BAB II

TINJAUAN PUSTAKA

2.1 Profil Tempat Kerja Praktek

2.1.1 Sejarah Instansi

Berawal dari visi Founder Metasistem Solusi, Setiabudi untuk memperkenalkan *OSS (Open Sources Software)* sebagai penerapan aplikasi perangkat lunak yang lebih *accountable, auditable* dan mampu menekan biaya investasi pelakunya maka dikembangkanlah berbagai solusi untuk beragam jenis institusi pengguna.

Selama perkembangan kampanye dan implementasi di lebih dari 100 proyek di berbagai jenis dan skala organisasi memberikan banyak pengalaman untuk menjadi bekal membangun sebuah *simbiosis mutualisma* antara pelaku bisnis dan komunitas pengembang OSS di negeri ini. Selama ini dalam pandangan Founder Metasistem Solusi kedua entitas tersebut cenderung berjalan sendiri-sendiri dengan membawa kepentingannya masing-masing. Pelaku bisnis OSS lebih berkesan sebagai bentuk lain dari bisnis *propietary independent software developer/vendor* dan komunitas OSS cenderung bersikap anti-dagang sehingga lebih mengandalkan donasi dalam mendukung kegiatannya. Padahal sebaiknya kedua entitas tersebut dapat saling bersinergi sehingga masing-masing dapat memetik keuntungan yang proporsional dan memberikan manfaat yang sebesar-besarnya bagi masyarakat.

Setelah jatuh-bangun membangun bisnis berbasis OSS sejak tahun 2000 dan mencari-cari format terbaiknya maka akhirnya pada tahun 2008 terbentuklah Metasistem Solusi yang mencoba menjadi simpul yang mensinergikan entitas bisnis dan komunitas berbasis OSS hingga dapat memberikan manfaat sebesar-besarnya kepada masyarakat luas.

2.1.2 Logo Instansi

Berikut ini adalah logo dari Metasistem Solusi :

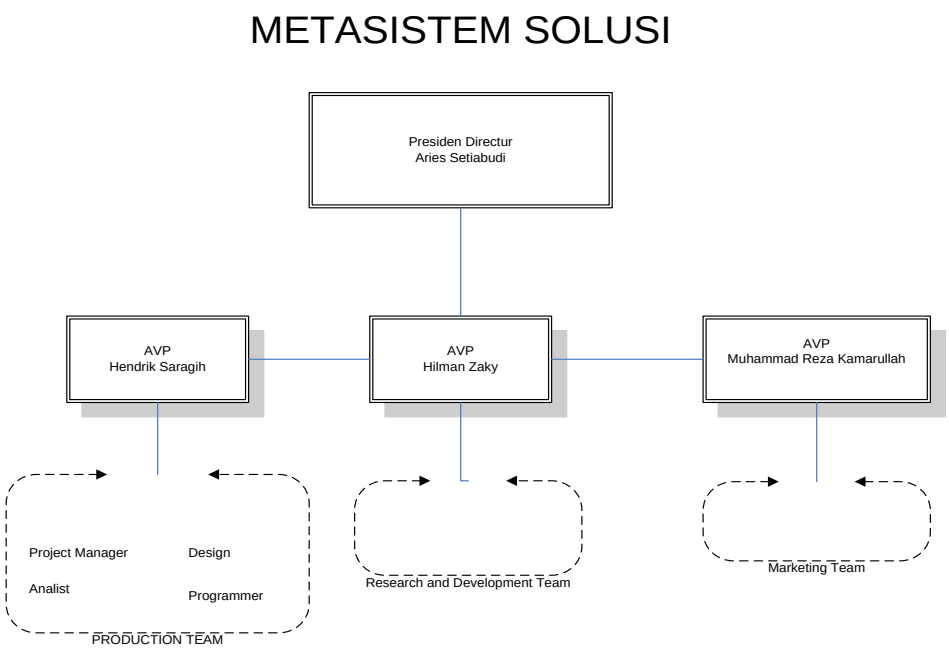
Gambar 2.1 Logo Metasistem Solusi

2.1.3 Badan Hukum Instansi

Sampai sekarang ini sejak berdirinya, Metasistem Solusi mempunyai badan yang resmi yaitu PT (Perseroan Terbatas).

2.1.4 Struktur Organisasi dan Job Description

Berikut ini adalah struktur organisasi yang ada di Metasistem Solusi :



Gambar 2.2 Struktur Organisasi PT.Metasistem Solusi

Metasistem Solusi kemudian secara serius mengembangkan *application framework* untuk

mendukung ERP (Enterprise Resources Planning). Pilihan tersebut diambil dengan mempertimbangkan bahwa setiap organisasi dalam berbagai skala maupun aktifitas bisnis intinya akan selalu membutuhkan aplikasi software yang mendukung ERP. Pilihan pertama adalah penajaman pada aplikasi software ERP bagi lembaga layanan kesehatan yang diberi nama **MetaCare – Healthcare Management System**. Langkah ini diambil setelah para founder Metasistem Solusi melihat bahwa sedikit sekali lembaga layanan kesehatan di Indonesia yang menggunakan aplikasi software untuk mendukung system informasi manajemen mereka. Padahal keberadaan aplikasi software yang mendukung ystem informasi manajemen tersebut berperan penting dalam mengakselerasi peningkatan pelayanan serta tumbuhnya lembaga-lembaga penyedia jasa layanan kesehatan tersebut, mulai dari praktek dokter pribadi, poliklinik hingga rumah sakit.

Akhirnya setelah melalui kerja keras selama 6 (enam) bulan, dirilislah **MetaCare – Healthcare Management System** yang secara kualitas dan kebermanfaatannya diakui oleh para penggunanya hingga memperoleh pengakuan sebagai *The Winner of Open Source Category* di ajang INAICTA 2010.

Sekarang kebanyakan para karyawan di Metasistem Solusi konsentrasi pada pengembangan aplikasi Metacare, dan sebagian lagi ditempatkan untuk menangani proyek-proyek yang kebanyakan berasal dari *e-gov*, termasuk penulis yang pada saat melakukan kegiatan kerja praktek diberikan amanah membantu menganalisa kebutuhan dan melakukan implementasi atau *coding* untuk menangani sebuah proyek pembuatan web pemerintahan kota gorontalo.

2.2 Landasan Teori

2.2.1 Ruby

Bahasa pemprograman Ruby merupakan bahasa pemprograman *open source* berorientasi objek yang dinamis, mudah dimengerti, dan produktif. Tujuan dari Ruby adalah menggabungkan kelebihan dari semua bahasa-bahasa pemprograman *scripting* yang ada di dunia . Secara sintak, Ruby banyak diinspirasi oleh bahasa pemprograman Perl dengan fitur yang mirip dengan Smalltalk. Secara garis besar Ruby banyak dipengaruhi oleh Smalltalk, Perl, Lisp, Scheme, Python, CLU, Eiffel, Ada, Dylan.

Ruby juga mendukung berbagai macam pemprograman, termasuk pemprograman fungsional, berorientasi objek, dan reflektif. Ruby juga memiliki kemampuan pengetikan dinamis (*dynamic typing*) dan manajemen memory secara otomatis.

Menurut Yukihiro Matsumoto (pembuat bahasa pemprograman Ruby), Ruby didesain untuk memenuhi produktivitas dan perasaan senang bagi sistem. Ruby lebih memperhatikan sisi kenyamanan dalam memprogram atau mengoperasikan aplikasi. Dengan demikian Ruby tidak membuat bahasa pemprograman sebagai tuan dan programmer sebagai budak.

Ruby dikatakan sebagai bahasa pemprograman yang mengikuti *principle of least astonishment* (POLA), yang berarti bahasa pemprograman Ruby harus memiliki perilaku yang dapat meminimalisir kebingungan bagi pengguna yang telah berpengalaman dengan bahasa-bahasa pemprograman lainnya.

2.2.2 PostgreSQL

PostgreSQL atau biasa disingkat Postgres, merupakan sebuah *Object-Relational Database Management System* (ORDBMS). Dirilis berdasarkan lisensi MIT yang bersifat *open source* dan tidak berbayar. PostgreSQL menawarkan skalabilitas, keluwesan, dan kinerja yang

tinggi, selain itu PostgreSQL juga didukung oleh banyak platform dan bahasa pemprograman.

PostgreSQL mendukung stardar SQL, selain itu PostgreSQL juga dilengkapi oleh banyak fitur-fitur yang membantu proses pengembangan basis data, diantaranya :

a. Complex queries
b. Foreign keys
c. Triggers
d. Views
e. Transactional integrity
f. Multiversion concurrency control

Selain itu PostgreSQL juga dapat diperluas penggunaanya dengan berbagai cara, diantaranya dapat menambahkan :

a. Tipe data
b. Fungsi-fungsi (functions)
c. Operator
d. Fungsi aggregate (aggregate functions)
e. Metode indeks (index methods)
f. Bahasa Prosedural (procedural languages)

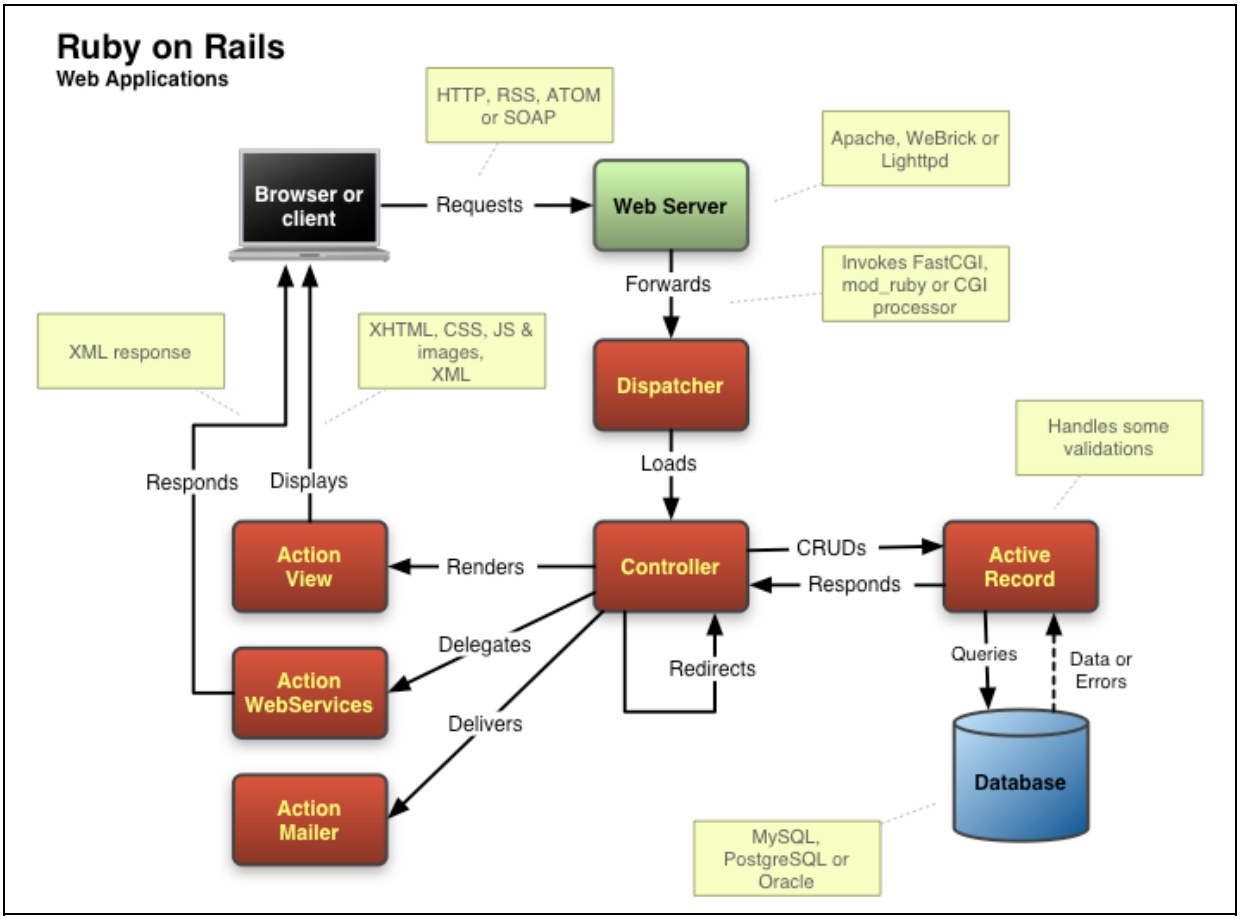
Dalam proses pengerjaan kerja praktek ini, PostgreSQL hanya digunakan sebagai aplikasi pendukung penyimpanan data. Sedangkan proses-proses penambahan foreign key, fungsi,

trigger, transactional, indexing, dan lain-lainnya diatur oleh ActiveRecord yang telah dipaketkan dalam framework Ruby on Rails. Hal ini ditujukan agar aplikasi dapat melakukan migrasi RDBMS dengan mudah, tanpa harus memodifikasi script SQL database sebelumnya.

2.2.3 Ruby on Rails

Ruby on Rails atau seringkali disingkat *Rails* atau *ROR* adalah sebuah *framework* aplikasi WEB yang bersifat *open source*. Framework ini ditujukan untuk bahasa pemprograman Ruby. Ditujukan untuk pengembangan aplikasi dengan metodologi *agile* yang biasa digunakan oleh para pengembang aplikasi web untuk proses pengembangan aplikasi secara cepat.

Berikut ini adalah arsitektur dari *framework* Rails :



[http://vvn.net/wp/2008/09/18/ruby-on-rails-architecture/]

Gambar 2.3 Arsitektur Framework Rails

Framework Rails dibangun untuk memenuhi beberapa kaidah yang memudahkan pihak developer dalam mengembangkan aplikasinya, diantaranya adalah :

Convention over configuration

Framework Rails lebih menekankan aturan penulisan dalam masa pengembangan dibandingkan dengan penekanan konfigurasi. Hal ini menyebabkan pengembang aplikasi memiliki pola yang sama, sehingga memudahkan pengembang aplikasi lainnya dalam memahami *source code* dalam aplikasi tersebut dengan tingkat kesulitan pemahaman yang kecil. Dengan memahami aturan-aturan penulisan pada *framework* Rails, pengembang aplikasi juga dapat mengurangi kode-kode konfigurasi aplikasi yang biasanya terus menerus berulang-ulang dilakukan sehingga meningkatkan kompleksitas kode aplikasi yang dikembangkannya. Aturan-aturan dan kaidah-kaidah penulisan yang digunakan dalam *framework* Rails meliputi *case-sensitive* (huruf besar dan kecil) dan *pluralization* (kata jamak dan tunggal) dalam menentukan Object Relational Mapping (ORM).

Less Code

Dengan aturan-aturan yang ditetapkan dalam *framework* Rails, ini juga berarti bahwa *framework* Rails mengeliminasi kode-kode konfigurasi. Ini juga berarti bahwa *framework* Rails menutupi kode-kode detail, sehingga pengguna *framework* ini tidak perlu melakukan pengkodean hingga ke tahap *low-level*. Dengan demikian maka proses pengembangan aplikasi menjadi sangat cepat dengan bugs yang sangat minim, dan juga membuat kode program yang dikembangkan mudah untuk dimengerti, dikelola, dan ditingkatkan.

Generators

Framework Rails juga menyediakan *generator script* yang bertujuan untuk mempercepat proses pengembangan aplikasi, sehingga pihak pengembang aplikasi dapat difokuskan untuk

berfikir lebih ke dalam kode aplikasi dan *business logic* tanpa harus menentukan struktur aplikasi yang dibuat.

Zero turnaround time

Secara umum siklus proses pengembangan aplikasi untuk melakukan pengetesan terhadap perubahan yang dilakukan memiliki tahapan seperti *configure*, *compile*, *deploy*, *reset*, dan *test*, tahapan-tahapan tersebut tentunya sangat memakan waktu. Namun *framework* Rails tidak demikian, dengan sederhana pengembang aplikasi dapat melakukan perubahan dan melihat perubahannya langsung pada browser tanpa harus melakukan tahapan-tahapan tersebut.

Scaffolding

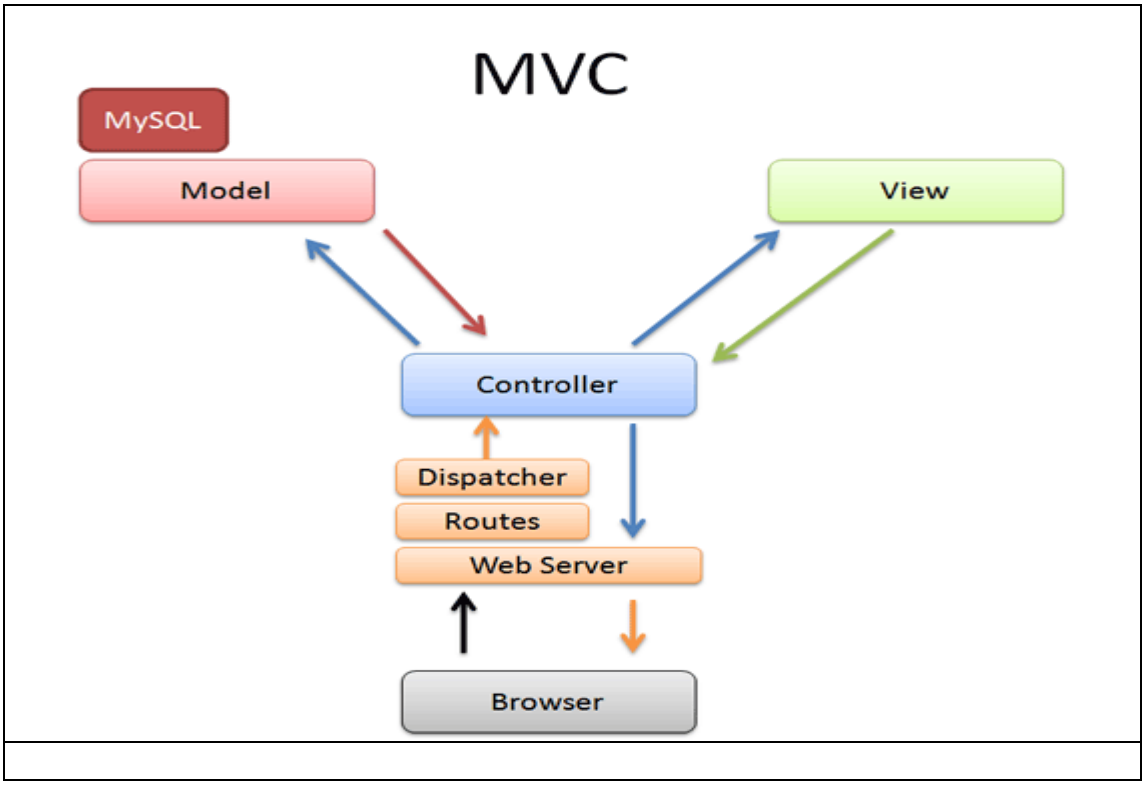
Framework Rails dapat secara otomatis membuatkan sekumpulan operasi CRUD (Create, Retrieve, Update, dan Delete) untuk setiap proses pada database, yang dapat secara langsung digunakan. Selain itu kode-kode hasil *Scaffolding* tersebut dapat dikostumisasi oleh pengembang aplikasi.

Dont Repeat Yourself

Framework Rails juga menyediakan kemampuan untuk meminimalisir pengulangan penulisan kode yang sama untuk operasi-operasi yang berbeda.

2.2.4 MVC (Model View Controller)

Berikut ini adalah konsep MVC (Model View Controller) yang digunakan dalam framework Ruby on Rails :



Gambar 2.4 Aristektur MVC

a. ***Browser*** membuat permintaan, contoh : "http://mysite.com/video/show/15"

b. ***WEB Server*** (mongrel, WEBrick, dan lain-lain) menerima permintaan tersebut. Pembacaan permintaan tersebut diatur oleh kaidah pembacaan yang tetapannya ditentukan dalam ***routes***. *Routes* memiliki peran untuk menentukan *pattern* dari suatu permintaan, secara *default pattern*-nya adalah "/controller/action/id" yang didefinisikan pada file config/routes.rb. Pada kasus permintaan sebelumnya ("http://mysite.com/video/show/15"), maka *controller*-nya adalah "video", *method*-nya adalah "show", dan *id*-nya adalah "15". Lalu *WEB Server* menggunakan ***dispatcher*** untuk membuat *controller* baru, memanggil *action*, dan mengirimkan *parameter*.

c. ***Controller*** melakukan pekerjaannya dengan mem-*parsing* permintaan-permintaan pengguna, submisi data, *cookies*, *sessions*, dan "perlengkapan browser". Dalam kasus

sebelumnya telah diketahui bahwa *method "show"* pada *controller "video"* digunakan untuk melihat sebuah video. Dan meminta ***model*** untuk mengambil video dengan *id* 15, dan dengan segera akan ditampilkan kepada pengguna.

d. ***Model*** merupakan suatu class yang berinteraksi langsung dengan database, dengan kebutuhan seperti penyimpanan dan validasi data, melakukan *business logic*, dan mengangkat suatu data ke dalam aplikasi. Dalam kasus sebelumnya *model* mengangkat data video dengan *id* 15 dari database.

e. ***View*** merupakan sesuatu yang dilihat oleh *end-user* (pengguna aplikasi), dapat berupa : HTML, CSS, XML, Javascript, JSON. *View* memiliki tugas untuk membaca segala sesuatu yang diberikan oleh *controller*. Dalam kasus sebelumnya *controller* memberikan data video dengan *id* yang telah didapatkan dari database untuk kemudian ditampilkan pada *view* "*show*". Di dalam *view* "*show*" kemudian dibuatkan elemen-elemen HTML untuk mempermudah *end-user* dalam membaca data, seperti : div, table, text, description, footer, dan lain-lain.

f. Terakhir, *controller* kembali mengirimkan respon berupa (HTML, XML, metadata) kepada server, kemudian server menggabungkan data-data mentah kedalam bentuk respon HTTP yang layak dan mengirimkannya kepada *end-user*.