BAB II
LANDASAN TEORI

II.1 Sistem Informasi

Pengertian sistem informasi tidak bisa dilepaskan dari pengertian sistem dan informasi. Definisi dari sistem adalah sekelompok dua atau lebih komponen-komponen yang saling berkaitan (*interrelated*) atau subsistem-subsistem yang bersatu untuk mencapai tujuan yang sama (*common purpose*). Sedangkan definisi dari informasi adalah data yang diambil kembali, diolah, atau sebaliknya digunakan sebagai dasar untuk peramalan atau pengambilan keputusan. Sumber dari informasi adalah data. Data adalah fakta dan angka yang tidak sedang digunakan pada proses keputusan, dan biasanya berbentuk catatan historis yang dicatatkan dan diarsipkan tanpa maksud untuk segera diambil kembali untuk pengambilan keputusan. Secara lugas sistem informasi didefinisikan sebagai suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan oleh pihak luar itu (Senn, 1989).

Sistem informasi memiliki tiga fungsi dasar :

1. Menerima data (*input*)
2. Mengubah data menjadi informasi (proses)

3. Untuk memproduksi dan mengkomunikasikan informasi ke dalam *timely fashion* bagi *user* untuk membuat keputusan (*output*).

II.1.1 Komponen Sistem Informasi

Sistem informasi mempunyai enam buah komponen, yaitu *input*, model, *output*, teknologi, basis data, dan kontrol. Keenam komponen ini harus ada bersama-sama dan membentuk satu kesatuan. Jika satu atau lebih komponen tersebut tidak ada, maka sistem informasi tidak akan dapat melakukan fungsinya, yaitu pengolahan data dan tidak dapat mencapai tujuannya, yaitu menghasilkan informasi yang relevan, tepat waktu, dan akurat. Komponen-komponen dari sistem ini dapat digambarkan sebagai berikut ini :

1. *Input*

   Input merupakan data yang masuk ke dalam sistem informasi. Sistem sistem informasi tidak akan dapat menghasilkan *output* jika tidak mempunyai komponen *input.*

2. *Output*

   Produk dari sistem informasi adalah *output* berupa informasi yang berguna bagi para pemakainya. *Output* dari sistem informasi dibuat dengan menggunakan data yang ada di basis data dan diproses menggunakan model tertentu.

3. Basis data

   Basis data adalah kumpulan dari data yang saling berhubungan satu dengan yang lainnya,

tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.

4. Model
   Model yang digunakan di sistem informasi dapat berupa model logika yang menunjukkan suatu proses perbandingan logika atau model matematik yang menunjukkan perhitungan matematika.
5. Teknologi
   Teknologi merupakan komponen yang penting di sistem informasi. Teknologi dapat dikelompokkan ke dalam dua macam kategori, yaitu teknologi sistem komputer (perangkat keras dan perangkat lunak) dan teknologi sistem telekomunikasi.
6. Kontrol
   Kontrol ini digunakan untuk menjamin bahwa informasi yang dihasilkan oleh sistem informasi sifatnya akurat.

II.2 *Service Oriented Architecture (SOA)*

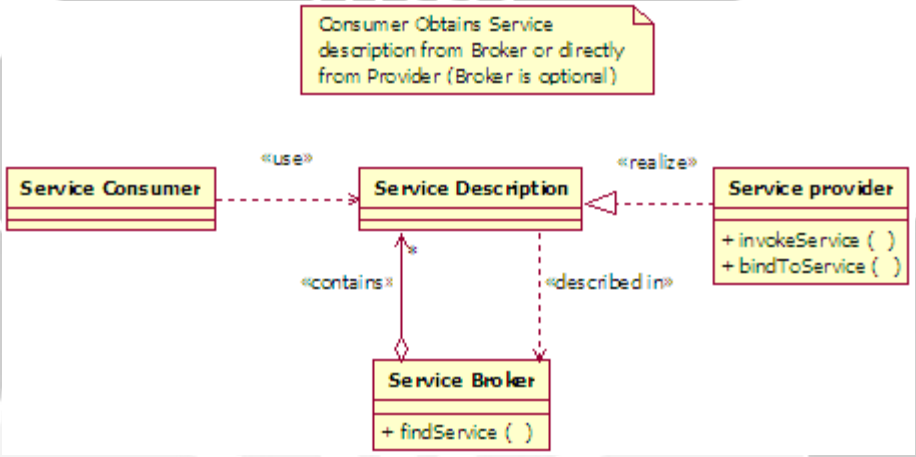
Service oriented architecture merupakan suatu arsitektur yang menyediakan prinsip-prinsip untuk pengelolaan konsep dan perubahan dalam pengembangan sistem dan integrasinya. Arsitektur ini, jika dipenuhi maka akan membungkus fungsionalitas sebagai sebuah layanan(*service)*.

Service sendiri merupakan suatu *unit* fungsionalitas yang tidak terasosiasi dan tidak mengandung pemanggilan ke *service* lain dalam *service*

itu sendiri. Satu *service* melakukan satu aksi, seperti pengisian formulir pendaftaran *account,* menampilkan *account* bank, atau pemesanan tiket untuk penerbangan.

II.2.1 *Conceptual Model SOA*

Konsep dari *SOA* dapat dilihat pada gambar di bawah ini.



Gambar 2.1 *Conceptual Model* Arsitektur *SOA*

Model di atas menggambarkan arsitektur *SOA.* Model di atas menggambarkan interaksi antara 3 komponen utama dari *SOA*: *Service provider,* sebagai pihak yang menyediakan deskripsi dari *service* serta implementasinya, service consumer, sebagai pihak yang dapat menggunakan *service* baik dengan menggunakan *URI(Uniform Resource Identifier)* untuk langsung mengakses layanan atau melalui *service registry* untuk mendapatkan deskripsi dari service dan memanggilnya. Komponen yang terakhir adalah *service broker,* yang menyediakan dan mengelola *service registry,* sehingga *service* dapat ditemukan dan dipanggil oleh *service consumer.*
(http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/)

II.2.2 Prinsip dari Model Arsitektur *SOA*

SOA merupakan arsitektur sistem informasi yang berskala *enterprise* untuk menghubungkan sumber daya yang satu dengan yang lainnya sesuai dengan permintaan. Di dalam *SOA*, sumber daya dibuat tersedia untuk para partisipan dalam sebuah bisnis. *SOA* terdiri dari *service* yang dapat memenuhi kebutuhan proses bisnis dan tujuan bisnis suatu organisasi atau perusahaan. Keluwesan suatu bisnis diperoleh dari sistem informasi yang fleksibel, dalam arti pemisahan antarmuka, implementasi, dan dapat dihubungkan satu dengan yang lain. Hal ini dapat direalisasikan dengan menerapkan *SOA,* yang dapat menyediakan pilihan *service* mana yang harus digunakan dalam waktu tertentu.

Agar *SOA* dapat diterapkan secara maksimal, ada prinsip-prinsip dasar yang harus diterapkan. Prinsip-prinsip dasar tersebut antara lain:

1. *reuse, granularity, modularity, composability, componentization* dan *interoperability.*
2. Sesuai dengan standar yang berlaku.
3. Identifikasi dan kategorisasi dari *service*, penyediaan dan pengiriman *service*, dan pengamatan *service*.

Prinsip-prinsip di atas mendefinisikan aturan-aturan dasar dalam pembangunan, pemeliharaan, dan penggunaan dari *service* dalam *SOA.*

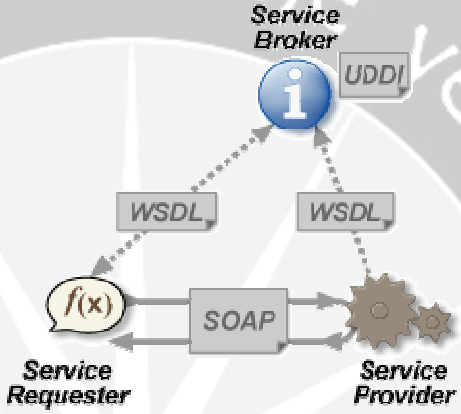
Dalam mengimplementasikan *SOA*, *web service* dapat digunakan untuk membuat *building block yang* diakses melalui standar *Internet protocol* yang mampu mengintegrasikan sistem informasi yang satu dengan sistem informasi lainnya sesuai dengan kebutuhan.

II.3 Web Service

Web service adalah komponen layakan aplikasi yang didesain untuk mendukung interaksi antar aplikasi dan integrasi aplikasi yang biasanya diserialisasi dengan menggunakan XML (eXtensible Markup Language) (dalam teknologi WSE (Web Service Enhancement) memungkinkan web service untuk diserialisasi sebagai binary) dan dapat diakses melalui protokol terbuka yaitu SOAP (Simple Object Access Protocol) dengan bahasa WSDL (Web Service Description Language) dan terdaftar dalam UDDI (Universal Discovery Description and Integration). XML sendiri adalah sebuah standart yang digunakan untuk mendefinisikan data dalam format yang sederhana dan fleksibel.

Web Service menjadi populer saat ini karena web service mampu mengintegrasikan aplikasi yang berbeda platform secara lebih sederhana dan mampu memperbaiki kelemahan dari teknologi sistem terdistribusi lainnya seperti RPC (Remote Procedure Call), Java RMI (Java Remote Method Invocation) maupun arsitektur CORBA (Common Object Request Broker Architecture). Kelemahan utama dari sistem terdistribusi tersebut adalah tidak mendukung heterogenoitas karena masing-masing teknologi mempunyai standar protokol sendiri-sendiri.

Web services menggunakan *protocol* HTTP (HyperText Transfer Protocol) yang merupakan *protocol* yang sangat umum digunakan. Dimana *protocol* tersebut sangat mendukung heterogenoitas dan interoperabilitas serta memudahkan integrasi. Untuk lebih jelasnya bisa dilihat ilustrasi dari cara kerja web service seperti yang diilustrasikan pada gambar 2.2 dibawah ini :



Gambar 2.2 Ilustasi cara kerja web service

Dalam perkembangannya, selain *SOAP web service* seperti yang telah dijelaskan di atas, dikembangkan juga *REST web service. REST web service* merupakan web service yang menggunakan arsitektur *REST* (*Representational State Transfer*). Dengan menggunakan operasi *HTTP* seperti *POST, PUT,* dan *DELETE, GET,* maka dapat dilakukan operasi penambahan, pengubahan, penghapusan, dan penampilan data. *REST web service* tidak membutuhkan pesan *XML* atau *WSDL.*

II.4 *Representational State Transfer (REST)*

REST merupakan suatu arsitektur perangkat lunak untuk sistem *hypermedia* yang terdistribusi. Contoh sistem yang mengimplementasikan REST adalah *world wide web. REST* pertama kali dikenalkan oleh *Roy T. Fielding*

di *University of California* pada tahun 2000. Arsitektur REST memiliki beberapa dasar antara lain : keadaan dan fungsionalitas aplikasi diabstraksikan menjadi *resource,* setiap *resource* secara unik diakses dengan menggunakan URI, setiap *resource* menggunakan antarmuka yang sama untuk pengiriman *state* antara klien dan *resource,* menggunakan *protocol* yang mendukung *client-server, stateless, cacheable, dan layered.* Dengan menerapkan dasar-dasar dari *REST*, maka implementasi komponen menjadi lebih sederhana, meningkatkan kinerja, dan meningkatkan skalabilitas komponen server.

Batasan yang perlu diterapkan dalam implementasi *REST* seperti yang telah disebutkan beberapa di atas, antara lain:

1. *Client-server*
   Klien terpisah dari *server*. Dengan pemisahan ini, berarti sisi klien tidak perlu memiliki hubungan secara langsung dengan tempat penyimpanan data, dan *server* tidak perlu mengurus antarmuka pengguna.
2. *Stateless*
   Keadaan klien dalam pemanggilan layanan di *server* tidak diperlukan. Hal ini berarti tiap permintaan yang dikirimkan klien berisi semua informasi yang dibutuhkan *server* untuk memberikan layanan yang dibutuhkan oleh klien.
3. *Cacheable*
   Klien dapat menyimpan *response* yang dikirimkan *server*. Sehingga, *response* yang dikirimkan *server* harus didefinisikan *cacheable* untuk

mencegah klien menggunakan data yang dikirimkan pada *response* tersebut dengan tidak sah.

4. *Uniform interface*
   Antarmuka yang seragam, menyederhanakan arsitektur *REST.*
5. *Layered system*
   Sebuah klien tidak bisa mengakses langsung ke *server* secara biasa. Klien harus melalui lapisan-lapisan sistem yang diperlukan untuk mengakses *server.*
6. *Code on demand* (pilihan, tidak harus ada)
   Server dapat mengubah fungsionalitas klien dengan mengirimkan suatu urutan logika yang dapat dikerjakan klien.

Konsep yang penting dari *REST* adalah adanya *resource,* yang mana tiap *resource* yang ada diidentifikasikan dengan suatu tanda pengenal(contoh: *URI* pada *HTTP*). Untuk memanipulasi *resource* ini, representasi dari *resource* dikirimkan antara klien dan *server.* Sehingga *resource* yang ada tidak secara langsung dimanipulasi oleh aplikasi yang ada, sebagai contoh: basis data di sisi *server* tidak dikirimkan ke klien, yang dikirimkan adalah representasi dari data di basis data tersebut dalam *format* tertentu seperti *XML*.

Melalui konsep *REST* ini, sebuah aplikasi dapat berinteraksi dengan *resource* yang ada dengan memperhatikan dua hal. Pengenal dari *resource* yang ada, aplikasi tidak perlu mengetahui apakah ada *proxy*, *gateway*, *firewall*, *tunnel*, atau apapun yang berada di antara aplikasi dengan *resource.* Namun aplikasi tetap harus mengetahui *format* dari representasi *resource* yang

dikirimkan, yang mungkin bisa berupa *XML*, *HTML*, atau *JSON.*

II.5 Basis Data

Saat ini peranan basis data sangat penting didalam pengembangan suatu sistem informasi. Pemrosesan basis data menjadi perangkat andal yang sangat diperlukan oleh berbagai instansi atau perusahaan. Basis data akan mempercepat proses perolehan informasi, dan juga dapat meningkatkan pelayanan dari badan yang terkait. Data merupakan fakta mengenai objek, orang dan lain-lain. Data dinyatakan dengan nilai tertentu, berbentuk angka, maupun simbol-simbol.

Basis data adalah suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media, tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data dengan cara-cara tertentu sehingga mudah untuk digunakan atau ditampilkan kembali; dapat digunakan oleh satu atau lebih program aplikasi secara optimal; data disimpan tanpa mengalami ketergantungan pada program yang akan menggunakannya; data disimpan sedemikian rupa sehingga penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol.

Secara tradisional, data diorganisasikan kedalam suatu hirarki yang terdiri atas:

1. Elemen Data

   Elemen data adalah satuan terkecil yang tidak dapat dipecahkan lagi sebagai unit lain yang bermakna.

2. Rekaman

   Rekaman adalah gabungan sejumlah elemen data yang saling terkait.

3. Berkas (File)

   Himpunan seluruh rekaman yang bertipe sama membentuk sebuah berkas.

Perkembangan teknologi basis data sendiri tidak terlepas dari perkembangan perangkat keras dan perangkat lunak komputer. Perkembangan teknologi jaringan komputer dan komunikasi data adalah salah satu penyumbang kemajuan penerapan basis data, yang kemudian melahirkan sistem basis data yang terdistribusi.

II.5.1 *Database Management System* (DBMS)

DBMS merupakan perangkat lunak yang memungkinkan user mendefinisikan, menciptakan dan memanajemen basis data. Fungsi utama dari DBMS adalah:

1. Mendefinisikan basis data dengan cara mendefinisikan tipe data, struktur dan constraint.
2. Membangun sebuah basis data yaitu proses untuk menyimpan data itu sendiri ke dalam media penyimpan.
3. Memanipulasi basis data yaitu suatu proses untuk melakukan query terhadap data tertentu di dalam basis data dan memperbaharui basis data.

Dalam perkembangan selanjutnya untuk membuat derajat kebebasan data yang tinggi sehingga program aplikasi tidak harus terpengaruh oleh perubahan representasi data internal dan menyediakan landasan yang kokoh yang berhubungan dengan masalah yang

berkaitan semantik data, konsistensi data dan redundansi data serta untuk memungkinkan pengembangan bahasa manipulasi data yang bersifat *set-oriented* dikembangkan Relational Database Management System (RDBMS). Di mana model relasional ini berdasarkan konsep relasi dalam matematika yang secara fisik direpresentaikan dalam bentuk tabel. Sebuah relasi adalah sebuah tabel dengan kolom dan baris. Informasi/data disimpan dalam tabel dua dimensi berupa: baris data (*row/record*) dan kolom (*column/field*)

Salah satu DBMS yang terkenal dan digunakan banyak orang adalah MySQL (http://www.mysql.com/). MySQL didistribusikan secara gratis dibawah lisensi General Public License (GPL). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat closed source atau komersial.

II.6 Bahasa Pemrograman

II.6.1 Bahasa Pemrograman Java

Java adalah sebuah platform teknologi pemrograman yang dikembangkan oleh Sun Microsystem. Pertama kali di-*release* tahun 1991 dengan nama kode **Oak**. Kemudian tahun 1995 nama kode **Oak** diganti menjadi **Java**. Yang memotivasi Java dibuat adalah untuk membuat sebuah bahasa pemrograman yang portable dan independent terhadap platform (*platform independent*). Java dapat membuat perangkat lunak yang dapat ditanamkan (*embedded*) pada berbagai mesin dan peralatan konsumer, seperti handphone, microwave, remote control, dan lain-lain. Hal ini kemudian Java

memiliki konsep yang disebut *write once run anywhere.*

Untuk membangun perangkat lunak menggunakan Java maka dibutuhkan Java Development Kit (JDK). JDK release pertama tahun 1996 yaitu JDK 1.1 yang diadopsi oleh Netscape. JDK terbaru sekarang (Desember-tahun 2009, ketika modul ini ditulis) adalah JDK 1.6.0_17. Semua program dan dokumentasi dari JDK ini bisa di-download secara gratis disitus www.sun.java.com.

Karakteristik Java

Sintaks Java merupakan pengembangan dari bahasa C/C++. Berikut adalah beberapa hal tentang pemrograman Java:

1. Bersifat *portable* dan *platform independent*. Program Java yang telah ditulis akan dapat dieksekusi di mesin apapun dan sistem operasi apapun tanpa harus mengubah sedikitpun dari program tersebut.
2. memiliki *garbage collection* yang dapat mendealokasikan memori secara otomatis.
3. menghilangkan sifat pewarisan berganda yang terdapat pada C++.
4. Mengurangi pointer aritmetika. Pengaksesan lokasi memori secara langsung dengan menggunakan pointer memungkinakn program untuk melakukan suatu tindakan yang tidak seharusnya atau tidak boleh dilakukan. Untuk mengurangi kemungkinan kesalahan seperti ini penggunaan

pointer pada Java telah dibatasi dengan menggunakan reference.

5. memiliki array sejati.
6. mengurangi kerancuan antara pemberian nilai pada *statement conditional*.

Java merupakan bahasa pemograman tingkat tinggi yang berorientasi objek atau lazim di sebut dengan istilah Object Oriented Programming (OOP).

II.6.2 Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek dimulai pertama kali dengan bahasa Simula yang dikembangkan di Scandinavia di pertengahan tahun 60-an. Simula utamanya digunakan untuk pemrograman simulasi, dimana adalah alamiah untuk memodelkan suatu entitas eksternal / diluar sistem perangkat lunak dan untuk memilih istilah-istilah untuk entitas-entitas tersebut dan tingkah lakunya. Simula memiliki sintak yang mirip dengan Pascal, tetapi programmer berfikir sedikit lebih berbeda ketika merancang suatu program yang akan dibuat dengan Simula.

Sebuah ide dasar yang diperkenalkan dalam Simula adalah *inheritance* (pewarisan). Dalam Simula juga sudah dikenal objek (entitas) yang ada dalam sistem yang dimodelkan. Ada beberapa objek yang dikumpulkan kemudian disebut ”Class”, dan tugas utama dari seorang perancang program dengan Simula adalah menentukan behaviour dari class tersebut.

II.7 Restlet

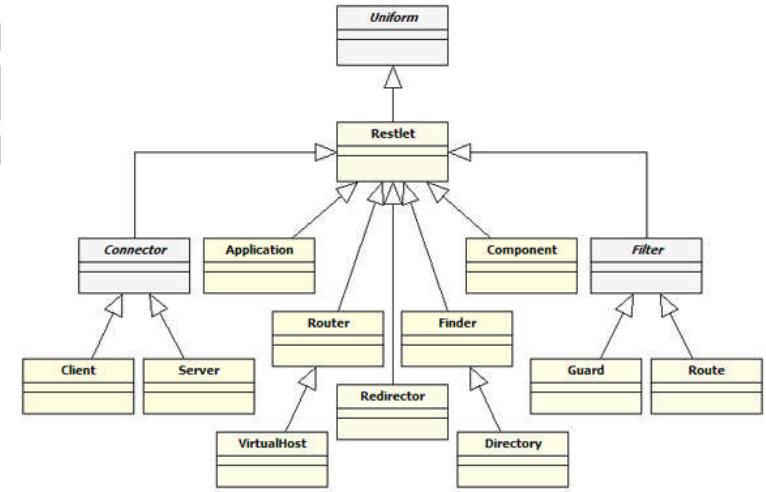
Restlet *framework* merupakan *framework* yang dikembangkan oleh *Jérôme Louvel* pada tahun 2005. Restlet adalah *framework* untuk bahasa pemrograman *Java* yang mendukung pembangunan sistem informasi yang menerapkan prinsip-prinsip *REST*. Alasan dikembangkannya Restlet adalah sedikitnya *framework* untuk bahasa pemrograman *Java* yang mendukung arsitektur *REST* (http://www.restlet.org).

Beberapa keunggulan dari Restlet *framework*:

1. Restlet secara ketat mengikuti arsitektur *REST* yang dikemukakan oleh *Roy T. Fielding,* sehingga aplikasi yang dibangun menggunakan *Restlet framework* dapat dipastikan juga mengikuti arsitektur *REST*. Setiap *class* yang ada di *Restlet framework* mewakili setiap konsep dari arsitektur *REST*.
2. Restlet mendukung berbagai macam protocol web. Hal ini dapat diterapkan untuk sisi klien maupun sisi server.
3. Restlet memisahkan *protocol* tingkat bawah dari aplikasi. Sebagai contoh, untuk mengakses *media type* dari sebuah *request*, kita tidak perlu mengakses ke *HTTP header* dari *request* tersebut, kita hanya perlu menggunakan *coding request.getEntity().getMediaType()*. Pemisahan ini memungkinkan pengaturan isi dari *request* yang dikirimkan ke *server*, dan mengijinkan *protocol connector* untuk melakukan optimisasi yang dibutuhkan tanpa mengganggu *coding* dari aplikasi.

4. Dengan menggunakan *class Routers* dan *Route*, Restlet memberikan kendali secara penuh untuk penanganan pemanggilan dari layanan yang disediakan.
5. Terakhir, *interface Representation* yang ada di Restlet dapat bekerja dengan baik, sebaik kinerja *BIO streams (java.io package) dan NIO channel (java.nio package)*. Hal ini memberikan peningkatan kinerja yang signifikan. Keuntungan yang utama adalah dari peningkatan skalabilitas dengan menghilangkan kebutuhan untuk mengasosiasikan satu *thread* ke setiap koneksi yang terbuka, yang dapat menimbulkan masalah *bottleneck*.

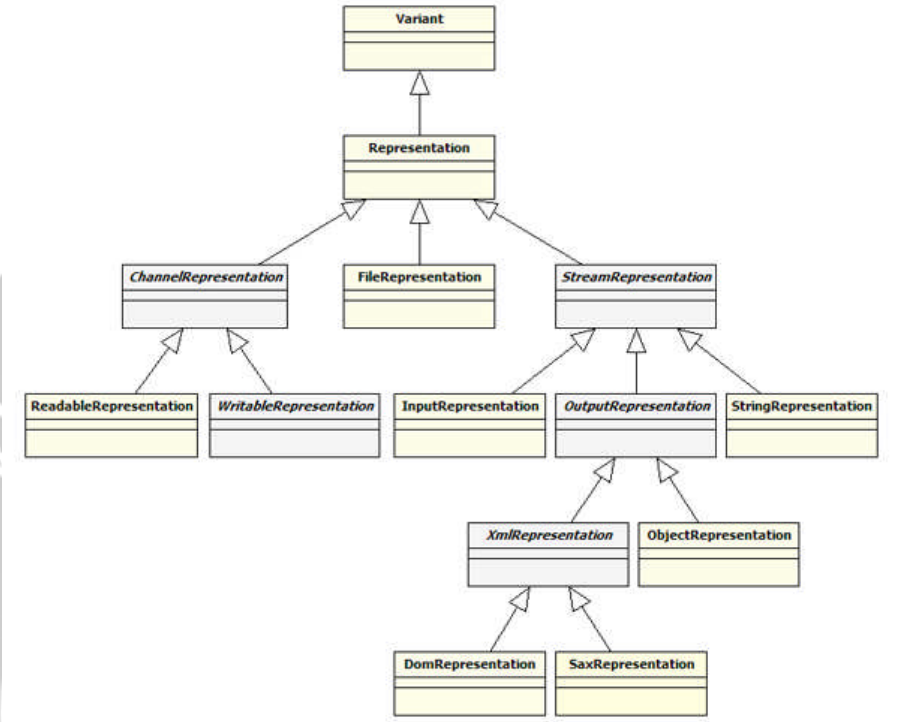
Framework Restlet terdiri dari banyak kelas yang mampu mengimplementasikan aritektur *REST* dalam membuat sebuah aplikasi *server*. Diagram di bawah ini menggambarkan kelas-kelas utama yang ada pada *framework* Restlet beserta dengan relasinya.



Gambar 2.3 Kelas-Kelas *Framework* Restlet

Selain itu, ada juga kelas-kelas yang digunakan sebagai representasi dari *resource* yang ada. Berikut diagram

yang menunjukkan kelas-kelas dari representasi keadaan dari *resource* yang ada.



Gambar 2.4 Kelas-Kelas Representasi *Resource* pada *Framework* Restlet

II.7.1 Kelas-Kelas Dasar Penting pada *Framework* Restlet

1. **Restlet**

   Merupakan kelas yang menyediakan *context* dan mendukung siklus hidup dari aplikasi. Kelas Restlet memiliki banyak *subclass* yang mampu menangani pemanggilan layanan yang lebih spesifik.

2. **Context**

   Data secara kontekstual dan layanan yang ada di kelas Restlet.

3. **Router**

   Kelas yang merupakan *subclass* dari kelas Restlet. Berfungsi untuk mengarahkan pemanggilan ke *resource* yang ada.

4. **VirtualHost**

   Merupakan *subclass* dari kelas Router. Kelas ini menjadi host dari Router yang akan mengakses *resource* yang ada.

5. **Application**

   Restlet yang dapat dipasangkan ke satu atau lebih VirtualHost.

6. **Component**

   Restlet yang mampu menangani beberapa Konektor, VirtualHost, dan Application.

7. **Resource**

   Kelas yang merepresentasikan *resource* yang ada.

8. **Representation**

   Kelas yang merepresentasikan keadaan dari *resource* yang ada.

9. **Variant**

   Kelas yang mendeskripsikan representasi apa saja yang dapat dikembalikan oleh sebuah *resource.*

10. **Request**

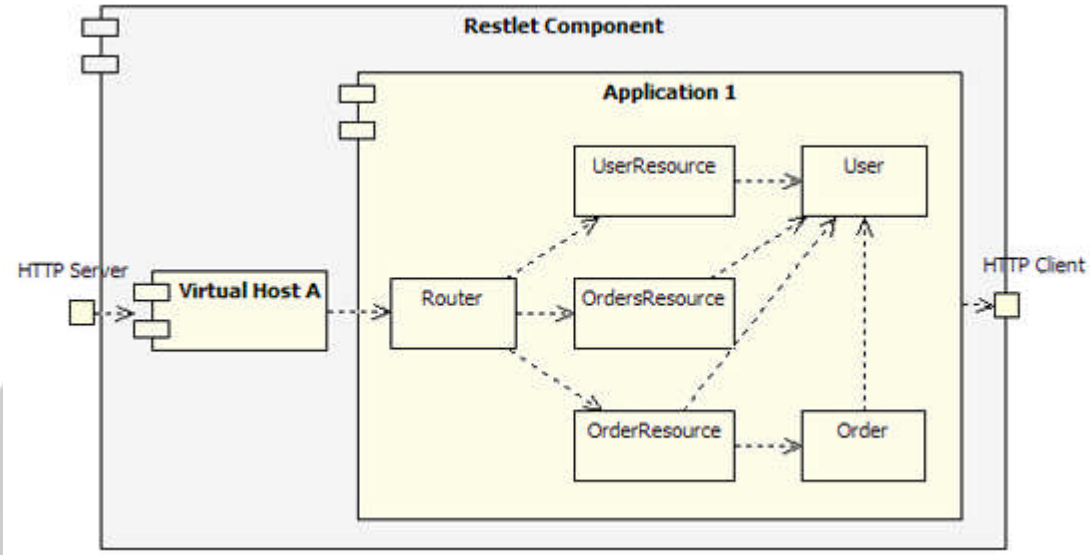
    Sesuai dengan namanya, kelas ini merupakan *request* yang dikirimkan ke *server.*

11. **Response**
    Sesuai dengan namanya, kelas ini merupakan *response* yang dikirimkan ke klien.
12. **Client**
    Kelas ini berperan sebagai klien yang mampu mengirimkan Request ke sisi *server.*

II.7.2 Bagaimana Restlet Menangani *Request*

Untuk mengakses *resource* yang ada, tentunya klien harus mengirimkan *request.* Sebuah *request* dalam *framework restlet* mengandung *URI* yang mengidentifikasikan *resource* yang menjadi menjadi target dari *request* tersebut. Informasi ini disimpan melalui *property* Request.resourceRef dan menjadi dasar atas pengarahan *resource* mana yang akan diakses. Sehingga, hal pertama yang dilakukan ketika *framework restlet* menangani *request* yang dikirimkan klien adalah menemukan *resource* yang ingin dituju dari *request* yang dikirim. Untuk melakukan hal ini, dapat digunakan kelas Finder, yang merupakan *subclass* dari kelas Restlet, yang mengambil referensi dari kelas Reference sebagai *parameter* dan secara otomatis menginisialisasi kelas tersebut, lalu pemanggilan yang dilakukan klien akan diteruskan ke *instance* dari kelas Finder yang diinisialisasikan tadi untuk ditangani *request*nya yang bisa jadi operasi *get, put, post,* atau *delete.* Secara singkat, *request* yang diterima sisi *server* akan diteruskan ke instansi dari kelas Router yang ada, lalu diteruskan ke Resource yang sesuai untuk diproses *request* yang dikirim tadi.

Diagram di bawah akan mengilustrasikan bagaimana *framework* Restlet menangani *request:*



Gambar 2.5 Penanganan Request dalam *Framework* Restlet

II.8 Tinjauan Pustaka

Integrasi sistem informasi merupakan suatu mekanisme yang banyak digunakan sebagai penjembatan antar aplikasi, sehingga aplikasi-aplikasi tersebut dapat berkomunikasi dan bertukar data/informasi. Terdapat beberapa mekanisme integrasi aplikasi, salah satunya adalah *Service Oriented Architecture* (Hery, 2009).

Service oriented architecture digunakan untuk menyediakan layanan(*service*) pada suatu sistem yang dapat digunakan sistem lain sesuai dengan kebutuhan. Arsitektur ini, jika dipenuhi maka akan membungkus fungsionalitas sebagai sebuah layanan(*service)*. *Service orientation* bertujuan untuk memberikan layanan yang dapat diakses sistem lain, sehingga mendukung integrasi antar aplikasi.

Dalam pendekatannya, *Service Oriented Architecture* menggunakan *web service* sebagai jembatan untuk memberikan layanan yang bisa diakses oleh sistem lain. *Web Service* mendukung integrasi antar sistem dengan kemampuannya dalam hal memberikan layanan yang dapat diakses oleh sistem lain.

Beberapa riset mengenai *Service Oriented Architecture* sudah pernah dilakukan. Salah satunya adalah Analisi dan Implementasi Integrasi Sistem Informasi Universitas Atma Jaya Yogyakarta Dengan *Service-Oriented Architecture*(*SOA*)(Hery). Dalam risetnya, penulis membangun sebuah sistem yang mengintegrasikan sistem informasi di Universitas Atma Jaya Yogyakarta bagian kepegawaian, aktivitas akademik dosen, dan penjaminan mutu. Sistem-sistem yang dapat diintegrasikan tersebut dapat bertukar informasi sehingga data yang diperoleh lebih *valid* dan dapat lebih mendukung pengambilan keputusan.

Riset lain mengenai *web service* yang pernah dilakukan adalah Pembangunan Sistem Informasi Geografis Rumah Sakit Wilayah DIY Berbasis Web(Christiana, Rika). Dalam riset ini, penulis membangun sebuah sistem informasi geografis dan sistem informasi untuk rumah sakit. Kedua sistem ini terintegrasi dengan menggunakan *web service.* Hasil dari riset ini adalah sebuah sistem informasi geografis yang mampu memberikan informasi klinik dan jadwal praktek dokter di rumah sakit tertentu.