BAB II

LANDASAN TEORI

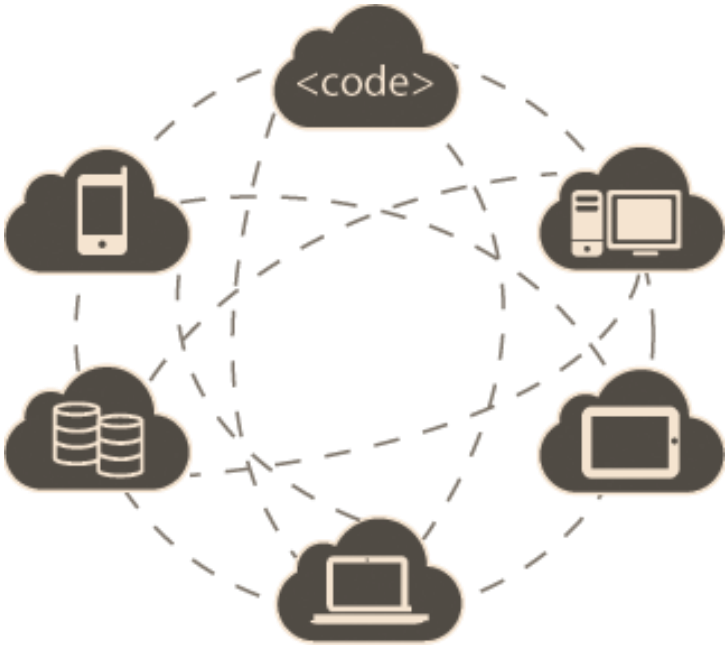
2.1. Cloud Computing

2.1.1. Pengertian *Cloud Computing*

Menurut Peter Mell dan Timothy Grance (2012:2) definisi *Cloud Computing* adalah sebuah model yang memungkinkan untuk *ubiquitous* (Diamanapun dan kapanpun), Nyaman, *On-demand* akses jaringan ke sumber daya komputasi (contoh: jaringan, server, *storage*, aplikasi, dan layanan) yang dapat dengan cepat dirilis atau ditambahkan. *Cloud Computing* sebagai suatu layanan teknologi informasi yang dapat dimanfaatkan oleh pengguna dengan berbasis jaringan/*internet*. Dimana suatu sumber daya, perangkat lunak, informasi dan aplikasi disediakan untuk digunakan oleh komputer lain yang membutuhkan. *Cloud computing* mempunyai dua kata "*Cloud*" dan "*Computing". Cloud* yang berarti *internet* itu sendiri dan *Computing* adalah proses komputasi.

Konsep *Cloud computing* biasanya dianggap sebagai *internet*. Karena *internet* sendiri digambarkan sebagai awan (*Cloud*) besar (biasanya dalam skema jaringan, *internet* dilambangkan sebagai awan) yang berisi sekumpulan komputer yang saling terhubung. *Cloud computing* datang sebagai sebuah evolusi yang mengacu pada konvergensi teknologi dan aplikasi lebih dinamis. Dimana terdapat perubahan besar memiliki implikasi yang menyentuh hampir setiap aspek komputasi. Untuk *end user*,

Komputasi awan menyediakan sarana untuk meningkatkan layanan baru atau mengalokasikan sumber daya komputasi lebih cepat, Berdasarkan kebutuhan bisnis.



Gambar 2.1. Gambaran umum *Cloud computing*

(Sumber Gambar: DataART)

Empat Model mengembangan *cloud*.

1. *Public Cloud*

   Jenis *Cloud* ini diperuntukkan untuk umum oleh penyedia layanannya.

2. *Private Cloud*

   Merupakan infrastruktur layanan *Cloud*, yang dioperasikan hanya untuk sebuah organisasi tertentu. Infrastruktur *Cloud* itu bisa saja dikelola oleh sebuah organisasi itu atau oleh pihak ketiga. Lokasinya pun bisa on-site

ataupun off-site. Biasanya organisasi dengan skala besar saja yang mampu memiliki/mengelola private *Cloud* ini.

3. *Community Cloud*

   Dalam model ini, sebuah infrastruktur *Cloud* digunakan bersama-sama oleh beberapa organisasi yang memiliki kesamaan kepentingan, misalnya dari sisi misinya, atau tingkat keamanan yang dibutuhkan, dan lainnya.

4. *Hybrid Cloud*

   yang menggabungkan baik public dan private. Untuk jenis ini, infrastruktur *Cloud* yang tersedia merupakan komposisi dari dua atau lebih infrastruktur *Cloud* (private, community, atau public). meskipun secara entitas mereka tetap berdiri sendiri, tapi dihubungkan oleh suatu teknologi / mekanisme yang memungkinkan portabilitas data dan aplikasi antar *Cloud* itu. Misalnya, mekanisme loadbalancing yang antar *Cloud*, sehingga alokasi sumberdaya bisa dipertahankan pada level yang Optimal.

2.1.2. Sejarah dan Perkembangan *Cloud Computing*

Ide awal dari cloud computing pada tahun 1960-an, saat John McCarthy, pakar komputasi MIT yang dikenal juga sebagai salah satu pionir intelejensia buatan, menyampaikan visi bahwa “suatu hari nanti komputasi akan menjadi infrastruktur publik, seperti halnya listrik dan telepon”. Namun baru di tahun 1995 lah, Larry Ellison, pendiri Oracle, memunculkan ide “*Network Computing*” sebagai kampanye untuk menggugat dominasi Microsoft yang saat itu merajai desktop *computing*

dengan Windows 95-nya. Larry Ellison menawarkan ide bahwa sebetulnya user tidak memerlukan berbagai *software*, mulai dari Sistem Operasi dan berbagai *software* lain, dijejalkan ke dalam PC Desktop mereka. PC Desktop bisa digantikan oleh sebuah terminal yang langsung terhubung dengan sebuah server yang menyediakan *environment* yang berisi berbagai kebutuhan *software* yang siap diakses oleh pengguna.

Kehadiran konsep ASP (*Application Service Provider*) di akhir era 90-an. Seiring dengan semakin meningkatnya kualitas jaringan komputer, memungkinkan akses aplikasi menjadi lebih cepat. Hal ini ditangkap sebagai peluang oleh sejumlah pemilik data center untuk menawarkan fasilitasnya sebagai tempat 'hosting' aplikasi yang dapat diakses oleh pelanggan melalui jaringan komputer. Dengan demikian pelanggan tidak perlu investasi di perangkat data center. Hanya saja ASP (*Application Service Provider*) ini masih bersifat "*private*", di mana layanan hanya dikastemisasi khusus untuk satu pelanggan tertentu, sementara aplikasi yang di sediakan waktu itu umumnya masih bersifat client-server.

Kehadiran berbagai teknik baru dalam pengembangan perangkat lunak di awal abad 21, terutama di area pemrograman berbasis *web* disertai peningkatan kapasitas jaringan internet, telah menjadikan situs-situs internet bukan lagi berisi sekedar informasi statis. Tapi sudah mulai mengarah ke aplikasi bisnis yang lebih kompleks.

Popularitas Cloud Computing semakin menjulang saat di awal 2000-an, Marc Benioff ex VP di Oracle, meluncurkan layanan aplikasi CRM (*Customer Relationship Management*) dalam bentuk *Software as a Service*, Salesforce.com.

Pada tahun 2005, mulai muncul inisiatif yang didorong oleh nama-nama besar seperti Amazon.com yang meluncurkan Amazon EC2 (*Elastic Compute Cloud*), Google dengan Google App Engine-nya, tak ketinggalan raksasa biru IBM meluncurkan Blue Cloud Initiative. Semua inisiatif ini masih terus bergerak, dan bentuk Cloud Computing pun masih terus mencari bentuk terbaiknya, baik dari sisi praktis maupun dari sisi akademis.

DiIndonesia yang menawarkan layanan komputasi awan salah satunya ialah PT Telekomunikasi Indonesia, Layanan yang ditawarkan berupa aplikasi berbasis *Software as a Service* (SaaS) dan *Infrastrukture as a Service* (IaaS). PT Telkom (Telekomunikasi Indonesia) menawarkan layanan *e-Office on Demand* untuk kebutuhan kolaborasi atau korespondensi di dalam suatu perusahaan atau organisasi. Teknologi informasi dalam negeri masih kurang "peminat" terhadap *Cloud Computing* ini, Beberapa faktor penyebabnya, di antaranya:

1. Penetrasi infrastruktur internet yang bisa dibilang masih terbatas, *bandwith* masih terbatas
2. Tingkat kematangan pengguna internet, yang masih menjadikan media internet utamanya sebagai media hiburan atau sosialisasi.
3. Tingginya investasi yang dibutuhkan menyediakan layanan cloud ini, karena harus merupakan kombinasi antara infrastruktur jaringan, *hardware* dan *software* sekaligus.

2.1.3. Karakteristik *Cloud Computing*

Lima kriteria yang harus dipenuhi oleh sebuah sistem untuk bisa di masukkan dalam keluarga *Cloud Computing*, yaitu :

1. *On Demand Self Service*

   Seorang pelanggan dimungkinkan untuk secara langsung "memesan" sumber daya yang dibutuhkan, seperti processor time dan kapasitas penyimpanan melalui *control panel* elektronis yang disediakan. Jadi tidak perlu berinteraksi dengan personil *customer service* jika perlu menambah atau mengurangi sumberdaya komputasi yang diperlukan.

2. *Broadband Network Access*

   Layanan yang tersedia terhubung melalui jaringan pita lebar, terutama untuk dapat diakses secara memadai melalui jaringan internet, baik menggunakan thin client, thick client ataupun media lain seperti smartphone.

3. *Resource pooling*

   Penyedia layanan *cloud*, memberikan layanan melalui sumberdaya yang dikelompokkan di satu atau berbagai lokasi data center yang terdiri dari sejumlah *server* dengan mekanisme *multi-tenant*. Mekanisme *multi-tenant* ini memungkinkan sejumlah sumberdaya komputasi tersebut digunakan secara bersama-sama oleh sejumlah user, di mana sumberdaya tersebut baik yang berbentuk fisik maupun virtual, dapat dialokasikan secara dinamis untuk kebutuhan pengguna/pelanggan sesuai permintaan. Dengan demikian, pelanggan tidak perlu tahu bagaimana dan darimana permintaan akan

sumberdaya komputasinya dipenuhi oleh penyedia layanan. Yang penting, setiap permintaan dapat dipenuhi. Sumberdaya komputasi ini meliputi media penyimpanan, *memory*, *processor*, pita jaringan dan mesin *virtual*.

4. Elastis (*Rapid elasticity*)

   Kapasitas komputasi yang disediakan dapat secara elastis dan cepat disediakan, baik itu dalam bentuk penambahan ataupun pengurangan kapasitas yang diperlukan. Untuk pelanggan sendiri, dengan kemampuan ini seolah-olah kapasitas yang tersedia tak terbatas besarnya, dan dapat "dibeli" kapan saja dengan jumlah berapa saja.

5. *Measured Service*

   Sumberdaya cloud yang tersedia harus dapat diatur dan dioptimasi penggunaannya, dengan suatu sistem pengukuran yang dapat mengukur penggunaan dari setiap sumberdaya komputasi yang digunakan (penyimpanan, memory, processor, lebar pita, aktivitas user, dan lainnya). Dengan demikian, jumlah sumberdaya yang digunakan dapat secara transparan diukur yang akan menjadi dasar bagi user untuk membayar biaya penggunaan layanan.

2.1.4. Model Layanan *Cloud Computing*

Berdasarkan jenis layanan *cloud computing* dibagi menjadi 3 model layanan yaitu:

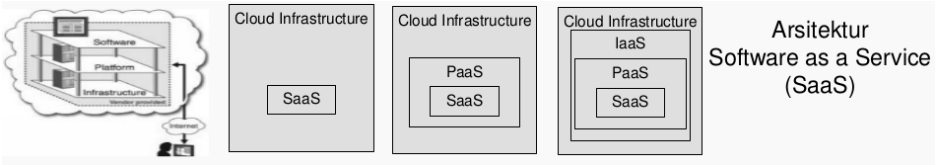


Gambar 2.2. Model *cloud computing*

(Sumber Gambar: Microsoft *Cloud Computing*)

2.1.4.1. Software As A Service

Software As A Service (SaaS) Menyediaan layanan berupa aplikasi yang dapat digunakan oleh konsumen yang berjalan pada infrastruktur *cloud.* Contoh penyedia layanan SaaS adalah gmail, google docs, office 365 dan SalesForce.



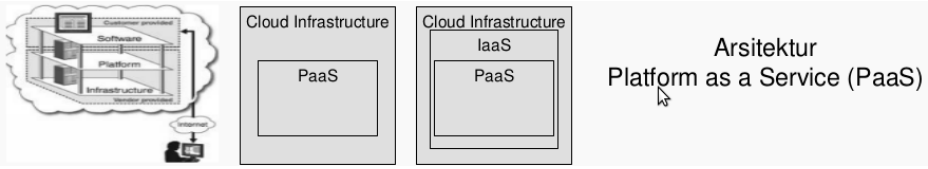
Gambar 2.3. Arsitektur model layanan SaaS

(Sumber Gambar: Slideshare.com)

2.1.4.2. Platform As A Service

Platform As A Service (PaaS) Menyediakan *platform* (Bahasa pemrograman, *Tools*, Web server, database) yang berguna untuk pengembangan aplikasi yang berjalan pada infrasturktur cloud dan hasilnya dapat dimanfaatkan untuk konsumen.

Contoh penyedia layanan PaaS, OpenShift, PHPCloud, AppFog, Heroku dan GoogleApp Engine.

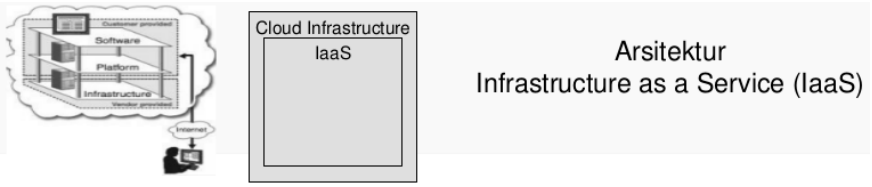


Gambar 2.4. Arsitektur model layanan PaaS

(Sumber Gambar: Slideshare.com)

2.1.4.3. Infrastruktur As A Service

Infrastruktur As A Service (IaaS) Menyediakan sumber daya pemroses, *storage*, kapasitas jaringan, dan sumber daya komputasi lainnya. Contoh penyedia layanan Amazon EC2 dan TelkomCloud.



Gambar 2.5. Arsitektur model layanan IAAS
(Sumber Gambar: Slideshare.com)

2.2. Pemodelan UML

2.2.1. Pengertian UML

UML (*Unified Modelling Language*) adalah keluarga notasi grafis yang didukung oleh meta model tunggal, yang membantu pendeskripsian dan desain sistem

perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OOP). Berikut klasifikasi model diagram pada UML:

2.2.2. Use Case Diagram

Menurut Munawar (2005 : 63) Use case diagram adalah deskripsi fungsi dari sebuah sistem dari perpektif pengguna (user). Menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem.

2.2.3. Activity Diagram

Menurut Munawar (2005:109) Activity diagram adalah teknik untuk mendeskripsikan logika procedural, proses bisnis dan aliran kerja dalam banyak kasus.

Menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Activity diagram merupakan state diagram khusus, dimana sebagian besar state adalah action dan sebagian besar transisi ditrigger oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity diagram tidak menggambarkan behaviour internal sebuah sistem (dan

interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

2.2.4. Sequence Diagram

Menurut Munawar (2005:87) Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Menunjukan sejumlah contoh obyek dan pesan yang diletakan diantara obyek-obyek ini didalam use case.

Memberikan gambaran detail dari setiap use case diagram yang telah dibuat sebelumnya. Setiap objek yang terlibat dalam sebuah use case digambarkan dengan garis putus-putus vertikal, kemudian message yang dikirim oleh objek digambarkan dengan garis horizontal secara kronologis dari atas ke bawah.

2.2.5. Class Diagram

Menurut Martin Fowler (2003:35) Class diagram menggambarkan jenis objek dalam sistem dan berbagai macam hubungan statis yang ada diatara mereka. Diagram kelas juga memiliki atribut dan operasi dari sebuah kelas.

Sebuah objek memiliki keadaan sesaat (*state*) dan perilaku (*behavior*). State sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam atribut. Sedangkan perilaku suatu objek mendefinisikan bagaimana sebuah objek bertindak dan memberikan reaksi. Kelas memiliki tiga area pokok : Nama (*stereotype*), Atribut dan Metode atribut dan metode dapat memiliki salah satu sifat berikut:

1. *Private*, tidak dapat dipanggil dari luar class yang bersangkutan

2. *Protected*, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya

3. *Public*, dapat dipanggil diluar kelas maupun didalam kelas.

2.2.6. Object Diagram

Menurut Munawar (2005:135) *Object diagram* adalah gambaran obyek-obyek secara ringkas disebuah sistem pada suatu waktu. Obyek diagram digunakan untuk menunjukan contoh konfigurasi dari obyek-obyek.

2.2.7. Component Diagram

Menurut Munawar (2005:199) *Component diagram* adalah bagian fisik dari sebuah sistem. Component diagram merepresentasikan dunia riil item yaitu komponen perangkat lunak. *Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*).

2.2.8. Deployment Diagram

Deployment diagram menurut Munawar (2005:125), menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian software yang berjalan pada bagian-bagian hardware. Deployment Diagram menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.

2.3. Perangkat Lunak Pendukung

Perangkat lunak pendukung merupakan perangkat lunak yang digunakan untuk membuat dan merancang aplikasi.

2.3.1. PHP

PHP (PHP *Hypertext Preprocessor*) merupakan tipe bahasa pemrograman *scripting language*, Yang dirancang untuk mengembangkan aplikasi web dinamis dan kode dapat ditanam HTML. PHP merupakan pengembang pertama yang mengembangkan tipe *scripting language* dan PHP dapat berjalan pada sisi *web server*.

2.3.1.1. Sejarah dan Perkembangan PHP

Pembangunan PHP dimulai pada tahun 1994 oleh *programmer* Rasmus Lerdorf awalnya menciptakan satu set script Perl yang ia sebut "*Personal Home Page Tools*" untuk mempertahankan *homepage* pribadinya. Scriptnya dapat melakukan tugas-tugas seperti menampilkan resume dan memonitor lalu lintas halaman web. Lerdorf awalnya mengumumkan peluncuran PHP pada forum diskusi pada tanggal 8 Juni, 1995. Lalu beliau menulis ulang script ini sebagai *Common Gateway Interface* (CGI) di binari C, Dengan memperluas untuk penambahan kemampuan untuk dapat bekerja pada *web* dan untuk berkomunikasi dengan basis data dan implementasi ini disebut "*Personal Home Page*/*Forms Interp*reter" atau disingkat PHP/FI. PHP /FI dapat digunakan untuk membangun aplikasi sederhana dan aplikasi *web* dinamis. Lerdorf mengeluarkan PHP/FI sebagai *Personal Home Page Tools* (PHP Tools) versi

1.0 di publikasikan pada tanggal 8 Juni 1995, untuk mempercepat lokasi bug dan memperbaiki kode.

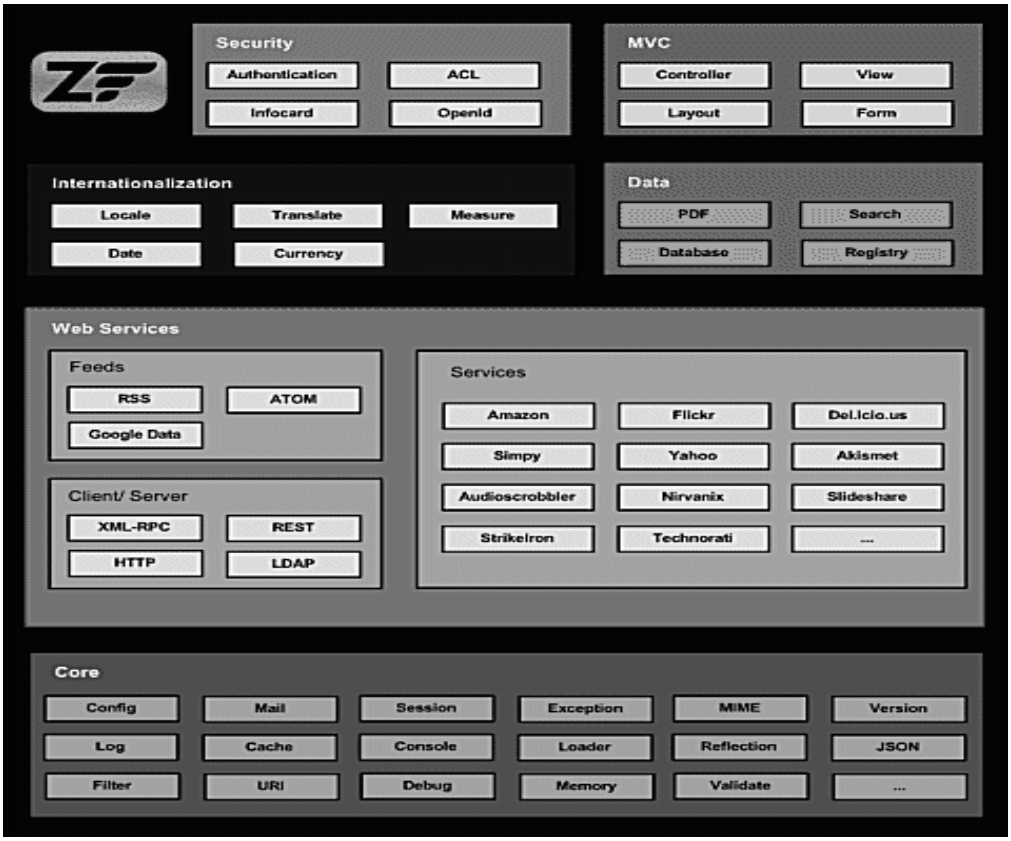
Zeev Suraski dan Andi Gutmans adalah dua pengembang dari Technion IIT yang menulis ulang parser pada tahun 1997 dengan membentuk dasar PHP 3. Hingga pada tahun 1999 terbentuk Zend Engine dan mendirikan Zend Technology sebagai pusat pengembangan PHP.

Pada tanggal 13 Juli 2004, PHP5 dirilis, Yang didukung oleh Zend Engine II. PHP 5 termasuk fitur baru seperti peningkatan untuk pemrograman berorientasi objek, ekstensi PHP Data Objects (PDO) dimana yang mendefinisikan antarmuka konsisten untuk mengakses database), dan berbagai peningkatan kinerja lainnya pada tahun 2008. PHP5 menjadi versi stabil dalam pengembangan. Sejak penelitian ini ditulis perkembangan versi PHP berada pada versi 5.3.14 dan 5.4.6 dengan status *Release supported*.

2.3.2. Zend Framework

Zend Framework versi terkhir pada versi 1.x adalah versi Zend Framework 1.12.0 yang merupakan *framework* PHP yang paling popular dikalangan pengembang PHP dan mempunyai komunitas yang aktif. Dimana *framework* ini didalamnya berisi class-class atau komponen yang dapat digunakan oleh pengembang untuk mempermudah pengembangan aplikasi web 2.0. *Framework* ini berbasiskan komponen yang dapat digunakan sesuai dengan kebutuhan masing-masing komponen atau *full stack framework*, Dimana pengembang dapat memanfaatkan semua

komponennya yaitu dengan memanfaatkan pengembangan berpola MVC (*Model View Controller*), dimana memisahkan antara logika aplikasi dan tampilan web.



Gambar 2.6. Zend Framework Komponen.

(Sumber Gambar: CodeDiesel)

a. Model

Kumpulan logika dalam bentuk class yang bias digunakan berulang, seperti *access data* dan *business rules* lainnya yang ditempatkan pada satu lokasi untuk mempermudah pengelolaan logika.

b. View

View bertanggung jawab hanya untuk menampilkan informasi kepada user. Informasi yang ditampilkan misalkan logo, form, gambar atau elemen HTML lainnya yang bias ditampilkan oleh *web browser*. View tidak mengikutkan logika pemrograman. Desain tampilan tidak akan berubah dikarenakan kesalahan logika pemrograman. Sehingga dapat mempermudah seorang desainer web untuk fokus pada desain web dan tidak berurusan dengan kode pemrograman yang kompleks.

c. Controller

Controller bertanggung jawab untuk merespons semua event, baik event yang dilakukan user, maupun event yang dijalankan oleh system. Ketika user menjalankan event controller akan mencari aksi yang sesuai untuk merespon event yang dilakukan oleh user tersebut. Controller dapat diartikan sebagai *application flow* untuk mencari respon atas *request* yang dilakukan user atau sistem dan berfungsi sebagai pengubung antara model dengan view.

2.3.3. Zend Server

Zend Server merupakan fungsionalitas aplikasi server untuk aplikasi berbasis PHP. Zend Server mengacu pada *application server* produk PHP yang ditawarkan oleh Zend Technologies, dirilis pada awal 2009 dengan dukungan produksi yang tersedia untuk Windows dan Linux. Tersedia dalam dua versi, Zend Server dan Zend Server *Community Edition*. Zend Server didukung pada mesin yang kompatibel x86 dan x86-64 dengan dukungan iSeries IBM di Zend Server 5.

Zend server mempermudah manajemen pengaturan aplikasi dengan menggunakan GUI (*Graphical User Interface*) seperti pengaturan php.ini dan sebagai alat pembantu tambahan, memanajemen aplikasi, mengaktifkan atau menonaktifkan ekstensi atau melihat file log. Serta adanya intergrasi web server seperti Apache atau IIS (*Internet Infotmasion Service*) pada windows.

2.3.4. Apache

Server HTTP Apache adalah web server Yang dapat dijalankan di banyak sistem operasi (Unix, BSD, Linux, Microsoft dan Windows) Yang berguna untuk melayani Dan memfungsikan situs web. Protokol Yang digunakan untuk melayani fasilitas *web* atau www inisial menggunakan HTTP. Apache memiliki fitur-fitur seperti pesan kesalahan canggih yang dapat dikonfigurasi, Autentikasi berbasis basis pengiriman berkas. Apache juga didukung oleh sejumlah antarmuka pengguna berbasis Grafik (GUI) yang memungkinkan penanganan server menjadi mudah. Apache merupakan perangkat lunak sumber terbuka dikembangkan oleh komunitas terbuka yang terdiri bahasa Dari pengembang-pengembang dibawah naungan Apache Software Foundation.

2.3.5. Basis Data MySQL

Basis data (*Database*) adalah kumpulan data yang terintegrasi satu sama lain. Setiap user akan diberi wewenang untuk dapat mengakses data di dalam database. Database biasanya terorganisasi dalam beberapa komponen yang terdiri dari satu atau lebih table. Table digunakan untuk menyimpan data yang terdiri dari baris dan kolom. Akses terhadap data dapat berupa menampilkan, memodifikasi, dan menambah atau

menghapus data yang telah tersimpan. Akses tersebut di atas dapat dilakukan oleh MySQL yaitu salah satu database yang memiliki koneksitas yang baik terhadap PHP.

MySQL Menurut situs resminya *mysql.com* adalah salah satu jenis database *server* yang sangat terkenal. MySQL menjadi sangat populer karena MySQL bersifat *free* (tidak perlu membayar dalam menggunakannya) pada berbagai jenis platform (unix/windows). Untuk mendapatkan MySQL dapat didownload dari *mysql.org* atau *mysql.com*. MySQL termasuk jenis RDBMS (*Relational Database Management System*). Pada MySQL, sebuah database mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom. Pada saat instalasi, secara *default* MySQL akan membuat sebuah database bernama mysql. Salah satu isi dari database ini adalah tabel user, tabel ini berisi nama dan password user yang dapat mengakses data pada database yang dibuat di mysql. Database ini juga berisi hak-hak yang diberikan pada setiap user. MySQL menerima berbagai macam tipe data, tipe-tipe data ini dibagi menjadi 3, yaitu tipe data untuk bilangan, tipe data untuk tanggal dan jam, dan tipe data untuk karakter.

2.3.6. JavaScript

JavaScript adalah bahasa skrip yang populer di internet dan dapat bekerja di sebagian besar penjelajah web populer seperti Internet Explorer (IE), Mozilla Firefox, Netscape dan Opera. Kode Javascript dapat disisipkan dalam halaman *web* menggunakan *tag script*.

2.3.7. Cascading Style Sheet

Cascading Style Sheet (CSS) merupakan salah satu bahasa pemrograman web untuk mengendalikan beberapa komponen dalam sebuah web sehingga akan lebih terstruktur dan seragam. Sama halnya *styles* dalam aplikasi pengolahan kata seperti Microsoft Word yang dapat mengatur beberapa *style*, misalnya *heading*, *subbab*, *bodytext*, *footer*, *images*, dan *style* lainnya untuk dapat digunakan bersama-sama dalam beberapa berkas. Pada umumnya CSS dipakai untuk memformat tampilan halaman web yang dibuat dengan bahasa HTML dan XHTML.

CSS dapat mengendalikan ukuran gambar, warna bagian tubuh pada teks, warna tabel, ukuran *border*, warna border, warna *hyperlink*, warna *mouse over*, spasi antar paragrap, spasi antar teks, *margin* kiri, kanan, atas, bawah, dan parameter lainnya. CSS adalah bahasa *style sheet* yang digunakan untuk mengatur tampilan dokumen.

2.3.8. NetBeans IDE

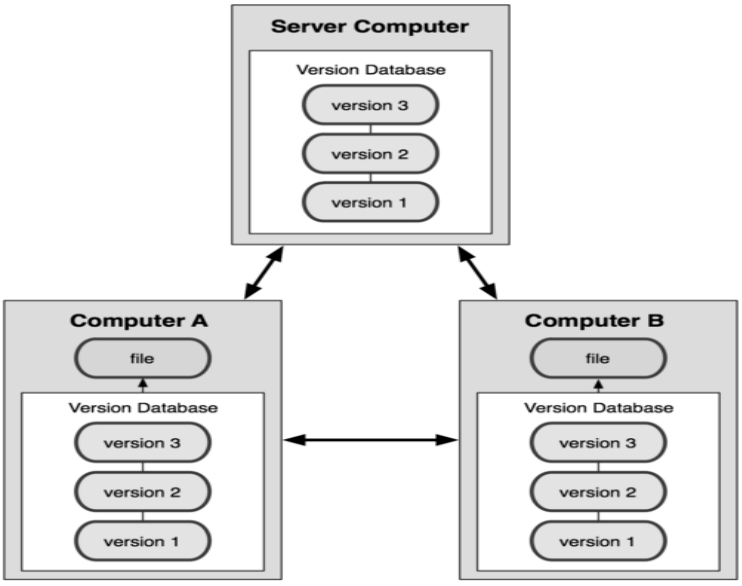
Menurut Miftakhul Huda (2011:423) NetBeans memiliki *Integrated Development Environment* (IDE). Ada juga yang menyebutkan *Integrated Design Environment* dan *Integrated Debbuging Environment*, yaitu sebuah program atau alat bantu yang terdiri atas editor, compiler, debugger dan design yang terintegrasi dala satu aplikasi. Berbasiskan Java dari Sun Microsystems yang berjalan di atas Swing. Swing sebuah teknologi Java untuk pengembangan aplikasi Desktop yang dapat bejalan di berbagai macam platforms seperti Windows, Linux, Mac OS X and Solaris. Netbeans support untuk berbagai bahasa pemrograman seperti Java,

Javascript dan PHP. Support untuk revision control seperti CVS, Git, Subversion dan Mercurial.

2.3.9. Version Control System Git

Version control adalah sebuah sistem yang mencatat setiap perubahan terhadap sebuah berkas atau kumpulan berkas sehingga pada suatu saat anda dapat kembali kepada salah satu versi dari berkas tersebut. Sebagai contoh menggunakan kode sumber perangkat lunak sebagai berkas yang akan dilakukan *version controlling*. Sistem ini memungkinkan untuk mengembalikan berkas pada kondisi atau keadaan sebelumnya, mengembalikan seluruh proyek pada keadaan sebelumnya, membandingkan perubahan setiap saat, melihat siapa yang terakhir melakukan perubahan terbaru pada suatu objek sehingga berpotensi menimbulkan masalah, siapa yang menerbitkan isu, dan lainnya. Dengan menggunakan Version Control System (VCS) dapat berarti jika telah mengacaukan atau kehilangan berkas, dapat dengan mudah mengembalikannya.

Git merupakan *Version Control System* (VCS)Terdistribusi atau *Distributed Version Control Systems* (DVCS) untuk mengambil tempat. dalam sebuah DVCS (seperti Git, Mercurial, Bazaar atau Darcs), klien tidak hanya melakukan *checkout* untuk *snapshot* terakhir setiap berkas, namun mereka (klien) memiliki salinan penuh dari repositori tersebut. Jadi, jika server mati, dan sistem berkolaborasi melalui server tersebut, maka klien manapun dapat mengirimkan salinan repositori tersebut kembali ke server. Setiap *checkout* pada DVCS merupakan sebuah *backup* dari keseluruhan data.



Gambar 2.7. Diagram *distributed version control*

(Sumber Gambar: git-scm.com)

Lebih jauh lagi, kebanyakan sistem seperti ini mampu menangani sejumlah *remote repository* dengan baik, jadi dapat melakukan kolaborasi dengan berbagai kelompok kolaborator dalam berbagai cara secara bersama-sama pada suatu proyek. Hal ini memungkinkan untuk menyusun beberapa jenis alur kerja yang tidak mungkin dilakukan pada sistem terpusat, seperti *hierarchical* model.