PERANCANGAN BACK-END APLIKASI RUMANTARA DENGAN GAYA ARSITEKTUR *REST* MENGGUNAKAN METODE *ITERATIVE INCREMENTAL*

Ikhsan Ahmad Faruqi[1], Soni Fajar Surya Gumilang[2], Muhammad Azani Hasibuan[3]
[1,2,3]Program Studi S1 Sistem Informasi, Fakultas Rekayasa Industri, Universitas Telkom
[1]faruqisan@gmail.com,[2]mustonie@telkomuniversity.ac.id,[3]muhammad.azani@gmail.com

Abstrak

Tempat penginapan merupakan kebutuhan wisatawan dalam berwisata, sebuah trend baru untuk penginapan adalah menginap di rumah penduduk lokal. Wisatawan lebih memilih menginap di rumah penduduk lokal karena harga yang lebih murah, dan ingin mendapatkan pengalaman merasakan kebudayaan penduduk secara langsung melalui aplikasi online. Pada penelitian ini akan dibangun aplikasi yang membantu wisatawan dalam mencari kamar penduduk lokal yang disewakan secara online. Penelitian ini befokus kepada pengembangan *backend* API menggunakan gaya arsitektur REST yang akan digunakan oleh tim pengembang internal *frontend web* dan *mobile*. Gaya arsitektur REST yang diterapkan pada pengembangan mempermudah pendistribusian data dari satu *backend* server ke berbagai *client* di platform *web* dan *mobile*. Metode pengembangan perangkat lunak yang digunakan adalah *iterative incremental*. Metode ini sesuai dengan kondisi tim pengembang rumantara yang kecil dan membutuhkan hasil yang bagus dari kebutuhan pengguna. Aplikasi *backend* API dikembangkan menggunakan Laravel sebagai *framework* bahasa pemrograman PHP yang digunakan.

Kata kunci: ***restful api,* laravel*, backend, iterative incremental***

Abstract

A place to stay is a need for traveller when travelling, a new trend for accomodation is to stay in local resident houses. Traveller choose to stay there because a cheaper price, and want to expreince local heritages directly using online application. On this research will be built an application that help traveller on finding local resident houses that rented. This research focused on backend API that will be used by internal frontend and mobile development team. REST architecture style that implemented on development make data distribution from one backend server into any client on any platform more easily. Software development method that used is iterative and incremental. This method match with development team condition wich is small and need the best result from user requirement. Backend API application that developed using Laravel as a PHP programming language framework.

Keyword: restful api, laravel, backend, iterative incremental.

1. Pendahuluan

Indonesia adalah negara yang kaya akan tempat wisata. Indonesia memiliki lebih dari 962 tempat wisata yang tersebar di seluruh daerah. Menurut Kementrian Pariwisata di tahun 2016, jumlah pengunjung mancanegara meningkat hampir di setiap bulan nya dibandingkan dengan tahun sebelumnya. [1]. Hal menarik yang terjadi pada sektor penginapan bahwa lebih dari 500.000 wisatawan memilih untuk tidak menginap di tempat penginapan biasa seperi hotel, tetapi memilih untuk menginap di rumah penduduk lokal secara online melalui perusahaan penyedia *platform* penyewaan kamar *online* [2]. Alasan wisatawan dalam memilih menginap di rumah penduduk lokal melalui *platform online* dibanding hotel, atau penginapan tradisional lainnya adalah 55% karena harga yang lebih murah, 35% karena lokasi, 31% karena ingin merasakan kebudayaan lokal saat menginap, 25% karena memiliki dapur sendiri, 24% karena keunikan tempat penginapan, 23% karena kemudahan penggunaan aplikasi [3].

Rumantara sebagai sebuah perusahaan *startup* hadir untuk menghadirkan sebuah aplikasi yang memungkinkan wisatawan untuk mencari kamar yang disewakan oleh pemilik kamar dengan harga yang terjangkau dan merasakan pengalaman menginap disertai budaya lokal yang otekntik, disertai aksesibilitas aplikasi yang baik sehingga dapat di akses dimana saja kapan saja selama ada jaringan internet yang memadai.

Untuk mendukung aksesibilitas bagi pengguna aplikasi Rumantara, diusung konsep arsitektur sistem *multi platform* agar pengembangan aplikasi dari Rumantara dapat dibuat di berbagai platform dengan menggunakan satu back-end server. Oleh karena itu akan dirancang dan membangun back-end untuk digunakan oleh aplikasi rumantara yang berbasis *web* dan *mobile* dengan konsep arsitektur *RESTful*, sehingga cukup membangun satu back-end untuk aplikasi aplikasi yang berjalan di berbagai *Platform* dengan sistem autentikasi API menggunakan Laravel Passport.

2. Tinjauan Pustaka

2.1. *Representational State Transfer* (REST)

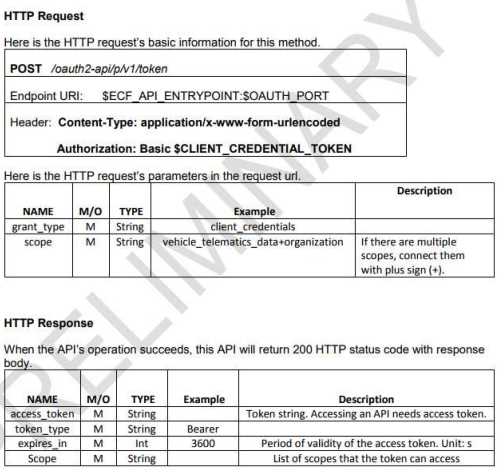
Representational State Transfer (REST) adalah sebuah gaya arsitektur perangkat lunak untuk sebuah sistem yang terdistribusi. REST secara spesifik menyediakan satu set prinsip arsitektur, yang berfokus kepada skalabilitas dari interaksi antar komponen dalam sistem dan generalitas dari sebuah *interface*. Penerapan REST yang bersifat independen dari setiap komponen bertujuan untuk mengurangi *latency*. Sistem yang mengikuti prinsip REST disebut sebagai RESTful [4].

2.2. Dokumentasi API

Pengembang Perangkat Lunak menggunakan Dokumentasi API untuk mengetahui bagaimana menggunakan fitur – fitur dari sebuah perangkat lunak. Seberapa cepat seseorang mempelajari bagaimana menggunakan fitur – fitur dari sebuah perangkat lunak menentukan seberapa cepat fitur – fitur tersebut teraplikasikan kepada sebuah aplikasi [5].

Dokumentasi API adalah referensi pertama yang digunakan oleh siapa saja yang akan mengimplementasikan API yang dibuat. Karena Dokumentasi API mendeskripsikan layanan – layanan yang disediakan oleh sebuah aplikasi dan bagaimana cara mengggunakan layanan – layanan tersebut [6].

Pada penelitian, format dokumentasi API mengikuti format *Interface Description for Partners Ericsson Connected Fleet 1.0* yang di rilis oleh Ericsson. Secara keseluruhan, bagian – bagian yang terdapat pada dokumen tidak jauh berbeda dengan penjelasan *Best Practices API Documentation*. Contoh dari format tersebut adalah mendeskripsikan *Request*, *Response*, dan contoh *Response* seperti Gambar 1:



HTTP Request

Here is the HTTP request's basic information for this method.

POST */oauth2-api/p/v1/token*
Endpoint URI: $ECF_API_ENTRYPOINT:$OAUTH_PORT
Header: **Content-Type: application/x-www-form-urlencoded** **Authorization: Basic $CLIENT_CREDENTIAL_TOKEN**

Here is the HTTP request's parameters in the request url.

NAME	M/O	TYPE	Example	Description
grant_type	M	String	client_credentials	
scope	M	String	vehicle_telematics_data+organization	If there are multiple scopes, connect them with plus sign (+).

HTTP Response

When the API's operation succeeds, this API will return 200 HTTP status code with response body.

NAME	M/O	TYPE	Example	Description
access_token	M	String		Token string. Accessing an API needs access token.
token_type	M	String	Bearer	
expires_in	M	Int	3600	Period of validity of the access token. Unit: s
Scope	M	String		List of scopes that the token can access

Gambar 1. Contoh Dokumentasi API

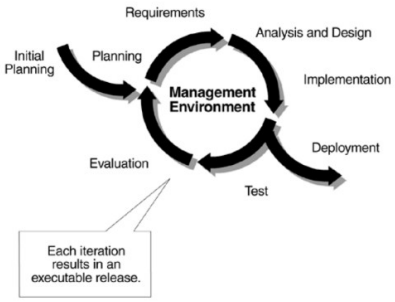
2.3. Laravel

Laravel adalah *Framework* Aplikasi *Web* berbasis PHP yang bersifat *Open Source* diciptakan oleh Taylor Otwell, menggunakan konsep MVC. Pada Maret 2015, Laravel dinobatkan sebagi *framework* PHP paling populer bersama Symfony2, Nette, Codeigniter, Yii2. Laravel mengedepankan *syntax* yang *Expressive, and Beautiful* [7].

Pada penelitian, Laravel digunakan sebagai *framework* untuk pembangunan aplikasi yang lebih cepat dan rapi. Beberapa fitur dari laravel berupa paket modular dengan *dedicated dependecy manager*. Kelebihan *framework* Laravel:

1. Laravel membantu dengan menerapkan batasan antara setiap objek database dengan bantuan *advanced query builder mechanism*.
2. Laravel mempunyai fitur *auto-loading* sehingga tidak perlu lagi untuk melakukan *maintenance* secara manual.
3. Laravel mempunyai fitur *database migration* yang sangat membantu pengerjaan pengembangan aplikasi jika dikerjakan oleh banyak orang dalam bentuk tim, sehingga setiap perubahan pada *database* dapat di kontrol.

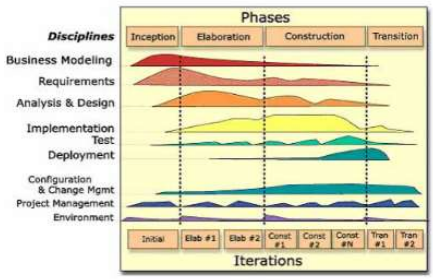
3.1. *Iterative Incremental*



Gambar 2 Tahapan Pada *Iterative Incremental (Kruchten, 2004)*

Pada Gambar 2 dijelaskan tahapan pada proses pengembangan aplikasi menggunakan metode *Iterative Incremental*. Tahapan pada metode *Iterative Incremental* dimulai dengan tahapan Inisiasi yaitu *Planning* dan berakhir dengan tahapan *Deployment* dimana terdapat interaksi di antara *planning, requirements, analysis and design, implementation, test and evaluation* [7].

Pada buku *Applying UML and Patterns* [8], dijelaskan bahwa metode pengembangan sistem dari *Iterative Incremental* terdiri dari beberapa fase, yaitu *inception, elaboration, construction,* dan *transistion* seperti pada Gambar 3:



Gambar 3 Fase Pada Iterative Incremental

Penjelasan dari tiap fase pada metode *Iterative Incremental* adalah:

1. *Inception Phase*
   Tujuan dari tahap *Inception* bukan untuk mendefinisikan seluruh kebutuhan, atau menghasilkan rencana proyek, estimasi biaya yang telah final. Tahap ini memutuskan apakah proyek bernilai dan membutuhkan investigasi yang serius, bukan melakukan investigasi ditahap ini.
2. *Elaboration Phase*
   Tahap *elaboration* adalah tahap awal dimana tim melakukan investigasi, implementasi, klarifikasi kebutuhan pengguna dan kebutuhan sistem, dan menangani kemungkinan – kemungkinan kesalahan dengan serius.
3. *Construction Phase*
   Fase ini dilakukan tahapan – tahapan yang dimulai dari rancangan arsitektural yang siap untuk diproduksi menjadi *code* yang dihasilkan dari analisis, perancangan, implementasi, dan menguji kebutuhan fungsional untuk di distribusikan kepada pengguna.
4. *Transition Phase*
   Fase ini befokus kepada mengenalkan produk yang dihasilkan kepada pengguna, di fase ini pengujian beta dilakukan, melatih pengguna, dan menguji *User Acceptance*. Hasil dari tahap ini adalah manual penggunaan kepada pengguna, aplikasi, atau sistem.

4. Implementasi

Aplikasi yang dibangun pada penelitian bernama Rumantara. Pengimplementasian berfokus kepada pengembangan *back end* API Rumantara. Dengan menggunakan gaya arsitektur REST akan memudahkan dalam mengembangkan sistem kedepannya karena *layer* arsitektur dipisah menjadi *frontend, mobile,* dan *backend*. Sehingga cukup dibutuhkan satu *backennd* untuk melayani banyak *client* di berbagai *platform*.

Pengembangan dari aplikasi *backend* Rumantara menggunakan metode pengembangan perangkat lunak *Iterative Incemental*. Berikut adalah penjelasan hasil dari setiap hasil iterasi.

4.1. Fase *Inception*

Hasil dari fase *inception* di setiap iterasi adalah analisa dari kebutuhan pengguna aplikasi seperti dijelaskan pada Tabel 1:

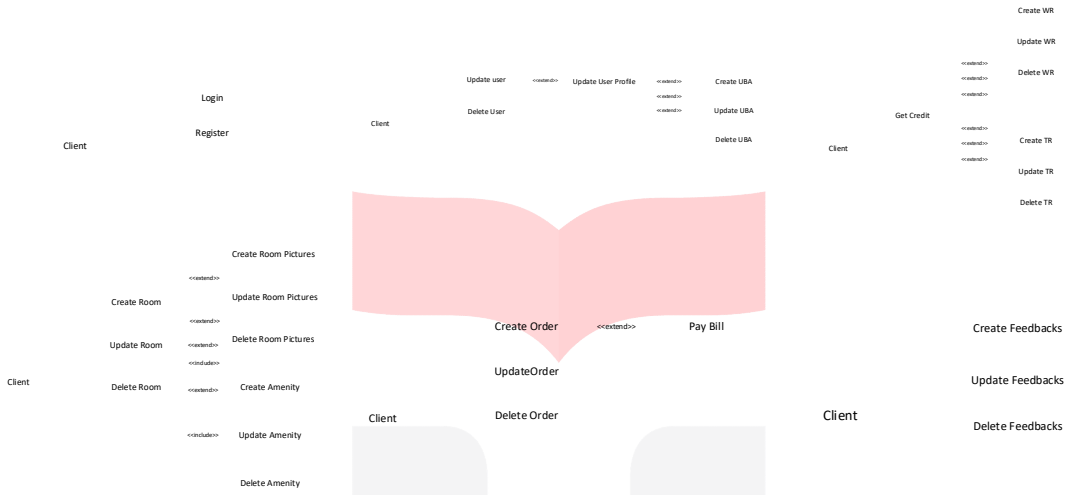
Tabel 1 Kebutuhan Fungsionalitas Sistem

ID	Grup Kebutuhan	Kebutuhan	Deskripsi
REQ-01.01	*Authentication*	*Login*	Proses mendapatkan API *token* untuk mengakses *endpoint* yang di proteksi.
REQ-01.02		*Register*	Proses untuk mendaftarkan pengguna aplikasi *web* dan *mobile*.
REQ-02.01	*Users*	*Get Users*	Proses untuk mendapatkan data seluruh pengguna.
REQ.02.04		*Update User*	Proses mengubah data pengguna tertentu.
REQ.02.05		*Delete User*	Proses menghapus data pengguna tertentu.
REQ.06.01	*Topup Request*	*Create Topup Request*	Proses untuk membuat permintaan penambahan saldo.
REQ.06.03		*Get User Topups*	Proses untuk mendapatkan data seluruh permintaan penambahan saldo pengguna dengapan parameter id pengguna.
REQ.07.01	*Withdraw Request*	*Create Withdraw Request*	Proses untuk membuat permintaan penarikan saldo.
REQ.07.03		*Get User Withdraws*	Proses untuk mendapatkan data seluruh permintaan penarikan saldo pengguna dengapan parameter id pengguna.
REQ.08.01	*Rooms*	*Create Room*	Proses untuk menambahkan data kamar yang akan disewakan.
REQ.08.02		*Get Rooms*	Proses untuk mendapatkan data seluruh kamar yang disewakan.
REQ.08.04		*Update Room*	Proses untuk mengubah data kamar tertentu.
REQ.08.05		*Delete Room*	Proses untuk menghapus data kamar tertentu.
REQ.10.01	*Amenities*	*Get All Amenities*	Proses Mendapatkan data seluruh fasilitas kamar.
REQ.11.01	*Room Amenities*	*Create Room Amenitiey*	Proses untuk menambahkan data fasilitas terhadap kamar.
REQ.12.01	*Room Availabilities*	*Get Room Availabilities*	Proses untuk mendapatkan data ketersediaan kamar berdasarkan parameter id kamar.
REQ.13.01	*Local Heritages*	*Get All Local Heritages*	Proses untuk mendapatkan data seluruh kebudayaan lokal.
REQ.14.01	*Orders*	*Create Order*	Proses untuk membuat pemesanan kamar.
REQ.14.02		*Update Order*	Proses untuk mengubah pemesanan kamar yang belum dikonfirmasi.
REQ.14.03		*Delete Order*	Proses untuk menghapus data pemesanan kamar yang belum dikonfirmasi.
REQ.14.06		*Get Order by ID*	Proses untuk mendapatkan data pemesanan kamar tertentu berdasarkan id.
REQ.15.01	*Bills*	*Get Bill by Id*	Proses untuk mendapatkan tagihan pengguna tertentu berdasarkan id.
REQ.15.04		*Pay Bill*	Proses untuk melakukan pembayaran tagihan oleh wisatawan.
REQ.16.01	*Feedbacks*	*Get All Feedbacks*	Proses untuk mendapatkan seluruh *feedback* dari seluruh kamar.
REQ.16.04		*Create Feedback*	Proses untuk membuat *feedback* kepada sebuah kamar.
REQ.16.06		*Delete Feedback*	Proses untuk menghapus *feedback* yang telah diberikan.

4.2. Fase *Elaboration*

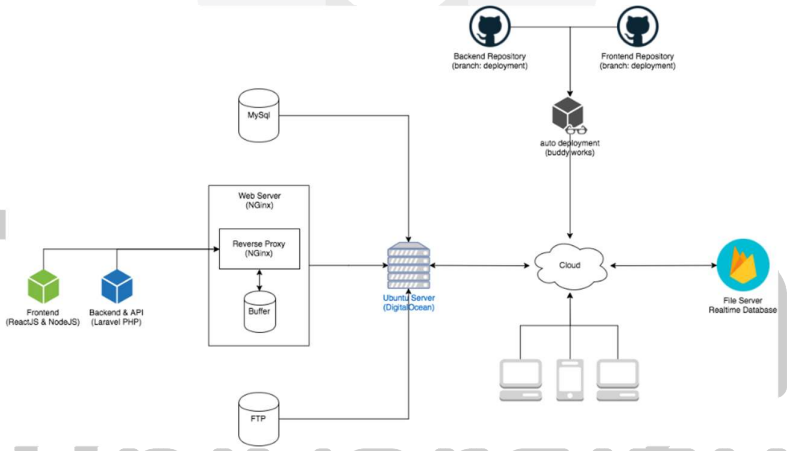
Pada fase ini *output* utama adalah rancangan dalam bahasa pemodelan yang didapat dari hasil analisa kebutuhan pengguna.

Use Case Diagram aplikasi *backend* Rumantara digambarkan pada Gambar 4:



Gambar 4 *Use Case Diagram* Rumantara

Arsitektur Sistem Rumantara digambarkan pada Gambar 5:



Gambar 5 Arsitektur Sistem Rumantara

4.3. Fase *Construction*

Pada fase ini dihasilkan *code* program dari aplikasi yang dibangun berdasarkan rancangan yang telah dibuat. Hasil dari aplikasi didokumentasikan dalam bentuk dokumentasi API. Dokumentasi API berguna untuk memberi informasi kepada pengguna bagaimana melakukan *request* dan mendapatkan *response* dari aplikasi *backend* API Rumanta. Berikut adalah contoh dokumentasi dari API yang dibangun pada Gambar 6:

B.4 *Authentication*

B.4.1 *Login*

HTTP *Request*

Environtment	
Method	POST
Endpoint	{{host}}/oauth/token
Header	
Body	
grant_type	password
Client_id	2
Client_secret	{{client_secret}}
Username	{{username}}
Password	{{password}}
Scope	

HTTP *Response*

Token_type	"Bearer"
Expires_in	31535999
Acces_token	eyJ0eXAiOiJKV1QiLCJhbGciOiJ
Refresh_token	WshsdSkCsIK1zQtPGfePbQ4

Contoh Response

Gambar 6 Contoh Dokumentasi API

4.4. Fase *Transition*

Bagian ini menjelaskan tentang fase *transition* dimana dilakukan pengujian terhadap aplikasi melalui pengujian unit. Berikut adalah hasil dari beberapa pengujian yang dilakukan pada Tabel 2:

Tabel 2 Hasil Pengujian *Unit*

No	Pengujian	*Output* Yang Diharapkan	Hasil
1	Register	HTTP Response dengan code 201 dan message Register Berhasil	Berhasil
2	Get Users	HTTP Response Code 200 dan JSON Object Users	Berhasil
3	Update User	HTTP Response dengan code 200 dan message Update Berhasil	Berhasil
5	Get User Credit	HTTP Response dengan code 200 dan JSON Object User Credit	Berhasil
6	Create Topup Request	HTTP Response dengan code 201 dan message berhasil	Berhasil
7	Get Topup Request	HTTP Response dengan code 200 dan JSON Object Topup Request	Berhasil
8	Create Withdraw Request	HTTP Response dengan code 201 dan message berhasil	Berhasil
9	Get Withdraw Request	HTTP Response dengan code 200 dan JSON Object Withdraw Request	Berhasil
10	Create Room	HTTP Response dengan code 201	Berhasil
11	Get Rooms	HTTP Response dengan code 200 dan JSON Object Room	Berhasil

5. Kesimpulan

Kesimpulan dari penelitian ini adalah:

1. Pembangunan aplikasi *backend* API Rumantara menggunakan gaya arsitektur REST membuat *client* lebih mudah dalam mengakses *endpoint* API dikarenakan penamaan *endpoint* yang sederhana dan penerapan HTTP *method* yang membantu dalam akses ke API.
2. Pembangunan aplikasi *backend* API Rumantara menggunakan *framework* Laravel mempercepat waktu pengembangan dikarenakan telah tersedia fitur Eloquent yang membantu *query* ke *database* dibandingkan dengan menggunakan *query* melalui *raw* sql.
3. Rumantara menggunakan autentikasi pada API untuk menjaga keamanan data sehingga API yangg diproteksi tidak dapat di akses tanpa menggunnakan *token* yang didapat dari autentikasi. Pada pengembangan Rumantara menggunakan Laravel Passport
4. Dokumentasi API menggunakan format Ericsson Interface Description for Partner 1.0 mempermudah *client* dalam memahami API karena setiap *request*, *response*, dan contoh *response* dijelaskan sangat detail.

Daftar Pustaka

[1] Badan Pusat Statistik, 2017. [Online]. Available: https://www.bps.go.id/Subjek/view/id/16#subjekViewTab3.

[2] A. Tsotsit, "Airbnb Hopes To Have Almost A Million Stays A Night By Summer," 27 May 205. [Online]. Available: https://techcrunch.com/2015/05/27/airbnb-hopes-to-have-almost-a-million-stays-a-night-by-summer/. [Accessed 07 Jan 2017].

[3] D. A. Guttentang, "Why tourists choose Airbnb: A motivation-based segmentation study underpinned by innovation concepts," 2016.

[4] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000.

[5] R. Watson, "Developing Best Practices for API Reference," 2012.

[6] D. Lakatos, "The Ten Essentials for Good API Documentation," 2017. [Online]. Available: https://alistapart.com/article/the-ten-essentials-for-good-api-documentation.

[7] B. Skvorc, "The Best PHP Framework for 2015; SitePoint Survey Result," 2015. [Online]. Available: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-result.

[8] P. Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley Proffesional, 2004.

[9] C. Larman, Applying UML and Patterns, 2012.