Membangun Web API dengan menggunakan JSON sebagai format serialisasi data

Muhammad Ghazali
Program Studi Teknik Informatika
Fakultas Teknik
Universitas Widyatama
<muhammad.ghazali@widyatama.ac.id>

1 April 2013

Daftar Isi

Daftar Gambar

Daftar Tabel

Ringkasan

LayangLayang Mobile (LLM) merupakan salah perusahaan yang bergerak di bidang *mobile application development*. Saat ini LayangLayang Mobile sedang mengembangkan sebuah produk bernama CampusLife. Produk yang dikembangkan tersebut merupakan aplikasi *mobile* yang bertujuan untuk membantu civitas kampus mengakses informasi relevan tentang kampus mereka.

Setiap informasi yang ditampilkan melalui aplikasi *mobile* CampusLife merupakan data yang sudah diolah dan diambil dari Web API[1] CampusLife. Saat ini LLM belum memiliki Web API tersebut. Berdasarkan kondisi tersebut, penulis bekerjasama dengan LLM untuk membangun Web API CampusLife. Web API yang akan dibangun bertujuan untuk membuka akses secara tidak langsung ke *data store*[2] yang tersimpan di salah satu layanan *Database as a Service*[3] yang digunakan oleh LLM di AppFog[4]. Seluruh data-data *event* yang tersimpan di *data store* akan diolah oleh Web API menjadi data dengan format yang dapat dikonsumsi dengan mudah oleh aplikasi *mobile* CampusLife. Proses pengelohan tersebut dinamakan serialisasi data[5].

Dalam penelitian ini penulis akan memilih format serialisasi data JSON untuk digunakan merepresentasikan setiap data-data *event* dalam format yang dapat dikonsumsi oleh aplikasi *mobile* CampusLife. Penulis memilih format serialisasi data JSON karena JSON lebih mudah dibaca ditulis dan dibaca oleh mesin (komputer) dan manusia. Selain itu JSON lebih mudah untuk diproses karena memiliki struktur yang lebih sederhana dibandingkan XML[6][3].

Kata kunci: Web API, JSON, Format Serialisasi Data

[1] http://en.wikipedia.org/wiki/Web_API
[2] http://en.wikipedia.org/wiki/Data_store
[3] http://en.wikipedia.org/wiki/Cloud_database
[4] http://www.appfog.com/
[5] Lihat bagian Landasan teori: Serialiasi Data

Bab 1

Pendahuluan

1.1 Latar Belakang dan Masalah

CampusLife adalah *mobile information directory application* yang dikembangkan oleh LayangLayang Mobile (LLM) untuk menyediakan informasi yang relevan kepada civitas kampus. Salah satu fitur utama yang akan dirilis dalam waktu dekat adalah menyediakan informasi *event-event* terbaru kepada civitas kampus.

Setiap informasi yang ditampilkan melalui aplikasi *mobile* CampusLife merupakan data yang sudah diolah dan diambil dari Web API[1] CampusLife. Saat ini LLM belum memiliki Web API dan penulis berniat untuk membangun Web API tersebut. Web API yang akan dibangun bertujuan untuk membuka akses secara tidak langsung ke *data store*[2] yang tersimpan di salah satu layanan *Database as a Service*[3] yang digunakan oleh LLM di AppFog[4]. Seluruh data-data *event* yang tersimpan di *data store* akan diolah oleh Web API menjadi data dengan format yang dapat dikonsumsi dengan mudah oleh aplikasi *mobile* CampusLife. Proses pengolahan tersebut dinamakan serialisasi data[5].

Di antara format serialisasi data yang sudah disebutkan di atas, XML[6] dan JSON[7] merupakan format serialisasi data yang paling terkenal saat ini[2]. Dalam penelitian ini penulis akan memilih format serialisasi data JSON untuk digunakan merepresentasikan setiap data-data *event* dalam format yang dapat dikonsumsi oleh aplikasi *mobile* CampusLife. Penulis memilih format serialisasi data JSON karena JSON lebih mudah dibaca ditulis dan dibaca oleh mesin (komputer) dan manusia.

[1] http://en.wikipedia.org/wiki/Web_API
[2] http://en.wikipedia.org/wiki/Data_store
[3] http://en.wikipedia.org/wiki/Cloud_database
[4] http://www.appfog.com/
[5] Lihat bagian Landasan teori: Serialiasi Data
[6] http://www.w3.org/XML/
[7] http://json.org/

Selain itu JSON lebih mudah untuk diproses karena memiliki struktur yang lebih sederhana dibandingkan XML[6][3].

Smartphone sebagai perangkat tempat aplikasi *mobile* CampusLife berjalan, merupakan salah satu *mobile computing devices* yang memiliki masa hidup baterai dan ketersediaan *bandwidth* yang terbatas.[1]. Dengan kedua keterbatasan tersebut, dalam penelitian ini penulis akan mengkaji penerapan format serialisasi data JSON yang efektif untuk menghasilkan ukuran data yang optimal saat pengiriman data berlangsung dari Web API ke aplikasi *mobile* CampusLife. Hasil akhir yang diharapkan adalah format serialisasi data JSON mampu menghasilkan ukuran data yang optimal untuk dikonsumsi oleh aplikasi *mobile*.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dipaparkan, maka masalah yang akan diteliti dalam penelitian ini dapat dirumuskan sebagai berikut:

1. Bagaimana membangun Web API dengan efektif agar data yang dikirimkan dapat memiliki ukuran data yang optimal untuk dikonsumsi oleh aplikasi *mobile* CampusLife?

2. Bagaimana JSON dapat diterapkan secara efektif agar data yang dihasilkan dapat memiliki ukuran data yang optimal untuk dikonsumsi oleh aplikasi *mobile* CampusLife?

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari pelaksanaan penelitian tugas akhir yang dilakukan penulis adalah:

1. Membangun purwa-rupa Web API yang mampu mengirimkan data dengan ukuran data yang optimal untuk dikonsumsi oleh aplikasi *mobile* CampusLife.

2. Menerapkan JSON dengan efektif untuk menghasilkan data dengan ukuran data yang optimal untuk dikonsumsi oleh aplikasi *mobile* CampusLife.

Penulis mengharapkan hasil penelitian ini akan membawa manfaat positif bagi kepentingan dunia akademik dan praktis dalam hal penerapan format serialisasi data JSON untuk menghasilkan ukuran data yang optimal untuk dikonsumsi oleh aplikasi *mobile*.

1.4 Batasan Masalah

Untuk memperjelas ruang lingkup pelaksanaan penelitian, penulis memiliki batasan masalah meliputi:

1. Pembangunan Web API hanya akan sampai pada tahap purwa-rupa.

2. Pembangunan Web API hanya akan meliputi API untuk mengambil data-data *event*.

3. JSON hanya mampu merepresentasikan data dalam bentuk teks, oleh karena itu data yang akan digunakan hanya terbatas pada data yang berbasis teks.

4. Skema data *event* akan disediakan oleh pihak LLM.

5. Penulis akan melakukan demo untuk mengakses Web API melalui Android[8] *smartphone* yang sudah terpasang aplikasi mobile CampusLife. Aktivitas demo akan difokuskan pada pengambilan data-data *event* dari purwa-rupa Web API yang dibuat oleh penulis.

6. Tidak membahas mengenai keamanan perangkat lunak, data dan jaringan.

7. Pengembangan perangkat lunak menggunakan sebagian praktek dari *Agile* dan tidak akan membahas *Agile* secara komprehensif.

1.5 Metodologi Penelitian

Adapun tahapan penelitian yang akan dilakukan oleh penulis tahap-tahap berikut:

1. Identifikasi masalah. Pada tahap ini penulis merumuskan masalah yang akan diteliti.

2. Perumusan hipotesis. Pada tahap ini penulis merumuskan jawaban sementara dari permasalahan yang diteliti.

3. Pengujian hipotesis. Pada tahap ini penulis menguji penerapan JSON terhadap Web API yang sudah dibangun dan pembangunan perangkat lunak akan dilakukan dengan menggunakan metodologi pengembangkan perangkat lunak *Agile*.

4. Kesimpulan. Pada tahap ini penulis menganalisa hasil pengujian hipotesis.

Adapun cara untuk menunjang penelitian dilakukan dengan melakukan studi kepustakaan, yaitu pengumpulan data dari artikel-artikel, jurnal dan buku-buku yang berhubungan dengan topik pembahasan di penelitian.

1.6 Sistematika Penulisan

Sistematika pembahasan laporan ini terdiri dari enam bab, yaitu:

Bab I Pendahuluan Pada bagian ini berisikan pendahuluan laporan yang berisi latar belakang, rumusan masalah, tujuan dan manfaat penelitian, batasan masalah, metode penelitian dan sistematika penulisan laporan.

[8]http://www.android.com/

Bab II Landasan Teori Pada bagian ini akan dibahas landasan teori yang berkaitan dan digunakan selama masa penelitian.

Bab III Analisis Sistem Pada bagian ini akan dibahas hasil analisis terhadap masalah.

Bab IV Desain Sistem Pada bagian ini akan dibahas hasil perancangan sistem.

Bab V Implementasi dan Pengujian Sistem Pada bagian ini akan dibahas proses implementasi dan pengujian sistem berdasarkan kriteria pengujian yang sudah ditentukan.

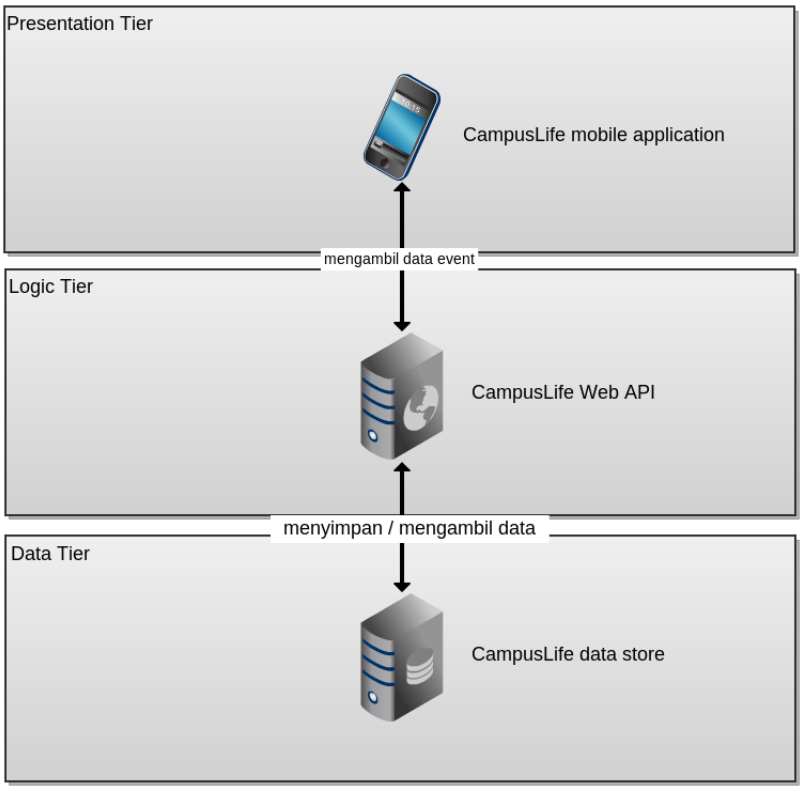
Bab VI Penutup Pada bagian ini berisikan kesimpulan dan saran yang didapatkan dari hasil penelitian.

Bab 2

Landasan Teori

2.1 High-level Architecture CampusLife

Secara umum CampusLife dibangun dengan menggunakan *three-tier architecture* yang terdiri dari 3 tingkat, yaitu *presentation*, *logic* dan *data*. CampusLife dibangun dengan menggunakan *three-tier architecture* dengan tujuan agar pengembang aplikasi bisa melakukan perubahan dengan relatif mudah. Berikut adalah gambaran visual dari arsitektur CampusLife:



Gambar 2.1: High-level architecture CampusLife

2.1.1 Presentation Tier

Presentation tier adalah *tier* paling atas dari aplikasi CampusLife. *Tier* ini menampilkan informasi terkait dengan layanan seperti menampilkan daftar event dan menampilkan daftar pengunjung event kepada pengguna melalui *mobile application*. Layanan informasi seperti ini akan disediakan melalui Web API yang berada di *logic tier* dan dapat diakses melalui *client* seperti *mobile application* dan *browser*. Secara sederhana *presentation tier* adalah tingkat dimana pengguna dapat mengakses aplikasi secara langsung. Dalam konteks pengerjaan penelitian tugas akhir ini, *mobile application* CampusLife akan berada di *tier* ini[7].

2.1.2 Logic Tier

Logic tier adalah *tier* yang berfungsi untuk mengontrol fungsionalitas aplikasi dengan melakukan pengolahan rinci. Contoh pengolahan rinci yang bisa dilakukan di *tier* ini adalah mengambil daftar event. Dalam konteks pengerjaan penelitian tugas akhir ini, Web API akan berada di *tier* ini[7].

2.1.3 Data Tier

Data tier adalah *tier* yang terdiri dari satu atau lebih *server* basis data. Disini setiap informasi disimpan dan diambil. *Tier* ini akan menjaga data tetap netral dan independen dari *application server* atau *business logic*. Selain itu dengan memberikan data di *tier* sendiri akan meningkatkan *scalability* dan *performance*. Dalam konteks pengerjaan penelitian tugas akhir ini, penulis menggunakan MongoDB sebagai basis data di *tier* ini[7].

2.1.4 Keuntungan dari Arsitektur Three-tier

CampusLife dibangun dengan menggunakan arsitektur three-tier untuk mendapatkan beberapa keuntungan berikut:

1. Lebih mudah untuk memodifikasi atau mengganti *tier* apapun tanpa mempengaruhi *tier* lainnya.
2. Memisahkan aplikasi dan fungsi database berarti *load balancing* yang lebih baik.

2.2 Web API

2.2.1 Apa itu Web API

Application programming interface (API) adalah sebuah protokol yang dimaksudkan untuk digunakan sebagai antarmuka oleh komponen perangkat lunak untuk berkomunikasi satu sama lain. Ketika digunakan dalam konteks pengembangan *web*, API biasanya didefinsikan sebagai satu set *Hypertext Transfer Protocol* (HTTP) *request message*, dan disertai dengan *response message* dalam *Extensible Markup Language* atau *JavaScript Object Notation*.

Praktek penerbitan Web API telah memungkinkan masyarakat *web* seperti pengembang aplikasi berbasis *web* untuk membuat arsitektur terbuka untuk berbagi konten dan data antara masyarakat dan aplikasi. Dengan cara ini, konten yang dibuat dalam satu tempat dapat dikirimkan secara dinamis dan diperbaharui di beberapa lokasi di *web* [9].

2.2.2 Kenapa Web API

Ketika Anda menjalankan aplikasi *web*, pada saat tertentu Anda ingin menyediakan Web API. Berikut adalah berapa alasan yang umum kenapa perlu menyediakan Web API [11]:

1. Untuk memungkinkan pengguna Anda mengakses data mereka melalui aplikasi pihak ketiga. Pertimbangkan berapa banyak pihak ketiga Twitter klien yang ada[1], semua menggunakan API Twitter, daripada mengakses langsung ke situs *web*. Hal yang sama berlaku untuk Amazon[2] dan eBay[3], antara lain, yang memungkinkan pengguna untuk mengakses data katalog mereka dan bahkan mengeksekusi penjualan, semua melalui API.

2. Untuk memungkinkan pengembang aplikasi mobile untuk mengakses situs Anda dengan mengirim dan mengambil data menggunakan HTTP.

3. Untuk memungkinkan aplikasi Anda sendiri untuk mengakses data sendiri melalui panggilan Ajax[4]. Bila Anda membuat panggilan JavaScript di latar belakang menggunakan Ajax, kemungkinan besar Anda akan ingin menggunakan panggilan API, menerima XML atau JSON, bukan HTML yang membutuhkan *parsing* lebih lanjut.

[1] http://en.wikipedia.org/wiki/List_of_Twitter_services_and_applications
[2] http://www.amazon.com/
[3] http://www.ebay.com/
[4] https://developer.mozilla.org/en/docs/AJAX

2.2.3 RESTful Web API

Representational State Transfer (REST) mendefinisikan seperangkat prinsip arsitektur dimana penulis dapat merancang layanan Web yang berfokus pada sistem *resource*, termasuk bagaimana keadaan *resource* dipanggil dan ditransfer melalui HTTP oleh berbagai macam klien yang ditulis dalam bahasa (pemrograman) yang berbeda. Jika diukur dengan jumlah layanan Web yang menggunakannya, REST telah muncul dalam beberapa tahun terakhir sebagai model desain layanan Web yang dominan. Bahkan, REST memiliki dampak besar di Web yang telah sebagian besar menggantikan desain antarmuka berbasis SOAP-WSDL dan karena REST memiliki gaya yang jauh lebih sederhana untuk digunakan [12].

Bibliografi

[1] Deepak, G., and Dr. Pradeep B S. "Challenging Issues and Limitations of Mobile Computing." *International Journal of Computer Technology and Applications* 3.1 (2012): Academic Journals Database. Web. 8 Jan. 2013.

[2] Audie Sumaray dan S. Kami Makki. "A comparison of data serialization formats for optimal efficiency on a mobile platform". *6th International Conference on Ubiquitous Information Management and Communication* (2012): Artikel No. 48. ACM Digital Library. Web. 24 Jan 2013.

[3] *Debate: JSON vs. XML as a data interchange format* http://www.infoq.com/news/2006/12/json-vs-xml-debate diakses pada 20 Januari 2012.

[4] *Serialization* http://en.wikipedia.org/wiki/Serialization diakses pada 24 Januari 2012.

[5] *Agile software development* http://en.wikipedia.org/wiki/Agile_software_development diakses pada 24 Januari 2012.

[6] *JSON: The Fat-Free Alternative to XML* http://www.json.org/xml.html diakses pada 20 Januari 2012.

[7] *Wikipedia: Multitier architecture* http://en.wikipedia.org/wiki/Multitier_architecture diakses pada 15 Maret 2013.

[8] *Three-Tier Architecture* http://www.linuxjournal.com/article/3508 diakses pada 18 Maret 2013.

[9] *Application Programming Interface* http://en.wikipedia.org/wiki/Application_programming_inte diakses pada 22 Maret 2013.

[10] *Web API* http://en.wikipedia.org/wiki/Web_API diakses pada 20 Januari 2013.

[11] *APIs* http://www.linuxjournal.com/content/apis diakses pada 20 Januari 2013.

[12] *RESTful Web services: The basics* http://www.ibm.com/developerworks/webservices/library/ws-restful/ diakses pada 14 September 2012.

[13] *Introducing JSON* http://www.json.org/ diakses pada 20 Januari 2013.

[14] *How REST replaced SOAP on the Web: What it means to you* http://www.infoq.com/articles/rest-soap diakses pada 14 September 2012.