

Analisis Penerapan *Single Page Application* Menggunakan Teknologi *AJAX* dan *REST API* (Studi Kasus : Sistem Informasi Reservasi Wisma Tamu UKSW)

Adib Luqman Azhari^{#1}, Radius Tanone^{#2}

^{#1}Teknik Informatika, Universitas Kristen Satya Wacana
Jl.Diponegoro no.52-60, Kota Salatiga

¹azhariadib@gmail.com

^{#2}Teknik Informatika, Universitas Kristen Satya Wacana
Jl.Diponegoro no.52-60, Kota Salatiga

²radius.tanone@staff.uksw.edu

Abstract — Developments of web for information system today are really popular. We can access the system from anywhere by internet connection. This paper focus on design of the web application that using *Single Page Application* concept. *Single Page Application* is a concept to design architecture of web application working like desktop application. Using *AJAX* technology the application will provide an interactive user experience, the effective and efficiency usage of bandwidth also be considered. In this paper, the system will evaluate using alpha and beta testing. The result of this paper shows that implement *Single Page Application* concept still make the web bandwidth usage effective and efficient if accessible by user.

Keywords— *Single Page Application*, *AJAX*, *REST API*.

I. PENDAHULUAN

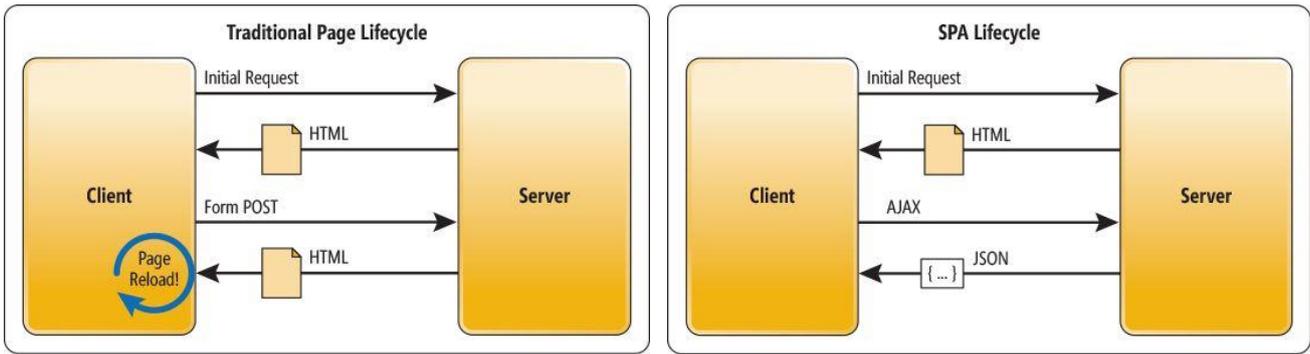
Wisma Tamu UKSW Salatiga merupakan anak perusahaan dari PT.Satya Mitra Sejahtera (SMS), milik YPTKSW. Dalam proses pengolahan data reservasi, Wisma Tamu UKSW telah memiliki sistem berbasis *desktop*. Sistem ini dapat memproses data kamar, tamu, booking hingga proses pembayaran, serta membuat laporan bulanan maupun tahunan. Sistem berbasis *desktop* ini telah digunakan sejak tahun 2013 oleh pihak Wisma Tamu UKSW. Pada akhir tahun 2015 terjadi kerusakan pada sistem berbasis *desktop* tersebut, sehingga tim pengembang dari PT.SMS telah membuat sistem baru yang berbasis web.

Sistem yang baru ini memiliki banyak permasalahan. Karena sistem tersebut berbasis web, penggunaan *bandwidth* yang boros menjadi persoalan utama dalam proses *request* dan *response* data dari *web server*. Selain itu, lama proses *loading* setiap halaman terasa lambat dan berat karena dipengaruhi oleh kecepatan jaringan internet yang digunakan. Setiap proses seperti *input*, edit, hapus, dan cari mengharuskan sistem untuk berpindah ke halaman yang baru dan proses ini menggunakan *bandwidth* yang tidak sedikit. Pengguna yang menggunakan sistem merasa tidak

nyaman karena proses *loading* data dan halaman web terasa lama dan tidak interaktif seperti sistem sebelumnya yang berbasis *desktop*. Sistem *desktop* sebelumnya dikategorikan tidak interaktif karena sistem tidak memberikan interaksi saat pengguna melakukan proses. Contoh saat melakukan proses *input* data, jika data tersebut sebelumnya sudah ada dalam *database*, sistem tidak memberitahukan pada pengguna sehingga redudansi data sering terjadi. Berdasarkan daftar masalah yang dideskripsikan diatas, dapat disimpulkan bahwa permasalahan utama hanya terletak pada borosnya penggunaan *bandwidth* dan lama waktu proses *loading* dari halaman web. Dibutuhkan suatu solusi untuk kedua masalah tersebut.

Single Page Application (SPA) merupakan konsep arsitektur aplikasi web dimana tujuannya membuat cara kerja suatu web bekerja layaknya seperti aplikasi *desktop*. Kelebihan *SPA* dibanding web tradisional pada umumnya terletak pada efisiensi penggunaan data dan lebih interaktif digunakan oleh pengguna. Penerapan konsep *SPA* pada sistem Wisma Tamu UKSW merupakan solusi untuk permasalahan yang dihadapi. Penelitian ini akan berfokus pada implementasi konsep *SPA* pada sistem yang sedang berjalan. Setelah itu akan dilakukan analisis untuk mengukur tingkat efektifitas dan efisiensi penggunaan data dan lama waktu *loading* setiap proses dari sistem setelah dibentuk menjadi *SPA*.

Rumusan masalah yang dapat ditarik dari penelitian ini yaitu, bagaimana merancang sistem Wisma Tamu UKSW yang interaktif menggunakan konsep *SPA* tetapi tidak boros dalam penggunaan *bandwidth* dan tetap cepat saat proses *loading*. Adapun tujuan dari penelitian ini yaitu merancang sistem yang lebih efisien dalam hal penggunaan *bandwidth* dan proses *loading* dibanding sistem sebelumnya dengan menggunakan konsep *SPA*. Manfaat dari penelitian ini diharapkan dapat meningkatkan kualitas pelayanan Wisma Tamu UKSW pada pelanggannya.



Gambar 1. Web Tradisional vs Single Page Application [1]

II. KAJIAN PUSTAKA

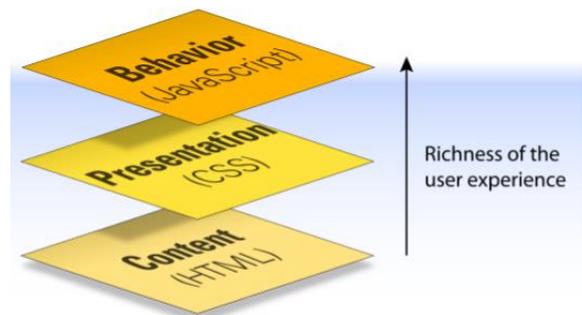
Penelitian yang berjudul “Single Page Application using AngularJS”, permasalahan yang terjadi pada penelitian ini yaitu tidak efisiennya penggunaan *bandwidth* untuk melakukan proses yang berulang-ulang pada halaman web. Solusinya menggunakan *Single Page Application (SPA)* menggunakan teknologi *JavaScript*, *HTML5* dan *AJAX*. Hasil penelitian menunjukkan penggunaan *bandwidth* menjadi efisien dan membuat *loading* web lebih cepat [2]. Selanjutnya, penelitian yang berjudul “Impact of AJAX in Web Applications”, permasalahan yang terjadi yaitu bagaimana mengurangi beban data dari *server* pada sebuah web dan juga tetap membuat web interaktif terhadap pengguna. Penelitian ini menggunakan teknologi *AJAX*. Hasil penelitian menyimpulkan bahwa teknologi *AJAX* memberikan dampak yang besar dalam pengembangan sebuah web dibandingkan dengan web tradisional yang tidak menggunakan teknologi *AJAX* [3]. Selanjutnya, penelitian yang berjudul “Use of AJAX to Improve Usability of Online Information Systems”, pada penelitian ini dijelaskan manfaat dari penggunaan teknologi *AJAX* untuk meningkatkan penggunaan sistem informasi online. Contoh manfaatnya yaitu penanganan kesalahan input, verifikasi pengguna menggunakan *CAPTCHA*, menampilkan data terbaru tanpa harus melakukan *reload* halaman web [4].

Single Page Application (SPA) merupakan konsep arsitektur aplikasi web yang bertujuan untuk membuat sistem lebih interaktif digunakan oleh pengguna. Aplikasi yang menerapkan konsep *SPA* bisa hanya sebatas aplikasi *CRUD (create, read, update, delete)* atau bisa sampai aplikasi yang kompleks, seperti menggunakan *view, library, template, validasi, dll*. Selain interaktif, tujuan utama dari *SPA* yaitu efisiensi penggunaan data saat proses *request* dan *response* dari web *server* [5]. Cara kerja dari *SPA* dimulai dengan proses *loading* halaman web (*HTML, JavaScript, CSS*) dari web *server*, proses ini hanya dilakukan sekali saja saat halaman web pertama kali dibuka. Selanjutnya untuk proses mengolah data, seperti *input, edit, hapus* dan cari tidak perlu lagi melakukan *loading* atau *reload* halaman. Proses pengolahan data akan dikerjakan oleh *AJAX* pada sisi *client* dan *REST API* pada sisi *server*. Perbedaan web

tradisional dan web yang menggunakan konsep *SPA* dapat dilihat pada gambar 1.

API adalah sebuah bahasa dan format pesan yang digunakan oleh program aplikasi untuk berkomunikasi dengan sistem lainnya. *REST API* merupakan sebuah *API* yang dikembangkan menggunakan teknologi *REST*. *REST* sama seperti arsitektur *client/server* web pada umumnya, *request* dan *response* digunakan untuk mentransfer *resources* atau data. Format data yang digunakan oleh *REST* adalah *JSON (JavaScript Object Notation)*. Setiap link *URI* yang terdapat pada *REST API* dianggap sebagai sebuah *resource* [6]. Kelebihan penggunaan teknologi *REST* untuk pertukaran data dibanding *SOAP* yaitu lebih ringan dan cepat. Kelebihan yang dimiliki *REST* tersebut cocok digunakan dalam penerapan *Single Page Application*.

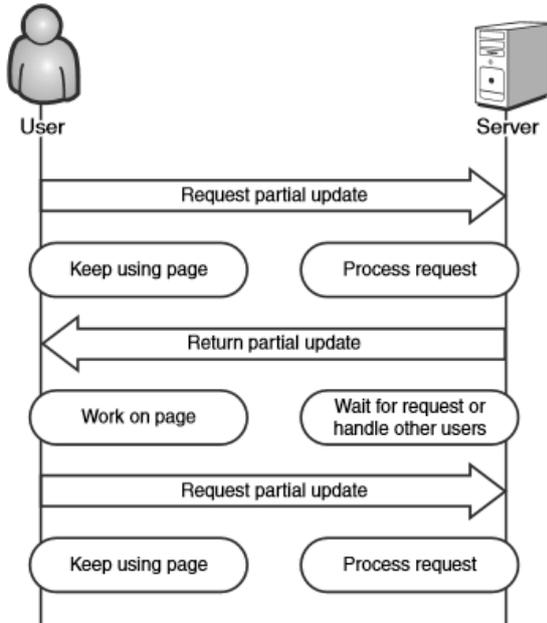
JavaScript merupakan bahasa *script* untuk sisi klien pada *browser*. *JavaScript* didesain untuk membuat aplikasi web yang dinamis dan dapat mengintegrasikan data antara *client* dan *server* [7]. Pada gambar 2 merupakan susunan *layer* dari arsitektur web yang *user experience* pada sisi klien. *JavaScript* merupakan *layer* untuk menangani *behavior* dari suatu web.



Gambar 2. Tiga layer dari web pada sisi klien [8]

AJAX merupakan singkatan dari *Asynchronous JavaScript and XML*. *AJAX* merupakan teknologi baru dalam dunia web yang menggunakan teknologi dari *JavaScript* untuk mengirim file *XML* ke halaman web tanpa harus melakukan *reload* halaman web tersebut [9]. *AJAX* menggunakan beragam teknologi web untuk menghasilkan

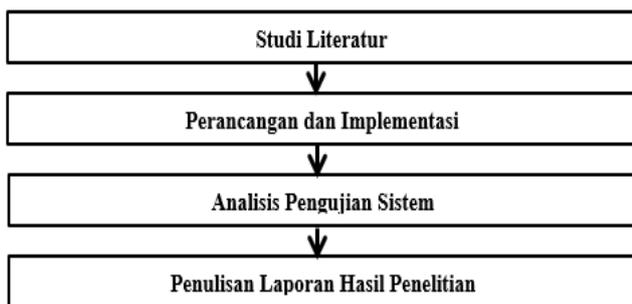
aplikasi web yang responsif dan membuat *user experience* dari aplikasi web hampir seperti aplikasi desktop [10]. *AJAX* hanya berjalan pada sisi *client* atau di *web browser*, oleh karena itu *AJAX* membutuhkan sebuah *API* untuk pengolahan datanya pada sisi *server*. *REST API* merupakan teknologi yang cocok dikombinasikan dengan *AJAX*. Cara kerja dari teknologi *AJAX* dapat dilihat pada gambar 3.



Gambar 3. Model request/response Asynchronous AJAX [11]

III. METODE PENELITIAN DAN PERANCANGAN SISTEM

Penelitian yang dilakukan dilakukan melalui tahapan penelitian yang terbagi dalam 5 tahapan, yaitu : 1) Studi literatur, 2) Perancangan dan Implementasi, 3) Analisis pengujian sistem, 4) Penulisan laporan hasil penelitian.



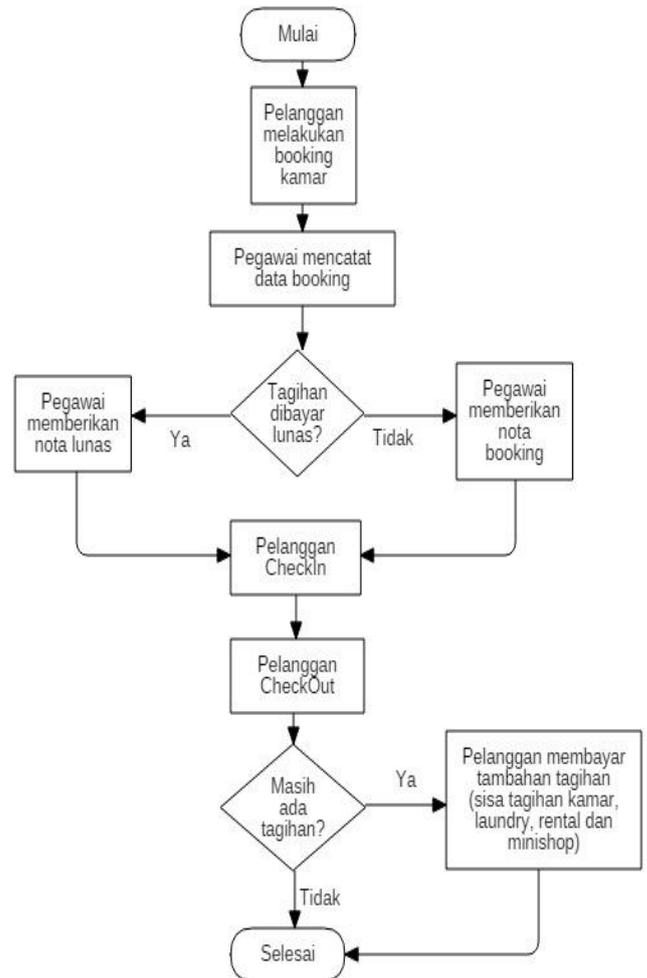
Gambar 4. Tahapan Penelitian

Tahapan penelitian pada Gambar 4 dapat dijelaskan sebagai berikut :

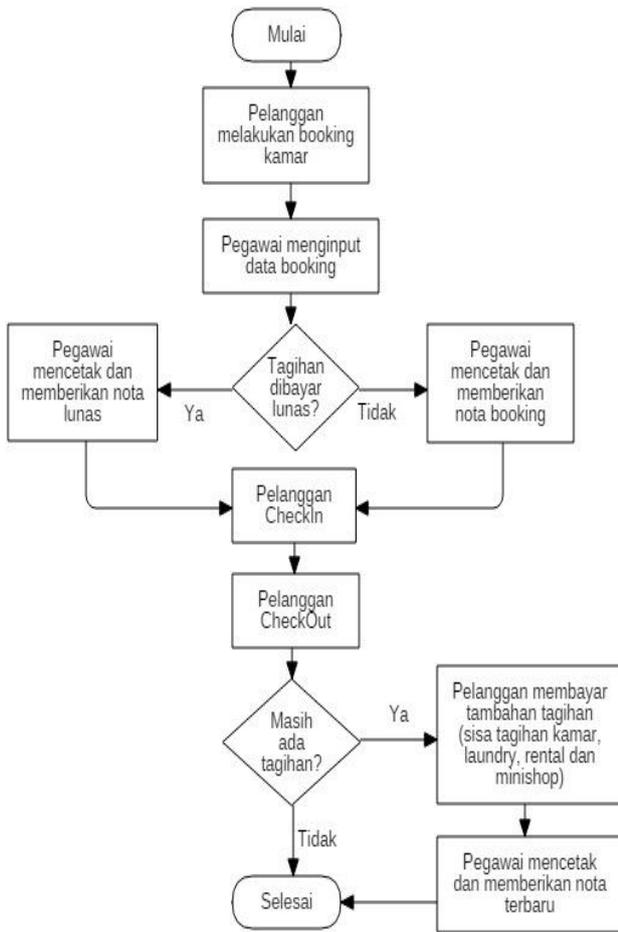
1. *Tahap pertama* adalah studi literatur. Dalam tahap ini dilakukan kajian pustaka terkait teknologi *SPA* yang

akan digunakan dan melakukan review dari penelitian-penelitian yang terkait.

2. *Tahap kedua* adalah perancangan dan implementasi. Dalam tahap ini dilakukan perancangan konsep *SPA* yang nantinya akan diimplementasikan pada sistem Wisma Tamu UKSW. Teknologi *SPA* yang digunakan yaitu *AJAX* pada sisi *client* dan *REST API* pada sisi *server*. Metode pengembangan perangkat lunak yang digunakan adalah Model Prototipe.
3. *Tahap ketiga* adalah analisis pengujian sistem. Dalam tahap ini dilakukan analisis untuk mengukur tingkat efektifitas dan efisiensi penggunaan data dan lama waktu *loading* setiap proses dari sistem Wisma Tamu UKSW yang sudah berbentuk *SPA*.
4. *Tahap keempat* adalah penulisan laporan hasil penelitian. Dalam tahap ini dilakukan dokumentasi proses dari tahap awal sampai tahap akhir dalam bentuk tulisan.



Gambar 5. Proses bisnis sebelum menggunakan sistem

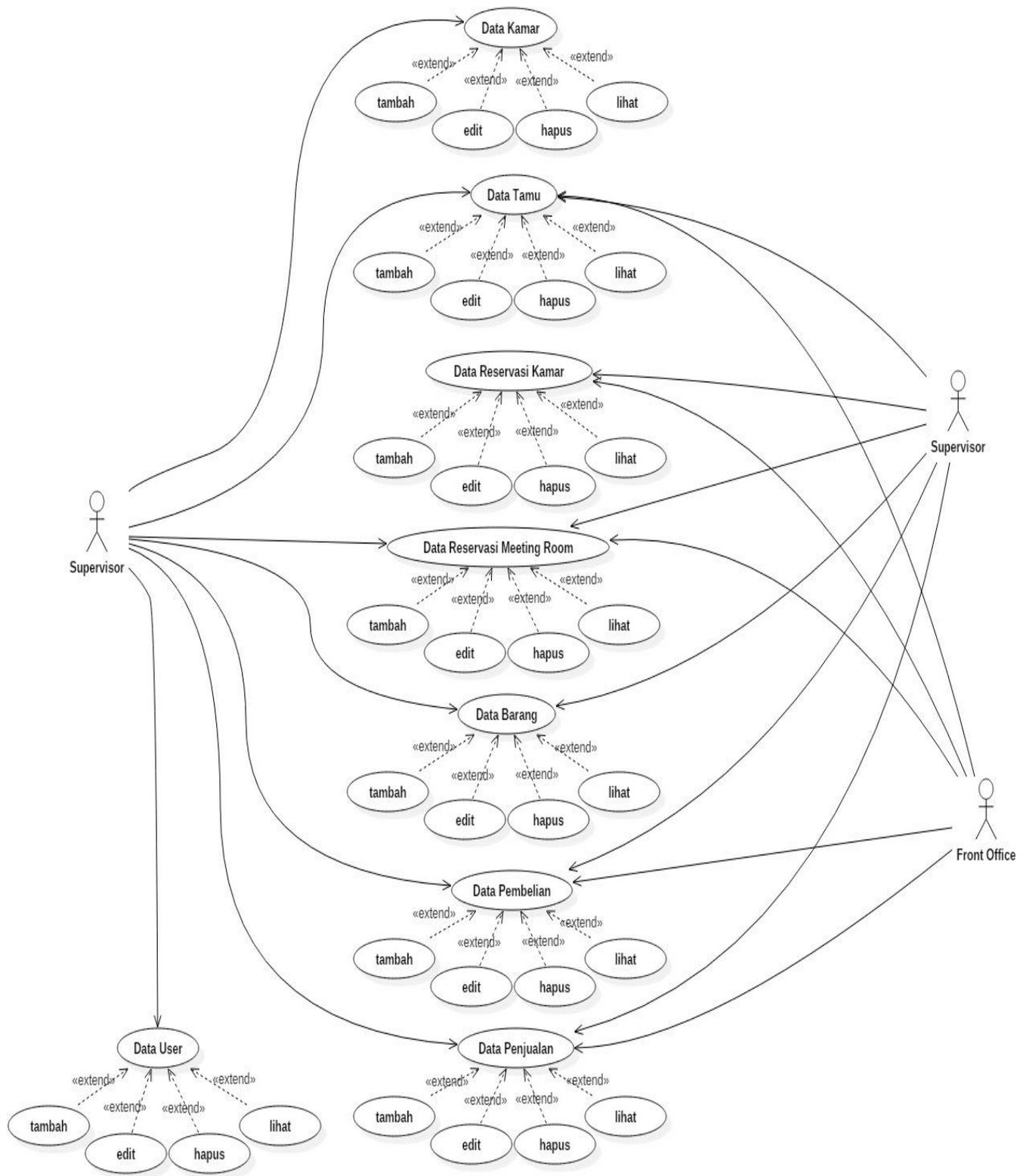


Gambar 6. Proses bisnis setelah menggunakan system

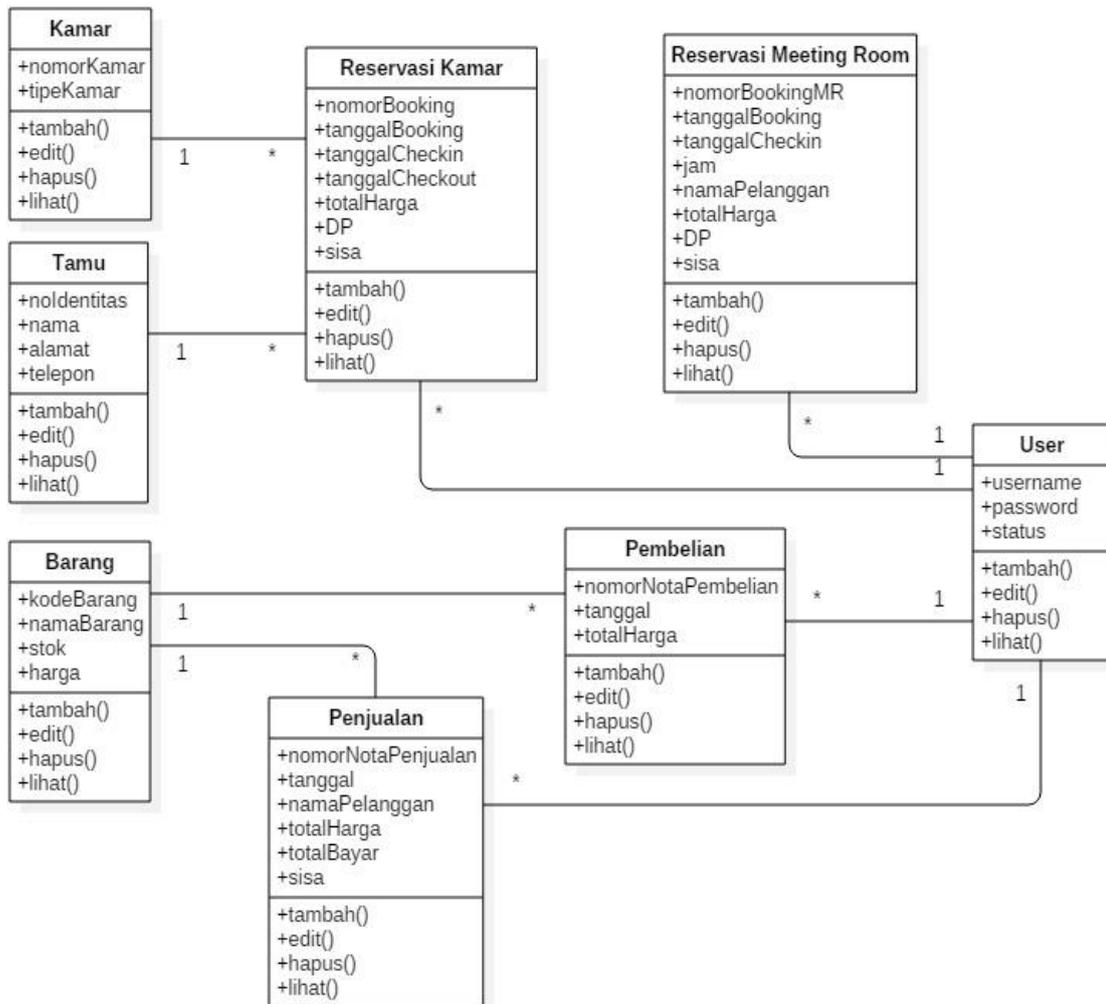
Gambar 5 menunjukkan proses bisnis lama yang terjadi di Wisma Tamu UKSW sebelum menggunakan sistem. Gambar 6 menunjukkan proses bisnis baru yang terjadi di Wisma Tamu UKSW setelah menggunakan sistem. Dapat dilihat perbedaan antara kedua proses bisnis tersebut yaitu saat pelanggan melakukan booking pegawai harus menginput data ke dalam sistem. Saat pelanggan *checkout*, pegawai dapat mencetak ulang nota terbaru jika terdapat tambahan tagihan didalamnya.

Sistem Wisma Tamu UKSW merupakan sistem yang dibangun menggunakan teknologi web. Bahasa pemrograman menggunakan *PHP* dan *database* menggunakan *MySQL*. Arsitektur sistem ini masih berbentuk web tradisional karena setiap proses seperti input, edit, hapus, dan cari mengharuskan sistem untuk berpindah ke halaman yang baru. Hal ini menyebabkan borosnya penggunaan *bandwith* saat proses *request* dan *response* data dari *web server*.

Pada gambar 7 merupakan desain *use case* sistem Wisma Tamu UKSW. Terdapat 3 aktor dalam sistem ini yaitu *admin*, *supervisor* dan *front office*. Ketiga aktor tersebut memiliki hak akses yang berbeda-beda untuk setiap fungsi yang ada dalam sistem. *Admin* merupakan pengguna yang dapat mengakses semua fungsi dalam sistem, pengguna ini hanya dimiliki oleh pihak PT.SMS. *Supervisor* merupakan pengguna untuk seorang *supervisor* wisma tamu yang ditunjuk oleh PT.SMS, pengguna ini memiliki akses yang terbatas seperti pada gambar 7. *Front Office* merupakan pengguna untuk karyawan yang ditempatkan di *lobby* wisma, tugas pengguna ini hanya menginput data tamu, reservasi kamar (*booking*, *checkin*, *checkout*), reservasi *meeting room*, pembelian dan penjualan barang.



Gambar 7. Use Case Sistem Wisma Tamu UKSW

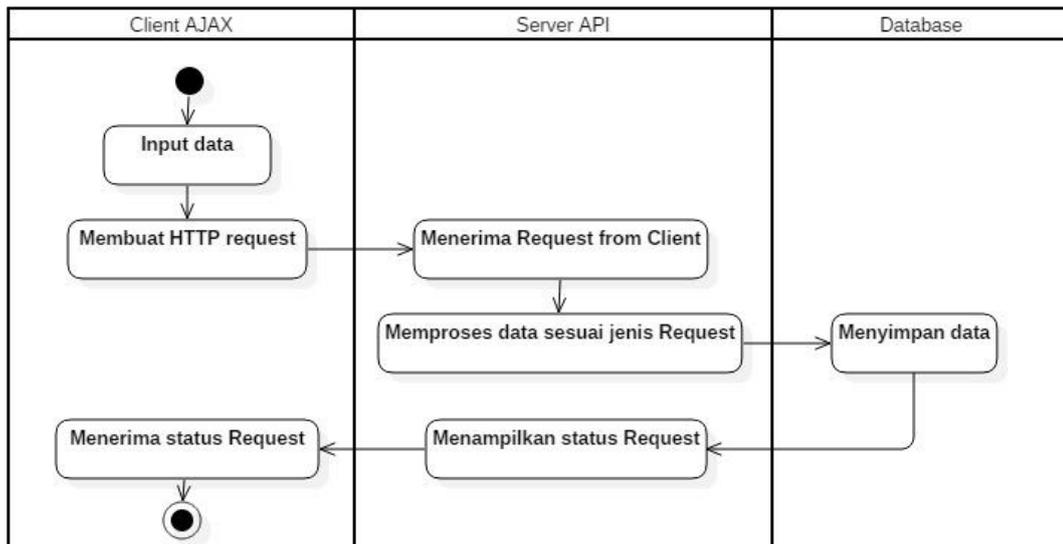


Gambar 8. Class Diagram Sistem Wisma Tamu UKSW

Pada gambar 8 merupakan desain *class diagram* sistem Wisma Tamu UKSW. Terdapat 8 entitas yang masing-masing memiliki fungsi tambah, edit, hapus dan lihat. Setiap entitas saling berkaitan, seperti entitas Kamar dengan entitas Reservasi Kamar atau entitas Barang dengan entitas Penjualan.

Pada gambar 9 merupakan desain *activity diagram* proses request data dari AJAX (client) ke REST API (server).

Proses ini digunakan oleh semua fungsi di sistem Wisma Tamu UKSW. Proses dimulai saat *client* membuat *HTTP Request* dan mengirimnya ke *REST API*. *Server* kemudian memproses *request* tersebut dan menyimpan datanya ke database, kemudian menampilkan status dari *request* dalam format *JSON*. *Client* kemudian menangkap status dari *request* tersebut agar dapat mengetahui apakah prosesnya berhasil atau gagal.

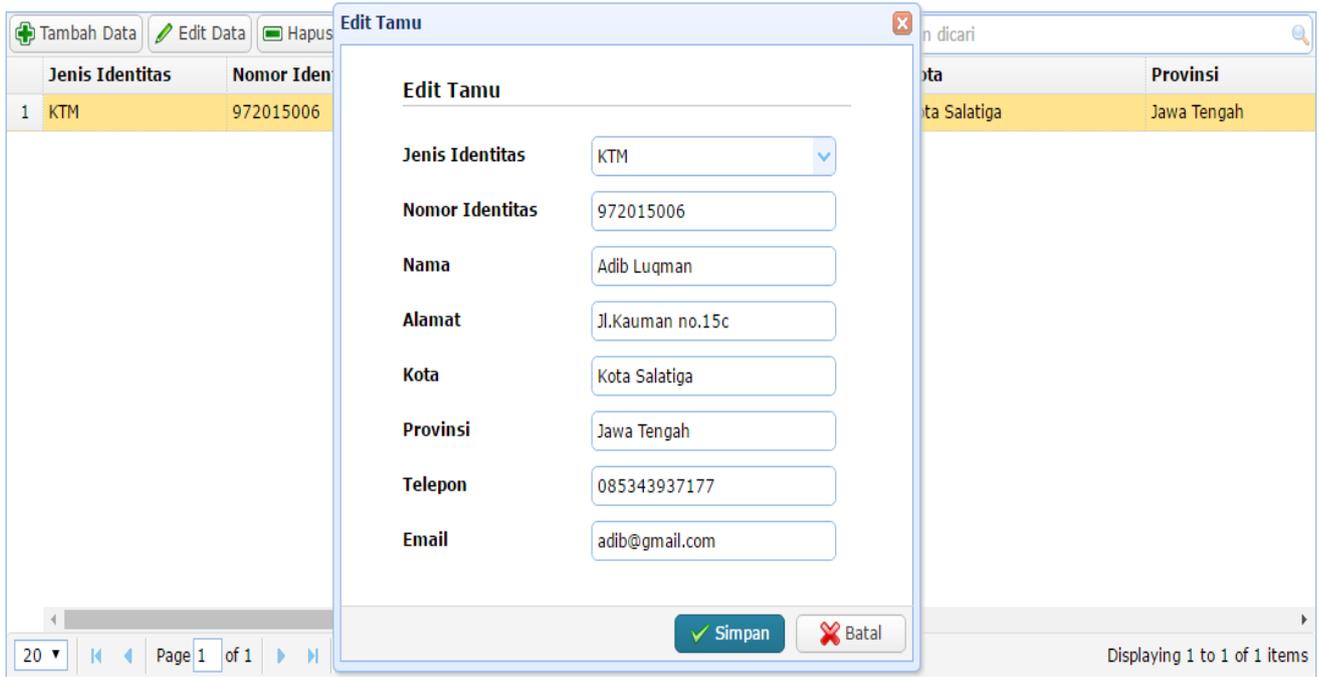


Gambar 9. Activity Diagram Sistem Wisma Tamu UKSW

Masukkan Nama Tamu yang akan dicari						
Jenis Identitas	Nomor Identitas	Nama	Alamat	Kota	Provinsi	
1	KTM	972015006	Adib Luqman	Jl.Kauman no.15c	Kota Salatiga	Jawa Tengah

20 Page 1 of 1 Displaying 1 to 1 of 1 items

Gambar 10. Komponen Datagrid framework jEasyUI



Gambar 11. Komponen *Dialog framework jEasyUI*

Implementasi SPA pada sistem Wisma Tamu UKSW menggunakan teknologi yang sudah lama ada tetapi masih sangat populer digunakan sampai saat ini, *AJAX* dan *REST API*. Teknologi *AJAX* digunakan pada sisi *client* dalam hal ini browser yang digunakan oleh pengguna. Agar mempermudah dan mempercepat proses pengembangan *interface* dan fungsi *CRUD SPA*, maka digunakan *framework javascript* yaitu *jQuery EasyUI* sering disingkat *jEasyUI*. *Framework* ini telah menyediakan semua kebutuhan untuk mengembangkan sistem SPA baik itu dari komponen *interface* maupun fungsi *CRUD SPA*.

Dalam proses *view data* atau menampilkan data digunakan komponen *datagrid*, seperti terlihat pada gambar 10. Selain menampilkan data, komponen ini juga telah mendukung fungsi *paging* atau *pagination* dimana sistem menampilkan data per blok baris (jumlah baris data tergantung dari konfigurasi) agar proses loading data tidak terlalu berat.

Proses tambah (*create*), edit (*update*) dan hapus (*delete*) data menggunakan komponen *dialog*, seperti terlihat pada gambar 11. *Dialog* digunakan untuk menempatkan komponen-komponen inputan dalam proses tambah dan edit seperti *textbox*, *combobox*, *button*, dan lain sebagainya. *Dialog* tampil ke pengguna saat fungsi yang terkait memanggilnya, seperti fungsi edit akan menampilkan *dialog* edit. Saat *dialog* dipanggil atau muncul, sistem tetap berada

di halaman yang sama. Hal ini membuat sistem tidak harus berpindah halaman lagi.

Pada sisi *server*, digunakan teknologi *REST API* sebagai layanan (*services*) penerima dan penyedia data untuk setiap *request* dari *AJAX (client)*. Pada penelitian ini *REST API* dibangun menggunakan bahasa pemrograman *PHP*. *AJAX* berkomunikasi dengan *REST API* melalui proses *request*, setiap *request AJAX* memiliki *HTTP Method* yang menandakan jenis fungsi *CRUD* yang dikirim.

TABEL I
JENIS *HTTP METHOD AJAX* DAN *REST API*

Fungsi	HTTP Method	
	AJAX	REST API
Tambah	POST	POST
Edit	PUT	PUT
Hapus	DELETE	DELETE
Cari / Lihat	GET	GET

Pada tabel I menunjukkan jenis fungsi beserta *HTTP Method* yang digunakan di *AJAX (klien)* dan *REST API (server)*. Pada umumnya jenis method antara klien dan server sama saja, seperti fungsi tambah akan menggunakan method *POST*. Standar dari jenis *method* ini telah diatur oleh *World Wide Web Consortium (W3C)*.

```
{
  "total": "3",
  - "rows": [Array[3]
    - 0: {
      "idTipeKamar": "3",
      "kodeTipeKamar": "DXL",
      "tipeKamar": "Deluxe",
      "hargaKamar": "205000"
    },
    - 1: {
      "idTipeKamar": "1",
      "kodeTipeKamar": "ECO",
      "tipeKamar": "Economy",
      "hargaKamar": "140000"
    },
    - 2: {
      "idTipeKamar": "2",
      "kodeTipeKamar": "SPR",
      "tipeKamar": "Superior",
      "hargaKamar": "165000"
    }
  ],
}
```

Gambar 12 Salah satu fungsi menampilkan data (Method GET) pada REST API

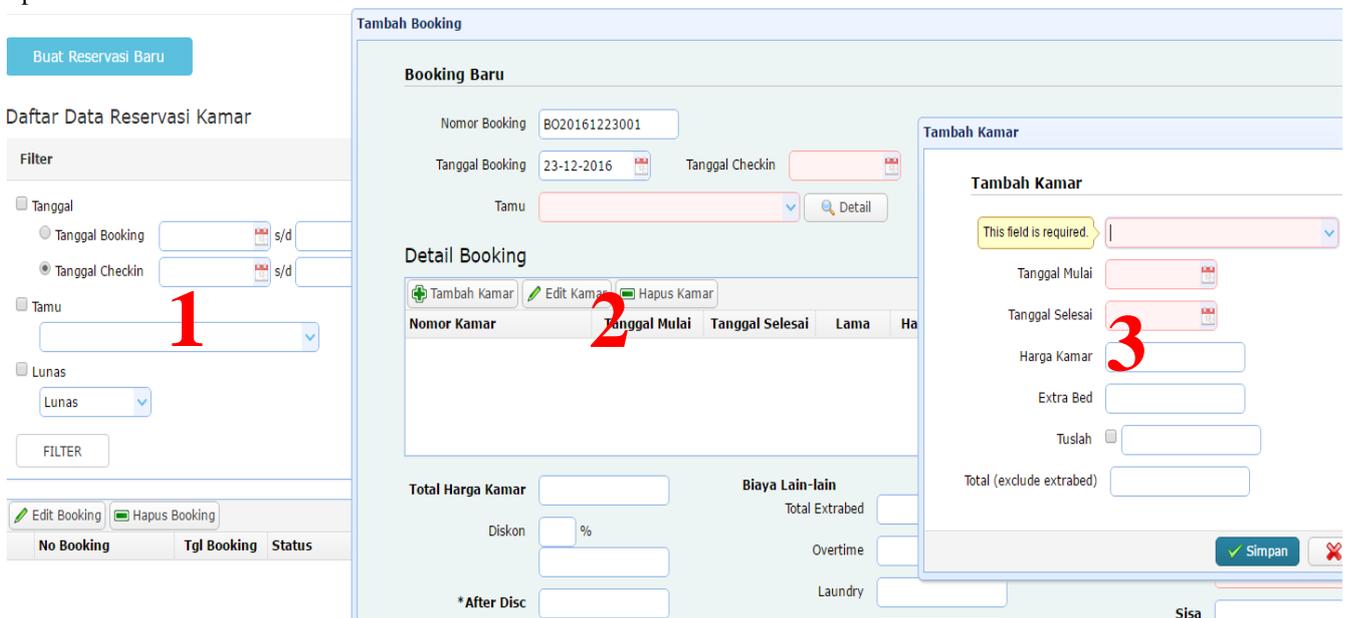
REST API menggunakan format JSON sebagai format pertukaran datanya. Format ini sangat cocok dikombinasikan dengan AJAX. Pada gambar 12 menunjukkan salah satu contoh response berformat JSON, fungsi ini merupakan fungsi untuk menampilkan data daftar tipe kamar di Wisma Tamu UKSW.

IV. HASIL DAN PEMBAHASAN

Sistem yang dirancang telah mengikuti arsitektur Single Page Application. Sebagian besar fungsi yang terdapat dalam sistem lama telah dikembangkan menjadi sistem yang menggunakan konsep SPA. Salah satu fungsi yang paling sering digunakan adalah reservasi kamar.

Gambar 13 menunjukkan proses reservasi kamar yang telah menggunakan konsep SPA. Terlihat beberapa dialog saling tumpang tindih dalam 1 halaman web dan dialog tersebut saling terkait. Prosesnya dimulai saat melakukan reservasi di dialog 1 maka akan memanggil dialog 2, kemudian saat menambah kamar di dialog 2 maka akan memanggil dialog 3. Selain fungsi ini, masih banyak fungsi lain yang juga telah dikembangkan menggunakan konsep SPA. Fungsi yang telah dikembangkan menggunakan konsep SPA kemudian dianalisa untuk mengukur tingkat efektifitas dan efisiensi penggunaan data (bandwith) dan lama waktu loading setiap prosesnya. Hasil analisa yang bisa dikatakan efektif dan efisien bila dapat mengurangi > 60% penggunaan bandwidth [12].

Tabel II merupakan daftar fungsi yang telah menggunakan konsep SPA. Pada tabel tersebut dipaparkan Size of page, yang menunjukkan berat setiap halaman saat dibuka pertama kali dalam satuan kilobytes. Juga dipaparkan Loading time dalam satuan milisecond, yang menunjukkan lama waktu setiap halaman saat dibuka pertama kali. Jika dianalisa nilai-nilai tersebut masih dikategorikan ringan, ini terlihat dari ukuran beratnya yang tidak sampai 1024 kilobytes (1 megabytes) dan lama waktu loading rata-rata masih dibawah 1000 milisecond (1 detik).



Gambar 13 Desain sistem menggunakan SPA

TABEL II
DAFTAR FUNGSI YANG TELAH MENGGUNAKAN KONSEP SPA

Fungsi	Size of page (kilobytes)	Loading time (millisecond)
Reservasi kamar		
Data tipe kamar	768 kb	535 ms
Data kamar	773 kb	704 ms
Data tamu	756 kb	632 ms
Data reservasi	805 kb	1016 ms
Cek status kamar	746 kb	522 ms
Reservasi meeting room		
Data reservasi	801 kb	919 ms
Mini shop		
Data barang	768 kb	809 ms
Data pembelian	789 kb	975 ms
Data penjualan	795 kb	902 ms
User		
Data user	758 kb	670 ms
Data user log	751 kb	613 ms

Tabel III menunjukkan daftar fungsi beserta jumlah penggunaan data dalam satuan *bytes*, jumlah yang dimaksud disini adalah jumlah data yang diterima oleh *AJAX* dari *REST API*. Rata-rata ukuran data untuk proses *create* (tambah) hampir sama untuk setiap fungsi yaitu 449 *bytes* karena bentuk *response* untuk proses ini hanya berupa teks *success*, kecuali untuk proses reservasi, cek kamar dan penjualan bentuk *responsenya* berbeda. Untuk proses *update* dan *delete* kasusnya hampir sama dengan proses *create*. Untuk proses *read*, setiap fungsi pasti akan berbeda ukurannya karena nilainya tergantung pada jumlah data yang ada di dalam tabel.

TABEL III
PENGUNAAN DATA PROSES REQUEST DAN RESPONSE SPA

Fungsi	Penggunaan Data (bytes)			
	Create	Read	Update	Delete
Reservasi kamar				
Data tipe kamar	449 bytes	836 bytes	449 bytes	571 bytes
Data kamar	449 bytes	3900 bytes	449 bytes	571 bytes
Data tamu	449 bytes	940 bytes	449 bytes	571 bytes
Data reservasi	569 bytes	2100 bytes	571 bytes	571 bytes
Cek status kamar	817 bytes	-	-	-
Reservasi meeting room				
Data reservasi	801 bytes	891 bytes	573 bytes	570 bytes
Mini shop				
Data barang	449 bytes	1123 bytes	449 bytes	571 bytes
Data pembelian	449 bytes	1700 bytes	449 bytes	571 bytes
Data penjualan	575 bytes	1800 bytes	577 bytes	571 bytes
User				

Data user	449 bytes	1200 bytes	571 bytes	571 bytes
Data user log	-	1230 bytes	-	-
Rata-rata	546 ms	1572 ms	504 ms	571 ms

Tabel IV menunjukkan daftar fungsi beserta lama waktu proses *request* dan *response* dari *AJAX* ke *REST API*. Setiap fungsi memiliki lama waktu yang bervariasi karena masing-masing memiliki alur program yang berbeda-beda. Lama waktu juga bisa dipengaruhi oleh ukuran data yang di kirim (tabel III). Lama waktu yang dipaparkan dalam daftar tersebut masih dikategorikan sangat cepat, rata-rata masih dibawah 200 *milisecond* (0,2 detik).

TABEL IV
LAMA WAKTU LOADING PROSES REQUEST DAN RESPONSE SPA

Fungsi	Penggunaan Data (bytes)			
	Create	Read	Update	Delete
Reservasi kamar				
Data tipe kamar	210 ms	71 ms	79 ms	206 ms
Data kamar	88 ms	67 ms	94 ms	212 ms
Data tamu	200 ms	73 ms	97 ms	229 ms
Data reservasi	302 ms	190 ms	212 ms	210 ms
Cek status kamar	187 ms	-	-	-
Reservasi meeting room				
Data reservasi	229 ms	72 ms	85 ms	216 ms
Mini shop				
Data barang	95 ms	73 ms	97 ms	210 ms
Data pembelian	210 ms	102 ms	201 ms	256 ms
Data penjualan	243 ms	109 ms	250 ms	270 ms
User				
Data user	218 ms	158 ms	202 ms	221 ms
Data user log	-	168 ms	-	-
Rata-Rata	198 ms	108 ms	146 ms	225 ms

Berdasarkan hasil analisa diatas, saat pertama kali membuka suatu halaman (fungsi), proses *loading* komponen dan *library* pendukung masih relatif ringan dan tidak memakan waktu yang lama, proses ini juga hanya dilakukan sekali per halaman (tabel II). Setelah itu, untuk proses *CRUD* data akan memanfaatkan *AJAX* dan dapat dilihat bahwa penggunaan data relatif sangat ringan (tabel III). Sedangkan untuk lama waktu *loading* juga relatif sangat cepat (tabel IV). Hal ini membuat aplikasi berjalan tidak terlalu berat saat melakukan proses *CRUD* data. Jika dibandingkan dengan sistem sebelumnya, sistem baru ini menggunakan *bandwidth* 70% lebih efektif.

Evaluasi sistem dilakukan dengan cara melakukan *software testing*, terdapat 2 jenis pengujiannya yaitu *alpha testing* dan *beta testing*. *Alpha testing* adalah pengujian yang dilakukan oleh *tester* dari pihak *developer*. *Beta testing* adalah pengujian yang dilakukan oleh *end user* tanpa didampingi *developer*. Pengujian dilakukan dengan cara mengecek semua fungsionalitas sistem apakah sudah berjalan sesuai yang diharapkan atau tidak. Untuk pengujian *beta testing*, *user* yang dilibatkan sebanyak 3 orang dari pihak Wisma Tamu UKSW (2 *front office* dan 1 *supervisor*).

TABEL V
HASIL PENGUJIAN ALPHA TESTING

Fungsi	Hasil	Status Uji
Reservasi kamar		
Data tipe kamar	Data tipe kamar terinput	Valid
Data kamar	Data kamar terinput	Valid
Data tamu	Data tamu terinput	Valid
Data reservasi	Data reservasi terinput	Valid
Cetak nota	Nota tercetak	Valid
Cek status kamar	Status kamar ditampilkan	Valid
Reservasi meeting room		
Data reservasi	Data reservasi terinput	Valid
Cetak nota	Nota tercetak	Valid
Mini shop		
Data barang	Data barang terinput	Valid
Data pembelian	Data pembelian terinput	Valid
Data penjualan	Data penjualan terinput	Valid
Cetak nota	Nota tercetak	
User		
Data user	Data user terinput	Valid
Data user log	Log user terekam	Valid

TABEL VI
HASIL PENGUJIAN BETA TESTING

Fungsi	Hasil	Status Uji
Reservasi kamar		
Data tipe kamar	Data tipe kamar terinput	Valid
Data kamar	Data kamar terinput	Valid
Data tamu	Data tamu terinput	Valid
Data reservasi	Data reservasi terinput	Valid
Cetak nota	Nota kadang tidak bisa tercetak (permasalahan terletak pada versi browser yang digunakan).	Belum Valid
Cek status kamar	Status kamar ditampilkan	Valid
Reservasi meeting room		
Data reservasi	Data reservasi terinput	Valid
Cetak nota	Nota kadang tidak bisa tercetak (permasalahan terletak pada versi browser yang digunakan).	Belum Valid
Mini shop		
Data barang	Data barang terinput	Valid
Data pembelian	Data pembelian terinput	Valid
Data penjualan	Data penjualan terinput	Valid
Cetak nota	Nota kadang tidak bisa tercetak (permasalahan terletak pada versi browser yang digunakan).	Belum Valid
User		
Data user	Data user terinput	Valid
Data user log	Log user terekam	Valid

Hasil pengujian *alpha* (Tabel V) dan pengujian *beta* (Tabel VI) menunjukkan keseluruhan sistem sudah berjalan dengan baik, walaupun pada pengujian *beta* yang dilakukan oleh pihak *end user* terdapat fitur yang tidak dapat berjalan, seperti fungsi cetak nota. Kesalahan tersebut dikarenakan pengguna masih menggunakan versi *browser* yang lama.

V. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan dapat disimpulkan bahwa sistem Wisma Tamu UKSW dapat mengimplementasikan konsep *SPA* untuk mengatasi permasalahan yang sebelumnya telah diidentifikasi. Tingkat efektifitas dan efisiensi penggunaan data dan lama waktu *loading* setiap proses telah dihitung dan dianalisa. Hasilnya menunjukkan setiap proses *SPA* masih relatif ringan dan cepat diakses, bisa dilihat pada tabel di pembahasan sebelumnya. Teknologi dari *SPA* juga membuat aplikasi lebih interaktif digunakan oleh pengguna terutama saat proses input dan edit. Saran dari hasil penelitian ini adalah meningkatkan keamanan dari sisi *server* terutama pada *REST API*, agar sistem lebih aman dari pihak yang tidak bertanggung jawab.

DAFTAR PUSTAKA

- [1] R. Saxena, "Single Page Application using AngularJS, Web API and MVC 5." [Online]. Available: http://www.c-sharpcorner.com/uploadfile/rahul4_saxena/single-page-application-spa-using-angularjs-web-api-and-m/. [Accessed: 20-Jan-2017].
- [2] M. Jadhav, S. Balkrishna, and A. Deshmukh, "Single Page Application using AngularJS," *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, no. 3, pp. 2876–2879, 2015.
- [3] S. Khanna and M. Mistry, "Impact of AJAX in Web Applications," *Int. J. Adv. Eng. Technol.*, vol. 3, no. 1, pp. 144–145, 2012.
- [4] Sachdeva, Ravi Kumar, and S. Singh, "Use of AJAX to Improve Usability of Online Information Systems," *Glob. J. Comput. Sci. Technol. Softw. Data Eng.*, vol. 13, no. 8, 2013.
- [5] F. Monteiro, *Learning Single-page Web Application Development*. Brimingham: Packt Publishing, 2014.
- [6] L. Richardson and S. Ruby, *RESTful Web Services*. Sebastopol: O'Reilly Media, 2007.
- [7] C. Easttom, *Advanced JavaScript*, 3rd ed. Texas: Wordware Publishing, Inc., 2008.
- [8] K. Yank and C. Adams, *Simply JavaScript*, 1st ed. Sitepoint, 2007.
- [9] C. Schmitt and K. Lawver, *Adapting to Web Standards : CSS and Ajax for Big Sites*. Berkeley: New Riders, 2008.
- [10] T. A. Powerll, *The Complete Reference AJAX*. Mc Graw Hill Osborne, 2008.
- [11] B. Hoffman and B. Sullivan, *Ajax Security*. Boston: Addison-Wesley, 2007.
- [12] C. L. Merrill, "Performance Impacts of AJAX Development," 2006. [Online]. Available: <http://www.webperformance.com/library/reports/ajax-bandwidth-testing/>. [Accessed: 04-Mar-2017].