Laman web jurnal: http://ejournal.stikom-db.ac.id/index.php/processor

Processor: Jurnal Ilmiah Sistem Informasi, Teknologi Informasi dan Sistem Komputer

P-ISSN: 1907-6738 | E-ISSN: 2528-0082

Optimalisasi Proses Sinkronisasi Data Akademik dan Web Services PDDIKTI Menggunakan Fitur Queues Pada Framework Laravel

Abdul Rahim[1], Mulyadi[2]

[1,2]STIKOM Dinamika Bangsa, Jl. Jendral Sudirman, Kel, Thehok, Kec. Jambi Selatan, Jambi, 36138, Indonesia.

ABSTRACT

Previous research has succeeded in mapping the needs of tables and designing applications for database synchronization between STIKOM Dinamika Bangsa Jambi academic information systems and higher education data base feeders (pddikti). The functions possessed by this application include data processing for new students, student value data, and student activity data. This database synchronization application has been used and implemented by STIKOM Dinamika Bangsa as academic reporting. Based on the results of the implementation, the authors found the main problem is that the application cannot be run through a web browser because the process required in synchronization is very long (according to the amount of data), resulting in a web browser and web server relationship being not responding, so this application must be run using php-cli or command line interface. The optimization carried out in this study is the optimization of the user interface and the process of synchronizing academic data to the secondary, using database-based Queue on the Laravel (Laravel Queues) framework. By using the Queue feature and application supervisor, the user does not need to wait for the synchronization process to complete, because the synchronization process has been stored in the queue and run as background services.

Keywords: web services, feeder dikti, laravel, queues, optimization of feeders

ABSTRAK

Pada penelitian sebelumnya telah berhasil memetakan kebutuhan tabel dan merancang aplikasi untuk sinkronisasi database antara sistem informasi akademik STIKOM Dinamika Bangsa Jambi dengan feeder pangkalan data pendidikan tinggi (pddikti). Fungsi yang dimiliki aplikasi ini meliputi pengolahan data mahasiswa baru, data nilai mahasiswa, dan data aktifitas mahasiswa. Aplikasi sinkronisasi database ini sudah digunakan dan diimplementasikan oleh STIKOM Dinamika Bangsa sebagai pelaporan akademik. Berdasarkan hasil implementasi tersebut, penulis menemukan permasalahan utama yaitu aplikasi ini tidak bisa dijalankan melalui web browser karena proses yang dibutuhkan dalam sinkronisasi sangat lama (sesuai dengan jumlah data data), sehingga mengakibatkan hubungan web browser dan web server menjadi timeout atau web browser menjadi not responding, sehingga aplikasi ini harus dijalankan menggunakan php-cli atau command line interface. Optimalisasi yang dilakukan dalam penelitian ini adalah optimalisasi user interface dan proses sinkronisasi data akademik ke pddikti, menggunakan Queue berbasis database pada framework Laravel (Laravel Queues). Dengan menggunakan fitur Queue dan aplikasi supervisor maka pengguna tidak perlu menunggu proses sinkronisasi selesai, karena proses sinkronisasi telah disimpan di queue dan dijalankan sebagai background services.

Kata Kunci: web services, feeder dikti, laravel, queues, optimalisasi feeder

1. PENDAHULUAN

Perkembangan bahasa pemrograman web merupakan salah satu perkembangan dalam dunia teknologi informasi yang sangat besar pengaruhnya. Hal ini dapat dilihat dari semakin banyaknya situs-situs yang dibangung dengan bahasa pemrograman berbasis web. Laravel merupakan salah satu framework dengan bahasa pemrograman web php (*Hypertext Preprocessor*), yang dapat digunakan untuk membuat aplikasi berbasis web. Laravel hadir sebagai *platform* web development yang bersifat *open source*. Yang menarik dari Laravel adalah sintaksnya yang ekspresif dan elegan yang dirancang khusus untuk memudahkan dan mempercepat proses web development[1].

Sistem informasi akademik sebagai pendukung proses administrasi pendidikan pada sebuah perguruan tinggi merupakan suatu kepentingan yang mutlak saat ini. Seluruh data yang berhubungan dengan kegiatan akademik akan diolah melalui sistem informasi akademik dan disimpan di database. Seiring dengan kebijakan-kebijakan baru dari Kementrian Riset, Teknologi dan Pendidikan Tinggi tentang pelaporan hasil pembelajaran dari instansi ke DIKTI, mengharuskan instansi mengirim data akademik ke DIKTI melalui mekanisme web service atau *feeder* [2].

Penelitian ini merupakan lanjutan dari penelitian sebelumnya yaitu aplikasi sinkronisasi database antara sistem informasi akademik stikom dinamika bangsa jambi dengan feeder pangkalan data pendidikan tinggi (pddikti) [3]. Pada penelitian sebelumnya telah berhasil memetakan kebutuhan tabel dan merancang aplikasi untuk sinkronisasi. Pada penelitian sebelumnya, disimpulkan bahwa Penelitian yang telah dilakukan menghasilkan aplikasi sinkronisasi database antara sistem informasi akademik stikom dinamika bangsa jambi dengan *feeder* pangkalan data pendidikan tinggi (pddikti). Fungsi yang dimiliki aplikasi ini meliputi pengolahan data mahasiswa baru, data nilai mahasiswa, dan data aktifitas mahasiswa.

Aplikasi sinkronisasi database ini sudah digunakan dan diimplementasikan oleh STIKOM Dinamika Bangsa sebagai pelaporan akademik. Berdasarkan hasil implementasi tersebut, penulis menemukan masalah pada aplikasi ini yaitu aplikasi tidak bisa dijalankan melalui web browser karena proses yang dibutuhkan dalam sinkronisasi sangat lama, sehingga mengakibatkan web server menjadi *timeout* atau web browser menjadi *not responding*, sehingga aplikasi ini harus dijalankan menggunakan php-cli atau

command line interface. Solusi atas permasalahan ini adalah merancang *queue* data untuk melakukan proses sinkronisasi dan mengoptimalkan *user interface* sehingga aplikasi sinkronisasi dapat digunakan melalui web browser.

2. TINJAUAN PUSTAKA

2.1. Aplikasi Sinkronisasi Database Akademik Dan Feeder Dikti

Aplikasi sinkronisasi database akademik dan *feeder* dikti merupakan aplikasi yang dikembangkan oleh STIKOM Dinamika bangsa untuk melakukan pelaporan data akademik. Berdasarkan hasil pengujian pada penelitian yang dibuat oleh abdul rahim dan agus siswanto dikatakan bahwa :

1. Aplikasi yang dibangun memiliki fungsi dalam mengambil data dari database sistem informasi akademik stikom dinamika bangsa jambi dan diimport ke database feeder pangkalan data pendidikan tinggi (pddikti).
2. Beberapa fungsi pengambilan data yang dapat dilakukan di aplikasi ini adalah data mahasiswa baru, data nilai mahasiswa dan data aktifitas mahasiswa.
3. Aplikasi ini dapat memberikan kemudahan dalam penginputan data khusunya untuk operator pdpt yang ada di stikom dinamika bangsa jambi.
4. Proses pemindahan data dari database sistem informasi akademik ke database feeder pangkalan data pendidikan tinggi (pddikti) masih tergolong lambat.
5. Pengujian yang dilakukan pada penelitian sebelumnya menggunakan sampel data, dimana data yang digunakan tidak sampai 300 data.

Pada implementasinya, data mahasiswa yang berupa KRS dan Nilai yang harus dilaporan pada Ganjil 2017 yaitu sebanyak 20158 data, dan ini belum termasuk data aktivitas mahasiswa. Sehingga pada saat melakukan sinkronisasi menggunakan web browser, maka terjadi *timeout* dan web browser menjadi *crash* atau *not responding* hal ini terjadi dikarenakan adanya proses *request* yang lama.[3]

2.2. Laravel

Laravel merupakan framework PHP yang menekankan pada kesederhanaan dan fleksibilitas pada desainnya. Laravel dirilis dibawah lisensi MIT dengan sumber kode yang disediakan di *Github*. Sama seperti framework PHP lainnya, Laravel dibangun dengan basis MVC (*Model-View-Controller*). Laravel dilengkapi *command line tool* yang bernama "Artisan" yang bisa digunakan untuk *packaging bundle* dan *instalasi bundle*. [4]

2.3. Queues

Salah satu fitur yang terdapat di laravel yaitu laravel queues. Dalam situs resminya disebutkan bahwa "Laravel queues provide a unified API across a variety of different queue backends, such as Beanstalk, Amazon SQS, Redis, or even a relational database. Queues allow you to defer the processing of a time consuming task, such as sending an email, until a later time. Deferring these time consuming tasks drastically speeds up web requests to your application." [5]

Berdasarkan penjelasan diatas dapat disimpulkan bahwa, laravel queues merupakan fitur antrian data yang mendukung beanstalk, amazon ataupun database relasional. Antrian data memungkinkan untuk menunda proses yang terjadi saat request, kemudian proses akan dilakukan di background sehingga meringankan kerja web browser.

2.4. Laravel Job Status

Laravel job status merupakan paket atau library yang di publikasikan di github dengan nama akun imTigger. Di dokumentasi situs ini dijelaskan bahwa Laravel Job Status adalah "Laravel package to add ability to track Job progress, status and result dispatched to Queue." [6]. Laravel job status merupakan paket tambahan di laravel, dimana cara kerja dari laravel job status adalah dengan mengcopy isi dari job yang ada di tabel queue. Pada penelitian ini, penulis menggunakan paket laravel-job-status untuk menampilkan proses yang sedang terjadi pada antrian, sehingga pengguna dapat mengetahui jumlah data yang sudah terkirim.

2.5. NuSOAP

Pada penelitian yang ditulis oleh andri dkk menjelaskan NuSOAP adalah kumpulan class-class PHP yang memungkinkan user untuk mengirim dan menerima pesan SOAP melalui protokol HTTP. NuSOAP merupakan toolkit web service berbasis komponen. NuSOAP memilik sebuah class dasar yang menyediakan method [7]. Sedangkan SOAP merupakan kependekan dari Simple Object Access Protocol, SOAP merupakan standar format dokumen berbentuk XML yang digunakan untuk proses request dan response antara web servis dan aplikasi.

Aplikasi Pddikti menggunakan NuSOAP sebagai standar pertukaran data antara aplikasi pddikti dan aplikasi akademik yang ada di masing-masing perguruan tinggi, sehingga untuk bisa berkomunikasi dengan aplikasi Pddikti, aplikasi akademik harus menggunakan NuSOAP client dan menjalankan method-method yang tersedia, method-method ini terdapat pada manual penggunaan web services PDDikti yang bisa diambil di https://forlap.ristekdikti.go.id/files/feeder.

2.6. Supervisor

Supervisor merupakan aplikasi yang dapat digunakan untuk mengontrol dan melakukan monitoring proses pada sistem operasi linux. Di Situsnya dijelaskan bahwa "Supervisor is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems"[8]. Pada penelitian ini, penulis menggunakan supervisor untuk menjalankan *queue worker* laravel, sehingga status services proses antrian pada laravel dapat di monitoring.

2.7. AJAX

AJAX adalah singkatan dari *Asynchronous JavaScript* And XML. Ajax merupakan teknik dimana data pada halaman web di perbaharui secara asinkron dengan bertukar data dalam bentuk teks biasa atau dengan format json secara background. JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer [9].

2.8. Application Programming Interface (API)

API adalah antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program [10]. Api memungkinkan pengembang untuk memakai fungsi yang sudah ada dari aplikasi lain. Dengan menggunakan teknik API, maka pengembang aplikasi lain dapat menggunakan database tanpa harus terkoneksi secara langsung. Dalam aplikasi berbasis web, pemanggilan API menggunakan protokol Hyper Text Transfer Protocol (HTTP) melalui alamat tertentu dan akan memberikan respon data yang dapat berupa Hyper text markup language (html), Javascript Object Notation (JSON) ataupun Extensible Markup Language XML[11].

Dalam penelitian ini penulis menggunakan arsitektur REpresentational State Transfer (REST). Arsitektur REST merupakan model desain web servis yang dominan digunakan saat ini yang biasa disebut dengan RESTful[12]. REst digunakan penulis untuk menampilkan output proses sinkronisasi.

2.9. Penelitian Sejenis

Penelitian yang berjudul Aplikasi Sinkronisasi Database Antara Sistem Informasi Akademik Stikom Dinamika Bangsa Jambi Dengan Feeder Pangkalan Data Pendidikan Tinggi (PdDikti) [2] berhasil melakukan mapping basis data akademik dan web services feeder 2.0. Penelitian ini juga menghasilkan aplikasi yang dapat digunakan untuk melakukan sinkronisasi data berbasis PHP command line interface.

Penelitian lainnya berjudul Implementasi Web Service Pada Integrasi Data Akademik Dengan Replika Pangkalan Data Dikti [13]. penelitian ini berupa sebuah tools web service yang terdiri tiga modul utama yang dapat mengakomodir kebutuhan-kebutuhan pelaporan data akademik ke DIKTI yang sudah berjalan. Tools ini telah digunakan dalam dua tahun terakhir untuk keperluan pelaporan

3. METODOLOGI PENELITIAN

Tahapan penelitian dilakukan melalui kegiatan sebagai berikut :

1. Pengumpulan Data
   Pada tahapan proses ini, dilakukan pengumpulan data yaitu penulis melakukan pengumpulan data berupa referensi-referensi tentang queue pada framework laravel. Selain itu, penulis juga melakukan proses wawancara kepada operator pdpt STIKOM Dinamika Bangsa untuk mengetahui permasalahan yang terjadi
2. Analisis Data
   Pada tahapan proses ini, dilakukan analisis terhadap data yang diperoleh dari proses pengumpulan data. Selain itu, penulis menganalisis kebutuhan basis data untuk queues, melakukan analisis kesalahan pada proses sinkronisasi kelas quick study dan mata kuliah agama.
3. Pengembangan Sistem
   Pada tahap ini, penulis melakukan pengembangan sistem yang meliputi pembuatan user interface, perancangan queues serta melalukan perbaikan pada kode-kode yang di aplikasi sebelumnya. Proses pengembangan sistem menggunakan metode waterfall.
4. Pengujian
   Pada tahapan proses ini, dilakukan proses pengujian. Pengujian yang dilakukan meliputi pengujian sinkronisasi dengan menggunakan ragam data, mulai dari 300 data sampai dengan 100.000 data untuk mendapatkan hasil pengujian yang maksimal.
5. Penyusunan Laporan
   Pada tahapan proses ini, dilakukan proses penyusunan atau pembuatan laporan yang diperoleh dari hasil penelitian yang dilakukan oleh penulis. Tujuan dari tahap ini adalah agar penelitian ini dapat dibaca sehingga dapat diperoleh kritik maupun saran dari para pembaca. Serta dapat juga dijadikan sebagai bahan acuan dan referansi bagi pengembangan penelitian yang selanjutnya.

4. HASIL DAN PEMBAHASAN

4.1. Analisis Permasalahan

Aplikasi sinkronisasi Database Akademik Dan Feeder Dikti yang dimiliki oleh STIKOM DB telah di implementasikan untuk membantu pelaporan kegiatan akademik. Berdasarkan hasil implementasi, terdapat permasalahan utama pada aplikasi ini yaitu proses sinkronisasi hanya bisa dilakukan menggunakan *PHP command line interface*, selain itu, catatan sinkronisasi atau *log* masih belum ditampilkan sesuai periode sinkronisasi.

4.2. Analisis Proses Sinkronisasi

Pada analisis proses sinkronisasi, penulis menemukan permasalahan yaitu penggunaan *browser* dalam proses sinkronisasi menimbulkan permasalahan :

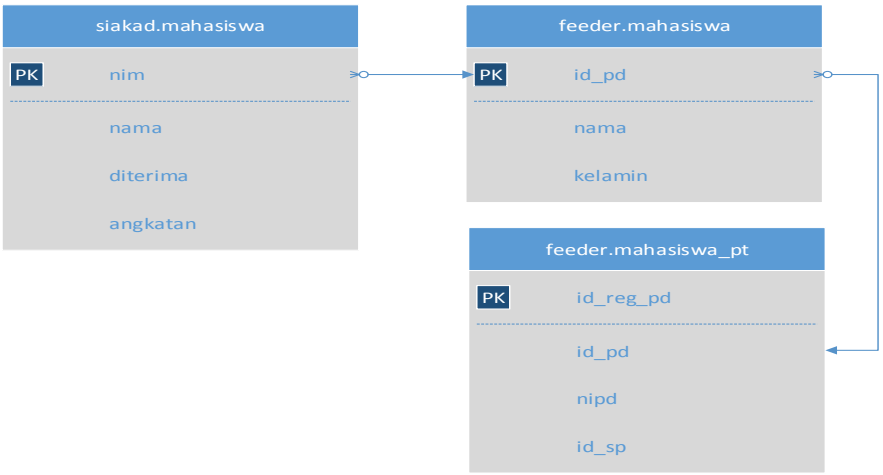
1. Proses sinkronisasi dengan jumlah data yang banyak akan membuat waktu tunggu menjadi lama dan *browser* tidak boleh ditutup selama proses sinkronisasi berlangsung.
2. Jika *browser client* ditutup/tertutup secara paksa maka proses sinkronisasi terhenti dan harus di ulang dari awal.
3. Proses sinkronisasi hanya bisa dilakukan satu persatu, jika proses pertama belum selesai, maka proses berikutnya tidak bisa dilakukan.
4. Proses sinkronisasi tidak memiliki *log* yang detail dan dapat di *sortir* sesuai kebutuhan

4.3. Analisis Data Sinkronisasi

Pada tahapan ini, penulis melakukan *mapping* tabel data yang terdapat pada sistem informasi akademik dan feeder dikti. Proses ini dilakukan untuk memetakan data yang terdapat di sistem informasi akademik dan feeder dikti.

Berikut adalah *mapping* database sisfo untuk proses sinkronisasi feeder :

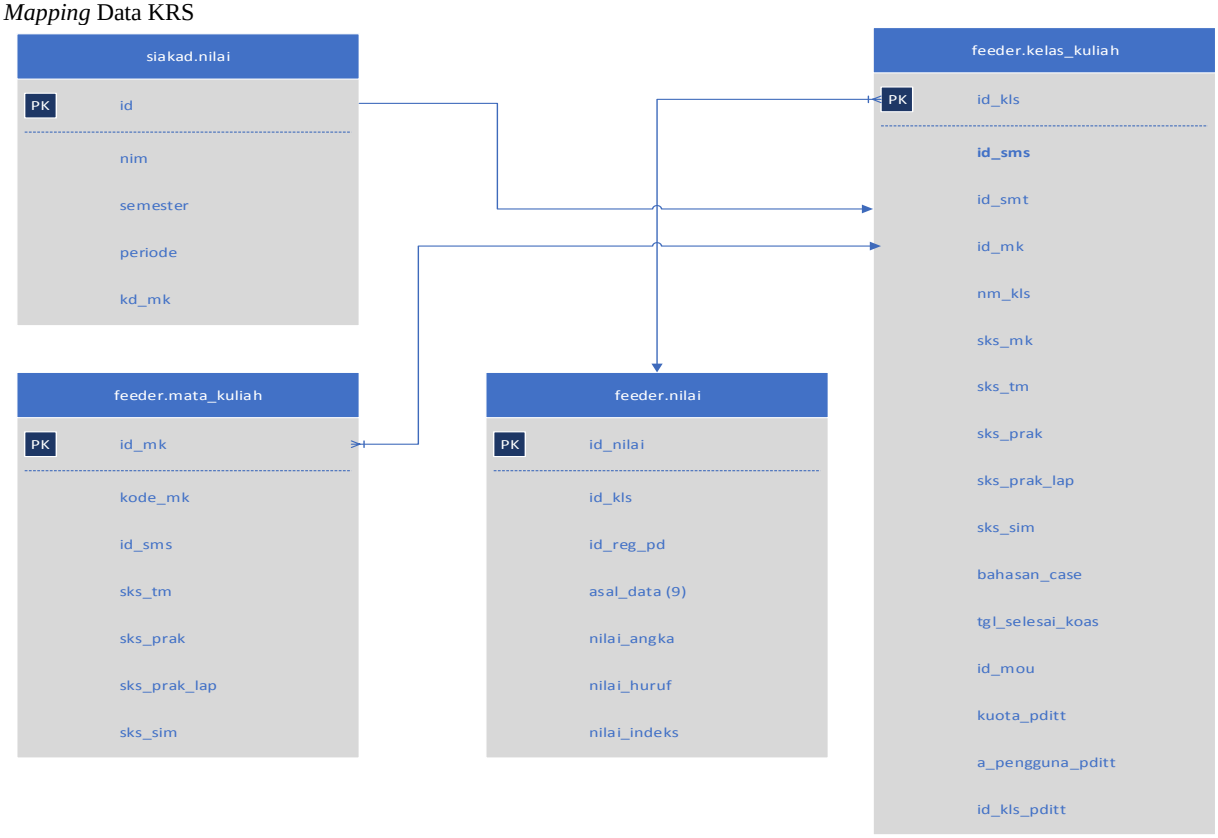
1. *Mapping* Data Mahasiswa



Gambar 1. *Mapping* Tabel Data Mahasiswa

Pada gambar 1 terdapat *mapping* tabel data mahasiswa. Sistem informasi akademik atau siakad memiliki tabel dengan nama mahasiswa dimana di tabel mahasiswa inilah semua data mahasiswa disimpan. Sedangkan pada sistem feeder dikti memiliki 2 tabel untuk data mahasiswa. Tabel mahasiswa pada feeder dikti digunakan untuk menyimpan data pokok yang terkait dengan mahasiswa, sedangkan tabel mahasiswa_pt digunakan untuk menyimpan data mahasiswa yang terkait dengan perguruan tinggi. Proses sinkronisasi data mahasiswa dilakukan dengan cara menyimpan data mahasiswa dari siakad ke mahasiswa feeder lalu, mendapatkan id_pd dari proses pertama untuk disimpan ke tabel mahasiswa_pt.

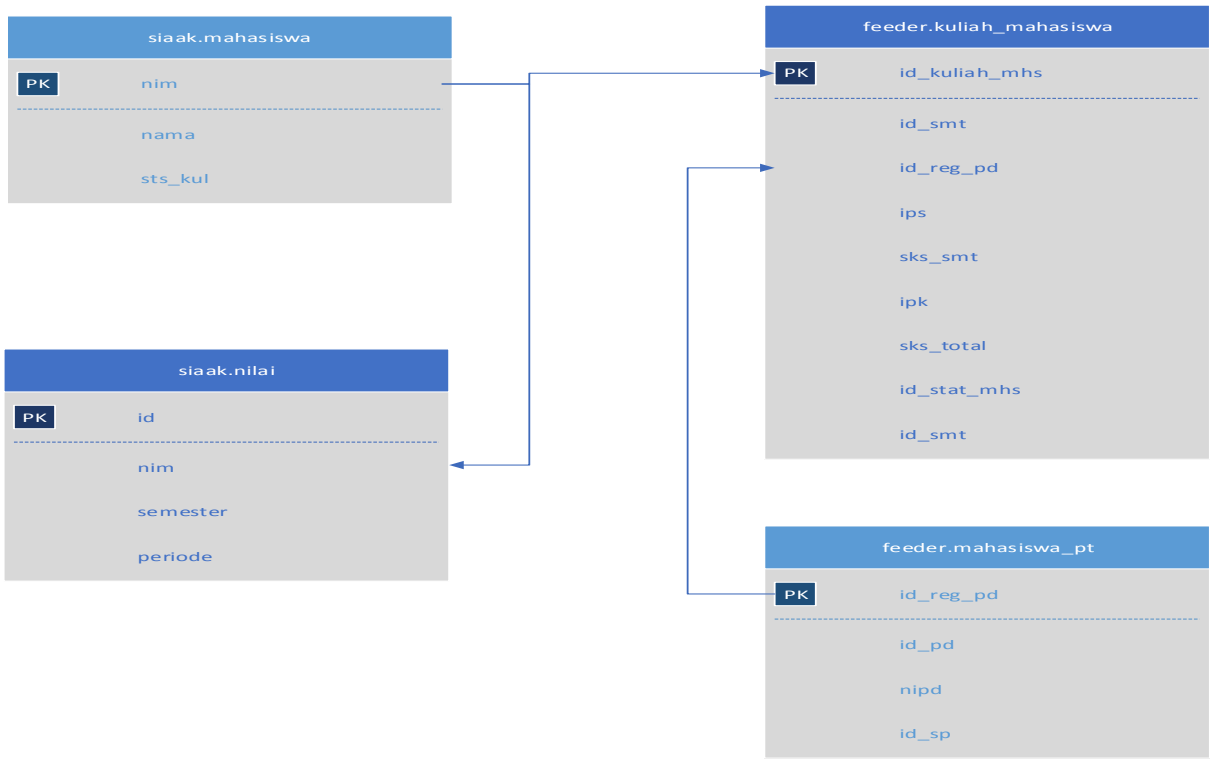
2. *Mapping* Data KRS



Gambar 2. *Mapping* Data KRS

Pada gambar 2 terdapat tabel yang digunakan pada proses sinkronisasi data KRS. Proses sinkronisasi KRS dilakukan di akhir semester setelah data nilai pada sistem informasi akademik lengkap. Proses sinkronisasi KRS mengambil data dari tabel nilai siakad, pada tabel nilai ini, terdapat semua data yang dibutuhkan untuk proses sinkronisasi.

3. *Mapping* Data Aktivitas (TRAKD)



Gambar 3 *Mapping* Data TRAKD

Pada gambar 3, proses sinkronisasi melibatkan tabel mahasiswa dan tabel nilai, proses ini dibutuhkan untuk melakukan menambahkan nilai hasil perkuliahan mahasiswa.

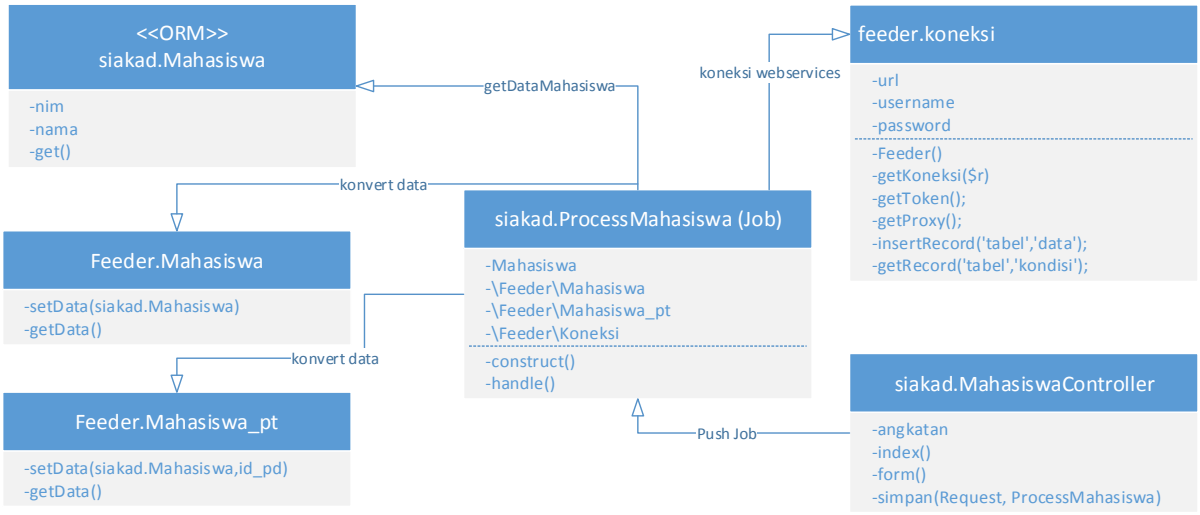
4.4. Solusi Permasalahan

Adapun solusi dari permasalahan yang telah disebutkan diatas yaitu :

1. Menggunakan fitur *queue* pada framework *laravel* berbasis *ajax* dan menjalankan *queue* data pada aplikasi *supervisor* sebagai *background* proses pada sistem operasi linux untuk mengoptimalkan proses sinkronisasi dengan *interface* web, sehingga proses sinkronisasi data tetap bisa dilakukan sekalipun *browser* ditutup oleh user.
2. Menambahkan fitur *log* untuk menampilkan hasil dari sinkrosnisasi data dan melengkapi fitur tersebut dengan periode sinkronisasi.

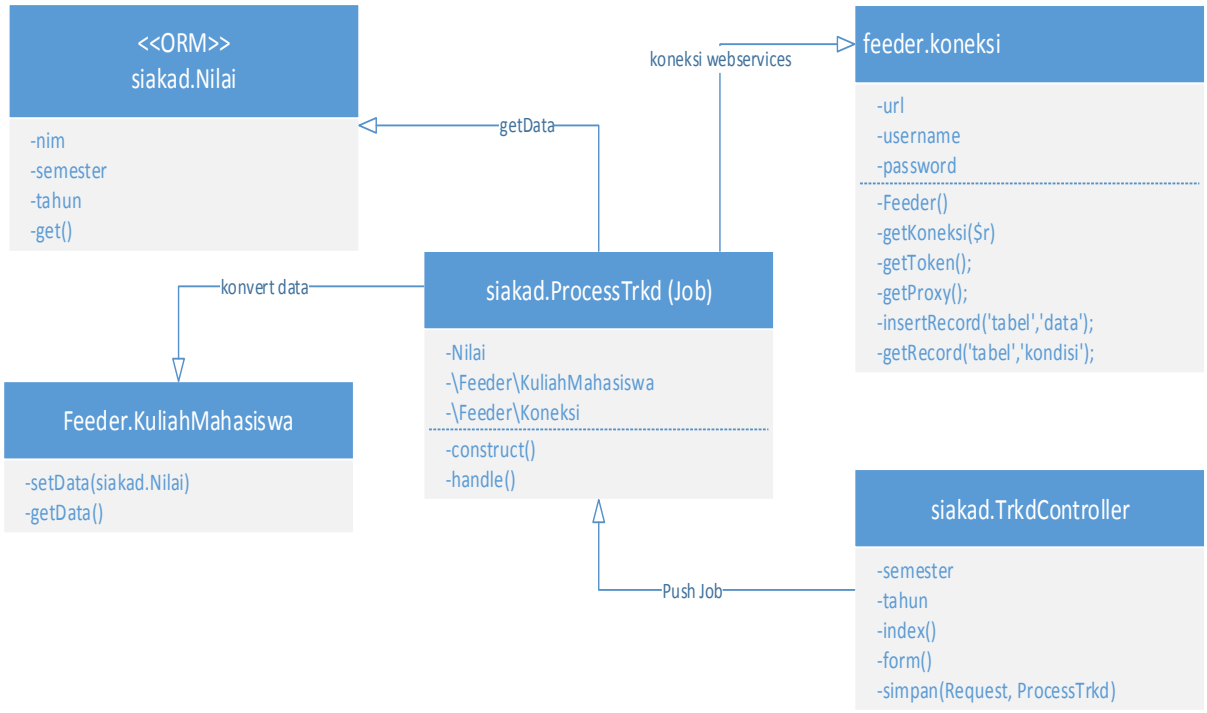
4.5. Perancangan Class Diagram

Untuk membantu memudahkan pada proses perancangan kode, penulis menggunakan class diagram seperti berikut :



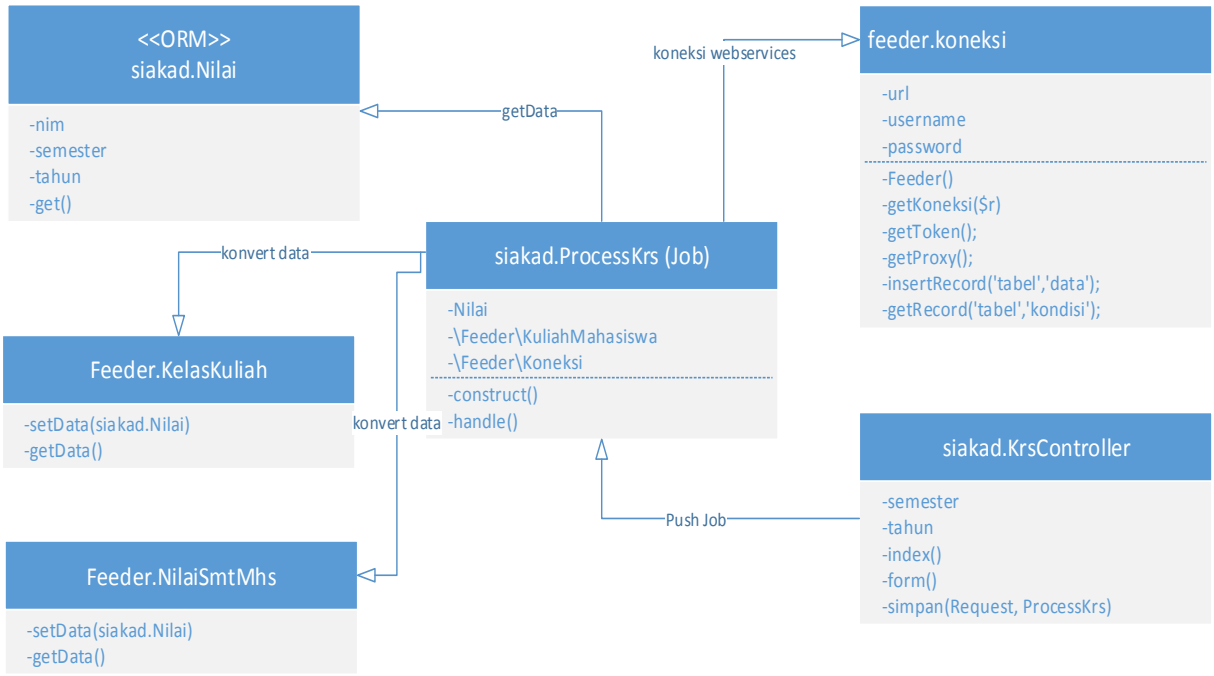
Gambar 4. *Class Diagram* Sinkroniasi Mahasiswa

Pada gambar 4 terdapat *class diagram* proses sinkronisasi data mahasiswa, proses sinkronisasi di awali dengan menampilkan *form* sinkronisasi mahasiswa oleh kelas *MahasiswaController method form*, lalu setelah user memilih angkatan, method simpan akan mengirim *job* ke *queue laravel* yang ada di kelas *ProcessMahasiswa*. Ketika *queue* dijalankan (*worker*) maka kelas *ProcessMahasiswa* akan mengambil data mahasiswa dari tabel siakad.mahasiswa, data mahasiswa lalu di konversi melalui kelas Feeder.Mahasiswa dan Feeder.Mahasiswa_pt, data yang telah di konversi akan disimpan oleh kelas Koneksi dengan *method insertRecord*.



Gambar 5. *Class Diagram* Sinkronisasi Aktivitas Mahasiswa (TRKD)

Pada gambar 5 terdapat *class diagram* proses sinkronisasi aktivitas mahasiswa, proses ini membutuhkan data nilai dari tabel siakad.nilai, data siakad.nilai diambil berdasarkan semester dan periode pilihan pengguna lalu di konversi melalui kelas Feeder.KelasKuliah kemudian dikirim ke laravel queue untuk di eksekusi sesuai antrian data.



Gambar 6. *Class Diagram* Sinkronisasi Data KRS

Pada gambar 6 terdapat *class diagram* proses sinkronisasi mahasiswa, kelas *KrsController* menerima input berupa tahun dan semester, lalu menjalankan fungsi simpan dengan mengirim *job* ke *laravel queue*. Ketika *ProcessKrs* dijalankan maka sinkronisasi data akan dilakukan dengan mengambil data di tabel siakad.nilai berdasarkan semester dan tahun lalu mengkonversi menjadi data kelas kuliah dan menyimpan ke tabel feeder.kelas_kuliah. Proses selanjutnya yaitu kembali mengambil siakad.nilai untuk kemudian di koversi ke *NilaiSmtMhs* yang kemudian disimpan ke tabel feeder.nilai_smt_mhs.

4.6. Optimalisasi Sinkronisasi

Untuk mengoptimalkan proses sinkronisasi yang digunakan, penulis memanfaatkan fitur *queue*/antrian pada *laravel*. Pada penelitian ini, penulis menggunakan database untuk menyimpan data antrian. Data antrian ini disebut dengan *Job*. Dalam penggunaan *laravel queue*, kita akan mengenal istilah berikut :

1. *Queue* pada *laravel* merupakan antrian yang berisi kelas *job*, kelas *job* akan disimpan ke database *queue* sebelum di eksekusi.
2. Kelas *Job*, Kelas *Job* pada *laravel* merupakan sebuah file kelas yang berisi kode-kode yang akan ditunda eksekusinya. Kelas *Job* ini akan dimasukkan ke *queue* yang kemudian akan dieksekusi.
3. *Queue worker* merupakan proses menjalankan semua kelas *job* (kode sinkronisasi) yang ada pada tabel *queue*.

Unuk membuat tabel *queue* data pada *laravel* dengan menggunakan perintah berikut :

- php artisan queue:table
- php artisan migrate

Perintah diatas akan membuat tabel dengan nama *jobs*, tabel inilah yang nantinya akan menyimpan setiap antrian *job*. Untuk membuat *job* pada *laravel* menggunakan perintah :

- php artisan make:job ProcessMahasiswa

Perintah diatas akan membuat file kelas \App\Jobs\ProcessMahasiswa.php yang berisi perintah berikut :

```
<?php
namespace App\Jobs;
use Illuminate\Bus\Queueable;
use Illuminate\Queue\SerializesModels;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use \App\Feeders\Mahasiswa;
class ProcessTes implements ShouldQueue
{
 use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
 /**
 * Create a new job instance.
 *
 * @return void
 */
 private $angkatan;
 public function __construct($angkatan)
 {
 $this->angkatan = $angkatan;
 }

 /**
 * Execute the job.
 *
 * @return void
 */
 public function handle()
 {
 $siakad_mahasiswas = \App\Mahasiswa::getDataByAngkatan($this->angkatan);
 foreach ($siakad_mahasiswas as $siakad_mahasiswa){
 $feeder_mahasiswa = new \App\Feeders\Mahasiswa($siakad_mahasiswa);
 $koneksi->insertRecord('mahasiswa',$feeder_mahasiswa);
 }
 }
}
```

Job pada *laravel* di representasikan pada sebuah kelas proses seperti yang telah dibuat yaitu kelas *ProcessMahasiswa.php*, kelas *ProcessMahasiswa* memiliki *method handle()* dimana semua kode yang akan diproses di antrian harus dimasukkan ke method ini. Dan untuk menjalankan kelas *ProcessMahasiswa* dari *MahasiswaController* dapat menggunakan perintah

```
$job = new \App\Jobs\ProcessMahasiswa($angkatan);
$this->dispatch($job);
```

Method dispatch($job) akan mengiri kelas *ProcessMahasiswa* ke tabel siakad.jobs untuk kemudian di eksekusi oleh worker.

id	queue	payload	attempts	reserved_at	available_at	created_at
1	default	{"displayName":"App\\Jobs\\ProcessMahasiswa","job"...	0	*NULL*	1540280810	1540280810

Gambar 7. Data *Jobs Queue*

Agar proses sinkroniasi dapat dimonitor maka penulis menggunakan paket laravel-job-status yang dapat di install melalui composer dengan cara :

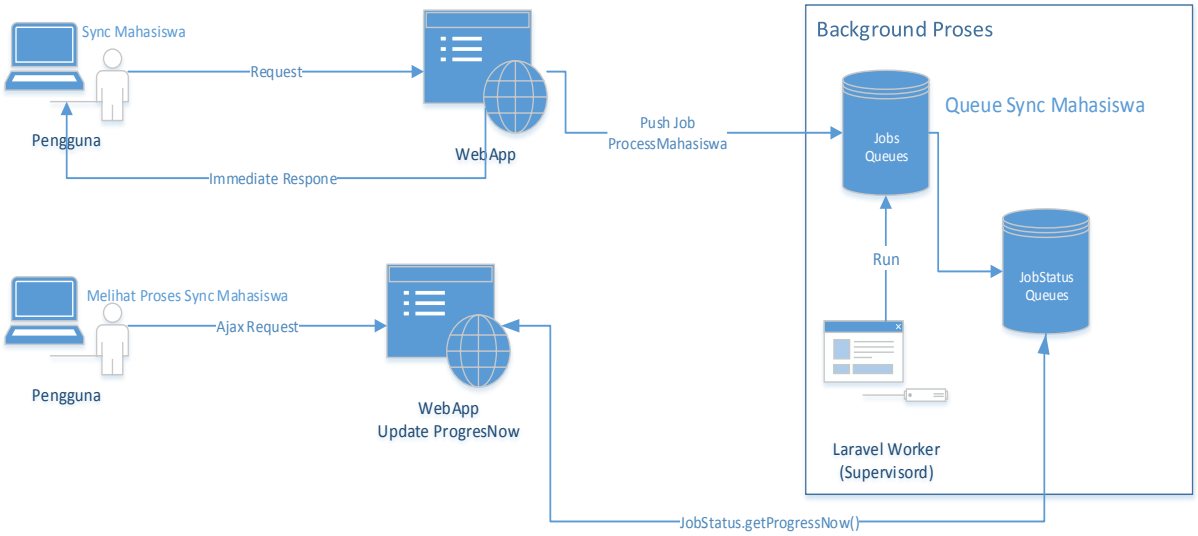
- compuser require imtigger/laravel-job-status

Untuk menggunakan laravel-job-status, kita perlu memodifikasi kelas ProcessMahasiswa.php seperti berikut :

```
<?php

namespace App\Jobs;

use Illuminate\Bus\Queueable;
use Illuminate\Queue\SerializesModels;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use \App\Feeders\Mahasiswa;
use Imtigger\LaravelJobStatus\Trackable;
class ProcessTes implements ShouldQueue
{
use Dispatchable, InteractsWithQueue, Queueable, SerializesModels, Trackable;

/**
* Create a new job instance.
*
* @return void
*/
private $angkatan;
public function __construct($angkatan)
{
$this->prepareStatus();
$this->angkatan = $angkatan;
}

/**
* Execute the job.
*
* @return void
*/
public function handle()
{
$siakad_mahasiswas = \App\Mahasiswa::getByAngkatan($this->angkatan);
$this->setProgressMax($siakad_mahasiswa->count());
$i=1;
foreach ($siakad_mahasiswas as $siakad_mahasiswa){
$feeder_mahasiswa = new \App\Feeders\Mahasiswa($siakad_mahasiswa);
$koneksi->insertRecord('mahasiswa',$feeder_mahasiswa);
$this->setProgressNow($i);
$i++;
}
$this->setOutput(['total' => $siakad_mahasiswa->count()]);
}
}
```

Untuk menggunakan *Laravel-job-status*, kita harus meng-implementasikan kelas *trait Trackable*, *Laravel-job-status* menggunakan *trait* pada kelas *Trackable* yang memiliki method *prepareStatus()*, method ini akan mengambil baris data yang ada di tabel *jobs* kemudian disimpan di tabel *jobstatus*. Selain itu, pada *method handle*, kita bisa menginformasikan jumlah data yang akan di sinkronisasikan dengan *method setProgressMax($total_data)* lalu pada setiap perulangan data kita menggunakan *method setProgressNow(i)* untuk mengirim perubahan jumlah data yang berhasil di sinkronisasi.

Secara keseluruhan, proses antrian dapat dilihat pada gambar berikut :



Gambar 8. *Laravel Queue Job Status*

Pada gambar 8, pengguna melakukan sinkronisasi data siakad.mahasiswa, sistem kemudian menyimpan proses sinkronisasi data mahasiswa ke tabel *jobs*, sistem kemudian memberikan respon ke pengguna bahwa data sudah disimpan, sistem juga akan mengirimkan *job* kelas *ProcessMahasiswa* ke tabel *jobs* untuk di eksekusi sesuai antrian. Dengan cara ini maka pengguna tidak perlu menunggu proses sinkronisasi sampai selesai, karena proses sinkronisasi mahasiswa akan dilakukan secara *background* oleh aplikasi *supervisor*.

Pada saat user membuka halaman index mahasiswa, maka sistem akan melakukan *request* menggunakan *ajax* ke tabel jobstatus untuk menampilkan sudah berapa banyak data yang berhasil di sinkronisasi kan ke feeder dikti.

4.7. Optimalisasi Proses Supervisor

Supervisor merupakan aplikasi *monitoring* proses yang berjalan pada server linux, untuk melakukan instalasi *supervisor* pada debian 8 menggunakan perintah :

- apt-get install supervisor

Selanjutnya membuat file konfogurasi dengan nama laravel-worker.conf agar *supervisor* menjalankan *laravel artisan queue:worker*,

- # nano /etc/supervisor/conf.d/laravel-worker.conf

Isi dengan perintah berikut :

```
[program:laravel-worker]
process_name=%(program_name)s_%(process_num)02d
command=php /var/www/pdpt-sync.stikom-db.ac.id/web/artisan queue:work --tries=3 --timeout=0
autostart=true
autorestart=true
user=stikom
numprocs=1
redirect_stderr=true
stdout_logfile= var/log/worker.log
```

Lalu melakukan update dan menjalankan ulang services supervisor,

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl start laravel-worker:*
```

4.8. Optimalisasi Web Interface

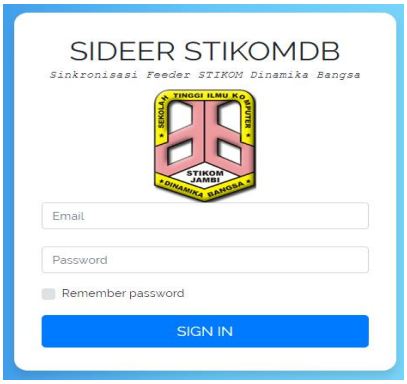
Perancangan *interface* web menggunakan *ajax* agar proses dilakukan secara *background*. Pada penelitian ini penulis menggunakan *bootstrap* dan *library jQuery*.
Pada penelitian ini, penulis menggunakan *ajax* untuk melihat status dari proses sinkronisasi, berikut adalah kode untuk melakukan request :

```
<script>
$(document).ready(function() {
 $("#form-sync").hide();
 @if(!empty($job_status_id))
 var interval = setInterval(function(){
 var elem = document.getElementById("progress-bar");
 $.get('{{ url('/job/statuskrs') }}/?job='+{{ $job_status_id }}, function (data){
 elem.style.width = data.progress_percentage + '%';
 $("#progress-bar").html(data.progress_percentage + '%');
 });
 },1000);
 @endif
});
</script>
```

Pada kode diatas, penulis melakukan *request* berdasarkan job_id ke url http://domain.com/job/statuskrs/?job=job_id, berikut adalah kode pada route *statusKrs* :

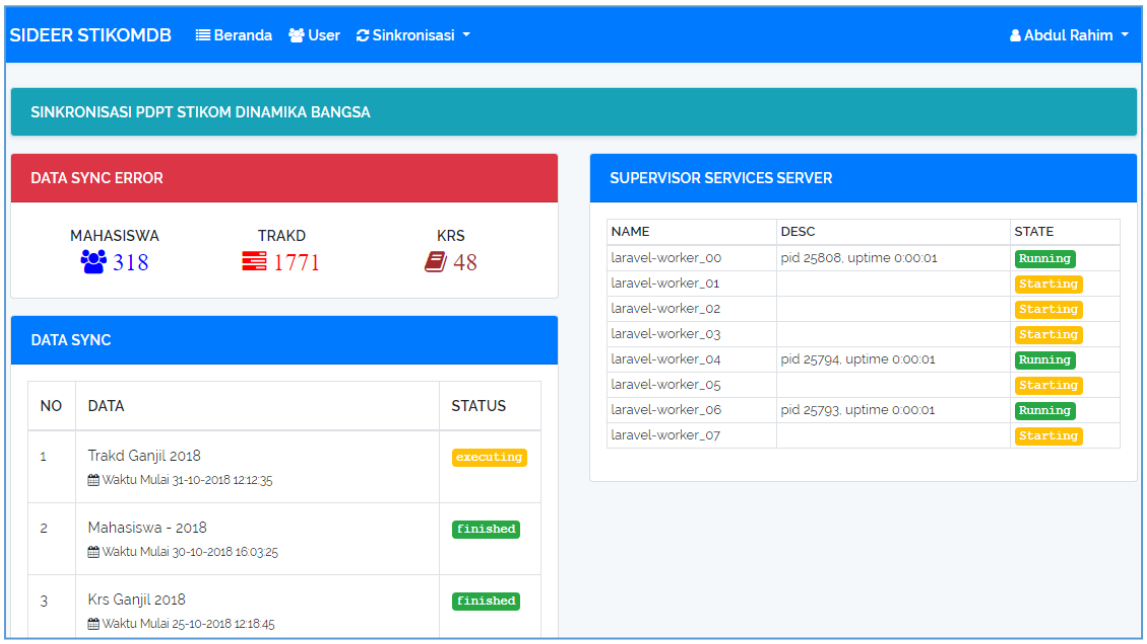
```
public function statusKrs(Request $request)
{
 $job_id = $request->input('job');
 $jobStatus = \App\JobStatus::find($job_id);
 $jobStatusr['total_error'] = $log->count();
 $jobStatusr['progress_percentage'] = $jobStatus->progress_percentage;
 $jobStatusr['progress_now'] = $jobStatus->progress_now;
 $jobStatusr['is_finished'] = $jobStatus->is_finished;
 $jobStatusr['status'] = $jobStatus->status;
 return response()->json($jobStatusr);
}
```

Pada method *statusKrs* diatas, penulis mengambil data dari tabel *jobstatus*, adapun data yang diambil adalah *progress_percentage* untuk menampilkan persentase proses, progress_*now* untuk menampilkan jumlah data yang berhasil di sinkronisasikan dan status untuk menampilkan status dari *job*.



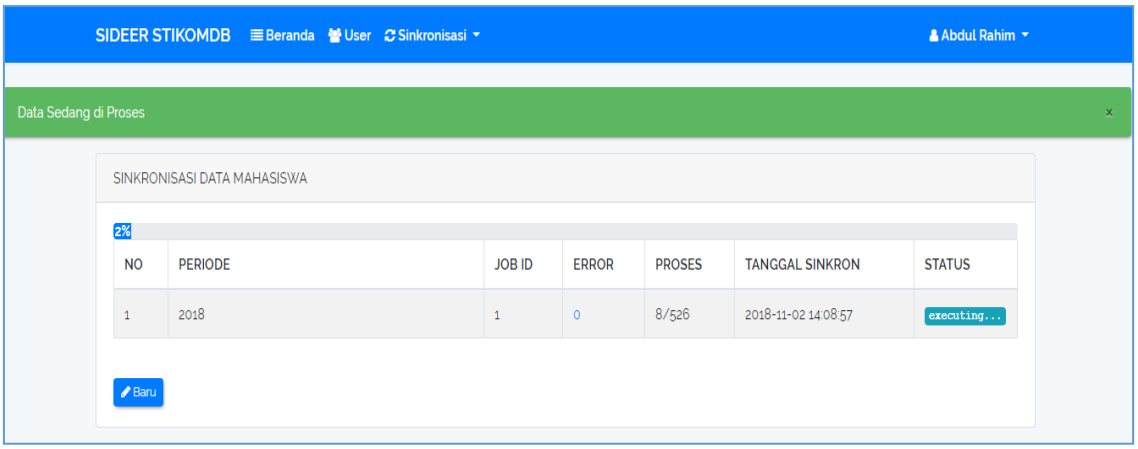
Gambar 9. Halaman *Login*

Gambar 9 merupakan halaman login untuk operator aplikasi SIDEER atau sinkronisasi feeder.



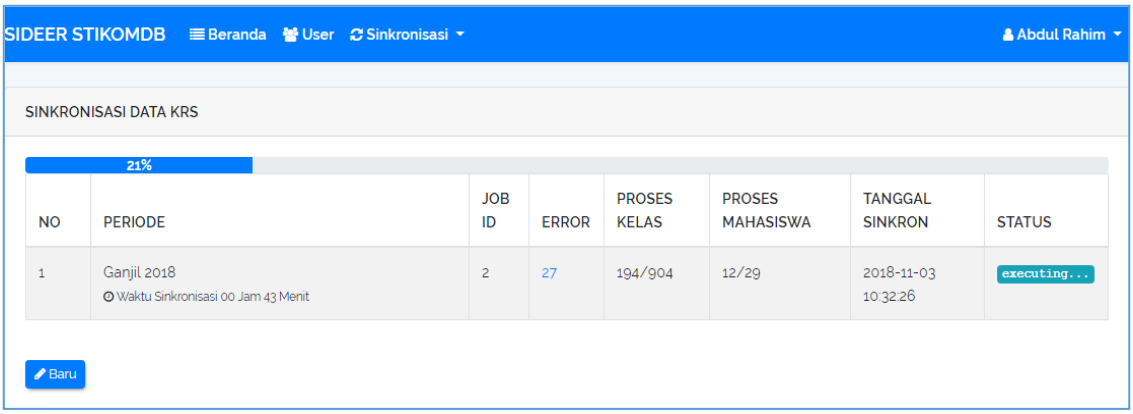
Gambar 10. *Dashboard SIDEER*

Gambar 10 merupakan dashboard utama setelah pengguna berhasil login, di halaman dashboard menampilkan informasi kesalahan-kesalahan di proses sinkronisasi dan proses *queue worker* pada *supervisor*.



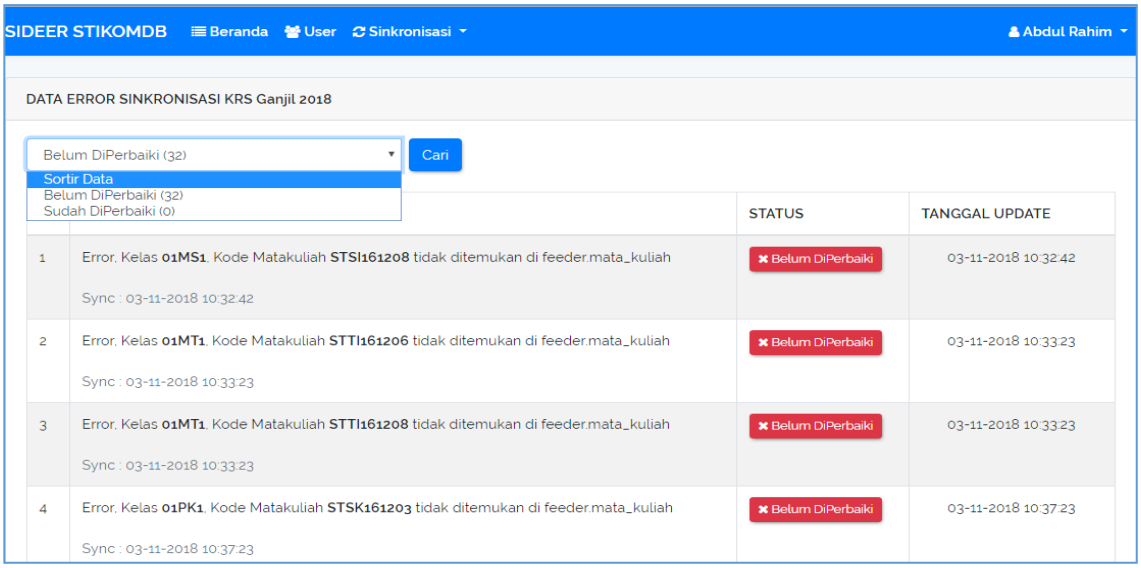
Gambar 11. Proses Sinkronisasi Data Mahasiswa

Gambar 11 merupakan tampilan proses sinkronisasi data mahasiswa, pada gambar 11 terdapat loading secara realtime yang menggambarkan proses sinkronisasi, selain itu terdapat infomasi data yang berhasil di simpan ke feeder yaitu 8 dari total 526 data dan status dari sinkronisasi yaitu executing atau sedang di eksekusi.



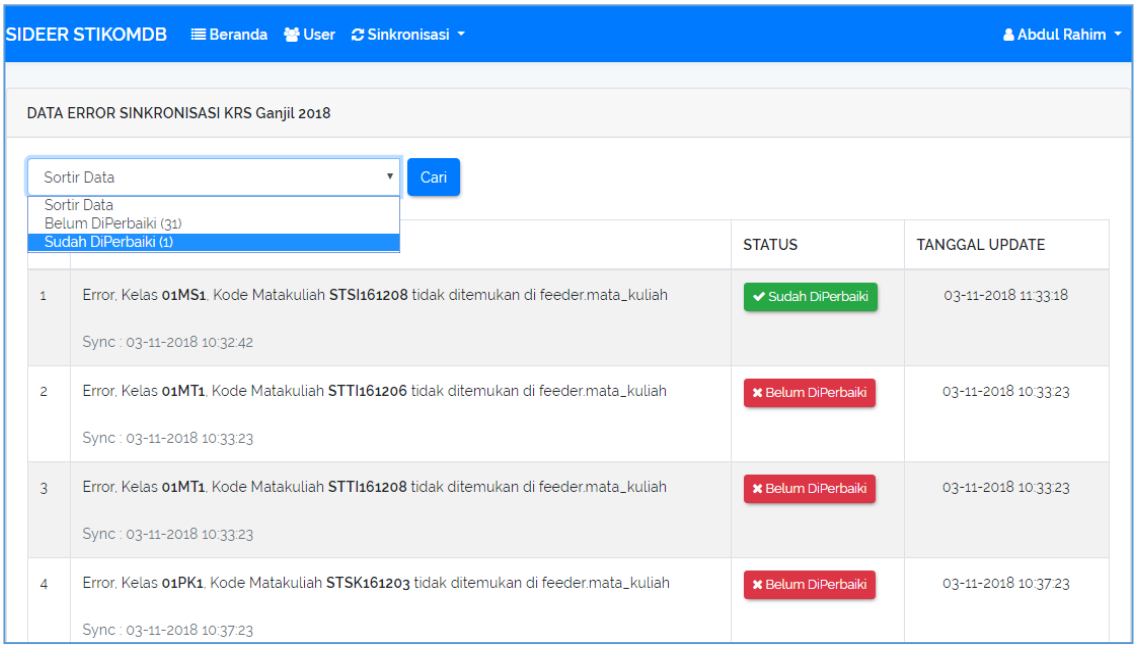
Gambar 12. Proses Sinkronisasi KRS

Gambar 12 merupakan tampilan proses sinkronisasi KRS, tampilan proses sinkronisasi KRS terdapat proses menyimpan data kelas dan proses menyimpan data mahasiswa didalam kelas tersebut secara realtime. Pada contoh diatas terdapat error dalam proses sinkronisasi yang bisa dilihat dengan cara mengklik jumlah error tersebut.



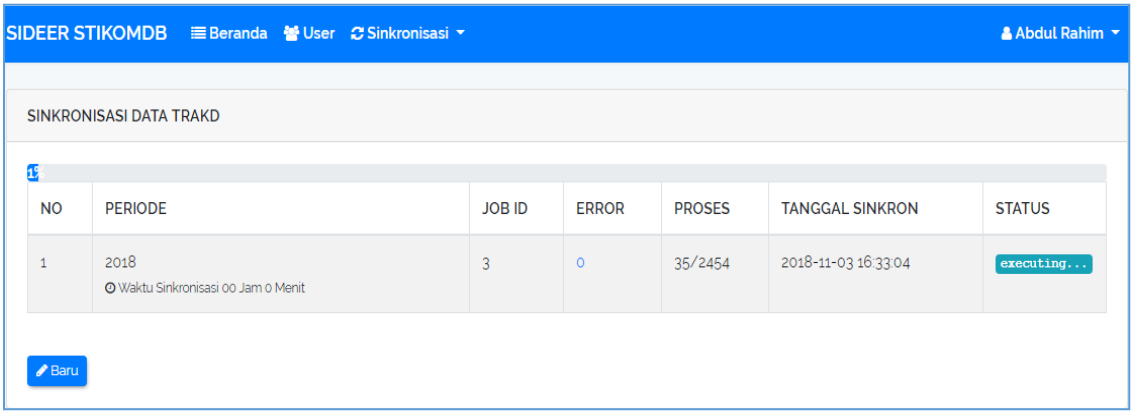
Gambar 13. Tampil Data *Error Proses* Sinkronisasi

Pada gambar 13 terdapat tampilan error dari proses sinkronisasi KRS, error ini disebabkan data matakuliah yang ada di sisfo akademik belum sinkron dengan matakuliah di feeder. Pengguna dapat mengklik Status untuk mengubah.



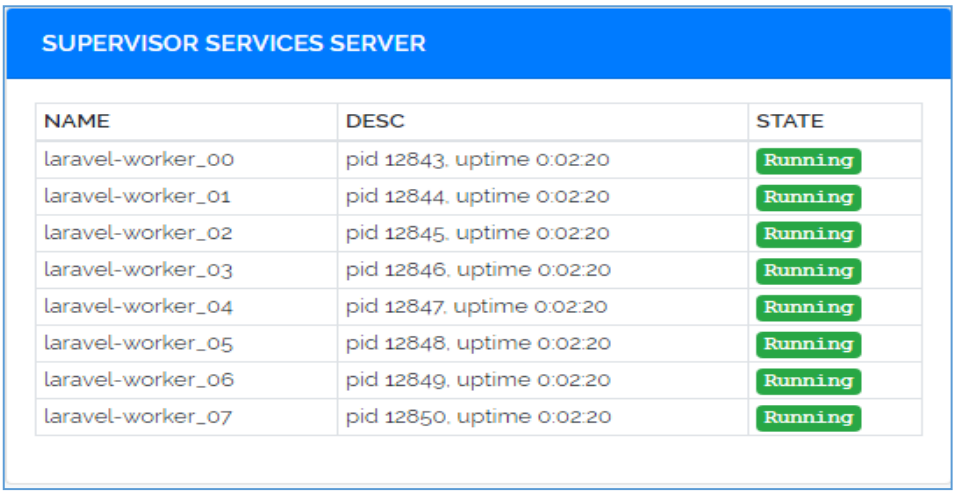
Gambar 14. Perbaikan *Error*

Gambar 14 merupakan perbaikan error yang telah dilakukan pengguna. Data yang sudah diperbaiki memiliki status berwarna hijau sebegai pembeda dengan yang belum di perbaiki.



Gambar 15. Proses Sinkronisasi TRAKD

Gambar 15 merupakan proses sinkronisasi data TRAKD atau aktivitas mahasiswa, proses ini akan memindahkan data sebanyak 2454 baris data.

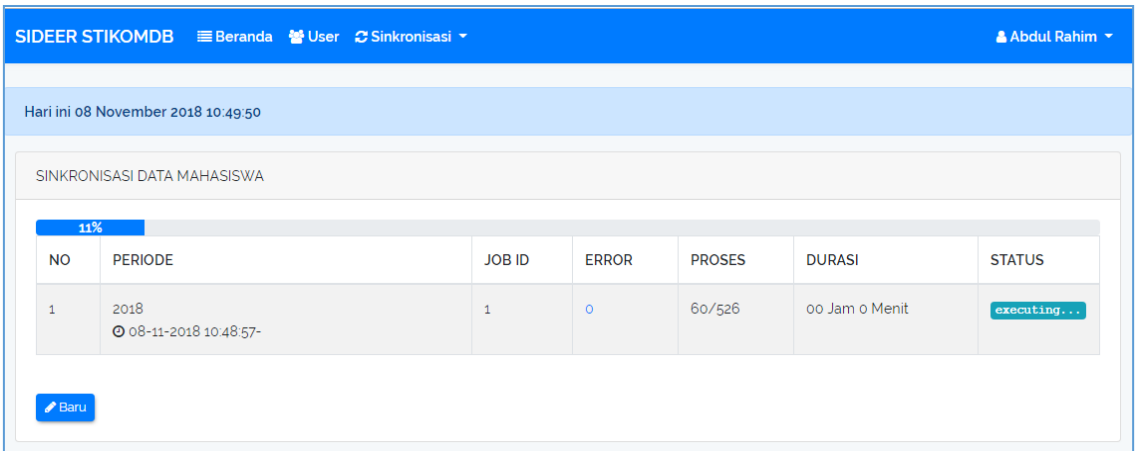


Gambar 16. Servis Supervisor

Gambar 16 merupakan tampilan dari supervisor, di gambar 16 kita bisa melihat status dari queue worker yang sedang di eksekusi oleh supervisor.

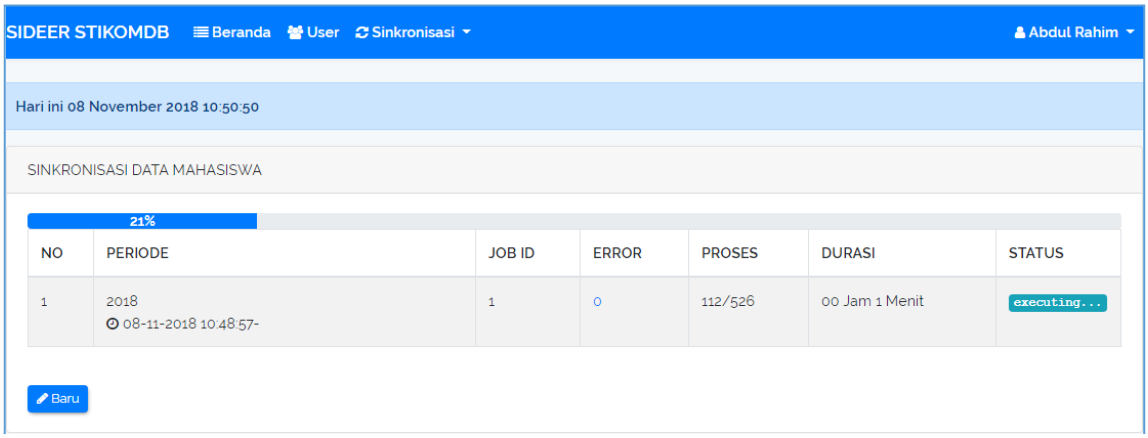
4.9. Proses Pengujian

Pada tahap ini, penulis melakukan pengujian terhadap optimalisasi yang telah dilakukan. Proses pengujian dilakukan dengan melakukan sinkronisasi data kemudian menutup *browser*. Pengujian ini dinyatakan berhasil jika proses sinkronisasi masih tetap dilakukan ketika *browser* membuka halaman sinkronisasi.



Gambar 17. Proses Sinkronisasi mahasiswa

Pada gambar 17 penulis melakukan sinkronisasi data mahasiswa, ketika proses mencapai 11% atau sudah 60 data berhasil di sinkronisasikan, maka penulis menutup *browser* untuk melakukan pengujian apakah proses tetap dilanjutkan secara background.



Gambar 18. Proses Sinkronisasi

Gambar 18 penulis membuka *browser* untuk melihat proses sinkronisasi mahasiswa, terlihat bahwa proses sinkronisasi masih terus berjalan dan sudah mencapai 21% atau sudah 112 data yang berhasil di sinkronisasikan ke aplikasi feeder. Pengujian ini dinyatakan berhasil karena proses sinkronisasi data tidak lagi bergantung pada browser yang digunakan oleh pengguna sehingga pengguna hanya perlu melakukan input data awal dimana data ini akan masuk pada antrian lalu aplikasi supervisor akan menjalankan antrian pada laravel.

Tabel 1. Pengujian Sinkronisasi

Jenis Sinkronisasi	Jumlah Data	Waktu Mulai	Waktu Selesai	Durasi
Mahasiswa	526	03-11-2018 09:55:53	03-11-2018 10:02:46	6 Menit
KRS Kelas/Mahasiswa	904/20166	03-11-2018 10:32:29	03-11-2018 14:35:15	04 Jam 2 Menit
TRAKD	2454	03-11-2018 16:33:04	03-11-2018 17:14:00	00 Jam 40 Menit

Pada tabel 1 terdapat proses pengujian sinkronisasi, proses sinkronisasi dengan jumlah data terbanyak adalah proses sinkronsasi KRS karena proses ini akan menyimpan semua data kelas serta data mahasiswa yang ada di tiap-tiap kelas tersebut. Pada Proses sinkronisasi KRS, terdapat 904+20166 data yang harus dipindahkan dari siakad ke aplikasi feeder. Proses ini membutuhkan waktu 04 jam 2 menit sebagaimana pengujian yang telah dilakukan oleh penulis. Pada rentang waktu pengujian, penulis juga melakukan percobaan menutup dan membuka browser untuk melihat proses yang sedang berjalan.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Adapun kesimpulan dari penelitian ini adalah :

1. Fitur Queues pada framework laravel merupakan fitur yang dapat digunakan untuk menunda proses atau menjalankan proses secara background.
2. Proses sinkronisasi data akademik dan feeder dikti dapat dilakukan dengan menggunakan fitur Queue pada framework laravel.
3. Dengan menggunakan fitur Queue dan aplikasi supervisor maka pengguna tidak perlu menunggu proses sinkronisasi selesai, karena proses sinkronisasi telah disimpan di queue dan dijalankan sebagai background services.
4. Template ini dibuat untuk konsistensi format artikel yang diterbitkan oleh Jurnal MEDIA PROCESSOR. Kerjasama dan kesediaan penulis mengikuti acuan penulisan sangat diharapkan.

5.2. Saran

Adapun saran untuk penelitian selanjutnya adalah diharapkan agar melakukan pengembangan sinkronisasi untuk modul-modul lain yang terdapat di sistem informasi akademik.

DAFTAR PUSTAKA

[1] Anugrah Sandi. “Alasan Mengapa Kamu Harus Menggunakan Framework Laravel.” https://www.codepolitan.com/alasan-mengapa-kamu-harus-menggunakan-framework-laravel-5a08d435ddcfb#,. [Mar. 10, 2019].

[2] Rifi Indra Perwira, Budi Santosa. “Implementasi Web Service Pada Integrasi Data Akademik dengan Replika Pangkalan Data DIKTI.” *Telematika*, Vol. 14 No. 01, April 2017, Pages 1-11.

[3] Agus Siswanto, Abdul Rahim. “Aplikasi Sinkronisasi Database Antara Sistem Informasi Akademik STIKOM Dinamika Bangsa Jambi Dengan Feeder Pangkalan Data Pendidikan Tinggi.” *PdDikti*, 2017.

[4] Abdul Rohman. “Mengenal Framework “ Laravel ” (Best PHP Frameworks For 2014).” Ilmu Teknologi Informasi. http://ilmuti.org/wp-content/uploads/2014/03/Abdul_Rohman-Mengenal_Framework_Laravel.pdf

[5] Laravel, 2017, Queues - Introduction., https://laravel.com/docs/5.7/queues#introduction, [Oct. 23, 2018].

[6] ImTigger. "imTigger/laravel-job-status: Add ability to track Job progress, status and result dispatched to Queue." https://github.com/imTigger/laravel-job-status., [Oct. 23, 2018].
[7] Andri, Zuer Zamzami. M., & Nasir. "Perancangan Dan Implementasi Sistem Akademik Berbasis Web Service Pada Akbid Budi Mulia Palembang." Seminar Nasional Teknologi Informasi Dan Komunikasi (SNASTIKOM 2014), (12), pp. 0–6
[8] "Supervisor: A Process Control System." http://supervisord.org/, [Oct. 23, 2018]
[9] JSON. "Pengenalan JSON." https://www.json.org/json-id.html, Tanggal diakses [Oct. 23, 2018].
[10] Rianto, Sarwosri. "Rancang Bangun Aplikasi Perangkat Bergerak berbagi foto berbasis android menggunakan API Facebook, Flickr dan Picasa." *Jurnal Teknik Pomits*, Pages 1-4. 2012
[11] Abdul Rahim. "Perancangan Aplikasi E-informasi dan Jadwal Perkuliahan Berbasis Mobile Android." *Jurnal Processor*, 12(1), Pages 1039–1049. 2017.
[12] Wisnu Nurdiyanto. "Perbandingan SOAP dan REST sebagai Web Service." http://pusdiklat.bps.go.id: http://pusdiklat.bps.go.id/index.php?r=artikel/view&id=206, [Oct. 23, 2018].
[13] Perwira, I. P. & B. S. "Implementasi Web Service Pada Integrasi Data Akademik Dengan Replika Pangkalan Data Dikti. Implementasi Web Service Pada Integrasi Data Akademik Dengan Replika Pangkalan Data Dikti." *Jurnal Teknik ITS*, Vol. 1, No. 1 (Sept. 2012). Pages A154-A159, 2017, ISSN: 2301-9271.