ini nantinya paling sering digunakan oleh *service* lain dalam mendapatkan data master, sehingga *service* master dikembangkan terlebih dahulu dibandingkan dengan *service* lainnya. *Service* master ini akan dibuat dengan menggunakan *Framework* Lumen karena SDM pada PT SEVIMA banyak yang berfokus pada bahasa pemrograman PHP.

2. METODOLOGI PENELITIAN

Waktu penelitian dilakukan pada Semester Genap Tahun Ajaran 2021/2022. Studi ini dilaksanakan di PT SEVIMA yang beralamat di Medokan Asri Tengah MA-2 Blok Q No 16, Medokan Ayu, Kec. Rungkut, Kota SBY, Jawa.Timur.

Studi ini dikembangkan dengan memakai bahasa pemrograman PHP [1] dengan *Framework* Lumen [2], untuk basis data sistem menggunakan Database PostgreSQL [3], di studi ini metode yang dipakai yaitu metode "waterfall" yang menekankan di fase berurutan dan sistematis [4], dalam penulisan kode program menggunakan tools Visual Studio Code, sistem yang dibangun diuji dengan pengujian *black box testing* yang diuji secara segi spesifikasi fungsional perangkat lunak [5]. Data yang dipakai di studi ini ialah data yang didapat dari pencarian data serta informasi lewat dokumen-dokumen, baik tertulis, gambar, foto, atau dokumen elektronik yang berkaitan dengan REST API [6] maupun micro*service* yang dapat mendukung dalam proses pengembangan sistem. Selanjutnya data juga diperoleh dari Observasi yang dilakukan dengan cara mempelajari aplikasi yang telah dibuat sebelumnya seperti arsitektur, database, struktur, fitur-fitur yang ada dan juga gate yang sering diakses oleh pengguna.

Ada tahapan guna melaksanakan studi ini, yakni: kerangka penelitian, metode pembuatan sistem, analisis kebutuhan sistem, penulisan kode program serta pengujian sistem.

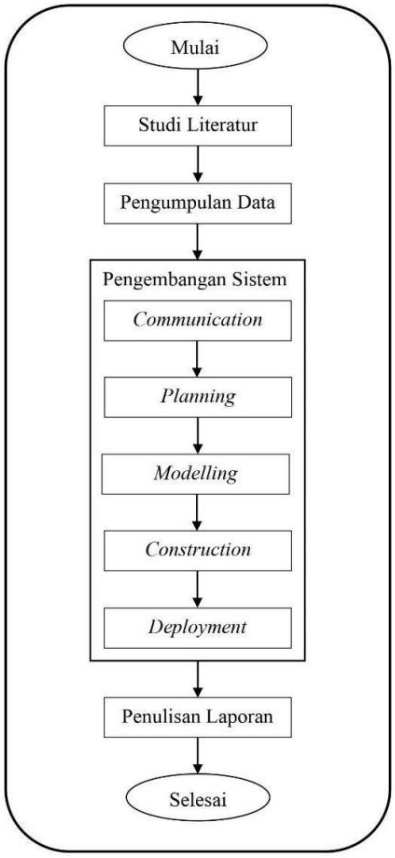
2.1. Kerangka Penelitian

Studi ini dilakukan awalnya dari permasalahan yang muncul pada aplikasi SiakadCloud versi sekarang yang penggunanya sudah mencapai angka 213.379.484 user/tahun ditambah aplikasi nya sudah cukup besar dan mempunyai banyak fitur di dalamnya yang mana aplikasi tersebut dibangun dengan menggunakan arsitektur monolitik. Hal ini mengakibatkan *server down* pada beberapa waktu dimana banyak user yang menggunakan SiakadCloud secara bersamaan. Kasus yang sering terjadi adalah ketika perkuliahan di kampus memasuki masa pengisian KRS. Ketika semua mahasiswa mengakses fitur KRS secara bersama-sama, maka akan terjadi server down, karena terlalu banyak load ataupun *request* yang dilakukan dalam satu waktu. Hal ini menyulitkan bagi mahasiswa lain yang hanya ingin sekedar melihat nilai ataupun data-data mereka yang terdapat di dalam aplikasi.

2.2. Metode Pembuatan Sistem

Di tahapan ini dilaksanakan metode pembuatan sistem. Metode pembuatan sistem di studi ini menggunakan metode "waterfall", yang melakukan pendekatan dengan sistematis dimulai dari tahap:

Communication, Planning, Modeling, Construction dan *Deployment*. Tahap metode ini ditampilkan pada Gambar 1.



Gambar 1 Diagram Alir Penelitian

a. Communication

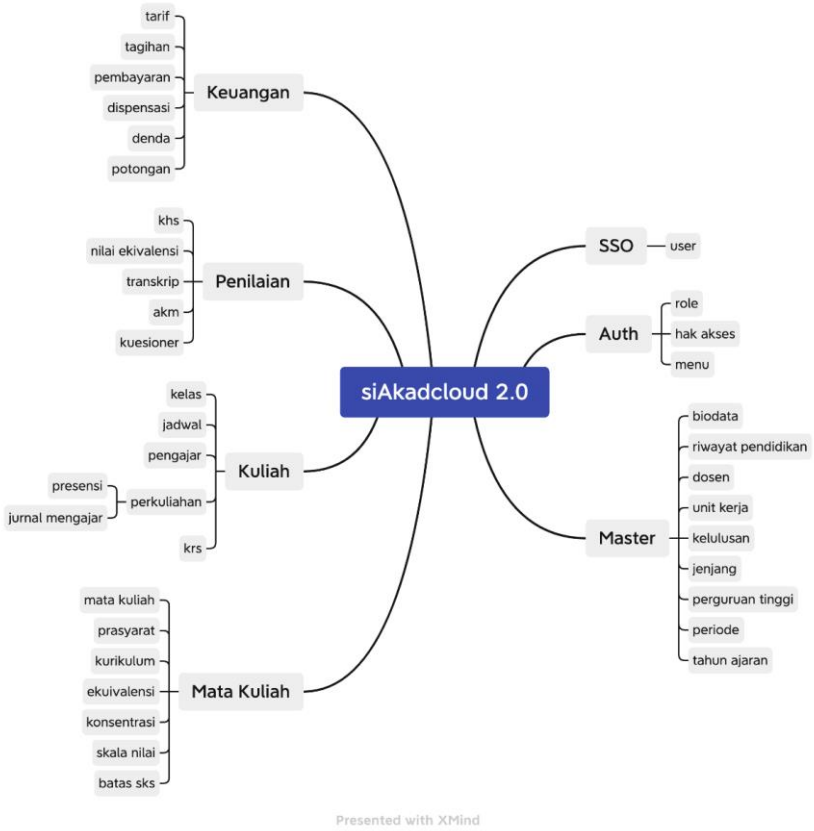
Tahap ini dilakukan dengan cara berdiskusi dengan programmer sebelumnya dan juga menghimpun data-data tambahan baik yang ada di dalam jurnal, artikel, atau dari internet. Dari tahapan communication didapatkan hasil yaitu:

- Analisis masalah

  Pada penelitian ini permasalahan yang ada pada aplikasi SiakadCloud versi sekarang adalah penggunanya yang sudah mencapai angka 213.379.484 user/tahun ditambah aplikasi nya sudah cukup besar dan mempunyai banyak fitur di dalamnya yang mana aplikasi tersebut dibangun dengan menggunakan arsitektur monolitik. Hal ini mengakibatkan server down pada beberapa waktu dimana banyak user yang menggunakan SiakadCloud secara bersamaan.

- Analisis Kebutuhan Sistem

  Dilaksanakan guna selesaikan masalah yang sudah dijelaskan sebelumnya. Untuk mengatasi masalah tersebut maka akan SiakadCloud versi sekarang atau SiakadCloud 2.0 perlu dilakukan refactoring arsitekturnya dari yang sebelumnya menggunakan arsitektur monolitik diubah ke arsitektur micro*service*. Pembagian *service* pada SiakadCloud 2.0 ditunjukkan di Gambar 2.



Gambar 2 Pembagian *service* pada SiakadCloud 2.0.

b. *Planning*

Planning pada penelitian ini jabarkan estimasi tugas-tugas teknis yang akan dijalankan, produk kerja yang diharapkan, penjadwalan kerja yang akan dilaksanakan, dan tracking tahap pengerjaan sistem. Di tahap ini dilaksanakan perencanaan yang dimulai dari minggu pertama September hingga minggu ke-4 Januari.

c. *Modeling*

Modeling sendiri menjadi gambaran kasar sistem sebelum diimplementasi pada sebuah bahasa pemrograman. Modeling ini berfokus pada perancangan use case diagram, activity diagram, entity relationship

diagram. Tujuan dalam tahap modelling ini adalah guna lebih pahami bayangan besar dari apa yang akan dijalankan.

d. *Construction*

Setelah dilakukannya perancangan atau modeling, maka selanjutnya dilakukan lah penerjemahan bentuk desain atau model yang sudah dibuat menjadi bentuk kode atau bahasa yang dapat dimengerti oleh mesin. Setelah melakukan koding, maka dilakukan pengujian dengan metode *black box testing*.

e. *Deployment*

Deployment merupakan tahapan membuat *service* yang sudah dikembangkan menjadi live atau biasa disebut dengan hosting, sehingga memudahkan programmer *service* lain dalam penggunaanya jika dibutuhkan dalam proses pengembanganya.

2.3. Analisis Kebutuhan Sistem

Analisis Kebutuhan Fungsional pengembangan *service* master pada aplikasi SiakadCloud 2.0

a. *Service* dapat memanajemen data-data master.
b. *Service* dapat mengirimkan data master ke *service* lain.
c. Terdapat pengecekan hak akses.

Analisis Kebutuhan Non Fungsional pengembangan *service* master pada aplikasi SiakadCloud 2.0

a. *Service* memberikan respon yang cepat dan tepat.
b. Terdapat dokumentasi api yang dapat memudahkan frontend engineer.
c. Memiliki tingkat keamanan yang baik.

2.4. Desain Sistem

Di tahapan ini dibentuk guna permudah dalam pengembangan.

a. Use Case Diagram
b. Activity Diagram
c. Entity Relationship Diagram (ER Diagram)

2.5. Penulisan Kode Program

Penulisan kode dalam penelitian ini memakai bahasa pemrograman PHP juga memakai PostgreSQL sebagai *database* sistem [3].

2.6. Pengujian Sistem

Pengujian sistem di studi ini memakai "*black box testing*" yang dilaksanakan dengan mencoba semua fungsi guna tahu kesesuaian pada

spesifikasi yang diperlukan. Metode ini bisa diketahui bila fungsionalitas masih bisa terima masukan data yang tidak diharapkan maka sebabkan data yang disimpan kurang valid [5].

3. HASIL DAN PEMBAHASAN

3.1. Hasil

Hasil studi memperlihatkan *service* master pada aplikasi SiakadCloud 2.0. *Resource* pada *service* ini akan digunakan oleh *service* lain dalam penggunaan data-data master. Terdapat 236 endpoint yang dapat digunakan dan setiap endpoint tersebut sudah ada di dalam dokumentasi API yang telah dibuat menggunakan Swagger.

3.1.1. Struktur *Framework*

Penelitian ini menghasilkan *service* master pada aplikasi SiakadCloud 2.0. *Resource* pada *service* ini akan digunakan oleh *service* lain dalam penggunaan data-data master. Berdasarkan hasil perancangan sistem yang sudah dibentuk, sistem ini diterapkan menggunakan *Framework* Lumen dan database PostgreSQL. *Service* ini mengimplementasikan gaya arsitektur REST dan menggunakan JSON sebagai bentuk datanya. Struktur *Framework* yang diimplementasikan pada *service* master terlihat di Gambar 4.

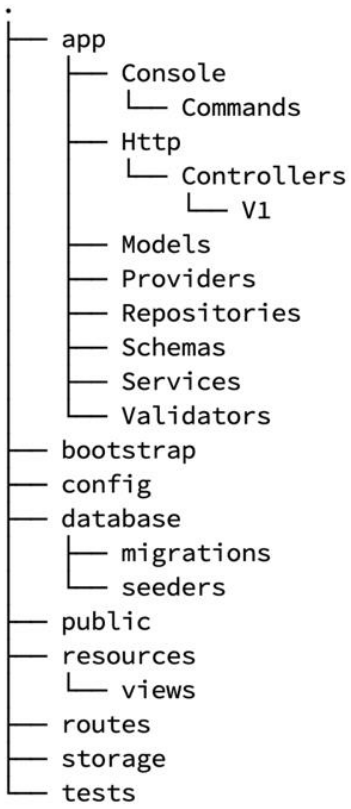
3.1.2. Struktur API

https://{host}/{version}/{resource}?query=string

Gambar 3 Desain URL pada *service* master.

Pada penelitian ini, terdapat dua pola klasifikasi *resource* berdasarkan *path* dan *query string* dalam URL serta *Request method* dalam HTTP *request*. Klasifikasi *resource* dibagi jadi dua pola:

a. Endpoint /{resource}?query=string dipakai guna dapatkan daftar dan menambahkan data anggota dari *resource* tersebut.
b. Endpoint /{resource}/{id}?query=string dipakai guna mendapatkan detail, mengubah dan hapus data dari anggota *resource* tersebut berdasarkan id.



Gambar 4 Struktur *Framework*.

Sementara klasifikasi sesuai *Request method* dibagi empat jenis yakni:

a. *Request method* GET dipakai guna dapatkan list dan detail anggota pada sebuah resource.
b. *Request method* POST dipakai guna menambahkan data anggota dari sebuah resource.
c. *Request method* PUT dipakai guna ubah data anggota dari sebuah resource.
d. *Request method* DELETE dipakai guna hapus data anggota dari sebuah resource.

Lalu klasifikasi dapat dipetakan pada resourcenya. Sebagian daftar API yang menjabarkan kaitan diantara *resource* dengan klasifikasinya pada *path* dan *query string* serta *Request method* terlihat di Tabel 1.

Tabel 1 Daftar API

***Request* Method**	**URI**	**Keterangan**
GET	/agama	Menampilkan daftar agama
POST	/agama	Menambahkan data agama
GET	/agama/{id}	Menampilkan detail agama
PUT	/agama/{id}	Mengubah data agama
DELETE	/agama/{id}	Menghapus data agama
GET	/almamater	Menampilkan daftar almamater

Pada *environment* sandbox, URL yang digunakan untuk mengakses *resource* pada *service* master adalah https://master.test/v1/{resource}?query=string. Setiap *Request* yang dikirim ke server harus memuat header yang berisikan authentication dengan jenis bearer token atau JWT.

3.1.3. Skenario Pengujian

Pengujian sistem di studi ini yaitu memakai metode "*black box testing*" dengan jenis *Equivalence Partitioning* yang merupakan metode pengujian pada "*black box testing*" dengan menggunakan skenario uji, hasil yang diharapkan dan hasil pengujian guna lihat apakah sistem berjalan sesuai atau tidak [7]. Tools yang akan digunakan dalam pengujian ini yaitu PHPUnit. Tabel 2 merupakan skenario dalam pengujian *service* master.

Tabel 2 Skenario pengujian *Equivalence Partitioning* pada *service* master

Kode Uji	**Daftar Pengujian**	**Kasus Uji**	**Hasil yang Diharapkan**
Path	Validasi *path*	*Request method* GET pada endpoint /mahasiswa	Sukses mendapatkan list mahasiswa
		Request method GET pada endpoint /mahasiswa/{id}	Sukses mendapatkan detail mahasiswa
		Request method GET pada endpoint /mahasiswa/{wrong_id}	Error dengan status kode 404 - not found
		Request method GET pada endpoint /mahasiswa/{id}/detail	Error dengan status kode 404 - not found
		Request method GET pada endpoint /salah	Error dengan status kode 500 - internal server error
Request method	Validasi *Request* method	*Request method* GET pada endpoint /mahasiswa	Sukses mendapatkan list mahasiswa
		Request method GET pada endpoint /mahasiswa/{id}	Sukses mendapatkan detail mahasiswa
		Request method POST pada endpoint /mahasiswa	Sukses menambahkan mahasiswa
		Request method PUT pada endpoint /mahasiswa/{id}	Sukses mengubah data mahasiswa
		Request method DELETE pada	Sukses menghapus

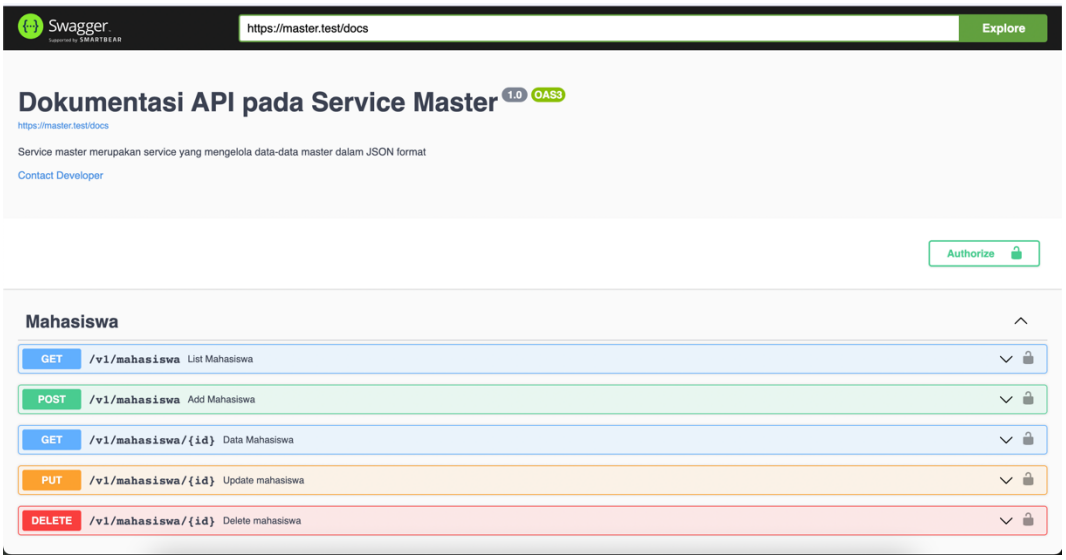
Kode Uji	Daftar Pengujian	Kasus Uji	Hasil yang Diharapkan
		endpoint /mahasiswa/{id}	mahasiswa
Request body	Validasi *Request* body	*Request method* POST pada endpoint /mahasiswa tanpa body	Error dengan status kode 400 - bad *request*
Security	Validasi Token	*Request method* GET pada endpoint /mahasiswa dengan invalid token	Error dengan status kode 401 - Unauthorized

3.2. Implementasi

Berdasarkan hasil perancangan sistem yang sudah dibentuk, sistem ini lalu diterapkan menggunakan *Framework* Lumen serta database PostgreSQL. *Service* ini mengimplementasikan gaya arsitektur REST dan menggunakan JSON sebagai bentuk datanya. Dokumentasi dari hasil penelitian dapat dilihat sebagai berikut:

3.2.1. Dokumentasi API pada *Service* Master

Dokumentasi ialah hal terpenting yang perlu ada di pengembangan API. Pada penelitian ini terdapat dokumentasi API pada *service* master. Dokumentasi dibuat dengan menggunakan Swagger. Gambar 5 merupakan dokumentasi API dari *service* master.



Gambar 5 Dokumentasi API pada *service* master

3.2.2. Dokumentasi API menampilkan data mahasiswa

Dokumentasi API menampilkan daftar mahasiswa berisikan informasi *Request* apa saja yang harus dikirimkan ke server dan juga *response* yang dikembalikan oleh server. *Request* yang harus dikirim yaitu *header* yang berisikan token JWT. *Response* yang didapatkan yaitu berupa daftar

mahasiswa, info mengenai *pagination* serta *debug* mengenai query yang dijalankan ketika *endpoint* tersebut dipanggil.

3.3. Hasil Pengujian

Pengujian sistem di studi ini dibagi dua tahap, yaitu *Unit test* dengan *Equivalence Partitioning* dan *performance test* dengan jenis stress test. Pengujian dilakukan pada tanggal 22 maret 2022. Sistem diuji langsung oleh pengembang dan tidak melibatkan user atau pengguna karena tidak ada GUI.

3.3.1. Unit Test

Unit test dilakukan untuk memastikan fungsi atau prosedur yang ditulis dapat berfungsi sesuai yang diinginkan. *Unit test* dilaksanakan manual dengan memakai tools Insomnia, tapi ada juga pengujian otomatis dilaksanakan memakai tools PHPUnit.

3.3.2. Performance Test

Performance test dilakukan untuk memastikan sistem atau aplikasi yang akan digunakan dapat memenuhi tingkat performa tertentu yang diminta oleh pengguna. Pengujian performa dilakukan dengan jenis *stress testing* dengan menggunakan tools k6. Berikut adalah hasil pengujian *stress testing* terlihat di Gambar 8.

```
checks.........................: 99.91% ✓ 45547     ✗ 41
data_received..................: 53 MB  69 kB/s
data_sent......................: 2.4 MB 3.1 kB/s
group_duration.................: avg=2m52s    min=1m0s     med=2m55s max=4m32s  p(90)=4m11s p(95)=4m12s
http_req_blocked...............: avg=1.41ms   min=0s       med=1µs   max=4.33s  p(90)=1µs   p(95)=2µs
http_req_connecting............: avg=771.54µs min=0s       med=0s    max=1.91s  p(90)=0s    p(95)=0s
http_req_duration..............: avg=3.11s    min=265.95ms med=2.58s max=3m18s  p(90)=4.83s p(95)=4.94s
  { expected_response:true }...: avg=2.94s    min=265.95ms med=2.58s max=1m58s  p(90)=4.83s p(95)=4.93s
http_req_failed................: 0.08%  ✓ 41        ✗ 45548
http_req_receiving.............: avg=1.13ms   min=0s       med=73µs  max=2.7s   p(90)=747µs p(95)=2ms
http_req_sending...............: avg=807.16µs min=13µs     med=141µs max=29.52s p(90)=263µs p(95)=304µs
http_req_tls_handshaking.......: avg=641.79µs min=0s       med=0s    max=4.07s  p(90)=0s    p(95)=0s
http_req_waiting...............: avg=3.11s    min=265.72ms med=2.58s max=3m18s  p(90)=4.83s p(95)=4.93s
http_reqs......................: 45589  59.254958/s
iteration_duration.............: avg=2m52s    min=1m0s     med=2m55s max=4m32s  p(90)=4m11s p(95)=4m12s
iterations.....................: 719    0.934531/s
vus............................: 1      min=1       max=480
vus_max........................: 480    min=480     max=480
```

Gambar 8 Hasil Pengujian *stress testing* menggunakan tools k6

Gambar 8 merupakan hasil pengujian *stress testing* menunjukkan bahwa terdapat total http *Request* sebanyak 45589 dengan rata-rata per detiknya sebanyak 59 *request*. Dari total tersebut *Request* yang berhasil sebanyak 45548 dan total yang gagal sebanyak 41 *request*. Waktu yang dibutuhkan dalam setiap *Request* rata-rata selama 3.11 detik dengan waktu terlama yaitu 198 detik.

4. SIMPULAN

Penelitian ini berhasil mengembangkan *service* master pada aplikasi SiakadCloud 2.0 dengan menggunakan *Framework* Lumen dan basis data PostgreSQL. *Service* master merupakan sebuah *service* yang dapat mengelola data-data master dalam aplikasi. Data pada *service* master akan digunakan oleh *service* lainnya. Terdapat 236 endpoint yang dapat digunakan dan setiap endpoint tersebut sudah ada di dalam dokumentasi API yang telah dibuat menggunakan Swagger. Hasil pengujian dari *Unit test* menunjukkan bahwa terdapat 518 test case yang sudah sesuai dan dari *performance test* menunjukkan bahwa terdapat 45548 *Request* yang berhasil dari total 45589 *Request* yang dikirim, dimana setiap *Request* dapat mencapai paling maksimal 480 VUs dan test dijalankan dengan durasi waktu 10 menit.

DAFTAR PUSTAKA

[1]. Haviluddin, A. T. Haryono and D. Rahmawati, Aplikasi Program PHP dan MySQL, Kalimantan Timur: Mulawarman University Press, 2016.

[2]. F. Surahman, S. H. A. Ikhsan and F. S. F. Kusumah, "Rancang Bangun Web *Service* untuk Transaksi Data pada Aplikasi Jasa dengan Metode REST," in Seminar Nasional Teknik Informatika, Bogor, 2018.

[3]. S. Munawaroh, "Mengeksplorasi Database PostgreSQL dengan PgAdmin III," Jurnal Teknologi Informatika DINAMIK , vol. X, no. 2, pp. 103-107, 2005.

[4]. R. Pressman and B. Maxim, Software Engineering: A Practitioner's Approach 9th Edition, Kybernates, 2020.

[5]. W. N. Cholifah, Yulianingsih and S. M. Sagita, "Pengujian *Black box testing* pada Aplikasi Action & Strategy Berbasis Android dengan Teknologi Phonegap," Jurnal String, vol. 3, no. 2, pp. 206-210, 2018.

[6]. M. A. K. Perdana, "Pengembangan REST API Layanan Penyimpangan menggunakan Metode Rapid Application Development," Jurnal Nasional Informatika dan Teknologi Jaringan, vol. 3, no. 1, pp. 100-104, 2018.

[7]. B. A. Pranata, A. Hijriani and A. Junaidi, "Perancangan Application Programming Interface (API) Berbasis WEB Menggunakan Gaya Arsitektur Representational State Transfer (REST) untuk Pengembangan Sistem Informasi Administrasi Pasien Klinik Perawatan Kulit," Jurnal Komputasi, vol. 6, no. 1, pp. 33-42, 2018.