BAB 2
LANDASAN TEORI

2.1 Portal Pembelajaran

2.1.1 Definisi dan Tipe-tipe Portal

Portal merupakan salah satu contoh dari aplikasi berbasis web. Aplikasi ini menyediakan akses suatu titik tunggal dari informasi *online* terdistribusi, seperti dokumen yang didapat melalui pencarian, kanal berita, dan link ke situs khusus. Untuk memudahkan pengguna, biasanya disediakan kemampuan pencarian dan pengorganisasian informasi-informasi[1].

Menurut PortalsCommunity.com, portal dapat dikelompokkan menjadi empat kategori yaitu[2]:

- Portal *Corporate/Enterprise* (Intranet)
- Portal *e-Business* (Extranet)
- Portal Personal (WAP)
- Portal Mega/Publik (Internet)

2.1.1.1 Portal *Corporate/Enterprise* (Intranet)

Sering disebut Portal Perusahaan atau Business2Employee (B2E). Contoh dari portal ini adalah:

- *Business Intellegent Portal* ~ Portal perusahaan yang memungkinkan pengguna mengakses dan membuat laporan untuk tujuan pembuatan keputusan pada perusahaan yang biasanya menggunakan *database* yang besar.
- *Business Area (Intranet) Portal* ~ Portal ini menyediakan fungsi spesifik atau proses dan aplikasi-aplikasi dalam suatu perusahaan.
- Portal Horizontal ~ Portal ini biasa digunakan dalam hubungan antar organisasi/perusahaan. Contoh dari portal ini diantaranya *Enterprise Collaborative Portal (ECP), Enterprise Expertise Portal (EEP),*

Enterprise Knowledge Portal (EKP), Content Management, dan Manajemen Dokumen.

- *Role Portals* ~ Portal ini mengembangkan pelayanan terhadap 3 model bisnis yaitu *Business to Employee (B2E), Business to Corporate (B2C), dan Business to Business (B2B)*.

2.1.1.2 Portal *e-Business* (Extranet)

Portal ini mempunyai tiga sub-kategori, yaitu:

- *Extended Enterprise Portal* (Portal Perluasan Perusahaan) ~ Portal ini digunakan sebagai perluasan perusahaan untuk para pelanggannya dengan tujuan pemesanan, tagihan, layanan pelanggan, *self service*, dsb. Perusahaan juga dapat memanfaatkan portal ini untuk memperluas perusahaan pada para *supplier* (pemasok barang) dan rekan kerjanya.

- *e-Marketplace Portal* (Portal Pemasaran melalui *net*) ~ Portal ini menyediakan perniagaan yang berhubungan dengan pelayanan kepada komunitas pembeli, penjual, maupun pembuat pasar melalu *net*. Portal akan menyajikan informasi yang dibutuhkan secara cepat dan membeli produk-produk serta jasa secara *online*.

- *ASP Portal* ~ ASP Portal merupakan portal B2B yang mengijinkan pelanggan bisnis menyewa produk maupun jasa.

2.1.1.3 Portal Personal (WAP)

Portal Pribadi ini dapat dibagi menjadi dua tipe utama, yaitu:

- *Pervasive portal atau mobility portal* ~ Portal ini terintegrasi dalam telepon web, telepon seluler, *wireless* PDA, pager, dan sebagainya.

- *Appliance portals* ~ Portal yang tergabung dalam TV (Web TV), otomobil (OnStar), dan sebagainya.

2.1.1.4 Portal Mega/Publik (Internet)

Organisasi yang masuk dalam kategori portal ini akan berfokus membangun pemirsa *online* yang besar dengan demografi yang luas atau berorientasi secara profesional. Ada dua tipe utama Portal Publik, yaitu[2]:

- *General public portals atau Mega Portals* ~ Portal ini sepenuhnya beralamatkan di Internet dengan masing-masing memiliki kepentingan khusus, contohnya seperti Yahoo, Google, MSN, dll.
- *Industrial Portals* ~ Portal ini berfokus pada komunitas yang sempit seperti konsumen suatu barang, bank, asuransi, dsb.

Berdasarkan uraian tipe-tipe portal di atas, maka Portal Pembelajaran *Object Oriented Programming* (OOP) Berbasis Ruby on Rails yang akan dibuat dalam tugas akhir ini termasuk ke dalam kategori *General Public Portals* atau disebut juga *Mega Portals*.

2.1.2 Komponen Teknik Portal

Portal melibatkan berbagai macam komponen teknik yang mendukung berfungsinya portal tersebut[1]. Sebuah portal komprehensif menggabungkan berbagai macam Internet dan aplikasi yang terkait dengan komponen-komponen teknologi. Karena tujuan dari portal adalah menyediakan tampilan tunggal untuk *end user* terhadap informasi yang datang dari berbagai sumber, daftar dari teknologi yang mungkin digunakan tak ada habis-habisnya. Beberapa dari teknologi yang digunakan (seperti *web service*) secara konstan berkembang, sementara yang lainnya (seperti *database* portal, yang biasanya tergabung dalam SQL *database*) secara relatif merupakan teknologi yang stabil[2].

Komponen-komponen teknik portal diantaranya adalah *server* aplikasi, *server web*, *database*, taksonomi, *Crawler* (perayap), gudang metadata, portlet, *Categorization Engine*, *filter*, *index*, *virtual card*, *web service*, protokol dan standar *development* (XML, XSL- XSLT, DTD dan XSD, WDSL, SOAP, UDDI, WSUI), profil pengguna, *Content Management System* (CMS), dan *Enterprise Application Integration* (EAI)[2].

2.1.3 Portal dan Infrastruktur

Portal terdiri dari berbagai macam infrastruktur yang mendukung portal itu berdiri. Tanpa infrastruktur ini portal tidak akan dapat memberikan layanan[1]. Beberapa infrastruktur yang terkait dengan portal diantaranya adalah sebagai berikut[2]:

- *Hosting Server Provider*
- *Service Level Agreement*
- *Platform* (lapisan klien, *server* presentasi, *server* aplikasi, *server* integrasi, *server* data)
- Manajemen Sistem
- *Network* (jaringan)
- *Interoperability* (lintas operasi)
- Penyebaran Teknologi (performansi, ketersediaan)
- *Unified Development Environment* (penyeragaman dalam lingkungan pembangunan)
- *Unity* (kesatuan)

2.1.4 Portal dan Keamanan

Portal adalah kumpulan layanan dari banyak *provider* dan menempatkannya ke dalam presentasi yang terorganisasi yang sesuai atau cocok dengan aliran kerja para pelanggannya (*user*). Para *provider* menggunakan berbagai sistem dan paradigma aplikasi yang berbeda, dimana semuanya mempunyai *hardware* yang berbeda, sistem operasi (*operating system*) yang berbeda, dan paradigma aplikasi yang berbeda untuk mengatur keamanannya[2].

Teknologi-teknologi yang dapat dimanfaatkan untuk mengatur keamanan pada aplikasi portal diantaranya adalah sebagai berikut [2]:

- *Single Sign-on* (SSO) ~ Teknologi *Single Sign on* sangat penting bagi portal. Singkatnya, portal membutuhkannya untuk mengkoordinasikan informasi dari beberapa *website*, penyimpanan data, XML, dan sistem transaksi lainnya. Kesemua ini mempunyai paradigma keamanan yang berbeda dimana solusi *Single Sign-on* akan diterapkan. Contoh dari *vendor*

yang memanfaatkan teknologi ini adalah Netegrity, Oblix, IBM, dan Entrust.

- Pendelegasian Manajemen ~ Sebuah evolusi dari teknologi *Single Sign-on*. Ketika SSO mencoba utuk memfasilitasi aktivitas, Sistem Pendelegasian manajemen mencoba untuk bertindak sebagai titik tunggal bagi pengaturan semua aplikasi dan sistem operasi tingkat keamanan. Sistem Pendelegasian manajemen pada akhirnya akan menggantikan sistem SSO seiring pendewasaannya. Contoh dari *vendor* yang memanfaatkan teknologi ini adalah Netegrity dan IBM.
- *Firewalls* ~ *Firewalls* adalah komputer yang menjalankan *software* yang menganalisis dan menyaring paket jaringan dan membuat keputusan sekuritas berdasar rekomendasinya.
- *Intrusion Detection* (Pendeteksian Penyusup) ~ *Software* ini menganalisis pola aktifitas dalam suatu jaringan untuk mengetahui jika sistem "diserang".
- Kriptografi ~ Ilmu pengetahuan ini menyediakan perhitungan matematis yang teliti dalam rangka otentifikasi, enkripsi, dan *non-repudiation*. Keamanan portal yang tinggi menerapkan kriptografi untuk semua kemampuannya.
- *Access controls* ~ Sistem kendali akses menyediakan aturan berdasar daftar identitas untuk menetapkan sebuah identitas, dimana bagian dari sebuah peranan/tugas dari group/kelompok, harus mempunyai level akses yang tepat untuk menjalankan operasi yang menggunakan sumber daya. Ilmu pengetahuan keamanan komputer adalah kombinasi dari kendali akses dan teknologi kriptografi. Semua portal umumnya menggunakan kendali akses.
- Otentifikasi ~ Otentifikasi mempunyai dua macam format, yaitu format kriptografi dan format kendali akses. Format kriptografi dalam otentifikasi menggunakan skema dasar sertifikasi untuk memastikan identitas. Format

kendali akses lebih sederhana, dimana biasanya menggunakan mandat seperti *user-id* (identitas pengguna) dan *password*.

- *Non Repudiation* ~ Tindakan mempercayakan data pada suatu sistem kunci yang susah dirusak disebut *non repudiation* yang menggunakan teknologi *public key* dan fungsi-fungsi lama dari kriptografi. Portal Finansial, Portal Perawatan Kesehatan akan memperoleh manfaat lebih dari teknologi ini.
- Otorisasi ~ Sebuah fungsi kendali akses yang sangat penting dimana portal akan memelihara daftar otorisasi (daftar kontrol akses) untuk menetapkan level akses yang tepat pada setiap identitas yang bisa mengakses sumber daya. Seperti sebuah sistem akan menetapkan jika *user* diotorisasi terhadap tindakan yang dilakukan terhadap sumber daya.
- *Policy* ~ Sebuah kebijakan keamanan perlu ditetapkan dalam organisasi. Outline kebijakan keamanan dalam bisnis dibutuhkan untuk keamanan dan prosedur organisasi untuk mempertemukan kebutuhan-kebutuhan bisnis. Seperti sebuah kebijakan yang digunakan untuk mendefinisikan kendali akses dan kebijakan sertifikasi.
- *Groups* ~ *Groups* adalah kumpulan identitas yang diorganisasikan. *Groups* diatur atau dikonfigurasikan oleh personil administratif dan diatur di atas basis harian (hari per hari). Portal selalu butuh untuk mengatur group sebagai alat ekonomi untuk mengatur privasi, integritas, dan aksesibilitas yang tepat dari data.
- *Roles* ~ *Roles* mengorganisasikan kumpulan kemampuan. Kumpulan dari kemampuan ini cenderung dipelihara oleh *developer*. *Roles* dapat berupa group dan atau *user* sebagai anggota yang mempunyai akses untuk kemampuan yang didefinisikan oleh *developer*. Keanggotaan *roles* cenderung diatur oleh administrator.
- *Lightweight Directory Access Protocol* (LDAP) ~ Sebuah struktur direktori umum yang diterima oleh sebagian besar industri. Portal

menggunakan ini untuk mengatur informasi pengguna, informasi organisasi, sebagai kontrol akses dan informasi sertifikasi kriptografi.

- *Certificates Authorities atau CA* (Otoritas Sertifikasi) ~ Otoritas sertifikasi adalah penentu keputusan dalam pembuktian, identitas digital, meskipun pada kenyataannya cenderung tidak dapat dipertanggungjawabkan atas hasil kerjanya. Berdasarkan hal ini, dan aksi tanda tangan digital, otoritas sertifikasi tidak dapat diadopsi secara luas. Otoritas sertifikasi dapat menghasilkan sertifikasi. Walaupun ada publik CA, seperti Valicert dan Verisign, tetap banyak perusahaan yang membuat sertikasinya sendiri. CA bermanfaat bagi portal yang menyediakan servis perdagangan dengan nilai tinggi atau jasa pelayanan kesehatan. Mereka juga menyediakan mekanisme *third party* (pihak ketiga) untuk memvalidasi identitas. Aplikasi portal yang lebih kecil akan membuat sertifikasinya sendiri. Tanda tangan digital memungkinkan sertifikasi-diri sendiri. Sertifikasi diri sendiri (*self certified*) ini legal dan valid untuk bertransaksi.
- *Validation Authorities atau VA*(Otoritas validasi) ~ Standar X509 meragukan dan tidak semua sertifikat yang dibuat dari semua *vendor* sama. Dalam hal ini, ketika perusahaan-perusahaan saling bertukar sertifikat sebelum melakukan e-Business, perusahaan “sumber“ membuat sertifikat dan akan mengontrol pemeliharaan sertifikat. Dengan kata lain, jika sumber pengguna “menjadi jahat“ sumber perusahaan pengguna akan meninjau kembali sertifikat tersebut. Otoritas validasi memungkinkan perusahan tujuan melakukan “penerbitan sertifikasi lokal“, hal ini mengurangi kebutuhan komunikasi organisasi yang kuat diantara dua perusahaan yang menerapkan sertifikasi transaksi secara kriptografi.
- *Public Key Infrastructur* (Infrastruktur Kunci Publik) ~ Kriptografi kunci publik menyediakan implementasi elegan (elok) dari implementasi enkripsi, *non repudiation*, dan otentifikasi yang membutuhkan aktifitas manajemen kunci yang minimum. Ini membuat infrastruktur kunci publik lebih efisien untuk diatur bila dibandingkan dengan infrastruktur kunci

simetrik yang tradisional. Portal yang membutuhkan kriptogafi akan menggunakan *Public Key Infrastructure(*PKI).

- *Secure Socket Layer* (SSL) ~ Standar bagi transaksi-transaksi yang butuh keamanan dengan menggunakan kriptografi public key dan X509. Ini dikhususkan bagi otentifikasi (dua arah) dan enkripsi informasi yang dikirim melalui socket TCP/IP. Portal yang membutuhkan transaksi finansial atau data perawatan kesehatan akan menggunakan SSL.
- *Secure Access Markup Languange* (SAML) ~ Terinspirasi oleh Netegrity, bahasa ini dibangun untuk memfasilitas Strategi Manajemen Pendelegasian. Berisi transaksi-transaksi yang tak bereputasi untuk mengatur kontrol aksesnya. Dengan ini diharapkan *vendor-vendor software* akan merangkul SAML untuk memfasilitasi strategi SSO miliknya (segera akan dikenal Manajemen Pendelegasian). Portal akan mengurangi biayanya dalam jangka menengah dengan mengadopsi SAML, sebagai sebuah integrasi dengan paradigma keamanan lainnya yang akan semakin sederhana.
- *Digital Signature* (Tanda tangan Digital) ~ Tanda tangan digital memanfaatkan kemampuan kunci (*non repudiation*) yang susah ditembus dari *public key infrastructure* untuk menyediakan kriptografi yang memastikan data dikelola sebagai integritas.

2.2 Ruby on Rails

2.2.1 Pemrograman Ruby

Bahasa Ruby lahir pada tanggal 23 Februari 1993 dan masuk ke Amerika pada tahun 2000. Pembuat bahasa ini adalah **Yukihiro "Matz" Matsumoto** yang semasa kuliahnya sangat menyenangi pemrograman berorientasi objek dan bahasa *scripting*. Matsumoto kemudian melakukan riset terhadap bahasa *scripting* Perl dan Python, Menurutnya, bahasa Perl kurang *powerful* dan bahasa Python kurang *object-oriented*. Maka lahirlah bahasa Ruby (berarti sejenis batu permata) yang lebih *powerful* daripada Perl dan lebih *object-oriented* dari Python[3].

Fitur-fitur Ruby diantaranya adalah sebagai berikut:

- Merupakan bahasa interpreter.
- Memiliki sintak yang sederhana, mudah dipelajari dan dipahami.
- Mendukung *exception handling* seperti halnya Java dan Python.
- Mendukung *single-inheritance* dan modul *mix-in* yang serupa dengan interface di Java. (Multiple-inheritance dapat digantikan dengan menggunakan fitur *mix-in* ini).
- Memiliki portabilitas yang tinggi antar-*platform* sehingga dapat berjalan di berbagai sistem operasi seperti UNIX, Linux, DOS, Windows 95/98/Me/NT/2000/XP, MacOS, BeOS, OS/2, dan sebagainya.
- Mendukung *dynamic-typing*, seperti halnya pada Python tidak diperlukan pendeklarasian tipe untuk suatu variabel.
- Mendukung *garbage collection* seperti halnya pada Java dan Python tidak diperlukan membebaskan memory yang dialokasi (mis. `free()` di C). Variabel yang tidak lagi digunakan akan segera dibebaskan oleh *garbage collector* sehingga tidak diperlukan manajemen memori.
- Mudah dikembangkan dengan bahasa C seperti halnya Python misalnya dengan menggunakan interface SWIG.
- Lahir dari komunitas, sehingga Ruby memiliki dukungan komunitas yang siap membantu jika menemui kesulitan.
- Gratis, bahkan untuk aplikasi komersial.

2.2.2 *Framework* Ruby on Rails (RoR)

RoR ditulis dalam bahasa pemrograman Ruby. Menurut Russel Kay RoR adalah *full-stack web application framework open-source*[10]. *Full-stack* mengacu pada tingkat kemampuan yang disediakan oleh *framework* RoR. *Web application* adalah *software* aplikasi yang menggunakan *browser* (IE, Firefox, Opera, dll) sebagai aksesnya dalam suatu jaringan Internet atau Intranet. *Framework* dapat diasumsikan sebagai pondasi dari sebuah *software* aplikasi web.

RoR diciptakan oleh David Heinemeier Hansson[11] asal Denmark, seorang mahasiswa perguruan tinggi. RoR pertama kali ditulisnya pada sebuah *project*

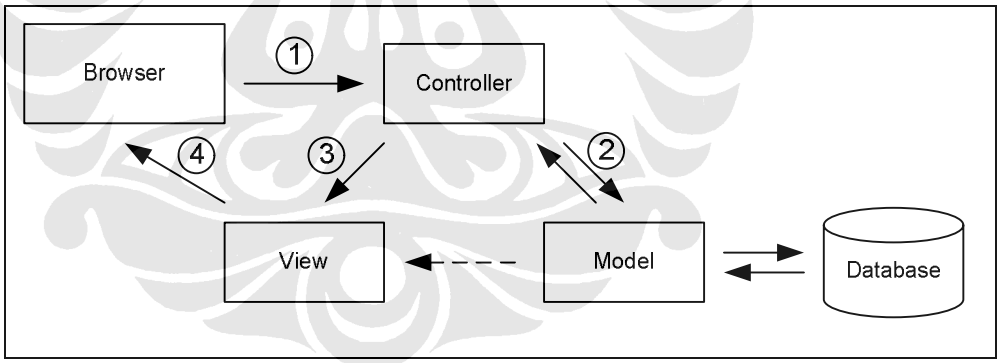
management tool Basecamp[12]. RoR versi beta pertama dilepaskan kepada publik pada bulan Juli 2004, kemudian diikuti dengan versi 1.0 pada tanggal 13 Desember 2005. Hingga saat ini lebih dari 300.000 *framework* RoR telah di *download* dan terus meningkat.

RoR juga termasuk *framework open-source*, dirancang untuk membuat pekerjaan para pengembang web menjadi lebih ringan dengan menyediakan segala hal yang perlu untuk membuat aplikasi berbasis web. RoR dirancang menjadi kolaborasi dari *quick-n-dirty* PHP dengan metodologi *slow-n-clean* JSP[13]. Ini berarti performansi RoR cepat seperti PHP dengan struktur sintak seperti JSP.

2.2.3 Arsitektur Rails

2.2.3.1 *Model, View,* dan *Controller*

Pada tahun 1979, Trygve Reenskaug datang dengan arsitektur yang baru untuk membangun aplikasi yang interaktif. Dalam rancangannya, aplikasi dibagi menjadi tiga komponen, yaitu *model*, *view*, dan *controller*[4].



Gambar 2.1 Arsitektur Model-View-Controller

Dari Gambar 2.1 di atas dapat disimpulkan bahwa proses yang terjadi dalam model MVC secara umum adalah sebagai berikut:

1. Browser akan mengirimkan *request* ke *controller*.
2. *Controller* akan merespon dan berkomunikasi dengan *model*. Komunikasi ini dapat berupa mengakses data, ataupun mengubah data yang disimpan oleh *model*. (Namun tidak semua *model* harus berhubungan dengan *database*).

3. *Controller* akan membuat *view* yang bersesuaian.
4. *Browser* akan me-*render* view yang ada.

Rails adalah sebuah *framework* MVC untuk pengembangan aplikasi berbasis web. Rails ditulis dengan menggunakan Ruby yang dikenal sangat *Object Oriented*. Di dalam pengembangan aplikasi menggunakan Rails dikenal sebuah paradigma yang disebut *convention over configuration*. Misalnya, untuk penamaan tabel di *database*, nama tabel harus menggunakan kata benda/jamak dari nama model yang dimiliki. Sebagai contoh, model ***Article***, maka diberikan tabel articles di *database*. Jika tidak diikuti maka harus dilakukan konfigurasi secara manual untuk melakukan ***mapping*** dari model ***Article*** ke tabel *articles* di *database*.

Arsitektur dengan tiga komponen pada Gambar 2.1 biasa disebut dengan arsitektur MVC. Berikut penjelasan ketiga komponen tersebut:

a. *Model* - bertanggung jawab mengatur kondisi aplikasi. Terkadang kondisi ini bersifat sementara, hanya untuk berinteraksi dengan *user*. Terkadang juga bersifat permanen dan akan disimpan di luar dari aplikasi (biasa disebut *database*).
b. *View* - bertanggung jawab menampilkan *user interface* berdasarkan data yang ada pada *model*.
c. *Controller* - mengendalikan jalannya aplikasi. *Controller* menerima *event* dari luar (input *user*), berinteraksi dengan *model*, dan menampilkan *view* yang sesuai untuk *user*.

2.2.3.2 Prinsip Pengembangan Aplikasi RoR[5]

RoR lebih menonjol jika dibandingkan dengan *framework* lainnya. Hal ini disebabkan RoR memiliki prinsip pengembangan *framework* seperti berikut:

a. Convention Over Configuration

Prinsip ini mengacu pada fakta bahwa RoR diasumsikan sebagai kumpulan sejumlah aplikasi web yang khas yang sudah menjadi standarisasi. Pada *framework* lain seperti *Java-based Struts* atau *Python-based Zope*, sebelum dimulai diperlukan suatu konfigurasi yang panjang.

Informasi konfigurasi biasanya disimpan ke dalam *file* XML yang besar dan sulit untuk di-*maintain*. Atau jika akan memulai suatu proyek baru wajib mengulangi seluruh proses konfigurasi. Heinemeier Hansson dengan sengaja menciptakan RoR sedemikian sehingga tidak memerlukan konfigurasi berlebihan, sepanjang beberapa konvensi yang standar diikuti.

b. *Don't Repeat Your Self*

RoR mendukung prinsip DRY (*Don't Repeat Yourself*) *programming*. Untuk mengubah perilaku suatu aplikasi yang didasarkan pada prinsip DRY, tidak diperlukan memodifikasi terlalu banyak sintak. Contoh bagaimana RoR mendukung prinsip DRY adalah bahwa, tidak seperti Java, RoR tidak memaksa untuk mengulangi definisi *database* di dalam aplikasi yang dibuat.

RoR juga menerapkan teknik Ajax (*Asynchronous Javascript and XML*). Ajax adalah satu pendekatan yang mengizinkan aplikasi web menggantikan isi di dalam *browser* secara dinamis. Misalnya, untuk menukar data sebuah *form* tanpa harus men-*download* seluruh halaman dari *server*. Para pengembang sering kali harus menyalin sintak selagi menciptakan aplikasi-aplikasi Ajax. RoR membuatnya mudah untuk memperlakukan masing-masing generasi *browser* tanpa harus menyalin setiap sintak untuk masing-masing *browser*.

c. *Agile Development*

Pada pendekatan pengembangan *software* tradisional pembuatan sketsa perencanaan biasanya lebih panjang dan kaku. Pendekatan ini biasanya memakai pendekatan aplikasi dari bawah-ke atas, langkah pertamanya berdasarkan data.

Sebaliknya, metode *Agile Development* menggunakan pendekatan adaptif. Pendekatan adaptif ini hanya membutuhkan sejumlah kecil tim sebagai pengembang. Para pengembang *Agile* juga merancang aplikasinya dari atas-ke bawah yang dimulai dari desain *layout*.

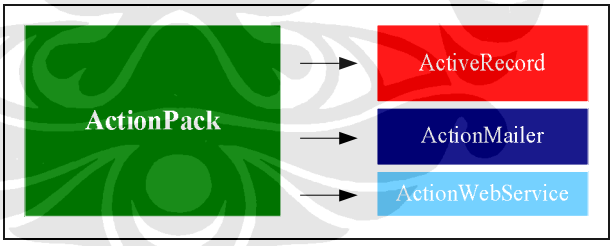
Menurut Patrick Lentz[14] ada beberapa contoh ilustrasi RoR menganut metode *Agile Development*.

- Pekerjaan dapat dimulai dengan mendesain *layout* aplikasi RoR sebelum memutuskan mengenai *database*. Tidak diperlukan membuat ulang *layout* tersebut ketika menambahkan fungsi ke desain, semuanya telah diatur secara dinamis berdasarkan kebutuhan.
- Tidak seperti sintak yang dibuat dalam bahasa C atau Java, aplikasi RoR tidak memerlukan kompilasi sehingga mempercepat proses pengembangan.
- RoR menyediakan otomatisasi uji coba sintak aplikasi pada proses pengembangan.
- *Refactoring (rewriting code with an emphasis on optimization)*.

Berdasarkan prinsip-prinsip di atas, RoR merupakan *framework* yang benar-benar menghemat waktu dalam usaha para pengembang membuat sebuah aplikasi web berbasis RoR.

2.2.3.3 Komponen Ruby on Rails

Secara umum garis besar, *framework* RoR terdiri dari beberapa komponen seperti pada Gambar 2.2 berikut ini:



Gambar 2.2 Komponen Ruby on Rails

a. *ActionPack*

Komponen ini merupakan jantung dari *framework* RoR. Terdiri atas dua sub-komponen:

- *ActionController*, modul ini memastikan pemakaian aksi yang tepat berdasarkan *web-request* setiap pengguna. Aksi ini diambil berdasarkan URL. Contohnya .../books/list akan memanggil metode 'list' pada *BookController-class*.

- *ActionView*, komponen ini digunakan oleh ActionPack untuk mengembalikan *request-view*. ActionView menggunakan *template* RHTML, RJS dan RXML. *Template* ini dapat digunakan untuk men-*generate* HTML/JS/XML.

b. *ActiveRecord*

ActiveRecord merupakan ORM(*Object Relational Mapper*) yang *powerful* dan salah satu kekuatan RoR. Kekuatannya adalah mudah digunakan, tidak memerlukan konfigurasi, hanya mewariskan ActiveRecord base class maka secara otomatis *object-model* dipetakan ke dalam *database-model*. Semua fungsi umum dari relasi *database* telah tersedia seperti proses pembuatan, modifikasi dan pencarian ke dalam *database*. Syaratnya untuk setiap tabel harus memiliki nama *field* id sebagai *primary key*.

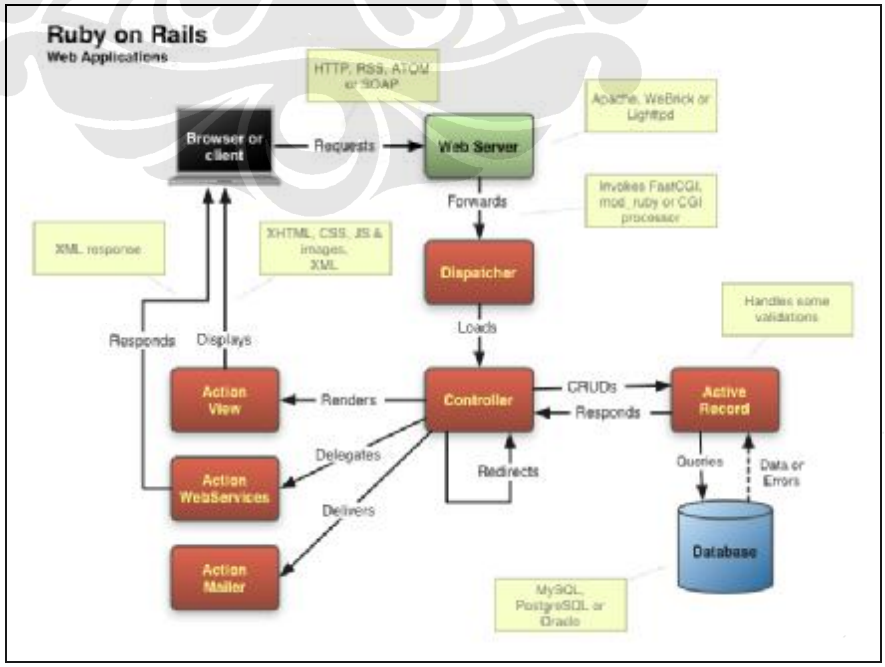
c. *ActionMailer*

Disediakan RoR untuk komunikasi melalui email.

d. *ActiveWebService*

Dukungan RoR terhadap XML-RPC dan SOAP *webservice*.

Berdasarkan komponen-komponen di atas, dapat digambarkan aliran kerja dari aplikasi web berbasis Ruby on Rails pada Gambar 2.3 berikut ini[5]:



Gambar 2.3 Diagram Ruby on Rails[5]

2.2.3.4 Struktur Direktori Aplikasi Rails

Gambar 2.4 di bawah ini menunjukkan struktur direktori dari aplikasi RoR.

Gambar 2.4 Struktur Direktori Rails

2.2.4 Radrails

RadRails pada Aptana merupakan salah satu *open source editor* (IDE) untuk Ruby dalam *framework* RoR. Ada banyak IDE yang dapat mendukung *framework* RoR, namun penulis lebih memilih RadRails karena RadRails memiliki fitur yang lebih lengkap jika dibandingkan dengan IDE lainnya. Perbandingan antara RadRails dengan IDE lainnya diberikan pada Tabel 2.1[6].

Tabel 2.1 Perbandingan Fitur IDE

General	RadRails	NetBeans	3rdRail
Harga	Gratis	Gratis	$399
Tipe Lisensi	*Open Source*	*Open Source*	Komersial
Available Standalone or as Eclipse Plugin	√	×	×
Interpreter Support/Bundling			
Bundled Jruby Interpreter	√	√	×
Interpreter Support	√ Ruby, Jruby, Rubinius	√ Ruby, Jruby	√ Ruby, Jruby
Scriptability/Extensibility			
Scriptable via Ruby	√	×	×

Debugging / Profiling			
Debugger	√ classic and ruby-debug for MRI; ruby debug bundled with Jruby	√ classic and ruby-debug for MRI; ruby debug bundled with Jruby	×
JavaScript Debugging	√	×	√
Profiler	√ (pro)	×	×
Editor			
Editor HTML	√	√	√
Editor CSS	√	√	√
Editor JavaScript	√	√	√
Editor JSON	√	√	×
Editor SQL	√	√	√
Editor YML	√	√	√
Editor RHTML/Erb	√	√	√
Editor XML	√	√	√
Ruby Editing			
Code Completion	√	√	√
Type Inferencing	√	√	√
Ruby-specific search engine (Find usages)	√	√	√
Code analysis(warnings/errors/hints)	√	√	×
Type Hierarchy View	√	×	√
Call Hierarchy View	√	×	√
Mylyn Integration	√	×	√
Reguler Expressions Tester	√	×	√
Quick Outline	√	×	√
Spell Checking Support	√	√	×
Smart Indent	√	√	×
Mark Occurences	√	√	×
Refactoring			
Rename	√ (1)	√	√
Konversi Variabel Local menjadi Field	√	×	×
Encapsulate Field	√	×	×
Extract Method	√	√	×
Extract Constant	√	√	×
Inline Class	√	×	×
Inline Local Variable	√	×	×
Inline Method	√	×	×
Merge Class Parts (internal to *file* & external)	√	×	×
Move Field	√	×	×
Move Method	√	×	×
Push Down Method	√	×	×
Pull Up Method	√	×	×
Split Local Variable	√	×	×
Pengujian			
Menampilkan Test :: Unit	√	√	×
Auto Test	√	√	×
Mendukung Rspec	√	√	×
Rails Specific Functionallity			
Integrated rails-specific “shell”	√	×	√
Log Tail View	√	×	√
Embedded browser	√	×	√

2.3 Web v2.0

2.3.1 Definisi AJAX

AJAX atau *Asynchronous JavScript XML* diperkenalkan oleh Jesse James Garret dari Adaptive Path pada tahun 2005. AJAX adalah teknologi *browser* (*client-side*) yang tidak tergantung pada *software web server* tertentu. Melalui AJAX,

JavaScript dapat dikomunikasikan secara langsung dengan *server* menggunakan obyek JavaScript XMLHttpRequest. Obyek JavaScript ini dapat men-*trade* data sebuah web *server* tanpa harus me-*reload* (*refresh*) halaman web[7].

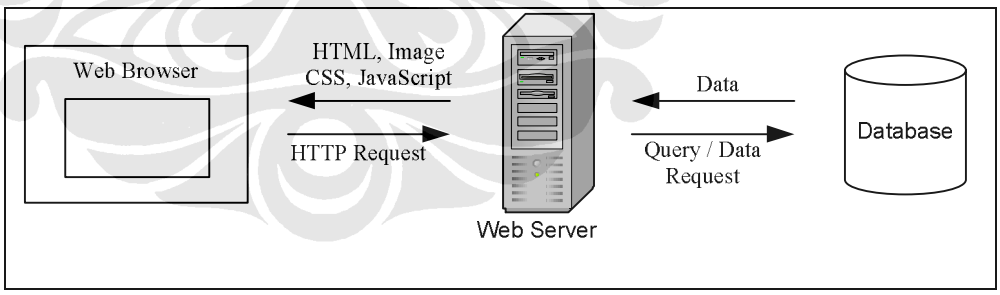
AJAX merupakan aplikasi web yang lebih baik. Aplikasi web menambah keuntungan dibanding aplikasi *desktop* karena hal-hal berikut ini:

- Dapat menjangkau pengguna yang luas
- Mudah diinstall
- Mudah dikembangkan
- Mudah dipelihara

Aplikasi Internet tidak selalu susah dan *user friendly* seperti aplikasi *desktop*. Dengan AJAX, aplikasi Internet menjadi semakin kecil, cepat dan mudah digunakan.

2.3.1.1 Model Tradisional

Pengembangan web secara tradisional bekerja secara *synchronous*, antara aplikasi dan *server*, setiap kali melakukan link atau melakukan operasi "submit" pada form. Model tradisional adalah model yang sering digunakan tanpa AJAX.

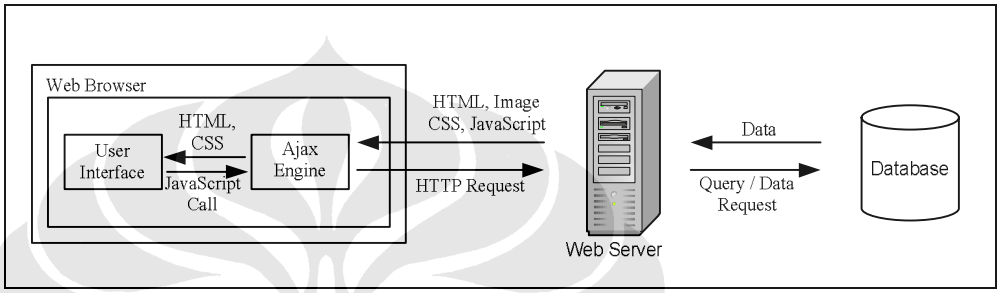


Gambar 2.5 Arsitektur Model Tradisional

Gambar 2.5 merupakan arsitektur model tradisional. Pada model ini (lihat Gambar 2.5), *server* mengirimkan respon berisi seluruh halaman termasuk *header*, logo, navigasi, *footer*, dll. Ketika mengklik *next* maka akan menampilkan halaman baru lagi (artinya *header*, logo, navigasi, *footer* dikirim ulang) dan seterusnya akan mengirimkan data halaman baru lagi setiap diminta *request* dari *user*.

2.3.1.2 Model AJAX

Aplikasi web yang bekerja dengan AJAX bekerja secara *asynchronous*, yang berarti mengirim dan menerima data dari *user* ke *server* tanpa perlu me-*load* kembali seluruh halaman, melainkan hanya melakukan penggantian pada bagian web yang ingin diubah. Dalam model AJAX, aksi dari sisi klien dibagi menjadi dua bagian, yaitu layer *User Interface* (UI) dan layer AJAX.



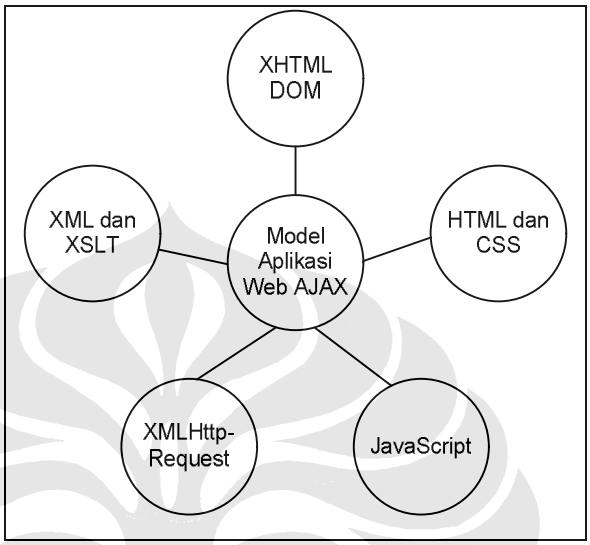
Gambar 2.6 Arsitektur Model AJAX

Gambar 2.6 merupakan arsitektur model AJAX. Berikut ini poin-poin yang penting untuk menggambarkan AJAX:

- Layer AJAX tidak memerlukan komunikasi dengan *server* (contohnya untuk validasi form karena dapat ditangani sepenuhnya oleh sisi klien).
- *Request* antara layer AJAX dan *server* berupa bagian kecil dari informasi (tidak lengkap satu halaman), maka sering digunakan untuk interaksi dengan *database* sehingga waktu *render* dan waktu pengiriman menjadi pendek.
- Layer UI secara langsung tergantung pada respon *server* sehingga *user* dapat melanjutkan interaksi dengan sebuah halaman selama aktivitas dikerjakan di *background* (*background process*). Berarti, untuk beberapa interaksi, waktu tunggu *user* hampir tidak ada.
- Komunikasi antar halaman dan *server* tidak selalu memerlukan AJAX untuk mengubah perubahan UI. Contoh, beberapa aplikasi menggunakan AJAX untuk notifikasi dengan halaman, tetapi tidak melakukan apapun terhadap respon dari *server*.

2.3.2 Teknologi di balik AJAX

Teknologi AJAX merupakan paduan dari beberapa teknologi yang telah dikenal sebelumnya. Adapun komponen-komponen yang terdapat di dalam teknologi ini digambarkan pada Gambar 2.6 berikut ini:



Gambar 2.7 Teknologi di balik AJAX

- ***HyperText Markup Language* (HTML)** digunakan dalam membuat halaman web dan dokumen-dokumen lain yang dapat ditampilkan dalam *browser*. HTML merupakan standar internasional dengan spesifikasi yang ditetapkan oleh *World Wide Web Consortium* (W3C).
- ***Extensible HyperText Markup Language* (XHTML)** adalah bahasa *markup* sebagaimana HTML, tetapi dengan gaya bahasa lebih baik.
- ***Cascading Style Sheets* (CSS)** adalah sebuah mekanisme sederhana untuk memberikan *style* (seperti font, warna, jarak spasi, dll) kepada dokumen web yang ditulis dalam HTML atau XML (termasuk beberapa variasi bahasa XML seperti XHTML dan SVG).
- **JavaScript** adalah bahasa *scripting* kecil, ringan, berorientasi-objek dan lintas platform. JavaScript tidak dapat berjalan dengan baik sebagai bahasa mandiri, melainkan dirancang untuk ditanamkan pada produk dan aplikasi lain seperti web.

- ***Document Object Model* (DOM)** adalah sebuah API (*Application Program Interface*) untuk dokumen HTML dan XML. DOM menyediakan representasi dokumen secara terstruktur, dimungkinkan untuk merubah isi dan presentasi visual. Pada dasarnya, DOM menghubungkan halaman web dengan script atau bahasa pemprograman.
- ***Extensible Markup Language* (XML)** adalah bahasa *markup* untuk keperluan umum yang disarankan oleh W3C untuk membuat dokumen *markup* keperluan khusus. Keperluan utama XML adalah untuk pertukaran data antar sistem yang beraneka ragam.
- ***Extensible Stylesheet Language Transformations* (XSLT)** adalah sebuah bahasa berbasis-XML untuk transformasi dokumen XML. Walaupun proses merujuk pada transformasi, dokumen asli tidak berubah melainkan dokumen XML baru dibuat dengan basis isi dokumen yang sudah ada. XSLT biasanya digunakan untuk merubah skema XML ke halaman web atau dokumen PDF.
- Objek **XMLHttpRequest** untuk melakukan pertukaran data secara asinkron dengan web *server*. Ajax menggunakan obyek XMLHttpRequest untuk melakukan pertukaran data dengan web *server*.
- ***JavaScript Object Notation* (JSON)** yaitu format pertukaran data komputer yang ringan dan mudah. Keuntungan JSON dibandingkan dengan XML adalah pada proses penterjemahan data menggunakan JavaScript. JavaScript dapat menterjemahkan JSON menggunakan *built-in* procedure **eval()**.

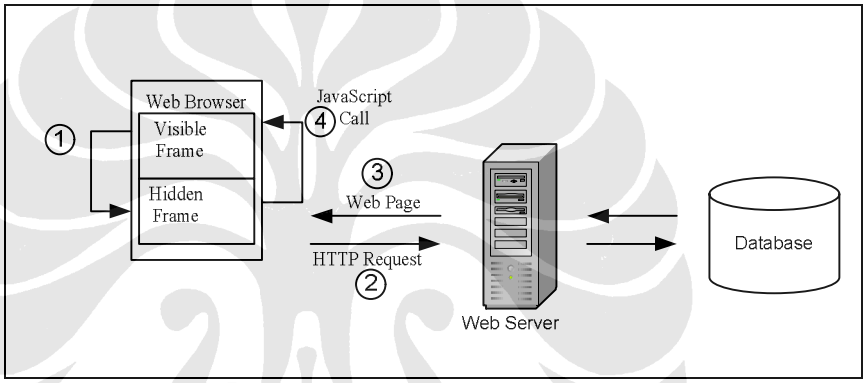
Dalam kenyataannya, semua teknologi dapat digunakan untuk AJAX, tetapi hanya tiga teknologi yang dibutuhkan yaitu HTML/XHTML, DOM dan JavaScript. XHTML diperlukan untuk menamplikan informasi, sedangkan DOM diperlukan untuk halaman XHTML tanpa di-*reload* ulang. JavaScript dibutuhkan untuk komunikasi *client-server*, sementara manipulasi DOM untuk meng-*update* halaman web. Teknologi yang lain digunakan agar AJAX lebih bagus, tetapi bukan merupakan hal yang pokok.

2.3.3 Teknik Penerapan AJAX

Teknik AJAX memungkinkan pengembang web membuat halaman seorang *user* tidak lagi menunggu untuk melakukan aksi selanjutnya. Hal ini berarti memungkinkan komunikasi ke *server* dalam setiap waktu. AJAX memiliki beberapa teknik untuk berkomunikasi yaitu: Hidden Frame dan IFrame, Cache Control (Cookie), dan HTTP Request.

2.3.3.1 Teknik Hidden Frame dan IFrame

Teknik ini secara spesifik memiliki empat pola, seperti terlihat pada Gambar 2.8 berikut ini:



Gambar 2.8 Teknik Hidden Frame AJAX

Metode ini memanfaatkan frame yang tersembunyi. Biasanya salah satu frame di-*set* dengan ukuran tinggi/lebar menjadi 0 pixel sehingga tidak terlihat di halaman. Frame tersembunyi inilah yang sebenarnya melakukan *request* ke dan menerima respon dari *server* sehingga frame yang tampil tidak tampak melakukan *post-back* ke *server*. JavaScript digunakan untuk mengambil data dan mengisi data yang ada di frame yang tersembunyi ini.

Setting hidden frame dapat dilihat pada skrip berikut ini:

```
<frameset rows="100%,*">
```

atau,

```
<iframe height="0" width="0" src="hidden.htm">
```

Hidden IFrame hampir sama dengan hidden frame. Perbedaannya terletak pada elemen yang digunakan, yaitu IFrame, bukan Frame.

2.3.3.2 Teknik Cache Control (Cookie)

Biasanya, ketika membuka halaman yang sama akan terlihat semakin cepat. Hal tersebut disebabkan ketika mengakses pertama kali, *browser* melakukan *caching*. Tujuan *caching* adalah menyimpan informasi yang telah dibuka untuk disimpan di *browser*. Teknik *caching* berfungsi untuk mempercepat dalam menampilkan halaman (yang telah di-*load* sebelumnya).

Penggunaan teknik pemanggilan AJAX beberapa kali akan dapat menyebabkan masalah. Dengan demikian, jalan terbaik adalah menambahkan *no-cache header* agar *browser* tidak melakukan *caching* (`Cache-Control:no-cache`).

2.3.3.3 Teknik HTTP Request

Metode HTTP Request memanfaatkan ActiveX Objek (IE) atau objek melakukan *post-back* ke *server* dan menerima respon balik berupa data (bukan halaman). Data yang didapat dari *server* kemudian diolah di klien untuk ditampilkan di halaman.

XMLHttpRequest adalah metode yang banyak digunakan AJAX untuk berkomunikasi karena memiliki dua fitur yang unik. Fitur utama adalah mempunyai kemampuan me-*load* isi data baru tanpa mengubah seluruhnya dan ini merupakan hal yang berbeda dibanding dengan cara konvensional. Fitur kedua adalah memperbolehkan JavaScript melakukan pemanggilan secara *asynchronous.*

2.3.4 Kelebihan dan Kekurangan AJAX

Dari teknik-teknik yang telah diutarakan pada sub-bab 2.4.3 di atas, terdapat kelebihan dan kekurangan dari masing-masing teknik tersebut. Tabel 2.2 menunjukkan kelebihan dan kekurangan tersebut.

Tabel 2.2 Teknik-teknik dalam AJAX beserta kelebihan dan kekurangannya

Teknik	Kelebihan	Kekurangan
XHR (XMLHttpRequest)	• Dapat mengirim dan menerima (send/get) semua HTTP *header*.	• Pada IE5 dan 6 harus ada ActiveX *request*. • Hanya berjalan pada Opera

	• Dapat membuat *request* HTTP menggunakan tipe (GET, POST, PROPEND). • Dapat mengontrol semua *request* POST, dan semua tipe.	dan Safari versi baru. • Tiap *browser* memiliki perbedaan cara implementasinya.
HF/IF (Hidden Frame/IFrame)	• Dapat menjalankan *request* GET dan POST. • Didukung oleh semua *browser* modern. • Mendukung *upload file* secara *asynchronous*.	• Tidak boleh menggunakan sistem *synchronous*. • Implementasi berbeda pada tiap *browser*. • Meninggalkan *history* tambahan (tergantung dari *browser* dan implementasinya) • Semua *request* data adalah *encode-URL*, dan dapat menambah ukuran.
CK (Cookie)	• Didukung oleh banyak *browser*. • Sedikit perbedaan pada implementasi tiap browser.	• Tidak boleh menggunakan *synchronous requests*. • Tidak boleh bekerja dengan *request* atau hasil yang besar. • Membutuhkan halaman *server* yang mendukung *cookie*. • Hanya menggunakan *request* GET HTTP.

Secara umum, teknik pemrosesan halaman ada dua, yaitu:

a. Pembuatan/manipulasi objek dokumen menggunakan JavaScript

   Klien mengirimkan data dalam format XML/JSON kepada *server* dan mendapatkan data dari *server* berupa XML/JSON. Data tersebut kemudian diolah untuk memanipulasi objek dokumen menggunakan DOM dan JavaScript.

b. Parsial Rendering

   User Interface (UI) dan perilaku UI tidak diproses di klien melainkan di *server*. Klien menerima UI dan perilakunya kemudian melakukan *rendering* pada bagian halaman tersebut.

Dengan AJAX, pertukaran data antara klien dan *server* lebih ringan karena hanya data yang dipertukarkan (bukan halaman) sehingga aplikasi web dapat berjalan lebih cepat.

2.3.5 Keistimewaan AJAX

AJAX memiliki beberapa keistimewaan, antara lain:

- Membuat permintaan kepada *server* tanpa memuat kembali (*reload*) kembali.
- Mengurai (*parse*) dan bekerja dengan dokumen XML dan atau JSON.
- Data yang dikirim sedikit sehingga menghemat *bandwidth* dan mempercepat koneksi.
- Proses dilakukan di belakang layar.
- Banyak didukung oleh *browser-borwser* moder yang popular.
- Aplikasi yang dibangun semakin interaktif dan dinamis.

Keuntungan-keuntungan yang akan didapatkan dalam penerapan AJAX diantaranya adalah:

- *High Usability*: *Update* data tidak me-*reload* keseluruhan halaman, melainkan hanya yang relevan.
- *High Speed*: Aplikasi AJAX lebih cepat dibandingkan dengan aplikasi web konvensional.

2.4 Alat-alat Pemodelan

Alat untuk memodelkan sistem yang digunakan dalam tugas akhir ini adalah *Use-Case Diagram* dan *Activity Diagram*.

2.4.1 *Use-Case Diagram*

Use-Case Diagram secara grafik menggambarkan perilaku sistem (*Use-Case*). Diagram ini digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem [8].

Komponen dari *Use-Case Diagram* ditunjukkan pada Tabel 2.3.

Tabel 2.3 Komponen *Use-Case Diagram*

Gambar	Penjelasan
	*Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang akan berinteraksi dengan sistem aplikasi.

	*Use-Case* merepresentasikan operasi-operasi yang dilakukan oleh *actor*.
	Interaksi atau hubungan antara *Actor* dan *Use-Case* di dalam sistem yang mencakup hubungan asosiasi dan generalisasi.

2.4.2 *Activity Diagram*

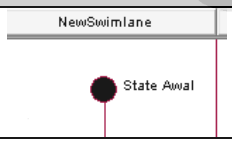
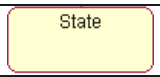
Activity Diagram menggambarkan cara untuk memodelkan *workflow* dari proses bisnis atau cara untuk memodelkan pengoperasian class. Diagram ini mirip dengan *flowchart* karena dalam memodelkan *workflow* dapat dilakukan dari *activity* ke *activity* atau dari *activity* ke *state*.

Workflow

Masing-masing *activity* menggambarkan performansi dari kelompok "*action*" dalam sebuah *workflow*. Ketika *activity* sudah lengkap, *flow control* pindah ke *activity* berikutnya atau kondisi transisi. Jika pada kondisi transisi tidak terpenuhi, maka akan dipenuhi oleh action lainnya di dalam *activity*. Keunikan fitur *Activity Diagram* adalah pada swimlane yang mendefinisikan "siapa" atau "apa" yang bertanggung jawab untuk menjalankan *activity* atau state. Workflow berhenti ketika transisi sampai pada state akhir.

Pada *Activity Diagram*, dapat ditampilkan banyak elemen pemodelan kecuali atribut, asosiasi, atau model elemen lainnya yang tampil dalam komponen tampilan. Komponen *Activity Diagram* ditunjukkan pada Tabel 2.4.

Tabel 2.4 Komponen *Activity Diagram*

Gambar	Penjelasan
	Swimlane yaitu pemisahan tanggung jawab dari setiap aktifitas.
	Kondisi Awal dan Kondisi Akhir dari proses aktifitas.
	*Activity* menggambarkan proses aktifitas yang terjadi.
	*State* menyatakan suatu kondisi.
	*Decisions*
	Transisi menggambarkan arah alur aktifitas.