BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perusahaan otobus (PO) merupakan salah satu jasa akomodasi angkutan darat yang melayani perjalanan dari satu kota menuju kota lainnya. Saat ini informasi mengenai jadwal keberangkatan bus dari perusahaan otobus bisa didapatkan dengan mengunjungi situs *online (website)*, baik itu melalui *website* resmi dari perusahaan otobus tersebut atau mencari melalui *blog* yang menyediakan. Akan tetapi, untuk dapat melakukan pemesanan tiket belum bisa dilakukan pada *website* tersebut. Adapun contoh perusahaan otobus yang sudah menyediakan layanan pemesanan tiket *online yaitu* PO kramatdjati, sedangkan untuk perusahaan otobus lainnya seperti PO Nusantara dan PO Rosalia hanya memberikan informasi mengenai jadwal keberangkatan bus saja.

Secara garis besar, *website* maupun *blog* yang tersedia saat ini mampu memberikan informasi mengenai jadwal keberangkatan bus secara *online* kepada para calon penumpang.Akan tetapi, *website* yang tersedia hanya memberikan informasi jadwal keberangkatan bus untuk satu perusahaan otobus saja. Adapun untuk pemesanan tiket masih dilakukan secara manual yaitu melalui agen yang disediakan pada kota tersebut.

Tampilan *website* yang berbeda-beda ketika proses mencari jadwal keberangkatan bus merupakan alasan munculnya gagasan untuk membuat sebuah *website* yang menyediakan informasi mengenai jadwal keberangkatan bus dari berbagai perusahaan otobus yang berbeda dalam satu tampilan *website* serta dapat menangani pengelolaan pemesanan tiket bus. Diharapkan *website* ini dapat membantu bagi calon penumpang untuk mendapatkan informasi jadwal keberangkatan bus dan pemesanan tiket serta membantu bagi perusahaann otobus untuk dapat menyediakan layanan pengelolaan pemesanan tiket secara *online* tanpa harus membuat *website* sendiri.

1.2 Rumusan Masalah

Adapun rumusan masalah yang disimpulkan dari permasalahan di atas adalah :

1. Bagaimana menyediakan data-data mengenai informasi jadwal keberangkatan bus yang dapat dimanfaatkan kembali ?
2. Bagaimana memberikan suatu solusi bagi perusahaan otobus untuk dapat menyediakan layanan pemesanan tiket ?
3. Bagaimana memberikan suatu solusi yang dapat membantu calon penumpang untuk dapat melakukan pencarian mengenai jadwal keberangkatan bus tanpa harus mengunjungi *website* dari masing-masing perusahaan otobus serta melakukan pemesanan tiket secara *online.*

1.3 Tujuan

Adapun tujuan yang ingin dicapai pada proyek akhir ini adalah :

1. Menyediakan API mengenai data-data jadwal keberangkatan bus yang dapat dimanfaatkan kembali oleh pihak ketiga.
2. Menyediakan suatu layanan *website* yang dapat digunakan oleh perusahaan otobus untuk memberikan jasa pemesanan tiket *online* tanpa harus membangun sendiri.
3. Membangun sebuah *website* yang menyediakan informasi jadwal keberangkatan bus dan pemesanan tiket yang terdiri dari berbagai perusahaan otobus dalam satu tampilan *website*.

1.4 Batasan Masalah

Adapun yang menjadi batasan masalah dalam pembuatan proyek akhir ini adalah sebagai berikut :

1. Bagi perusahaan otobus yang ingin memanfaatkan aplikasi ini harus melakukan proses pendaftaran, untuk mendapatkan layanan pemesanan tiket.

2. Sebagai batasan pada proyek akhir ini hanya menggunakan tiga perusahaan otobus saja yaitu perusahaan otobus kramatdjati, nusantara dan rosalia.
3. Data-data mengenai Jadwal keberangkatan bus yang ada di *website* ini merupakan data *dummie.*
4. Arsitektur *web service* yang digunakan adalah REST.
5. Belum menyediakan daftar pemesanan secara keseluruhan.
6. Metode pembayaran yang digunakan hanya melalui transfer antar bank.
7. Belum menanggani masalah *discount*.

1.5 Definisi Operasional

Aplikasi pengelolaan pemesanan tiket bus berbasis *web service* ini merupakan suatu *website* yang menyediakan informasi mengenai jadwal keberangkatan bus dan pemesanan tiket dari beberapa perusahaan otobus yang berbeda dengan memanfaatkan *web service*. Adapun penggunaan *web service* yang dibangun dengan menggunakan arsitektur REST. Pada arsitektur REST terdapat *resource - resource* yang dapat dikembangkan oleh pihak ketiga untuk membangun aplikasi lagi atau disebut dengan aplikasi *client*. sistem yang dibangun ini merupakan salah aplikasi *client* dalam bentuk *website.*

1.6 Metode Pengerjaan

Metode pengerjaan yang digunakan pada untuk membangun aplikasi ini adalah dengan waterfall [4], dengan tahap sebaagai berikut :

a. Analisi

   Pada tahapan ini, dilakukan analisa kebutuhan-kebutuhan apa aja yang dibutuhkan untuk membangun aplikasi ini. Adapun bentuk analisa yang dilakukan dengan mengumpulkan data-data mengenai jadwal keberangkatan bus dari *website-website* resmi perusahaan otobus maupun *blog-blog* yang menyediakan informasi mengenai jadwal tersebut.

b. Desain

Pada tahan ini, membuat rancangan desain sistem berupa desain alur pemrosesan data dari aplikasi yang dibuat, kemudian menuangkannya dalam bentuk desain rancangan *use case diagram, class diagram* dan *sequences.*

c. Coding

Tahap pembuatan kode atau coding ini adalah tahap pembuatan aplikasi berdasarkan hasil analisis dan alur pemrosesan data yang telah dilakukan di tahap-tahap sebelumnya.

d. Pengujian

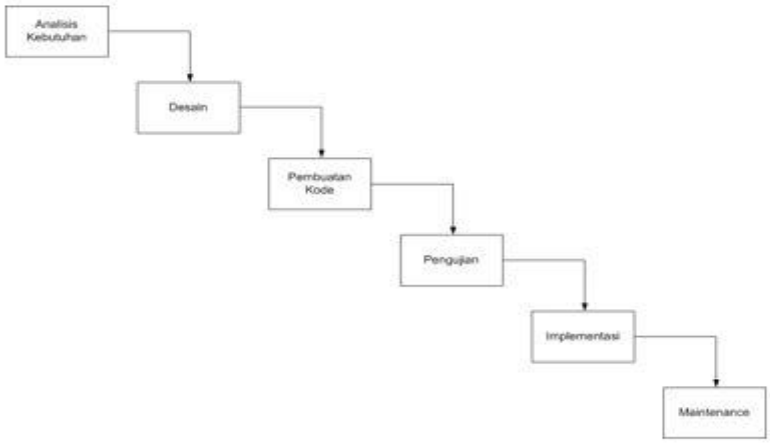
Melakukan pengujian yang bertujuan untuk mengidentifikasi kesalahan-kesalahan yang mungkin terjadi pada proses pengkodean kemudian dilakukan perbaikan serta memastikan bahwa aplikasi yang dibuat telah sesuai yang direncanakan.

e. Implementasi

Pada tahap ini, implementasi aplikasi hanya membahas pemodelan awal saja dan belum sampai pada tahap implementasi yang sebenarnya.

f. Maintenance

Aplikasi ini tidak akan membahas tahap *maintenance* dikarenakan pembangunan aplikasi ini belum diimplementasikan.



Gambar 1-1 Urutan Metode Pengerjaan dengan Waterfall [4]

BAB 2
TINJAUAN PUSTAKA

2.1 Service Oriented Architecture (SOA)

Dalam rekayasa perangkat lunak, sebuah *Service Oriented Architecture* (*SOA*) adalah metode untuk merancang dan mengembangkan perangkat lunak dalam bentuk layanan. Layanan didefinisikan dengan baik dan dibangun sebagai komponen perangkat lunak yang dapat digunakan kembali untuk berbagai tujuan.

SOA umumnya menyediakan layanan bagi konsumen seperti aplikasi berbasis *web*. Sebagai contoh, departemen yang berbeda dalam perusahaan dapat mengembangkan dan menyebarkan layanan *SOA* dalam implementasi yang berbeda, masing-masing konsumen akan mendapatkan keuntungan dari layanan yang terdefinisi dengan baik. XML sering digunakan untuk berinteraksi dengan layanan SOA begitu juga dengan JSON.

Sistem yang dibangun merupakan salah satu bentuk penggunaan SOA, yaitu menyediakan layanan *client service* dalam bentuk tampilan *website* yang dapat di implementasikan pada perusahaan yang berbeda. SOA juga mendefinisikan bagaimana cara untuk mengintegrasikan aplikasi yang berbeda secara luas [1].

2.2 Web service

Web service menyediakan komunikasi standar antar aplikasi yang berbeda, yang berjalan di platform atau framework yang berbeda pula. *Web service* adalah sebuah system software yang di desain agar mendukung interaksi antar mesin melalui jaringan. Memiliki antar muka dengan format yang dapat diproses kedua belah pihak (umumnya WSDL). Sistem lain berinteraksi dengan *Web service* biasanya disampaikan menggunakan protokol HTTP dengan serialisasi XML. Tujuan dari *Web service* adalah untuk menyediakan akses untuk menggunakan fungsi atau resource aplikasi penyedia *Web service* untuk digunakan oleh aplikasi lain di platform atau framework yang berbeda [8].

Web service pada awalnya menggunakan menggunakan SOAP (Simple Object Access Portocol), namun SOAP masih terlalu kompleks dan membutuhkan banyak

kode program sehingga tidak mudah untuk diimplementasikan. Perkembangan selanjutnya Web service adalah menggunakan arsitektur REST, REST dinilai lebih mudah diimplementasikan dibanding SOAP, sehingga banyak *Web service* mulai beralih menggunakan arsitektur REST. Pada masa kini, perkembangan *Web service* memicu lahirnya cloud computing.

Aplikasi yang dibangun ini merupakan salah satu contoh yang memanfaatkan *Web service.* Tujuan penggunaan *web service* pada aplikasi ini yaitu untuk menyediakan suatu fungsi atau *resource* yang dapat dikembangkan oleh aplikasi lain. Adapun pengembangan aplikasi lain atau biasa disebut aplikasi *client* dalam sistem yang dibangun ini merupakan aplikasi yang data-data nya memanfaatkan API yang dibangun dalam bentuk tampilan *website* dengan menggunakan *framework codeneigter*.

2.3 REST

Representational State Transfer atau disingkat REST adalah suatu arsitektur dalam *web service* untuk melakukan pertukaran data yang berorientasi pada sumber daya Informasi atau resource melalui protokol HTTP. Pemanfaatan Rest pada aplikasi ini digunakan untuk mengambil data-data atau sumber daya yang disediakan oleh *service*. Berikut penjelasan Roy Fielding mengenai Representational State Transfer: “Representational State Transfer is intended to evoke an image of how a well-designed Web Application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use” [3]. Roy Fielding mengidentifikasi empat prinsip REST yaitu :

1. Identification of resources.
2. Manipulation of resources through representations.
3. Self-descriptive messages.
4. Hypermedia as the engine of application state (abbreviated *HATEOAS)*.

Tabel 2-1
Perbandingan RESTless dengan RESTful

RESTless		Restfull	
VERB	HREF		
POST	/users/create	POST	/users
GET	/users/1	GET	/users/1
PUT	/users/1/update	PUT	/users/1
????	/users/1/delete	????	/users/1

2.4 Uniform Resources Identifiers

Uniform Resource Identifier (URI) adalah sebuah identifikasi dari *resource*. Identifikasi tersebut memungkinkan representasi *resource* melalui jaringan. Jika struktur penyimpanan *resource* di *server* berubah URI akan tetap sama selama konteks sumber daya tidak berubah [7].

2.5 API

Sebuah antarmuka pemrograman aplikasi (*API*) adalah kode sumber berbasis spesifikasi dimaksudkan untuk digunakan sebagai antarmuka dengan komponen software untuk berkomunikasi satu sama lain [6].

Contoh sebuah *website* yang memberikan ribuan informasi mengenai berbagai *Web API* yang ada, baik yang terbuka maupun yang dibatasi, baik yang gratis maupun yang berbayar. Yaitu *ProgrammableWeb.com*.

Adapun keunggulan yang dimiliki Ketika sebuah *API* digunakan dalam proyek, yaitu :

1. Memungkinkan untuk fokus pada proyek.
2. Menghemat waktu pengembangan.

2.6 Aplikasi

Menurut Jogiyanto aplikasi merupakan penerapan, menyimpan sesuatu hal, data, permasalahan, pekerjaan kedalam suatu sarana atau media yang dapat digunakan untuk menginplementasikan hal atau permasalahan yang ada sehingga berubah menjadi suatu bentuk yang baru tanpa menghilangkan niali-nilaidasar dari hal data, permasalahan, pekerjaan itu sendiri [5].

2.7 Codeigniter

CodeIgniter adalah Application Development Framework - toolkit - bagi orang-orang yang membangun aplikasi menggunakan PHP. Tujuannya adalah memungkinkan untuk mengembangkan proyek-proyek jauh lebih cepat daripada menulis kode dari awal, dengan menyediakan banyak *library* untuk tugas-tugas yang biasa diperlukan, serta antarmuka yang sederhana dan struktur logis untuk mengakses library [2].

CodeIgniter memungkinkan developer fokus pada proyek dengan meminimalkan jumlah kode yang dibutuhkan untuk tugas yang diberikan. Codeigniter menggunakan Uniform Resource Identifier (URI) untuk mengidentifikasi *resource* yang di akses. Sesuai dengan spesifikasi RESTful yang menggunakan URI daripada URL. Pada sistem ini codeigniter dipilih sebagai frmaework untuk membangun aplikasi ini.