p-ISSN : 2443-2210
e-ISSN : 2443-2229

Jurnal Teknik Informatika dan Sistem Informasi
Volume 5 Nomor 3 Desember 2019

Integrasi *Micro-Apps* Individual menjadi *One-Stop Services*: *Application Suite*

http://dx.doi.org/10.28932/jutisi.v5i3.1993

Joseph Sanjaya✉#1, Erick Renata#2, Vincent Elbert Budiman#3, Francis Anderson#4, Mewati Ayub#5

Program Studi Magister Ilmu Komputer, Universitas Kristen Maranatha
Jl. Surya Sumantri No.65, Bandung.

[1]mi1879011@student.it.maranatha.edu
[2]mi1879007@student.it.maranatha.edu
[3]mi1879010@student.it.maranatha.edu
[4]mi1879015@student.it.maranatha.edu
[5]mewati.ayub@it.maranatha.edu

Abstract — **The use of information systems at the university is required to support all the activities of the academic community. Aside from determining the smooth operations, information technology also maintains the competitiveness of the competitors by constantly updating the information technology so as not to miss. Trends in the development of technology will make strong connectivity between universities and stakeholders, governments, and partners. Some of the problems that occur when information systems are not yet fully integrated system, namely separate user management, applications are implemented on different platforms, and non-integrated dashboards for management. The university needs to provide a single and integrated application that is integrated into each of their services. The single and integrated applications are available at the web application/desktop/ mobile multi-platform, for which data is available in real-time, and there is no duplication of data in the system.**

Keywords— **Unified; Single; Multiplatform; Integrated Services**

I. PENDAHULUAN

A. Latar Belakang

Kampus wajib memiliki "Sistem Informasi Perguruan Tinggi yang Terintegrasi" mengapa hal ini menjadi penting? Alasannya utamanya yaitu "Sumber Daya Manusia" dan "*Customer*" perguruan tinggi saat ini adalah orang-orang yang mengerti teknologi informasi (*IT*). Para dosen dan tenaga kependidikan, bahkan mahasiswa menjadikan teknologi sebagai kebutuhan primer. Teknologi digunakan masing-masing pihak untuk dalam menjalankan aktivitas setiap hari dalam rangka menopang produktivitas masing-masing. Teknologi dijadikan sarana utama, baik dari sisi perguruan tinggi atau dari sisi *customer*, proses belajar mengajar sangat tergantung pada ketersediaan teknologi informasi. Penggunaan teknologi informasi sudah menjadi standar pengelolaan perguruan tinggi, selain untuk kelancaran operasional juga untuk menjaga daya saing dengan kompetitor, sangat penting untuk selalu memperbaharui teknologi informasi agar tidak ketinggalan. *Trend* perkembangan teknologi harus tetap dijaga agar konektivitas universitas dengan *stakeholders*, *government* dan *partners* tetap terjaga.

Merupakan tantangan perguruan tinggi saat ini, bagaimana kampus berpikir keras dan mencari strategi untuk dapat mencarikan solusi teknologi informasi yang dapat memenuhi kebutuhan "Sumber Daya Manusia"/ Mahasiswa dan *stakeholders*. Penting bagi universitas untuk tetap menjaga semangat asetnya, untuk tetap memberikan rasa percaya diri yang tinggi *customer* (mahasiswa, dosen, dan tenaga kependidikan) bahwa kampus kita tidak ketinggalan zaman, dan untuk tetap menjaga produktivitas dan meningkatkan daya saing di lingkungan universitas.

Saat ini, potret sistem informasi yang tersedia memiliki beberapa isu:

a. Belum terintegrasi secara menyeluruh, terdapat beberapa aplikasi utama untuk (akademik, *recruitment*, sumber daya manusia, keuangan, *Customer Relationship Management (CRM)*, dan perpustakaan) di bawahnya terdapat beberapa aplikasi penunjang.
b. Masing-masing aplikasi tersebut memiliki manajemen *user* yang terpisah atau *account* untuk masing-masing aplikasi, memiliki tantangan tersendiri untuk bisa mengingat berapa *account* dan *password*, belakangan beberapa aplikasi sudah memiliki *user login* yang terintegrasi.
c. Masing-masing aplikasi di implementasikan pada *platform* berbeda dengan beberapa standar misalnya: berbasis *web*, *desktop application*, *mobile application*.

d. *Dashboard* untuk manajemen yang tidak terintegrasi.

Sangat penting untuk perguruan tinggi menyediakan sebuah aplikasi *Single and Unified Apps* yang saling terintegrasi dalam setiap layanannya, kemudian *Single and Unified Apps* yang tersedia dalam *multi-platform web*/*desktop*/*mobile-apps*, [9],[10] dimana *data*-nya tersedia secara *real-time*, dan tidak terjadi duplikasi *data*. *Single and Unified Apps* untuk beberapa modul utama sebagai berikut:

a. Pengelolaan *data* Sumber Daya Manusia
b. Pengelolaan *data* Akademik
c. Pengelolaan *data* Kemahasiswaan
d. Pengelolaan *data* pendukung Operasional Perguruan Tinggi
e. *Dashboard Management*
f. *Quality Managemens System Tools* untuk bidang pengendalian manajemen mutu universitas
g. Konektivitas langsung dengan pangkalan *data* pemerintah
h. Konektivitas langsung dengan pihak pengelola Standardisasi ISO (International Organization for Standardization)
i. *Audit Reporting Tools* untuk Yayasan / Lembaga Pengawas
j. Konektivitas dan terintegerasi secara utuh dengan *Enterprise Resource Planning System*
k. Pengelolaan *Data* pelanggan *Customer Relationship Management* (*CRM)*
l. Pengelolaan data Alumni

B. Rumusan Masalah

Berikut ini merupakan 2 hal yang menjadi fokus permasalahan utama:

a. Dibutuhkan *one-stop services* untuk menintegrasikan *micro-apps*.
b. Dibutuhkan asumsi *PageRank* untuk menghitung banyaknya relevansi pekerjaan alumni program studi pada *dashboard* aplikasi.

C. Tujuan

Tujuan dibuatnya aplikasi ini antara lain:

a. Menghindari *redundancy Data*.
b. Menghindari *redundancy Activity* (pekerjaan yang diulang).
c. Mempermudah pihak *management* dalam membuat laporan dan mengakses informasi yang akurat dan tervalidasi.

II. KAJIAN LITERATUR

A. Agile

Metode *Agile* merupakan salah satu dari beberapa metode yang digunakan dalam pengembangan *software*. Metode *Agile* adalah jenis pengembangan sistem jangka pendek yang memerlukan adaptasi cepat dan pengembang terhadap perubahan dalam bentuk apapun. Perubahan yang dihadapi oleh metode *agile* bersifat tidak terantisipasi. Jenis perubahan harus direspon mungkin timbul dalam berbagai area yang berbeda [1]. Ramasesh (2001) menggolongkan perubahan menjadi tiga yaitu:

a. *Product maker or output related change*. Wujudnya berupa perubahan dalam permintaan produk, munculnya produk dan konsumen baru, hilangnya produk dan pelanggan yang telah ada;
b. *Factor-market or input-related change*, yaitu adanya sumber bahan mentah baru yang menawarkan keuntungan signifikan, rusaknya keberadaan sumber bahan mentah yang telah ada.
c. *Transformation-process-related change*, yang berupa adanya proses teknologi baru yang radikal penerapan peraturan baru mengenai lingkungan.

Dalam Metode *Agile* interaksi dan personal lebih penting daripada proses dan alat, *software* yang berfungsi lebih penting daripada dokumentasi yang lengkap, kolaborasi dengan *client* lebih penting daripada negosiasi kontrak, dan sikap tanggap terhadap perubahan lebih penting daripada mengikuti rencana. Oleh karena itu, keinginan dari konsumen merupakan prioritas utama yang harus diwujudkan dalam pengembangan menggunakan metode *agile* ini.

B. Semantic Web

Semantic Web bertugas untuk memroses informasi dari segala *level*, entah itu individu, kooperasi, maupun suatu grup, tergantung pada kebutuhannya. *Semantic Web* juga diharapkan dapat memprediksi hal – hal yang didapatkan melalui *data* – *data* dan informasi yang dimiliki. Maka dari itu, untuk mendapatkan *data* yang baik, tentu bukanlah *data* mentah yang digunakan, tetapi memilah dan menggunakan *data* dan informasi yang penting. [4]

Web 3.0 ini maka disebut oleh Berners – Lee sebagai *Semantic Web*, dimana tujuan pada *web* 3.0 adalah pengembangan *web* yang dapat mengerti konteks dan memiliki personalisasi pribadi terhadap *user*. *Semantic Web* dapat membantu *user* melalui "*DSS*" (*Decision support system*),[6] dimana suatu aplikasi dapat membantu *user* dalam melakukan pilihan, "*Information Sharing*", dan juga otomatisasi.

C. Big Data Analysis

Big Data di definisikan oleh 5 properti berikut, yaitu *Volume*, *Variety*, *Veracity*, *Value*, dan *Velocity*. Pertumbuhan *data* telah bersifat eksponensial dalam beberapa tahun terakhir, pernyataan ini dapat didukung oleh pernyataan yang dibuat oleh Eric Schmidt (ketua eksekutif google) "Dari awal peradaban hingga 2003, umat manusia menghasilkan lima *exabyte data*. [2] Sekarang kita menghasilkan lima *exabyte data* setiap dua hari". Setiap perusahaan terlepas dari latar belakang industrinya menyimpan *data*-nya untuk analisis sehingga dapat

mengambil keputusan cerdas bagi perusahaannya. Dua perusahaan pertama yang menghadapi tantangan *Big Data* ini adalah Google dan Facebook, dan kedua perusahaan ini yang mengarah ke fondasi *tools* pengolahan *Big Data* yang dapat digunakan saat ini. *Big Data* saat ini berada di mana-mana, bisa berupa *data* yang dikumpulkan oleh *sensor* di seluruh dunia atau lalu lintas langsung di internet. 204 juta dikirim setiap menit, 2 juta permintaan pencarian diterima oleh Google pada tahun 2012 dan *data* ini berlipat ganda pada tahun 2014. Jumlah *data* yang besar ini dikumpulkan dari berbagai sumber *data* menghasilkan pembentukan *Big Data*. Sekarang untuk menangani masalah *data* terkait ini diperlukan *tools* atau kerangka kerja baru yang dapat menangani *volume data* yang besar ini secara efisien dan hemat biaya. Google pertama kali memperkenalkan kerangka kerja *MapReduce*.

D. MapReduce

MapReduce adalah *model* pemrograman yang membantu dalam komputasi paralel dengan cara yang tidak rumit. *MapReduce* berkomunikasi dengan yarn *untuk* penjadwalan sumber daya.[3] Untuk membuat *model* pemrograman ini berfungsi, pengguna harus menyediakan dua fungsi. Diperlukan fungsi *map* () yang akan diterapkan ke setiap *node* dalam *cluster*, fungsi ini membaca *data* dan menyediakan *output* dalam pasangan kunci - nilai. *Reduce* () mengambil pasangan nilai kunci yang disediakan oleh *map* (), membacanya, dan mencoba merangkum elemen dengan cara tertentu.

III. PERANCANGAN

A. Sumber Data

Sumber *data* diperoleh dengan menggunakan *data* yang diambil dari *micro-services* yang sebelumnya berjalan secara individual. *Data-data* tersebut antara lain:

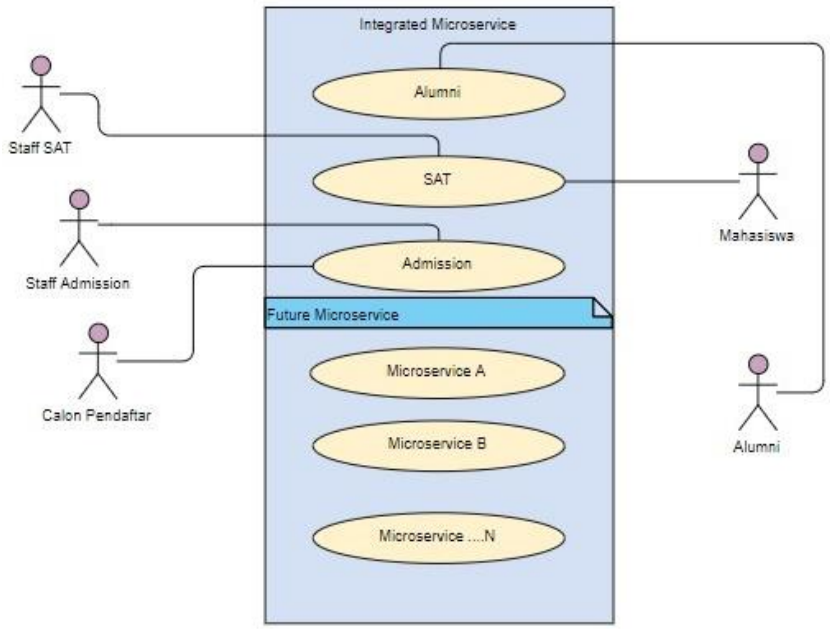
1. *Data Prospect*
2. *Data Student*
3. *Data Employee*
4. *Data* Alumni
5. *Data Organization*

B. Diagram Perancangan Basis Data

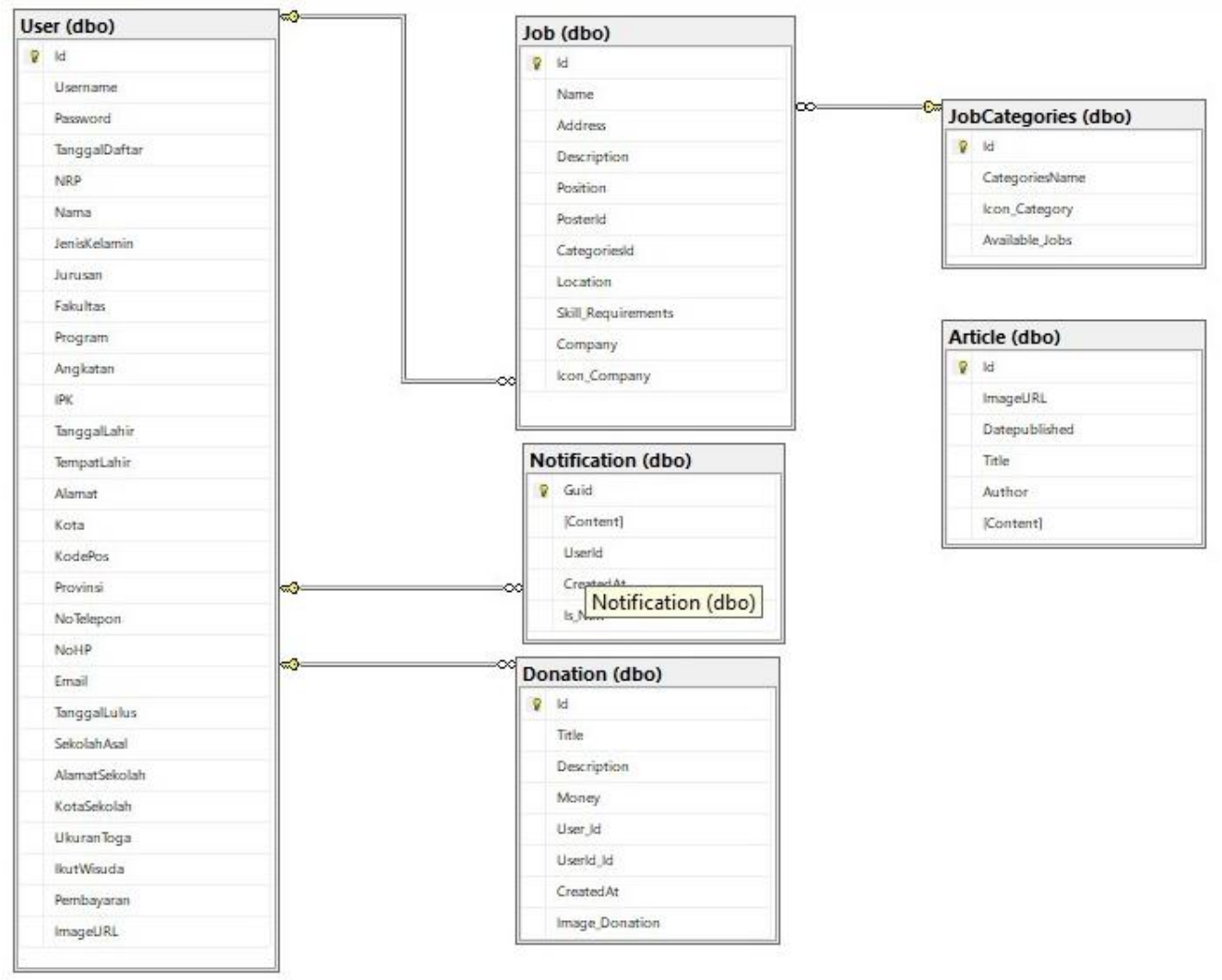
Berikut ini adalah *use case diagram* yang dapat dilihat pada Gambar 1.

Gambar 1 merupakan diagram *use case* pada aplikasi *mobile application* suite, dengan lima *roles* / peran, yaitu *staff* SAT, *staff admission*, calon pendaftar, mahasiswa dan alumni, serta terdiri dari 3 fitur dengan banyak sumber *micro-services*.

Rancangan *database* diagram dapat dilihat pada Gambar 2.



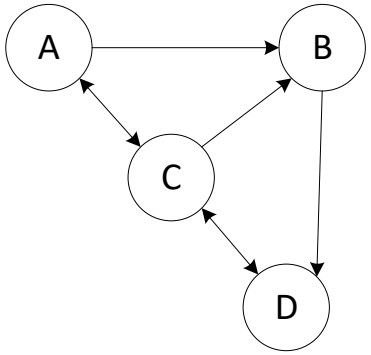
Gambar 1. *Use Case Diagram*



Gambar 2. *Database Diagram*

Gambar 2 merupakan pembuatan relasi untuk *database* yang dipakai pada aplikasi ini.

C. Perancangan PageRank Model untuk Dashboard



Gambar 3. Model *PageRank*

Gambar 3 adalah *model* program *dashboard* untuk menentukan penilaian program studi berdasarkan relevansi pekerjaan. Jika dimisalkan *node C* adalah program studi dan node yang lain adalah kategori pekerjaan dari alumni program studi tersebut. Maka didapatkan tingkatan *PageRank* [7],[11] untuk program studi. Dengan perhitungan 3 kali iterasi seperti pada Tabel I

TABEL I
DATA ITERASI PAGERANK

Node	Iterasi 0	Iterasi 1	Iterasi 2	PageRank
A	1/4	1/12	1.5/12	1
B	1/4	2.5/12	2/12	2
C	1/4	4.5/12	4.5/12	4
D	1/4	4/12	4/12	3

Dengan melihat data pada tabel I, kita akan mendapatkan *node* mana yang paling penting dalam rangkaian *node* tersebut. Sehingga diketahui banyaknya kategori pekerjaan yang relevan dengan program studi tersebut.

D. Perancangan API

API untuk implementasi pada aplikasi Alumni dibagi menjadi 6 *API* utama, yakni:

1. *Users*
   API ini digunakan sebagai akses *data* utama *user*. *User* dapat berupa mahasiswa yang ingin daftar, mahasiswa, maupun alumni. *API User* ini dirancangan sebagai *service* utama yang dapat diakses oleh implementasi terhadap aplikasi lainnya yang terintegrasi. Pada *API* ini terdapat beberapa fitur yaitu:
   - *Login*
     API ini digunakan sebagai *login* pada aplikasi *microservice* manapun yang terintegrasi.
   - *Add User*/*Register*
     API ini digunakan untuk registrasi.
   - *Edit Profile*
     API ini digunakan untuk melakukan *edit profile*.

2. *Jobs*
   API ini digunakan agar *user* dapat melakukan *posting job* agar *user* dapat melihat lowongan yang sedang terbuka. Pada *API* ini terdapat beberapa fitur yaitu:
   - *Get Job By Category Id*
     API ini digunakan untuk mencari pekerjaan berdasarkan kategori.

3. *Job Categories*
   API ini digunakan untuk mengatur lowongan pekerjaan tergantung dari kategorinya serta total ketersediaan lowongan pekerjaan. Pada *API* ini terdapat fitur yaitu:
   - *GET*/*PUT*/*POST Standard*
     API ini digunkan untuk *GET*/*PUT*/*POST standard* pada *job categories*.

4. *Notification*
   API Notifications digunakan agar dapat melihat *update* yang belum dilihat atau dibaca pada aplikasi. Pada *API* ini terdapat fitur yaitu:
   - *GET*/*PUT*/*POST Standard*
     API ini digunakan untuk *GET*/*PUT*/*POST standard* pada *Donations*.

5. *Donation*
   API ini digunakan sebagai donasi oleh alumni untuk beasiswa membantu mahasiswa baru. Pada *API* ini terdapat fitur yaitu:
   - *GET*/*PUT*/*POST Standard*
     API ini digunakan untuk *GET*/*PUT*/*POST standard* pada *Donations*.

6. *Articles*
   API ini digunakan untuk membuat *Article* / *News* agar *User* dapat mendapatkan berita terkini pada halaman *home page* atau *articles*. Pada *API* ini terdapat fitur yaitu:
   - *GET*/*PUT*/*POST Standard*
     API ini digunkan untuk *GET*/*PUT*/*POST standard* pada *Articles*.

API dibuat menggunakan ASP.net dan C#, menggunakan *database SQL Server*, lalu di *hosting* pada: http://bigdatamaranatha.azurewebsites.net.

Seluruh pembuatan *API* akan menggunakan JSON, sebagai contoh *API Login* pada Kode program 1.

API/*users*/*Login*

Tipe:*POST*
Json *Body*:

```
{
   "Username" : "Sample",
   "Password" : "Sample"
}
```

Json *Response*:

```
{
  "NRP": "121212",
  "Id": "5af2c0b0-db87-48dc-8940-
ee6e33dc5a5c",
  "Username": "sample string 3",
  "Password": "sample string 4",
  "TanggalDaftar": "2019-06-
23T16:22:02.207868+00:00",
  "Nama": "sample string 6",
  "JenisKelamin": "sample string 7",
  "Jurusan": "sample string 8",
  "Fakultas": "sample string 9",
  "Program": "sample string 10",
  "Angkatan": 11,
  "IPK": "sample string 12",
  "TanggalLahir": "2019-06-
23T16:22:02.207868+00:00",
  "TempatLahir": "sample string 14",
  "Alamat": "sample string 15",
  "Kota": "sample string 16",
```

p-ISSN : 2443-2210
e-ISSN : 2443-2229


```
  "KodePos": "sample string 17",
  "Provinsi": "sample string 18",
  "NoTelepon": "sample string 19",
  "NoHP": "sample string 20",
  "Email": "sample string 21",
  "TanggalLulus": "2019-06-
23T16:22:02.207868+00:00",
  "SekolahAsal": "sample string 23",
  "AlamatSekolah": "sample string 24",
  "KotaSekolah": "sample string 25",
  "UkuranToga": "sample string 26",
  "IkutWisuda": "sample string 27",
  "Pembayaran": "sample string 28",
  "ImageURL": "sample string 29"
}
```


Kode Program 1. Contoh Request dan Response JSON

Kode program 1 merupakan contoh *request* dan *response* daripada *API Login*, dimana seluruh *API* yang digunakan pada aplikasi ini akan menggunakan standar JSON.

Pada *update* ke depan, akan ditambahkan pula *API* dan *Microservice* untuk implementasi lainnya (Seperti SAT, *Admission*, dsb.). Penambahan dilakukan dengan membuat 1 *SQL Server* dan 1 *Web Application* di setiap *microservices*. Contoh *Microservice dummy* yang dibuat dapat diakses di

- SAT *Dummy*: https://satdummymaranatha.azurewebsites.net/
- *Admission Dummy*: http://admissiondummymaranatha.azurewebsites.net/

E. Perancangan implementasi Agile

Pada masa pembuatan aplikasi, metode *Software Development Lifecycle* yang digunakan yaitu adalah *Agile*. *Standup meeting* dilakukan secara 2 hari sekali untuk *update* terakhir dari pekerjaan pada hari tersebut. *Sprint Planning* dilakukan di awal setiap bulan sekaligus pembagian tugas dan kesulitan daripada tiap tugas yang akan dilakukan masing – masing *developer* (*Story Point*) [8]. Pada perancangan awal terdapat 3 kali *sprint* yaitu setiap *sprint* dijalankan sebulan sekali, diharapkan agar evaluasi dan perubahan yang penting dapat di identifikasi secara cepat. Pada setiap akhir *sprint* juga akan di implementasi *burnt down chart* untuk mengetahui berapa banyak pekerjaan yang tersisa serta yang sudah diselesaikan, serta untuk menentukan apakah suatu fitur masih mungkin untuk diselesaikan tepat waktu atau tidak.

F. Penggunaan OSS pada Aplikasi

Dalam pengembangan aplikasi, *tools* yang digunakan adalah Android Studio 3.5. Android Studio bersifat *open source* karena *developer* dapat menambahkan *plugin* dan *extension* yang dapat digunakan *developer* lain dalam mengembangkan aplikasi. Android Studio didasarkan pada versi dasar IntelliJ yang merupakan *open source*. Orang-orang dari Google menambahkan rasa Android di atas IntelliJ dan merilis ini di bawah lisensi Apache 2 yang juga *open source*. Selain itu Android Studio memiliki *hosted git repository* yang memungkinkan *developer* untuk mengembangkan Android Studio.

Aplikasi dibuat khusus *platform* Android yang mana Android merupakan projek *open source*. Android memakai sistem kernel dari dari Linux, namun *kernel* tersebut telah disesuaikan oleh pengembang Google Android untuk kebutuhan dari perangkat khusus Android. *Kernel* merupakan inti dari sebuah sistem yang dapat diibaratkan seperti struktur *DNA* pada makhluk hidup.

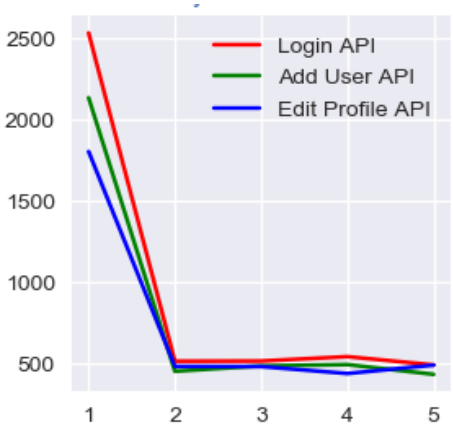
IV. HASIL EKSPERIMEN DAN EVALUASI

A. Tampilan Aplikasi Android

Pada halaman *login* ini pengguna dapat memasukkan nomor NRP dan *password*-nya untuk melanjutkan proses menuju halaman Home. Terdapat validasi dan pengecekan pengguna apakah pengguna tersedia pada *database* pengguna atau tidak. Terdapat *menu Forgot Password* dimana digunakan apabila pengguna lupa *password*-nya. Tampilan dari halaman *login* dapat dilihat pada Gambar 4.

Gambar 4. Halaman *Login*

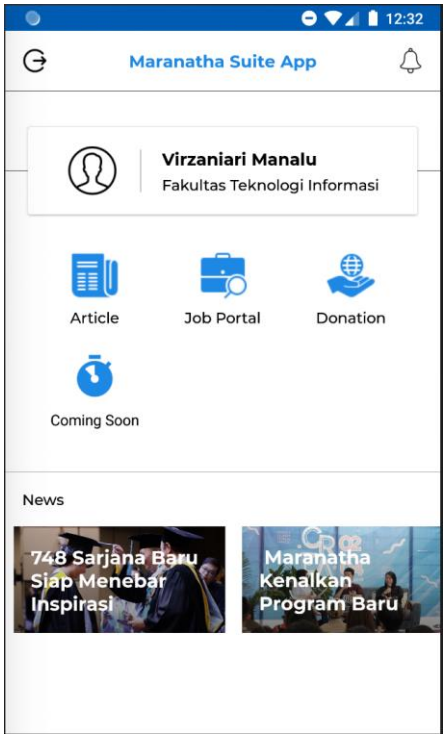
Gambar 1 merupakan tampilan dari halaman *login* yang memiliki fitur untuk *login* dan *forgot password*. Pada halaman ini digunakan *API*, untuk melakukan login, *add user*, dan *edit profile*. Pada gambar 5 dapat dilihat hasil *experiment response time* pada *API* yang bersangkutan.



Gambar 5. Visualisasi experiment API 1

Gambar 5 merupakan hasil *experiment response time* yang dilakukan pada *Login API, Add User API,* dan *Edit Profile API*.

Halaman *Home* merupakan halaman beranda dimana terdapat beberapa *micro service* yang disediakan mulai dari *article, job portal* dan donasi. Terdapat beberapa pengembangan fitur seperti *alumni group* yang akan dikembangkan ke depannya. Tampilan dari halaman *home* dapat dilihat pada Gambar 6.



Gambar 6. Halaman *Home*

Gambar 2 merupakan tampilan dari halaman *home* yang memiliki fitur *micro-services article, job portal* dan donasi.

Halaman *detail article* merupakan halaman dari detail artikel atau berita yang disediakan dari Maranatha. Sumber berita akan berasal *website news.maranatha.edu* sehingga pengguna cukup dengan satu aplikasi dapat menggunakan beberapa fitur di dalamnya. Tampilan dari halaman *detail article* dapat dilihat pada Gambar 7.

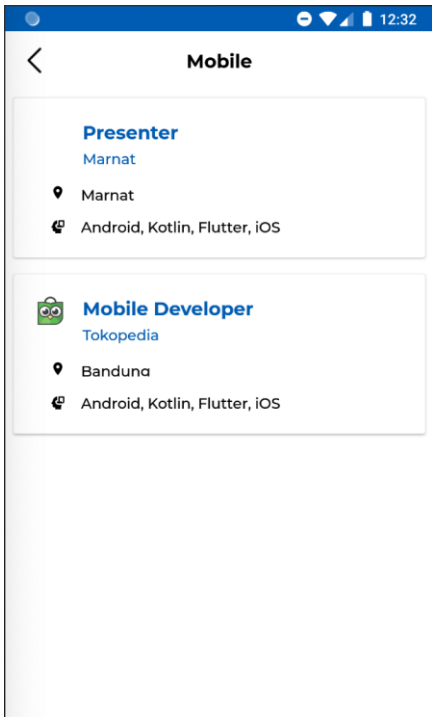
.

Gambar 7. Halaman *Detail Article*

Gambar 3 merupakan tampilan dari halaman *detail article* yang beritanya diambil dari *website news.maranatha.edu*

Halaman *Job Portal* merupakan halaman yang digunakan sebagai penyedia lowongan kerja yang berhubungan dengan jurusan dan fakultas pengguna. Pengguna dapat memilih lowongan kerja berdasarkan kategori yang diminati. Fitur ini merupakan *micro service* dari *website career.maranatha.edu*. Tampilan dari halaman *job portal* dapat dilihat pada Gambar 8.

p-ISSN : 2443-2210
e-ISSN : 2443-2229



Gambar 8. Halaman Job Portal

Gambar 4 merupakan tampilan dari halaman *job portal* yang merupakan *micro-service* dari dari *website career.maranatha.edu*. Pada halaman ini digunakan *API*, untuk melakukan *get job*, *edit job*, dan *post job*.

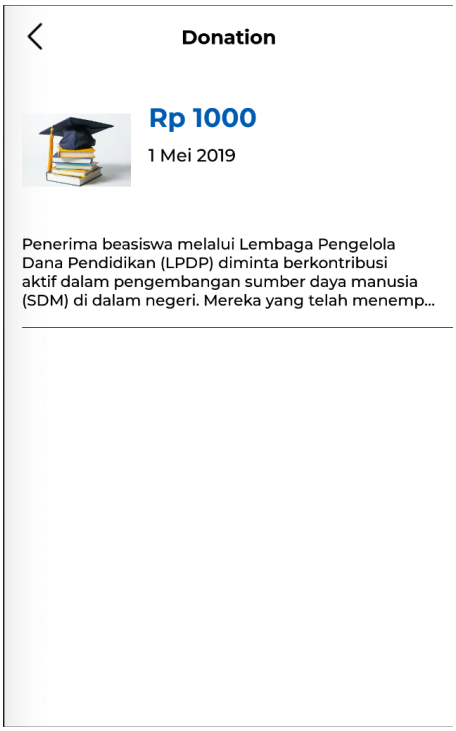
Pada gambar 9 dapat dilihat hasil *experiment response time* pada *API* yang bersangkutan.

Gambar 9. Visualisasi experiment API 2

Gambar 5 merupakan hasil *experiment response time* yang dilakukan pada *Get Job API*, *Edit Job API*, dan *Post Job API*.

Halaman *Donation* merupakan halaman dimana alumni dapat membuka donasi yang diperuntukkan kegiatan atau program seputaran kampus seperti beasiswa mahasiswa ataupun *event* yang berkaitan dengan kampus Maranatha

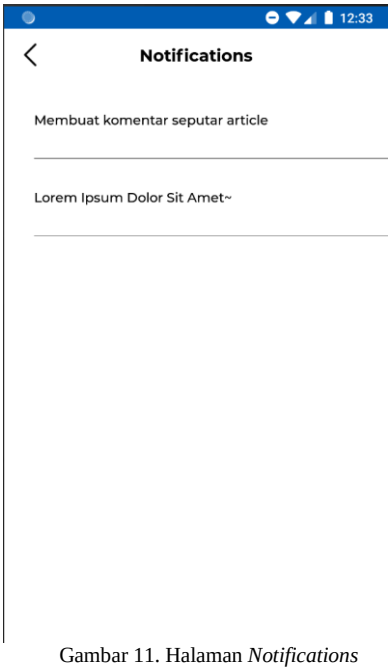
Tampilan dari halaman *donation* dapat dilihat pada Gambar 10.



Gambar 10. Halaman *Donation*

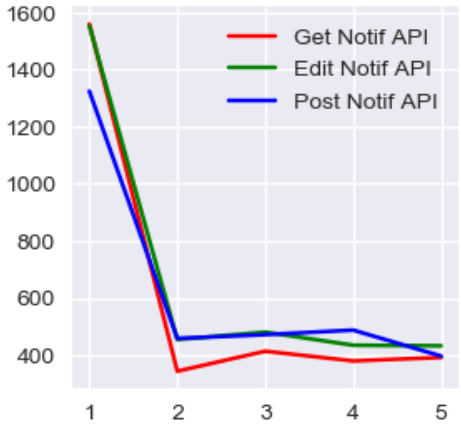
Gambar 6 merupakan tampilan dari halaman *donation* yang merupakan halaman dimana alumni dapat membuka donasi.

Halaman *Notifications* merupakan halaman yang berisi *list* atau daftar pemberitahuan untuk pengguna baik informasi umum seputaran kampus ataupun informasi personal tentang pengguna. Tampilan dari halaman *notifications* dapat dilihat pada Gambar 11.



Gambar 11. Halaman *Notifications*

Gambar 7 merupakan tampilan dari halaman *notification* yang merupakan halaman yang berisi *list* atau daftar pemberitahuan. Pada halaman ini digunakan *API*, untuk melakukan *get notification*, *edit notification*, dan *post notification*. Pada gambar 12 dapat dilihat hasil *experiment response time* pada *API* yang bersangkutan.



Gambar 12. Visualisasi experiment API 3

Gambar 8 merupakan hasil *experiment response time* yang dilakukan pada *Get Notif API*, *Edit Notif API*, dan *Post Notif API*.

Pada setiap pemanggilan *API* pertama kali terdapat "*spike*" atau lonjakan *response time* yang diakibatkan oleh koneksi *API* pertama kali dari tiap *API*. Pemanggilan *API* selanjutnya stabil diantara 397 hingga 540ms. Pemanggilan *API* memakan waktu yang cukup besar (diatas 250ms) berakibat dari digunakannya Server SQL dengan *platform* Microsoft Azure dengan hanya 10 *DTU* (*Tier Standard 0*), serta penempatan *server* yang tidak di Indonesia, melainkan *South East Asia*[5].

DTU merupakan *data transfer unit* yaitu satuan *data* yang dapat diproses di dalam satuan waktu. Semakin besar *DTU*, semakin besar kemungkinan *data* tidak mengalami *bottleneck* saat pemrosesan yang mengakibatkan waktu proses dan pengembalian lama.

B. Evaluasi Aplikasi

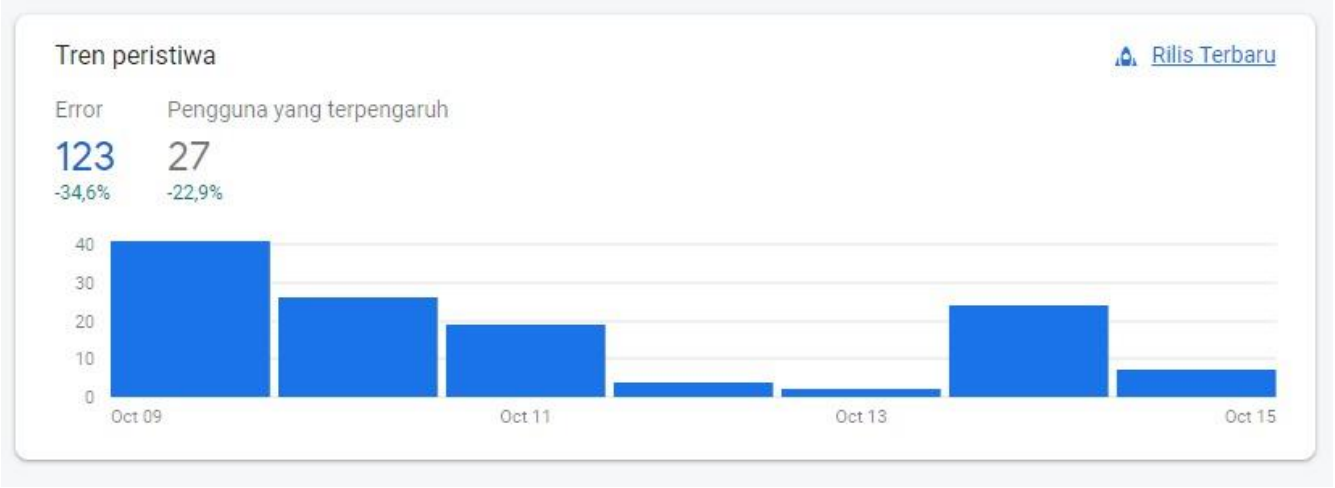
Metode yang digunakan dalam aplikasi untuk menunjukkan *data* statistik seputar penggunaan aplikasi yaitu dengan Crashlytics. Crashlytics merupakan salah satu fitur yang disediakan oleh Firebase untuk memantau aktifitas pada aplikasi. *Data* tingkat stabilitas aplikasi dapat dilihat pada gambar 13.

Untuk metode yang digunakan dalam aplikasi untuk menunjukkan *data* statistik seputar penggunaan aplikasi yaitu dengan Crashlytics. Crashlytics merupakan salah satu fitur yang disediakan oleh Firebase untuk memantau aktifitas pada aplikasi. *Data* tingkat stabilitas aplikasi dapat dilihat pada gambar 13.

Gambar 9 menunjukkan terjadi peningkatan *error* pada tanggal 9 Oktober dikarenakan pada tanggal tersebut ada beberapa fitur yang sedang dikembangkan yang menyebabkan terjadi lonjakan *error*.

Kemudian terjadi penurunan jumlah *error* setelah tanggal 9 Oktober karena aplikasi perlahan mulai stabil dari kemungkinan terjadinya *error*. Dalam seminggu per tanggal 9 - 15 Oktober telah terjadi 123 *error* pada 27 pengguna.

p-ISSN : 2443-2210
e-ISSN : 2443-2229



Gambar 13. Data statistik crash/error yang terjadi pada aplikasi

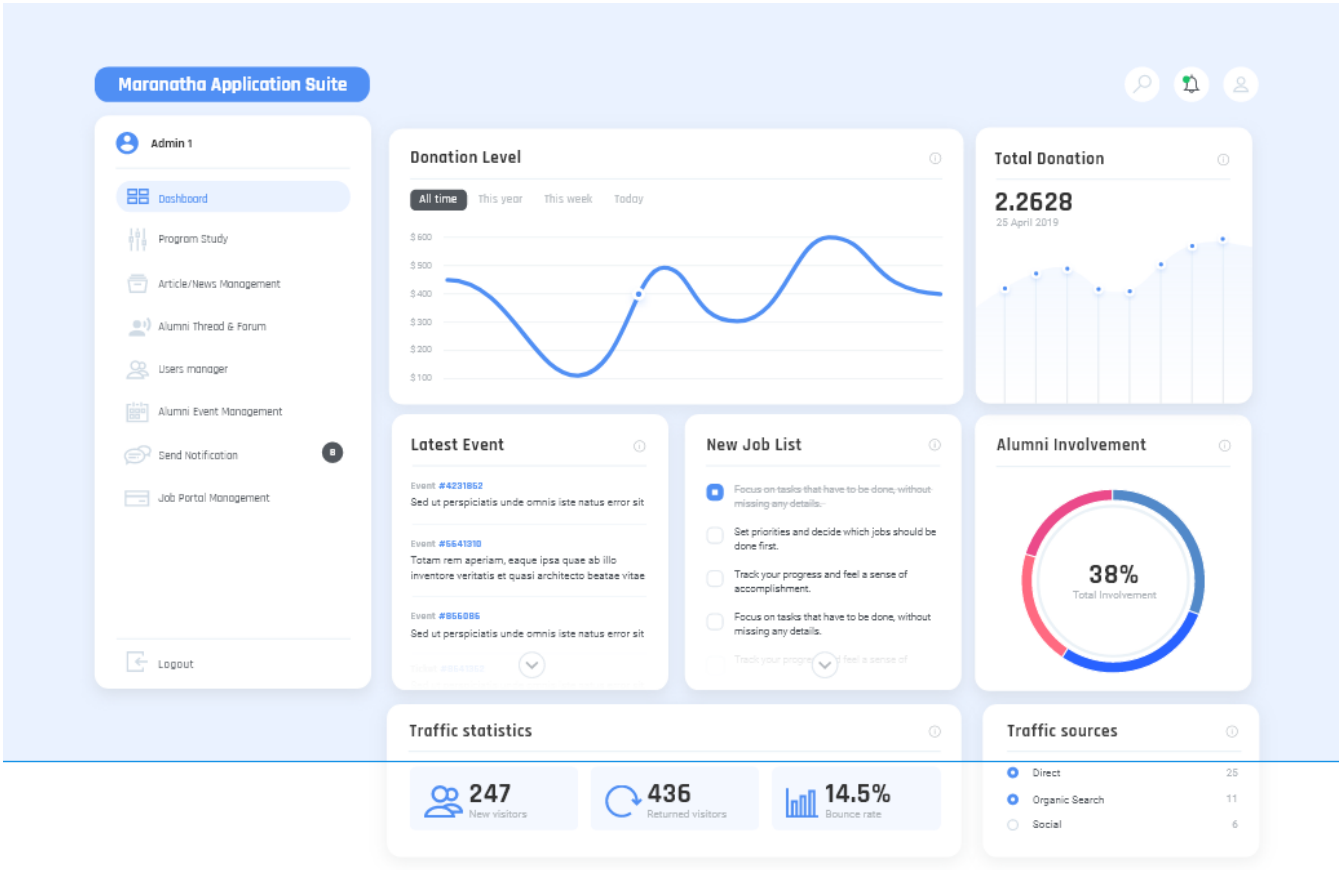
Gambar 14. Data statistik keberhasilan jaringan

Gambar 10 menunjukkan hasil statistik tingkat keberhasilan aplikasi untuk menggunakan jaringan dari tanggal 13 September sampai 14 Oktober 2019.

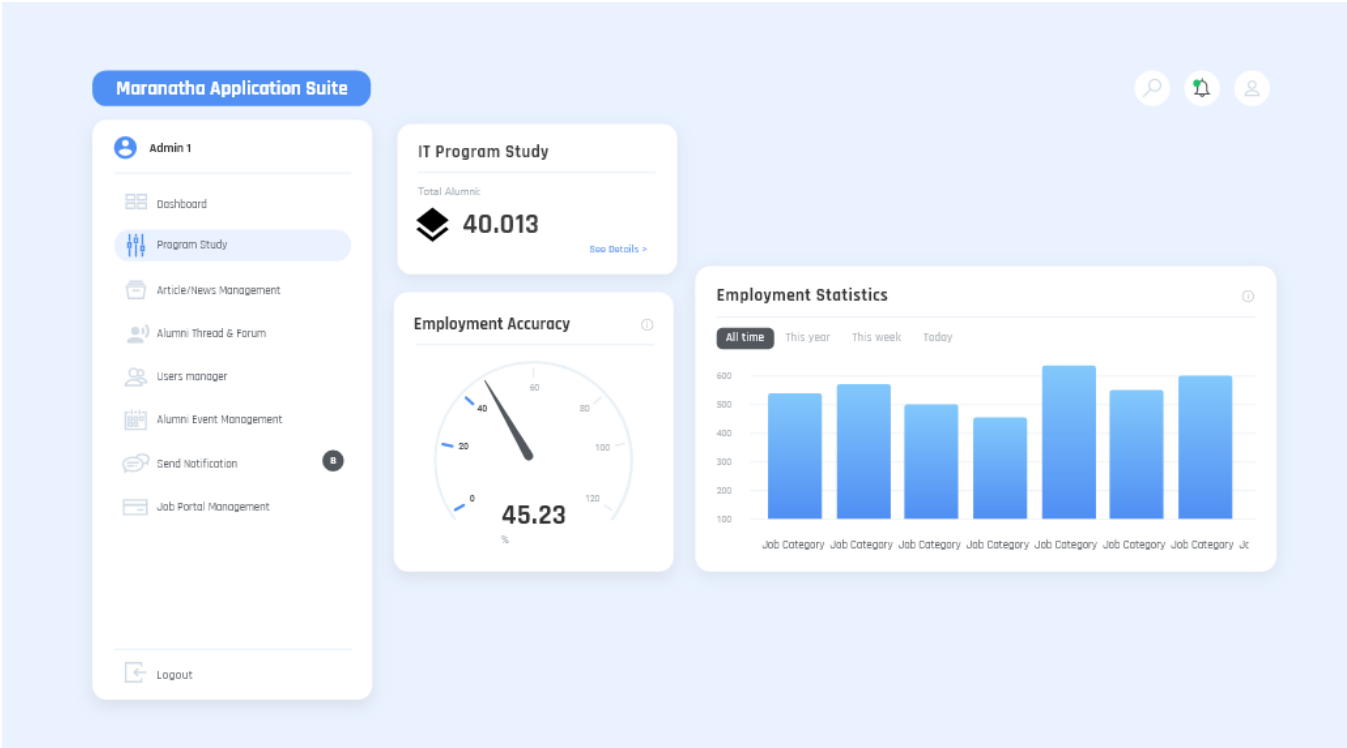
Tingkat keberhasilan bisa disimpulkan cukup baik karena dalam sebulan presentase keberhasilan jaringan minimal masih di atas 99.55%. Pada hal ini keberhasilan jaringan disebabkan oleh faktor *server* yang digunakan dan koneksi pada *device* pengguna.

C. Tampilan Dashboard Aplikasi

Pada halaman *dashboard*, *admin* dapat melihat statistik donasi, *event* alumni terbaru, *job* baru yang baru dipost, dan *traffic* sekitar aplikasi. Tampilan dari halaman *dashboard* dapat dilihat pada Gambar 15.

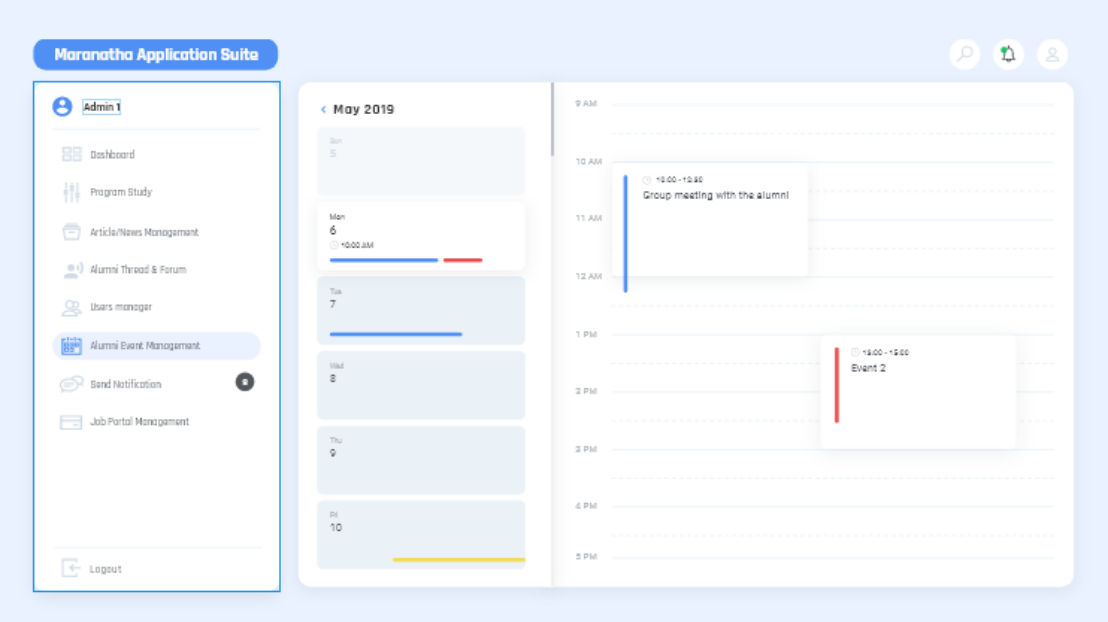


Gambar 15. Halaman *Dashboard*



Gambar 16. Halaman Dashboard Program Studi

p-ISSN : 2443-2210
e-ISSN : 2443-2229

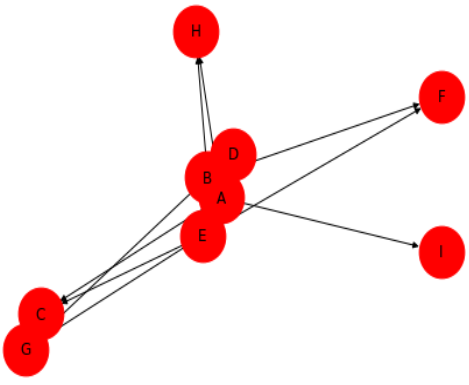


Gambar 17. Halaman *Alumni Event Management*

Gambar 11 merupakan tampilan dari halaman *Web Dashboard* yang merupakan halaman yang dapat melihat statistik donasi, *event* alumni terbaru, *job* baru yang baru dipost, dan *traffic* sekitar aplikasi.

Pada halaman *Dashboard Program Studi*, *admin* dapat melihat *employment statistic* dan *employment relevance meter*. Gambar 12 merupakan tampilan dari halaman *Web Dashboard Program Studi* yang merupakan halaman yang dapat melihat *employment statistic* dan *employment relevance meter*. Pada halaman ini, digunakan metode *PageRank* untuk memperhitungkan *employment statistic* dan *employment relevance meter*.

Percobaan dilakukan dengan membuat *nodes* yang melambangkan pekerjaan alumni seperti pada Gambar 18.



Gambar 18. Simulasi Percobaan Node Pekerjaan

Gambar 18 merupakan grafik simulasi percobaan untuk *node* pekerjaan *Pagerank*. Dengan *data nodes* diatas, digunakan nilai *alpha* untuk *damping factor* 0.5. Maka dihasilkan nilai seperti pada tabel II.

TABEL II
HASIL ANALISIS PAGERANK

Damping Factor: 0.5	
Nodes	*Value*
A	0.08362378260353447
B	0.1358880835477541
C	0.10801378971083697
D	0.08362378260353447
E	0.08362378260353447
F	0.12020915662029508
G	0.11672486989068043
H	0.17421610962745165
I	0.0940766427923784

Dari hasil pada tabel II, diketahui *nodes* yang merupakan ranking tertinggi. *Nodes* tersebut mewakili pekerjaan dari alumni dari setiap program studi. Jika *nodes* pekerjaan tidak sesuai dengan program studi maka *employment relevance meter* rendah.

Pada halaman *Alumni Event Management*, admin dapat melihat kalender jadwal *event* yang akan dilaksanakan sekaligus melakukan penambahan atau perubahan. Gambar 13 merupakan tampilan dari halaman *Alumni Event Management* yang merupakan halaman yang dapat melihat jadwal *event* yang akan dilaksanakan sekaligus melakukan penambahan atau perubahan.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Simplify & Unified IT Maranatha Core atau kami sebut sebagai M*aranatha application suite* bisa menjadi solusi untuk pengembangan Aplikasi Perguruan Tinggi *Modern*. Aplikasi universitas yang tradisional dalam pengertian tidak terintegrasi secara utuh dan tidak mengakomodasikan kebutuhan pihak *internal*, *external* dan para *stakeholder*. akan menjadi masalah dan akan berdampak pada aspek lain khususnya dengan pihak extenal dan *stakeholder*. Dengan *system* ini, diharapkan beberapa berikut bisa dicapai, yaitu r*edundancy data* dan duplikasi pekerjaan bisa dihindarkan, pihak manajemen memiliki kemudahan dalam membuat keputusan, membuat laporan karena informasi terpusat, pihak manajemen dapat mengeluarkan *data* yang *valid* dengan cepat dan akurat. Dengan adanya *engagement* melalui *system*, kita dapat meningkatkan *good experience* dari *Customer Internal*, yaitu mahasiswa, dosen, tenaga kependidikan, alumni, dan pengguna lulusan.

B. Saran

Dengan dukungan komitmen bersama dan keinginan yang besar agar perencanaan bisa terlaksana, kedepan diharapkan pengembangan aplikasi dapat mengimplementasikan semua proses besar yang ada di lingkungan universitas bukan hanya pada cakupan diatas.

UCAPAN TERIMA KASIH

Terima kasih diucapkan sebesar – besarnya kepada para pengajar dan staf Fakultas Teknologi Informasi, Prodi Magister Ilmu Komputer Universitas Kristen Maranatha Bandung atas dukungan dan bantuannya dalam penyusunan laporan penelitian ini.

DAFTAR PUSTAKA

[1] R. Ramasesh, S. Kulkarni, and M. Jayakumar, “Agility in manufacturing systems: an exploratory modeling framework and simulation,” Integrated Mfg Systems, vol. 12, no. 7, pp. 534–548, Dec. 2001.

[2] N. Gupta, R. K. Lenka, R. K. Barik, and H. Dubey, “FAIR: A Hadoop-based Hybrid Model for Faculty Information Retrieval System,” arXiv:1706.08018 [cs], Jun. 2017.

[3] K. S. Candan, P. Nagarkar, M. Nagendra, and R. Yu, “RanKloud: a scalable ranked query processing framework on hadoop,” in Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT ’11, Uppsala, Sweden, 2011, p. 574.

[4] R. Garcia, Ed., Semantic Web for Business: Cases and Applications. IGI Global, 2009.

[5] [A. Bora and T. Bezboruah, “Some Aspects of Reliability Evaluation of Multi Service Multi-Functional SOAP Based Web Services:,” International Journal of Information Retrieval Research, vol. 8, no. 4, pp. 24–38, Oct. 2018.

[6] W. E. Zhang, Q. Z. Sheng, Y. Qin, K. Taylor, and L. Yao, “Learning-based SPARQL query performance modeling and prediction,” World Wide Web, vol. 21, no. 4, pp. 1015–1035, Jul. 2018.

[7] A. Thalhammer and A. Rettinger, “PageRank on Wikipedia: Towards General Importance Scores for Entities,” in The Semantic Web, vol. 9989, H. Sack, G. Rizzo, N. Steinmetz, D. Mladenić, S. Auer, and C. Lange, Eds. Cham: Springer International Publishing, 2016, pp. 227–240.

[8] M. A. Hart, “Agile Product Management with Scrum: Creating Products that *Custome*rs Love by Roman Pichler: Book Reviews,” Journal of Product Innovation Management, vol. 28, no. 4, pp. 615–615, Jul. 2011.

[9] S. Cuccurullo, R. Francese, M. Risi, and G. Tortora, “MicroApps Development on Mobile Phones,” in End-User Development, vol. 6654, M. F. Costabile, Y. Dittrich, G. Fischer, and A. Piccinno, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 289–294.

[10] Y. C. Hsu, Y. S. Wu, T. H. Tsai, Y. P. Chiu, C. H. Lin, and Z. W. Chen, “MicroApp: Architecting Web Application for Non-uniform Trustworthiness in Cloud Computing Environment,” in 2014 IEEE Eighth International Conference on Software Security and Reliability-Companion, San Francisco, CA, USA, 2014, pp. 97–106.

[11] A. N. Langville and C. D. Meyer, Google’s PageRank and beyond: the science of search engine rankings. Princeton, N.J: Princeton University Press, 2006.