DESAIN DAN IMPLEMENTASI TEKNOLOGI *MOBILE BACKEND AS A SERVICE*(MBaaS) PADA APLIKASI LAYANAN WEB

Syahbudin[1)], **Amil A. Ilham**[2)], **Muh. Niswar**[3)]

[1), 2)] *Sistem Komputer, STMIK Handayani*
[2), 3)] *Universitas Hasanuddin*
Email : syahbudin@uin-alauddin.ac.id[1)], amil@unhas.ac.id[2)], niswar@unhas.ac.id [3)]

Abstrak

Penelitian ini bertujuan untuk merancang teknologi *Mobile Backend as a Sevice* dan diterapkan pada jaringan lokal menggunakan layanan web jenis REST.
Pada penelitian ini digunakan metode eksperimental dalam pengembangan arsitektur teknologi MBaaS. Sistem dibagi menjadi tiga aplikasi, yaitu aplikasi *server*, aplikasi lokal dan aplikasi mobile. Sistem diuji menggunakan sistem *point of sales* (POS) pada restoran atau kafe. Selanjutnya melakukan analisis standar pada jaringan lokal dan metode akses layanan. Pengujian selanjutnya adalah uji *response time* dan uji kinerja dari fitur aplikasi.
Hasil penelitian ini menunjukkan bahwa layanan web jenis REST dengan metode akses berbasis CORS serta penggunaan fix IP pada jaringan lokal dapat digunakan untuk membangun MBaaS.

Kata kunci: MBaaS, layanan web, REST, CORS.

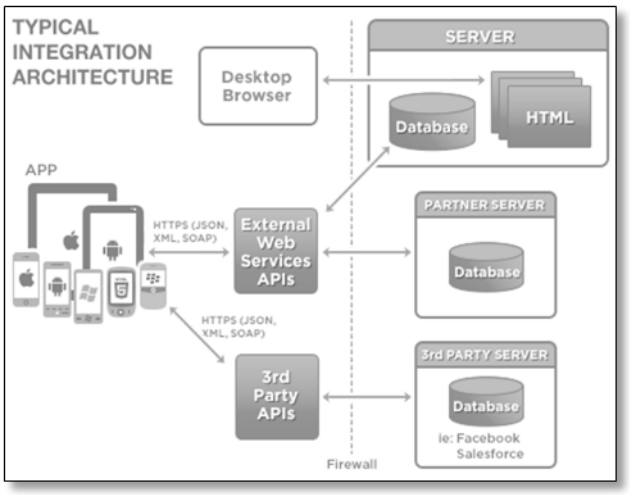
1. Pendahuluan

MBaaS telah muncul sebagai salah satu tren teknologi dan banyak diadopsi oleh perusahaan untuk proses bisnisnya. MBaaS berjalan dengan baik dan dapat diadopsi oleh perusahaan, jika mereka menyediakan mekanisme yang efisien untuk menengahi komunikasi antara aplikasi *mobile* dan sistem bisnisnya. Teknologi yang berkaitan dengan hal itu adalah penggunaan *RESTful web service*[1].

a. Tinjauan Pustaka

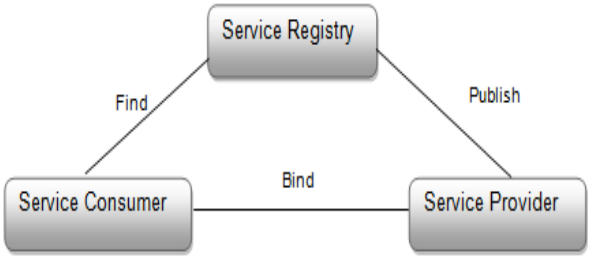
Ajax digunakan pada website yang berinteraksi dengan *server* melalui *javascript* secara *asinkron(background),* sehingga *browser* tidak perlu memuat keseluruhan isi halaman(*page*). Seiring perkembangan teknologi istilah AJAX tidak terbatas pada javascript dan XML, karena saat ini ada *ActionScript*(aplikasi Flash), Java(applet) dan VBScript yang mampu melakukan proses seperti javascript serta *Javascript Object Notation* (JSON) yang dapat merepresentasikan data seperti XML. Oleh karena itu Javascript lebih diasumsikan sebagai bahasa pemrograman yang bersifat *client-side* yang dapat melakukan proses *http-request* dan XML adalah proses representasi data[2].

MBaas adalah sebuah pendekatan untuk menyediakan web dan pengembang aplikasi *mobile* dengan cara menghubungkan aplikasi ke penyimpanan berbasis komputasi awan*(backend cloud storage)* dan pengolahan sementara serta menyediakan fitur-fitur umum seperti manajemen pengguna, pemberitahuan(*push notification*), integrasi jaringan sosial, dan fitur lainnya sesuai kebutuhan pengguna akhir[3].



Gambar 1. *Arsitektur integrasi sistem pada MBaas*

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan(Kurniawan, 2005). Sistem-sistem lainnya berinteraksi dengan *web service* menggunakan pesan SOAP yang umumnya dikirim melalui HTTP dalam bentuk XML[4]. Tetapi secara umum, *web service* tidak terbatas hanya pada standar SOAP saja[5].

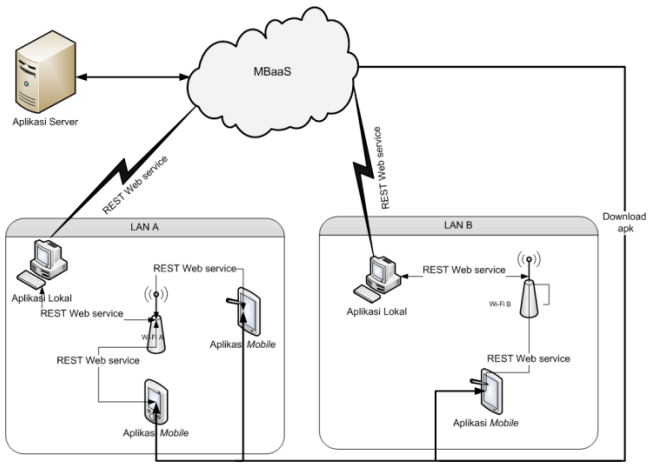


Gambar 2. *Arsitektur Web Service*

b. Penelitian Terkait

- Elgazzar Khalid, dkk. yang berjudul AppaaS: *offering mobile applications as a cloud service*, layanan kepada pengguna akhir diberikan berdasarkan lokasi, waktu akses, data perangkat akses dan tingkatan hak akses. Layanan pada suatu lokasi disimpan pada sistem *cloud* selanjutnya layanan tersebut dapat diakses oleh pengguna akhir. [6].
- Muhsin Shodiq, dkk. *Implementation of Data Synchronization with Data Marker using Web Service Data*, dijelaskan bahwa untuk sinkronisasi data antara server, klien dan pengguna akhir semuanya bisa dikontrol menggunakan *web service*[7].
- Julian Ohrt dan Volker Turau yang berjudul *Building-Linked Location-Based Instantaneous Services Sistem*(BLISS). Distribusi aplikasi dilakukan dengan membaginya kedalam dua area yaitu *server part(SP)* dan *client part(CP)*. Penerapan dari BLISS pada pengguna akhir dengan cara membuat aplikasi pada perangkat mobile yang mereka sebut sebagai *MultiApp*[8].

c. Kerangka Konseptual



Gambar 3. *Kerangka konsep sistem MBaaS*

Fitur-fitur pada lapisan aplikasi

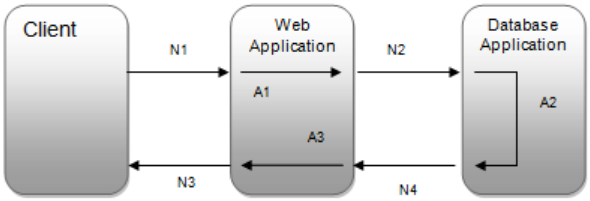
Tabel 1. *Fitur pada aplikasi*

Aplikasi	Fitur
Aplikasi Server	• Manajemen Aplikasi • Manajemen *frontend* dan pengguna akhir
Aplikasi Lokal	• Pemasangan aplikasi • Manajemen menu restoran • Manajemen promosi • Manajemen pelanggan
Aplikasi Mobile	• Pemesanan menu • Kepastian waktu pelayanan

d. Model Pengujian Sistem

Response time

Response time merupakan rentang waktu mulai saat user mengirim *request* ke server hingga server selesai merespon *request* tersebut. Pada aplikasi web, response time didefenisikan sebagai jumlah *network time*(N1, N2, N3,N4) dan jumlah *application time* (A1, A2, A3)[10]. (Pu and M. Xu, 2009).



Gambar 4. *Response time pada aplikasi web*

Uji kinerja dan fitur aplikasi

Untuk proses uji kinerja dan kelengkapan fitur aplikasi, pada penelitian ini menggunakan kuisioner dengan mengajukan pertanyaan terhadap sejumlah responden. Selanjutnya data-data kuisioner diolah menggunakan skala linkert. Format pertanyaan yang diajukan pada kuisioner terlihat pada tabel di bawah ini.

Tabel 2. *Format kuisioner*

No	Pertanyaan	Persepsi				MP (GB)
		TS	KS	S	SS	
1	Proses instalasi mudah dilakukan					
2	Ukuran aplikasi sesuai dengan sumber daya perangkat					
3	Aplikasi sudah bersifat responsive					
4	Validasi login sudah sesuai					
5	Untuk menampilkan feature(menu) dalam aplikasi tidak membutuhkan waktu yang lama					
6	Ketersediaan menu aplikasi sesuai untuk POS kafe/restoran					
7	Aplikasi mudah diterapkan pada kafe/restoran					
8	Aplikasi memudahkan proses pemesanan menu pada kafe/restoran					
9	Aplikasi membantu pengusaha restoran/kafe skala kecil dan menengah					
10	Aplikasi dapat meningkatkan kepuasan pelanggan					

MP = Kapasitas memori pada perangkat pengguna.

2. Pembahasan

a. Hasil Penelitian

Pada penelitian ini dikembangkan tiga aplikasi yang berjalan sesuai dengan layer dan arsitektur pada rancangan penelitian. Aplikasi tersebut adalah aplikasi server, aplikasi lokal dan aplikasi *mobile*.

Aplikasi Server

Fungsi utama dari aplikasi *server* adalah memastikan adanya layanan bagi aplikasi lokal dan aplikasi *mobile*. Layanan tersebut berupa akses untuk mendaftar, mengunduh aplikasi dan proses autentikasi. Aplikasi server berbasis model, view, dan controller (MVC) menggunakan Codeigniter Framework.

Aplikasi Lokal

Aplikasi lokal merupakan aplikasi yang berbasis web, sehingga membutuhkan pemasangan *web server* dan *database server* di dalamnya. Sebagaimana aplikasi server, aplikasi lokal juga dibangun menggunakan Codeigniter framework yang mengadopsi konsep MVC. Perbedaan utama antara keduanya adalah penggunaan *registry* dalam aplikasi. Registry yang digunakan pada sistem MBaaS tersimpan pada aplikasi lokal dalam folder data.
File-file pada folder data adalah format permintaan yang digunakan untuk proses autentikasi pengguna dan format layanan data bagi aplikasi *mobile* dalam bentuk JSON. Agar folder data pada aplikasi lokal dapat diakses oleh aplikasi *mobile* maka ada beberapa hal yang harus dibuatkan standarisasinya yaitu :

- Jaringan Internet dan IP
  Jaringan internet yang menjadi standar dalam MBaaS ini adalah jaringan lokal dengan tipe kelas C. Selain pada sisi jaringan, harus ditentukan pula satu fix IP yang digunakan untuk komunikasi data antara aplikasi lokal dan aplikasi *mobile*.
- Metode Access
  Metode akses yang digunakan adalah REST dengan jenis CORS. Agar akses data pada aplikasi lokal dapat dilakukan oleh aplikasi *mobile* maka harus ada proses *allow* pada semua *header HTTP request*. Format standar dari header tersebut adalah :

  *header('Access-Control-Allow-Origin: *')*

  Sementara di sisi aplikasi lokal, proses *allow* di tentukan pada *httpd.conf* yang berada dalam apache server.

Aplikasi Mobile

Aplikasi pada pengguna akhir dibangun menggunakan ionic framework dan bersifat *client-side scripting.*

b. Analisis Data dan Pembahasan

Uji *Response Time*

Sebelum melakukan ujicoba HTTP Request dengan *Apache Jmeter,* terlebih dahulu harus ditentukan nilai dari 3 variabel di bawah ini, yaitu *Number of Threads, Ramp-Up Periode* dan *Loop Count.*

- *Number of Threads* adalah pengunjung/user yang mengakses dalam satu periode;
- *Ramp-Up Periode* adalah jangka waktu setiap periode
- *Loop Count* : adalah jumlah pengulangan *thread*

Uji coba sistem dan pengambilan data analisis pada penelitian dilakukan pada aplikasi *backend* dan aplikasi *frontend*. Kedua aplikasi ini adalah aplikasi yang memberikan layanan web dalam MBaaS. Pengambilan data dilakukan dengan cara menaikkan jumlah *threads(user)* secara bertahap. Jumlah *threads(user)* yang digunakan pada simulasi adalah 50,100, 200, 300, 400, 500, 600, 700, dan 800. Sementara untuk jumlah *rump up periode* adalah 5 detik dan 10 detik serta *loop count* diset dengan jumlah 1.

Jaringan yang digunakan dalam testing adalah jaringan lokal dengan dua komputer. Satu komputer bertindak sebagai *server* dan yang satunya lagi bertindak sebagai *client*. Perangkat pengujian adalah sebagai berikut :

Tabel 3. *Perangkat pengujian*

Item	Aplikasi Lokal	Aplikasi Server
CPU	Intel atom	Intel Core I3
Memori	1 GB, free 17 MB	2GB, Free 102 MB
Disk	Local Disk 103 GB, Free 82,3 GB	Local Disk 99.9 GB, Free 4.7 GB
Network	54 Mbps	72.2 Mbps
Jarak dengan Wi Fi	1 Meter dan 5 Meter	1 Meter dan 5 Meter

Berdasarkan pengaturan variabel pada *Jmeter* dan jaringan yang digunakan maka diperoleh data hasil pengujian seperti pada tabel di bawah ini.

Tabel 4. *Data pengujian aplikasi server*

RUP	Sample	Jarak ke Wi Fi 1 Meter			Jarak ke Wi Fi 5 Meter		
		E (%)	T (s)	KB/Sec	E (%)	T (s)	KB/Sec
5	50	0	9.9	89.00	0	9.8	88.35
	100	0	19.3	173.58	0	19.3	173.84
	200	0	36.7	330.89	0	35.9	323.23
	300	0	41.3	371.62	0	46	414.62
	400	0	41.2	371	0	41	369.62
	500	0	33	297.24	0	37.9	340.91
	600	0	23.2	209.31	0	26.2	236.2
	700	0.6	21.3	191	0	21.8	196.23
	800	21.4	23.5	177.66	25.37	24.3	176.7
10	50	0	5	45.45	0	5	45.45
	100	0	9	88.96	0	9.8	88.57
	200	0	19.4	175.09	0	19.3	173.96
	300	0	27.1	243.82	0	28.3	254
	400	0	35.4	319	0	35.1	316.45
	500	0	29.6	266	0	28.9	260.3
	600	0	22.9	206.14	0	24.3	218.62
	700	0	24.2	217.84	0	21	189.48
	800	11	23.6	194.61	27.62	22	156.2

Tabel 5. *Data pengujian aplikasi lokal*

RUP	Sample	Jarak ke Wi Fi 1 Meter			Jarak ke Wi Fi 5 Meter		
		E (%)	T (s)	KB/Sec	E (%)	T (s)	KB/Sec
5	50	0	9.9	51.26	0	9.9	51.37
	100	0	19.1	98.74	0	19.3	99.86
	200	0	25.6	132.64	0	36.6	189.42
	300	0	43.4	224.83	0	32.8	169.93
	400	0	39.4	203.81	0	37.2	192.75
	500	0	35.6	184.23	0	32.5	170.49
	600	0	24.4	126.35	0	25.2	130.26
	700	0	22.3	115.35	0	23.1	115.72
	800	17	25.8	120.41	26.1	25.1	116.7
10	50	0	5.1	26.22	0	5	26.1
	100	0	9.9	51.14	0	10	51.59
	200	0	19.5	100.9	0	19.4	100.53
	300	0	28.1	145.32	0	28.1	145.31
	400	0	35.8	185.58	0	37.4	193.74
	500	0	30.2	156.59	0	28.3	146.47
	600	0	24.5	126.73	0	22.9	118.78
	700	0	25.8	133.84	0	25.3	130.95
	800	8.25	24	118.27	7.12	23.6	120.35

Keterangan tabel 4 dan 5 :

- Sample : Jumlah *threads*(*virtual user*)
- E : Persentasi *Error* yang terjadi.
- T : Throughput

Error

Error atau kegagalan dalam *http request response* sangat dipengaruhi oleh besarnya nilai *rump-up periode* yang diberikan. Pada pengujian pertama di aplikasi *server, error* terjadi pada saat *thread* dinaikkan menjadi 700 dan 800. Masing-masing dengan nilai *error* 0.57% dan 21.38% pada *rump-up periode* 5 detik. Sementara untuk pengujian menggunakan *rump-up periode* 10 detik, *error* terjadi pada saat *threads* dipasang 800 dengan nilai *error* 11%. Perbedaan nilai error dengan jumlah *threads* dan *rump-up periode* yang sama pada dua aplikasi *server* dan lokal disebabkan oleh perbedaan ukuran dari kedua aplikasi tersebut. Semakin besar ukuran file yang diminta(*request*) maka akan semakin meningkatkan penggunaan sumber daya pada komputer uji. Oleh karena itu, *error* pada *http request* juga sangat dipengaruhi oleh sumber daya komputer yang digunakan.

Throughput

Nilai *throughput* sangat dipengaruhi oleh *average bytes* dan nilai KB/Sec. Nilai *average bytes* adalah data yang bersumber dari *http response.* Ketika tidak terjadi error(0%) pada HTTP *response*, maka nilai *throughput* cenderung meningkat seiring dengan bertambahnya jumlah *threads*

Kalkulasi dari ketiga variable tersebut adalah sebagia berikut.

Misalkan V=*KB/sec*, AB=*Average bytes* dan T=*throughput* . Maka nilai T = V / AB.

Contoh pada tabel 5 di data pertama diketahui bahwa nilai V=51.26 KB/sec, nilai AB=5302 Byte => 5.18 KB. Maka nilai nilai *throuhput* adalah 9.9 sec. Data yang disajikan pada tabel di atas adalah hasil kalkulasi menggunakan sistem pembulatan.

KB/Sec

Apabila dihitung kecepatan rata-rata akses data(KB/sec) dari kedua tabel di atas, akan terlihat bahwa meningkatnya jumlah *threads* akan berpengaruh pada nilai kecepatan akses. Ini berarti bahwa semakin banyak *http request* yang dilakukan maka rata-rata waktu pelayanan(*response*) akan semakin lambat. Keterlambatan *response* dari sistem tidak berdasarkan pada urutan *threads* atau dengan kata lain bahwa *threads* pertama tidak lebih cepat dari *threads* terakhir ataupun sebaliknya. Kecepatan dan lambatnya proses pada *threads* sangat dipengaruhi oleh latensi dan waktu yang dibutuhkan untuk membangun koneksi. Hubungan antara *threads*, latensi dan waktu koneksi dapat dilihat pada tabel di bawah ini.

Tabel 6. *Latensi dan connect pada aplikasi server*

RUP	Sample	Threads Awal			Threads Akhir		
		NUT	L	C(s)	NUT	L	C(s)
5	50	1	14	8	50	8	3
	100	1	19	12	100	7	3
	200	7	15	5	200	40	11
	300	3	20	8	296	45	13
	400	28	25	11	397	842	5
	500	8	25	16	493	6583	8
	600	4	12	5	563	17043	3046
	700	1	21	13	690	24262	9063
	800	2	31	18	559	23131	9094
10	50	1	82	74	50	11	6
	100	3	25	18	100	12	6
	200	10	43	36	200	8	3
	300	2	16	8	249	8	3
	400	2	23	14	400	8	4
	500	1	17	11	487	5215	3
	600	36	15	6	596	14764	9036
	700	53	14	5	699	15452	9050
	800	21	51	36	710	22758	9018

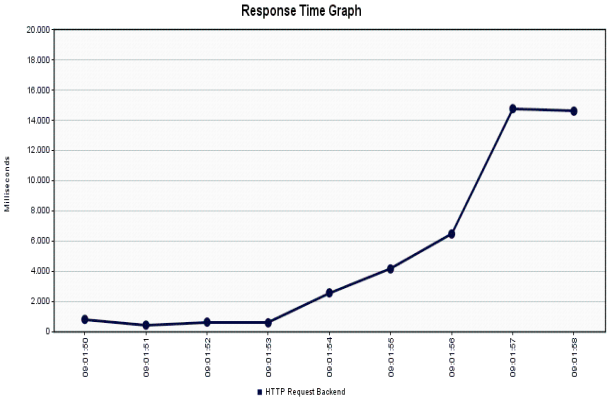
Tabel 7. *Latensi dan connect pada aplikasi lokal*

RUP	Sample	Threads Awal			Threads Akhir		
		NUT	L	C(s)	NUT	L	C(s)
5	50	6	50	13	7	13	6
	100	18	100	43	33	23	6
	200	5	174	52	47	12	6
	300	1	296	11	5	8	3
	400	39	340	27	8	1493	7
	500	16	484	22	5	5536	7
	600	41	587	29	17	16581	9162
	700	2	679	28	5	21199	9083
	800	1	703	21	12	21585	9089
10	50	1	50	26	18	11	6
	100	1	100	19	12	28	24
	200	1	200	20	14	8	4
	300	1	300	13	6	10	6
	400	2	399	23	15	68	4
	500	16	483	24	16	4484	5
	600	2	600	12	5	12654	4
	700	1	607	36	25	14296	9087
	800	1	727	21	13	21752	9019

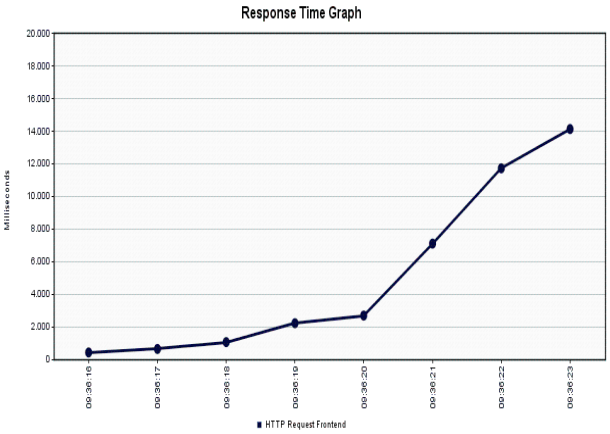
Keterangan tabel 6 dan 7**:**

NUT = Nomor dari thread
L = *Latensi*
C= *Connect*

Pada tabel 6 dan 7 ditemukan bahwa kenaikan nilai latensi sangat dipengaruhi oleh kemampuan dari sumber daya yang digunakan dalam proses pengujian. Hal ini dapat dilihat pada kenaikan nilai *latensi* yang signifikan ketika jumlahnya *threads* lebih besar dari 400. Untuk mengambarkan perubahan itu maka dalam penelitian disajikan dalam bentuk grafik *response time*.



Gambar 5. *Response time aplikasi server*



Gambar 6. *Response time aplikasi lokal*

Uji Kinerja dan Fitur Aplikasi

Aturan penilaian dan pemberian skor dengan menggunakan pendekatan skala Likert pada penelitian ini adalah sebagai berikut :

Tabel 8. *Pemberian skor*

No	Persepsi	Nilai
1	Tidak setuju	1
2	Kurang Setuju	2
3	Setuju	3
4	Sangat Setuju	4

Tabel 9. *Presentase nilai*

No	Persepsi	Nilai
1	0 % - 24.99 %	Tidak setuju
2	25 % - 49.99 %	Kurang Setuju
3	50 % - 74.99%	Setuju
4	80 % -100%	Sangat Setuju

Berdasarkan data dari hasil kuisioner dari 16 responden, maka didapatkan nilai skor tertinggi adalah 4x16 = 64, dan skor terendahnya adalah 1x4 = 4. Selengkapnya mengenai data kuesioner akan disajikan pada tabel di bawah ini.

Tabel 10. *Skor jawaban responden*

No Resp.	Nomor urut pertanyaan										MP (GB)
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
R1	4	3	2	3	3	3	3	3	3	3	1
R2	4	3	2	3	3	3	3	3	3	3	1
R3	3	3	3	3	2	2	2	3	3	3	2
R4	3	3	3	3	3	3	3	3	3	3	1
R5	3	3	3	4	4	3	3	4	3	3	2
R6	3	3	3	3	3	3	3	3	3	3	2
R7	4	4	3	3	4	3	3	3	3	3	8
R8	4	3	3	4	4	3	4	3	4	3	4
R9	4	4	3	3	4	3	2	3	3	2	3
R10	4	3	3	4	4	4	3	4	4	4	16
R11	3	3	3	3	3	3	3	3	3	3	2
R12	4	3	2	3	3	3	3	3	3	3	2
R13	4	2	3	3	4	3	3	3	3	3	2
R14	4	3	3	3	3	3	3	3	3	3	1
R15	3	3	4	4	4	3	4	4	4	4	4
R16	4	3	3	3	3	3	4	4	3	4	16

Berdasarkan persamaan index :
Index% = (Total Skor/Skor tertinggi) x 100
maka didapatkan index untuk setiap pertanyaan sebagai berikut.

Tabel 11. *Index pada tiap pertanyaan*

Nomor Pertanyaan	Jumlah Responde				Total Skor	Index (%)
	TS	**KS**	**S**	**SS**		
P1	0	0	6	10	58	90.63
P2	0	1	13	2	49	76.56
P 3	0	3	12	1	46	71.88
P 4	0	0	12	4	52	81.25
P 5	0	1	8	7	54	84.38
P 6	0	1	14	1	48	75.00
P 7	0	2	11	3	49	76.56
P 8	0	0	12	4	52	81.25
P 9	0	0	13	3	51	79.69
P 10	0	1	12	3	50	78.13

Dari nilai index untuk setiap pertanyaan dapat diketahui bahwa aplikasi *mobile* yang telah dikembangkan sangat mudah dipasang untuk semua perangkat responden. Untuk proses autentikasi atau validasi login bagi pengguna sebagian besar response sangat setuju dengan menu login yang ditampilkan pada aplikasi. Sementara terkait proses pemesanan ataupun meningkatkan kepuasan pelanggan, data menunjukkan bahwa sebagian besar responden juga setuju mengenai fitur yang disiapkan pada aplikasi *mobile.*

3. Kesimpulan

- Dalam membangun MBaaS berbasis layanan yang dioperasikan pada jaringan lokal maka harus ditentukan standar fix IP dan metode akses *service*.
- Metode akses berbasis CORS sangat tepat digunakan ketika aplikasi yang dibangun berbasis *client-side scripting.*
- Kecepatan akses pada aplikasi sangat dipengaruhi oleh kemampuan sumber daya yang digunakan.
- Fitur-fitur yang dibangun pada aplikasi lokal sudah sesuai dengan standar perangkat *mobile* yang memiliki memory lebih besar atau sama dengan 1 GB.

Daftar Pustaka

[1] Mazzara, M and Lucchi, R. *A Framework for Generic Error Handling in Business Processes*. Department of Computer Science, University of Bologna, Italy. Elsevier, 2004. (http://www.sciencedirect.com, diakses pada 23 Februari 2016).

[2] Rochardson, L dan Ruby, S. 2011. *RESTful WebService*. United States of America. O'Reilly Media, Inc., 2011.

[3] Kin Lane. *Overview Of The Backend as a Service (BaaS) Space*. 2013.

[4] *W3. Web Services Architecture, 2004. (http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/)*.

[5] Rochardson, L dan Ruby, S. *RESTful WebService*. United States of America. O'Reilly Media, Inc, 2011.

[6] E, Khalid. Dkk. *AppaaS: offering mobile applications as a cloud service*. School of Computing, Queen's University, Kingston, Canada. Springer. 2013.

[7] Muhsin Shodiq, M. dkk. *Implementation of Data Synchronization with Data Marker using Web Service Data*. Bina Nusantara University, Palmerah Jakarta Barat, Indonesia. Elsevier, 2015. (http://www.sciencedirect.com, diakses pada 23 Februari 2016)

[8] Julian, O dan T. Volker. *Building-Linked Location-Based Instantaneous Services Sistem.* Hamburg University of Technology, Institute of Telematics, Schwarzenbergstr, Hamburg, Germany. *Elsevier.*2014.(http://www.sciencedirect.com, diakses pada 24 Februari 2016).