BAB II
LANDASAN TEORI

2.1 Posisi Penelitian

Awal mula pengembangan sistem *web service* REST API UTC dilaksanakan dengan melakukan studi literatur terhadap penelitian terkait. Terdapat berbagai macam metode yang digunakan pada penelitian terdahulu dalam pengembangan *web service*.

Terdapat penelitian terdahulu tentang sistem administrasi yang hanya dibuat berdasarkan kebutuhan administrasi pada waktu dan tempat tertentu. Oleh karena itu, dibuat sebuah sistem administrasi dengan *back-end* yang baik. *Back-end* yang baik dapat dicapai dengan menerapkan konsep API dengan gaya arsitektur REST yang dikembangkan menggunakan metode *waterfall* (Komputasi, 2017).

Selanjutnya terdapat pengembangan *web service* yang didasari oleh masalah kurangnya pengembangan sistem informasi pariwisata. Terdapat beberapa sistem informasi pariwisata yang telah ada, namun belum dapat melakukan fungsi dengan lengkap. Dengan ini dibuatlah sistem informasi yang berfokus pada pengembangan REST API untuk mempermudah *web* dan *android* sistem informasi IndoExplore.id untuk menjalankan fungsi yang tersedia, mempermudah penyimpanan dan pengambilan data (Ardiansyah, 2017).

Setelah itu terdapat penelitian yang berbicara tentang *website* bank sampah yang belum cukup untuk menyelesaikan masalah di sisi masyarakat. Dengan ini dibutuhkan aplikasi platform *mobile* dan sebuah *web service* untuk integrasi data *web* dan *mobile*. Pengembangan *web service* pada penelitian ini menerapkan konsep REST API karena dirasa lebih efektif dalam pengembangan dan penggunaan. REST API dikembangkan dengan metode *iterative incremental* (Fakhrun & Gumilang, 2018).

Ketiga penelitian terdahulu yang telah dirangkum membantu pengembangan REST API UTC dalam melihat posisi penelitian. Tahap yang terjadi dapat berbeda-beda pada tiap pengembangan *web service*. Menurut Heather Kreger (2001) terdapat empat tahap siklus pengembangan, yaitu *build*, *deploy*, *run*, dan *manage* (Services & Architecture, 2001). Pengembangan *web service* REST API UTC dilakukan dengan metode *iterative incremental* dengan melalui enam tahap dan pengujian menggunakan aplikasi Postman. .

Tabel 2.1 menunjukkan posisi penelitian *web service* UTC.

Tabel 2.1 Tabel Posisi Penelitian

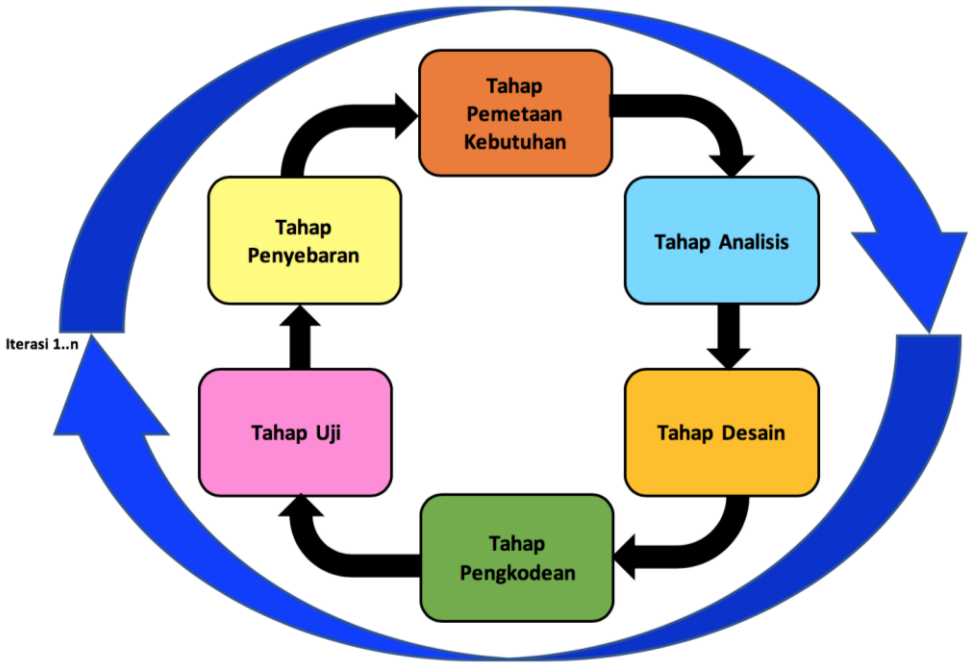
No	Parameter	Posisi Penelitian			
		REST API Sistem Informasi Administrasi (Komputasi, 2017)	REST API Sistem Informasi Pariwisata (Ardiansyah, 2017)	*Web Service* pada Bank Sampah (Fakhrun & Gumilang, 2018)	REST API UTC
1	Metode yang digunakan	*Waterfall*	*Scrum*	*Iterative Incremental*	*Iterative Incremental*
2	Jenis *web service*	API	API	API	API
3	Jumlah tahap pengembangan	5	4	4	6
4	Kakas pengujian fungsi *web service*	Postman	Postman	Postman dan SoapUi	Postman

2.2 Siklus Implementasi *Web Service*

Proses pengembangan *web service* cenderung dilakukan secara iteratif dan inkremental. Menurut Cockburn (2008) iteratif merupakan strategi revisi yang dilakukan secara berulang guna memperbaiki sistem. Sedangkan inkremental merupakan proses pengerjaan bagian sistem dari satu tahap ke tahap selanjutnya yang akhirnya akan diintegrasikan ketika semua bagian sistem telah selesai (Cockburn, 2008).

Iterasi terjadi pada semua tahap (Draft, 2005). Ketika satu fungsi telah berhasil dikembangkan hingga dapat diimplementasikan kepada pengembang aplikasi *mobile* (tahap penyebaran) maka pengembangan fungsi selanjutnya akan dimulai dari tahap awal yaitu tahap pemetaan. Apabila pada pengembangan fungsi pertama tahap pemetaan telah dilakukan untuk semua fungsi, maka dilakukan pengecekan ulang apabila terdapat perubahan pada basis data aplikasi. Ketika terdapat perubahan pada basis data maka akan berpengaruh pada semua tahap setelahnya. Hal ini yang menyebabkan iterasi harus terjadi pada semua tahap.

Pengembangan *web service* REST API UTC mengadopsi enam tahap siklus pengembangan seperti pada Gambar 2.1 (Draft, 2005). Penelitian menggunakan metode serta tahap ini karena dianggap paling sesuai dengan kebutuhan pengembang.



Gambar 2.1 Siklus Implementasi *Web Service*

Enam tahap pada siklus pengembangan *web service* UTC dengan penjelasan singkat sebagai berikut :

a. Tahap Pemetaan Kebutuhan

Tujuan dari tahap pemetaan adalah untuk memahami kebutuhan bisnis yang selanjutnya akan diterjemahkan ke dalam *web service* (Draft, 2005). Pemetaan kebutuhan atau seleksi persyaratan pada tahap ini dilakukan dengan metode pemetaan terhadap basis data *web* UTC.

b. Tahap Analisis

Analisis bertujuan memahami lebih dalam mengenai fungsi yang telah diidentifikasi pada proses sebelumnya dan membantu dalam menentukan prioritas pengembangan fungsi. Pada tahap ini fungsi yang telah diidentifikasi selanjutnya akan disempurnakan dan diterjemahkan ke model konseptual. Juga dalam tahap ini dilakukan analisis arsitektur dan mengidentifikasi antarmuka *web service* (Draft, 2005).

c. Tahap Desain

Sebelum membangun *web service*, pengembang perlu menentukan spesifikasi pengembangan. Desain memuat rancangan kebutuhan, analisis dari rancangan, dan yang

terpenting adalah desain antarmuka dan desain arsitektur sistem *web service* (Draft, 2005).

d. Tahap Pengkodean

Tahap pengkodean dikerjakan sesuai dengan yang telah didesain pada tahap sebelumnya. Pengkodean dapat dilakukan dengan bahasa yang beragam namun tetap harus mempertimbangkan tipe data pada bahasa tersebut (Draft, 2005).

e. Tahap Uji

Pengujian melibatkan pelaksanaan rangkaian tugas dan membandingkan hasil dengan *output* yang diharapkan. Menerapkan tes fungsional di tahap awal pengembangan berguna untuk mempercepat pengembangan, meningkatkan kualitas, dan mengurangi risiko pada tahap akhir (AppPerfectCorporation, n.d.).

f. Tahap Penyebaran

Tujuan dari penyebaran adalah untuk memastikan bahwa *web service* yang telah diuji dapat digunakan dengan baik. Penyebaran akan dilakukan setelah *web service* melewati tahap uji (Draft, 2005). Tahap penyebaran dilakukan dengan pengembangan sebuah katalog.

2.3 API

Application Programming Interface (API) dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya. Secara umum API merupakan dokumentasi dari fungsi, struktur, kelas, dan lain sebagainya yang dapat diakses oleh yang membutuhkan dengan cara tertentu sesuai yang ditentukan layanan. API memudahkan pengembang lain mengambil data untuk kemudian diintegrasikan dengan perangkat lunak lain (Santoso & Rais, 2015).

2.4 REST

Representational State Transfer atau yang lebih dikenal dengan sebutan REST merupakan arsitektur API yang menggunakan protokol HTTP dalam komunikasi data. Dalam arsitektur REST, *server* menyediakan akses ke sumber data dan klien yang mengambil data. Setiap sumber data diidentifikasi dengan *link* URI. Dalam menyajikan data REST menggunakan berbagai format seperti teks, XML dan JSON (Fielding, 2000). *Javascript Object Notation* (JSON) adalah salah satu format pertukaran data yang ringan. JSON bersifat mudah untuk dibaca dan ditulis oleh manusia, diterjemahkan dan dibuat (*generate*) oleh

komputer. Format teks JSON tidak bergantung pada suatu bahasa pemrograman tertentu (Crockford, n.d.). Penerapan JSON banyak digunakan pada basis data dan *web service*.

REST menggunakan protokol HTTP yang bersifat *stateless*. Cara REST berkomunikasi antar platform adalah dengan mengirim perintah dan menerima *response* berupa JSON dan kode HTTP. Perintah yang dikirim akan dikerjakan oleh *server* (Perkasa & Setiawan, 2018). Terdapat empat metode perintah HTTP yang bisa digunakan pada REST yaitu :

a. GET untuk membaca sumber data.
b. POST untuk membuat data baru.
c. PUT untuk memperbarui data.
d. DELETE untuk menghapus data.

Namun, pada penelitian ini hanya menggunakan GET dan POST sesuai dengan kebutuhan pengembang aplikasi UTC basis *mobile*.

Ketika menjalankan perintah GET, apabila tidak terdapat kesalahan maka *response* kode HTTP yang direpresentasikan dalam JSON akan dikembalikan oleh *server* berupa 200 (OK), namun apabila terdapat *error* maka kode status yang paling sering muncul adalah 404 (*not found*) dan 400 (*bad request*). POST merupakan perintah yang sering digunakan. Apabila pembuatan *resource* berhasil maka perintah POST yang dikirim ke *server* akan dikembalikan dalam JSON dengan status 201 (*successful creation*) (Fredrich, n.d.).

2.5 Kakas Pengujian API

Fungsi REST API yang telah berhasil dikembangkan akan didokumentasikan dalam sebuah katalog REST API. Dokumentasi minimal disediakan dalam bentuk *link, method* dan deskripsi singkat mengenai fungsi dari API. Katalog REST API bertujuan agar para pengembang lain dapat dengan mudah mengakses fungsi REST API yang dibutuhkan. Banyak *tools generator* yang dapat digunakan untuk pengujian dan pendokumentasian fungsi REST API seperti Postman, SoapUI, Swagger, dan lain-lain (Kusuma, 2017).

2.5.1 Postman

Postman adalah platform kolaborasi untuk pengembangan API. Dibuat oleh Abhinav Asthana, seorang *programmer* dan desainer yang berbasis di Bangalore, India, Postman memudahkan dalam menguji, mengembangkan, dan mendokumentasikan API. Fitur Postman yang sederhana membuat pengujian API dapat dilakukan dengan baik dan cepat. Cara kerja Postman dengan mengklasifikasi *request* berdasarkan *request method*, URL dan parameter-parameter *request* (Wagner, 2014).

2.5.2 SoapUI

SoapUI adalah aplikasi pengujian *web service* yang bersifat *open source* untuk SOA dan REST. Fungsionalitas aplikasi mencakup pengujian *inspection, invoking, development, simulation and mocking, functional testing, load and compliance testing*. SoapUI mendukung semua jenis protokol dan teknologi standar untuk menguji semua jenis API. Dengan antarmuka yang sederhana, SoapUI memungkinkan pengguna dapat mengoperasikan dengan mudah (SoapUI, 2019).

2.5.3 Swagger

Swagger adalah perangkat lunak untuk mendokumentasikan REST API. Swagger dapat digunakan untuk berbagi dokumentasi di antara manajer proyek, penguji dan pengembang. Swagger juga dapat digunakan oleh berbagai alat untuk mengotomatisasi proses terkait API. Swagger bersifat *real time* sehingga klien dan sistem dokumentasi bergerak dengan kecepatan yang sama dengan *server*. Deskripsi metode, parameter, dan model terintegrasi erat ke dalam kode *server*, sehingga sinkronisasi API dan dokumentasinya dapat terjaga (Ratovsky, 2015).

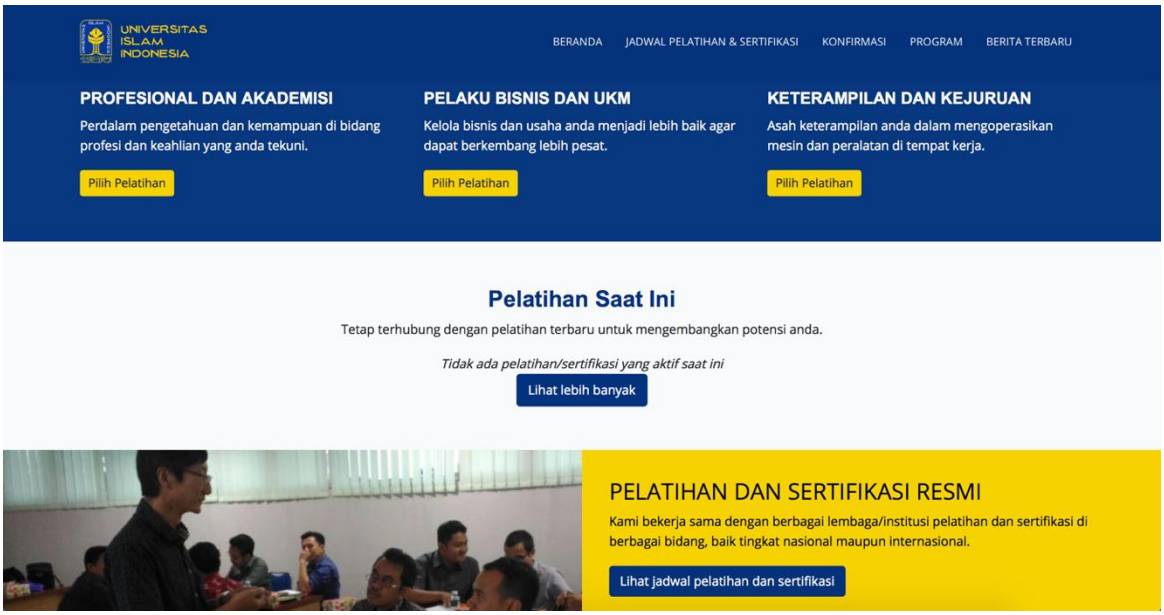
2.6 Platform *Web* UTC

Organisasi di UII, Simpul Tumbuh memiliki tujuan mewujudkan sinergi pengembangan dunia wirausaha. Cara Simpul Tumbuh mewujudkan hal tersebut salah satunya adalah dengan mensentralisasi berbagai macam *trainings*/pelatihan yang berada di seluruh Fakultas UII. Dalam hal ini Simpul Tumbuh tengah mengembangkan aplikasi *web* bernama UII Training Center (UTC) dengan pengguna *administrator*, pengelola lembaga, dan peserta. Dengan tujuan UTC yaitu memudahkan pengguna dalam mengadakan dan mendaftar pelatihan, yang saat ini masih dilakukan secara konvensional peserta harus datang ke salah satu jurusan yang mengadakan pelatihan.

Gambar 2.2 menunjukkan salah satu antarmuka dari *web* UTC dengan *login* yang dapat diakses menggunakan akun Google. Berbagai macam pelatihan akan ditampilkan pada tampilan awal *web*. UII dengan 8 fakultas dan 45 jurusan tentu memiliki banyak pelatihan. Segala jenis pelatihan akan ditampilkan secara umum pada antarmuka Gambar 2.3 dan secara lebih lengkap pada jadwal pelatihan & sertifikasi.

Gambar 2.2 Antarmuka *Web* UTC.

Dengan kemudahan yang berada pada aplikasi *web* UTC, tidak dapat dipungkiri bahwa aplikasi *mobile* telah menjadi tren masa kini. Oleh karena itu, pengembangan *web service* API dengan arsitektur REST ditujukan kepada pengembang platform lain untuk mempermudah dalam pengambilan data yang dibutuhkan.



Gambar 2.3 Antarmuka *Web* UTC.