BAB 2
LANDASAN TEORI

2.1. *Cloud Computing*

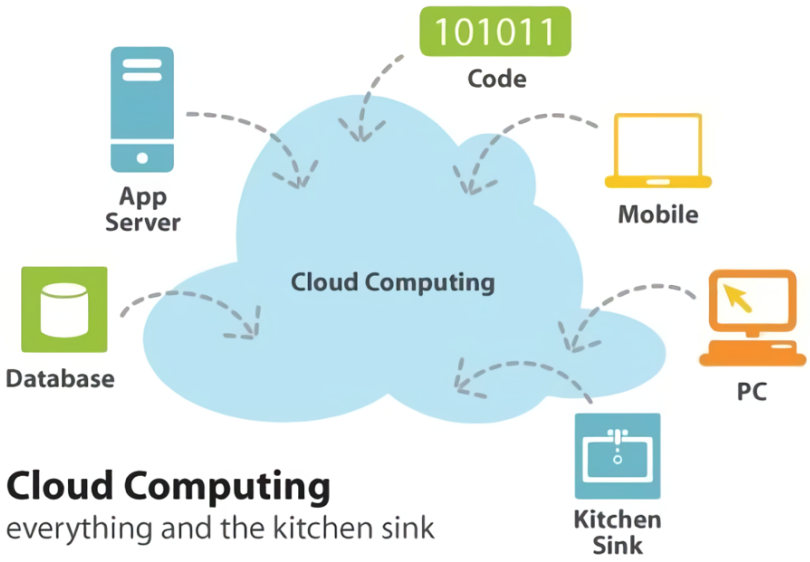
Cloud computing merupakan sebuah model yang memungkinkan adanya penggunaan sumber daya (*resource*) secara bersama-sama dan mudah, menyediakan jaringan akses dimana-mana, dapat dikonfigurasi dan layanan yang digunakan sesuai keperluan (*on-demand*) [8, 9].

Cloud computing merupakan metafora dari jaringan komputer / internet, dimana *cloud* (awan) merupakan penggambaran dari jaringan komputer / internet yang diabstraksi dari infrastruktur kompleks yang disembunyikan. Pada *cloud computing* sumber daya seperti *processor/computing power*, *storage*, *network*, *sofware* menjadi abstrak (*virtual*) dan diberikan sebagai layanan di jaringan / internet [10].

Cloud computing adalah evolusi selanjutnya dari internet "awan" pada *cloud computing* merupakan penyedia hal-hal yang berkaitan dari tenaga komputasi hingga infrastruktur komputasi, aplikasi-aplikasi, proses bisnis hingga kolaborasi yang muncul sebagai layanan yang dapat diakses pada saaat dibutuhkan kapanpun dan dimanapun [11].

Cloud computing adalah sebuah model *client-server,* di mana *resource* seperti *server, storage, network,* dan *software* dapat dipandang sebagai layanan yang dapat diakses oleh pengguna secara *remote* dan setiap saat. Pengguna dapat menikmati berbagai layanan yang disediakan oleh *provider cloud computing*, tanpa perlu terlalu banyak meminta bantuan teknis atau *support* dari pihak *provider* [12].

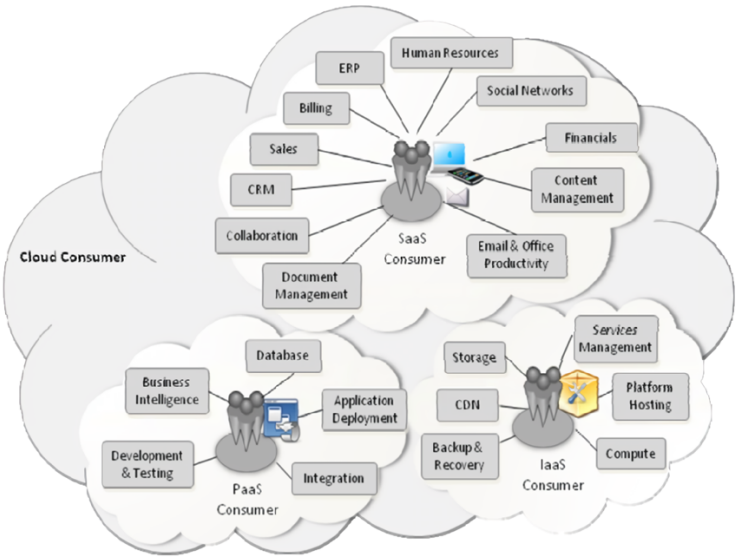
Dapat disimpulkan bahwa *cloud computing* mengacu pada layanan komputasi *on-demand* yang tersedia pada sebuah jaringan komputer / internet. Layanan *cloud computing* dapat di akses oleh beberapa pengguna secara bersamaan dimana semua informasi disimpan secara terpusat pada suatu *server. Cloud computing* memiliki manfaat yang baik untuk pengguna termasuk skalabilitas, jangkauan, dan pengelolaan.



Gambar 2.1. Ilustrasi *Cloud Computing*

2.1.1. Model Layanan *Cloud Computing*

Layanan pada *cloud computing* terbagi menjadi tiga model layanan, hal ini disesuaikan dengan kebutuhan dan keperluan dari pengguna. Ketiga model layanan tersebut yang dijelaskan oleh NIST (*National Institute Standards Technology*) yaitu *Software as a Service* (SaaS), *Platform as a Service* (PaaS), dan *Infrastructure as a Service* (IaaS) [8].



Gambar 2.2. 3 Model Layanan *Cloud Computing*

2.1.1.1. *Software as a Service* (SaaS)

Software as a Service (SaaS) merupakan jenis layanan yang diberikan oleh teknologi *Cloud Computing* kepada para penggunanya dalam bentuk pemakaian bersama perangkat lunak (aplikasi) yang umumnya disediakan dalam bentuk tatap muka web. SaaS merupakan jenis layanan *Cloud Computing* yang paling banyak digunakan oleh para pengguna komputer, khususnya pengguna akhir yang tidak terlalu membutuhkan pengetahuan teknis dalam instalasi dan konfigurasi. Cukup dengan sebuah komputer/perangkat mobile, sistem operasi, aplikasi *web browser*, dan koneksi internet atau intranet seorang pengguna komputer dapat dengan mudah menggunakan layanan *Cloud Computing* dengan model SaaS ini [9].

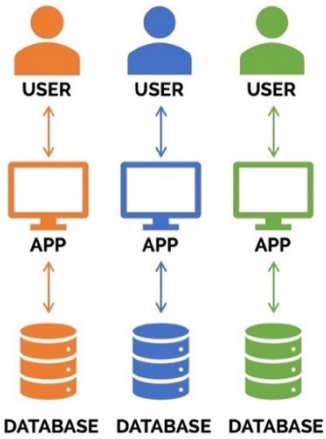
Kemampuan yang diberikan kepada konsumen untuk mengunakan aplikasi penyedia dapat beroperasi pada infrastruktur awan. Aplikasi dapat diakses dari berbagai perangkat klien melalui antarmuka seperti *web browser* (misalnya, *email* berbasis *web*). Konsumen tidak mengelola atau mengendalikan infrastruktur awan yang mendasari termasuk jaringan *server*, sistem operasi penyimpanan atau bahkan kemampuan aplikasi individu, dengan kemungkinan pengecualian terbatas terhadap pengaturan konfigurasi aplikasi pengguna tertentu [1, 8].

Dapat disimpulkan bahwa SaaS merupakan sebuah layanan teknologi *cloud computing* yang paling dekat dengan pengguna akhir. Layanan ini berupa perangkat lunak (aplikasi) yang dapat dijalankan secara bersama-sama oleh pengguna. Sehingga pengguna tidak lagi diharuskan untuk menginstall aplikasi dan membuat aplikasi, pengguna hanya dapat menggunakan ataupun menyewa perangkat lunak dari penyedia layanan SaaS. SaaS merupakan layanan yang bertujuan untuk memudahkan aktifitas komputasi pengguna.

SaaS memiliki tiga pendekatan model aplikasi dan pendekatan model *database* yang dapat digunakan sebagai pelayanan *service*-nya, yaitu model *separate application and separate database,* model *shared application and separate database,* dan model *shared application and shared database* [12].

2.1.1.1.1 Model *Separate Application and Separate Database*

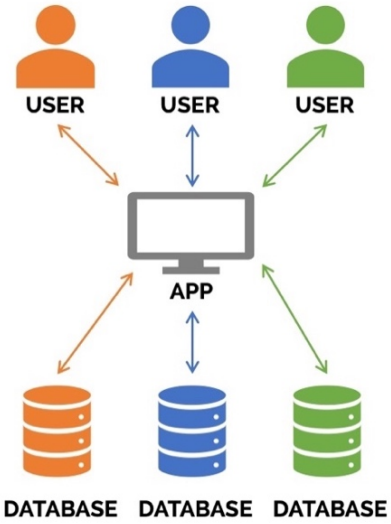
Model ini memisahkan aplikasi dan *database* yang digunakan oleh setiap pengguna. Dengan kata lain data terkait dari setiap pengguna terpisah dari pengguna lainnya. Model ini biasa digunakan oleh pengguna yang memiliki data-data yang bersifat khusus. Kelemahan dari model ini dalam segi biaya yang relatif mahal dalam perawatan aplikasi dan *database.* Ilustrasi model *separate application and separate database* dapat dilihat pada Gambar 2.3.



Gambar 2.3. Model *Separate Application and Separate Database*

2.1.1.1.2 Model *Shared Application and Separate Database*

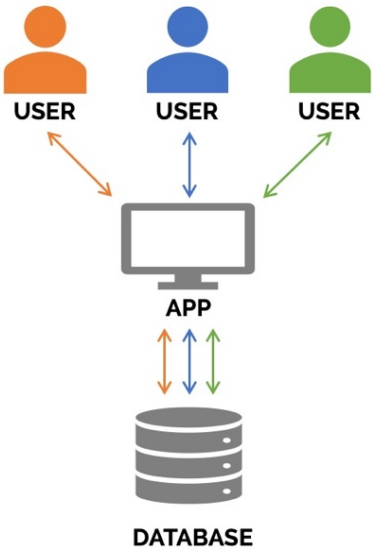
Model ini hanya memisahkan *database* yang digunakan oleh setiap pengguna. Dengan kata lain aplikasi yang digunakan oleh pengguna merupakan aplikasi yang sama, namun data terkait dari setiap pengguna terpisah dari pengguna lainnya. Kelemahan model relatif cukup sulit dalam perawatan aplikasi dan *database.* Ilustrasi model *shared application and separate database* dapat dilihat pada Gambar 2.4.



Gambar 2.4. Model *Shared Application and Separate Database*

2.1.1.1.3 Model *Shared Application and Shared Database*

Model ini berbeda dengan dua model lainnya. Pada model ini aplikasi dan *database* yang digunakan oleh pengguna adalah sama. Dengan kata lain pengguna menyimpan semua data pada sebuah aplikasi dan *database* yang sama. Model ini merupakan model yang memiliki biaya relatif murah dan merupakan model yang mudah dalam perawatan aplikasi dan *database*. Ilustrasi model *shared application and shared database* dapat dilihat pada Gambar 2.5.



Gambar 2.5. Model *Shared Application and Shared Database*

2.1.1.2. *Platform as a Service* (PaaS)

Platform as a Service (PaaS) merupakan jenis layanan pada *Cloud Computing* yang menekankan kepada penyediaan platform untuk membantu proses pengembangan perangkat lunak secara cepat dan mudah. Layanan platform yang disediakan oleh PaaS umumnya juga berbasis web, di mana didalamnya telah tersedia banyak fitur yang memudahkan programmer dan pengguna awam dalam mengembangkan aplikasi tanpa memerlukan banyak proses penulisan sumber kode (*coding*) [9].

Kemampuan yang diberikan kepada konsumen untuk menyebarkan aplikasi yang dibuat konsumen atau diperoleh ke infrasturktur komputasi awan menggunakan bahasa pemrograman dan peralatan yang didukung oleh provider. Konsumen tidak mengelola atau mengendalikan infrastruktur awan yang mendasari termasuk jaringan, *server*, sistem operasi, atau penyimpanan namun memiliki kontrol atas aplikasi disebarkan dan memungkinkan aplikasi melakukan hosting konfigurasi [1, 8].

Dapat disimpulkan bahwa PaaS merupakan sebuah platform yang dapat digunakan oleh pengguna dalam pengembangan aplikasi yang belum tersedia pada layanan SaaS. PaaS memungkinkan pengguna dapat membangun sebuah aplikasi baru yang dapat membantu dalam aktifitas komputasinya.

2.1.1.3. *Infrastructure as a Service* (IaaS)

Infrastructure as a Service (IaaS) merupakan jenis layanan pada *Cloud Computing* yang menekankan kepada layanan penyediaan sarana jaringan komputer (*computer network*), perangkat keras jaringan, komputer *server*, media penyimpanan (*storage*), processor, berserta dengan proses virtualisasi yang menunjang proses komputasi [9].

Kemampuan yang diberikan kepada konsumen untuk memproses, menyimpanan, berjaringan, dan komputasi sumber daya lain yang penting dimana konsumen dapat menyebarkan dan menjalankan perangkat lunak secara bebas dapat mencakup sistem operasi dan aplikasi. Konsumen tidak mengelola atau mengendalikan infrastruktur awan yang mendasari tetapi memiliki kontrol atas

sistem operasi penyimpanan, aplikasi yang disebarkan dan mungkin kontrol terbatas komponen jaringan yang dipilih [1, 8].

Dapat disimpulkan bahwa IaaS merupakan model layanan *cloud computing* yang paling dasar. Dimana pengguna yang memiliki kontrol penuh atas teknis sistem untuk menjalankan *website* yang akan dibangunnya. Pengguna sendiri yang melakukan pengkonfigurasian *server* untuk kebutuhan *website*-nya seperti instalasi *web server, database server*, bahasa pemograman, *email server* dan kebutuhan lainnya.

2.1.2. Karakteristik *Cloud Computing*

Ada lima karakteristik penting yang dimiliki *cloud computing* menurut NIST yakni *On-Demand Self-Service, Broad Network Access, Resource Pooling, Rapid Elasticity,* dan *Measured Service* [8].

2.1.2.1. *On-Demand Self-Service*

Pengguna mendapatkan kontrol penuh terhadap pemesanan ataupun pengelolaan layanan tanpa harus melalui interaksi manusia dengan penyedia layanan. Semua layanan dilakukan secara otomatis pada penyedia melalui sebuah aplikasi *website* [8].

2.1.2.2. *Broad Network Access*

Kemampuan yang tersedia melalui jaringan dan diakses melalui mekanisme standar yang mengenalkan penggunaan berbagai platform teknologi (misalnya, telepon seluler, laptop dan PDA) [8].

2.1.2.3. *Resource Pooling*

Cloud computing provider dapat melayani pengguna via *multi-tenant model*. Berbagai *resources*, seperti : *storage*, CPU, *memory*, *bandwidth*, dan mesin virtual, yang terdapat di berbagai lokasi dapat digunakan oleh banyak pengguna dalam waktu yang bersamaan [12].

2.1.2.4. *Rapid Elasticity*

Kemampuan yang dapat dengan cepat dan elastis ditetapkan, dalam beberapa kasus akan secara otomatis, dengan cepat dikeluarkan dan dengan cepat

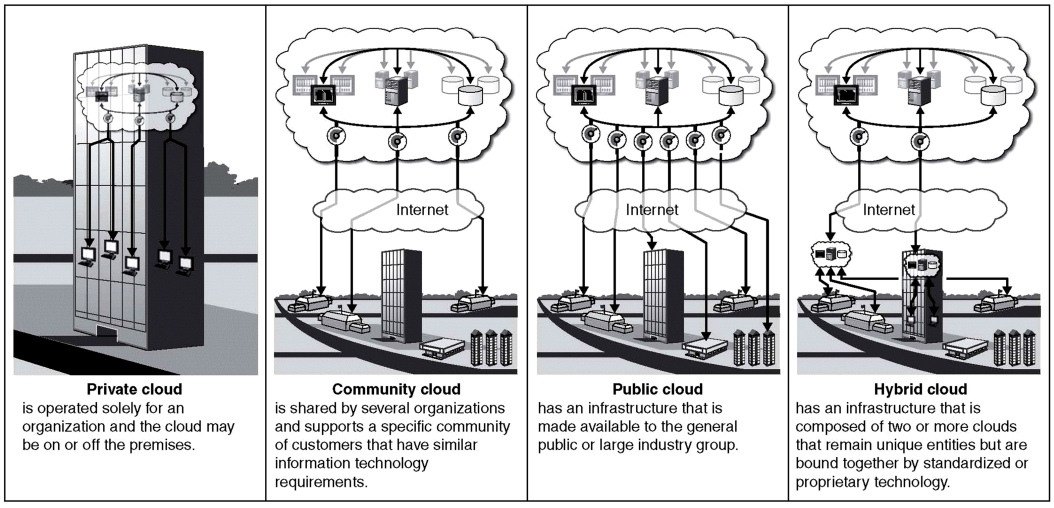
dilepaskan untuk meningkatkan kecepatan masuk. Untuk konsumen, kapabilitas tersedia bagi pengadaan sering muncul menjadi tidak terbatas dan dapat dibeli dalam jumlah berapapun setiap saat [8].

2.1.2.5. *Measured Service*

Services yang disediakan bersifat terukur. *Provider cloud computing* dapat mengendalikan dan memonitor *cloud services*, misalkan untuk keperluan *billing, access control, resource optimization, capacity planning*, dan sebagainya [12].

2.1.3. Model Pengembangan *Cloud Computing*

Dalam membantu menyesuaikan lingkungan, kondisi, dan keperluan dari pengguna agar dapat diterapkan dan dimanfaatkan dengan baik dan optimal NIST menjelaskan jika ada empat model pengembangan *cloud computing* yaitu *Private Cloud*, *Community Cloud, Public Cloud,* dan *Hybrid Cloud* [8].



Gambar 2.6. Empat Model Pengembangan *Cloud Computing* [14]

2.1.3.1. *Private Cloud*

Private Cloud merupakan model pengembangan yang diperoasikan pada suatu perusahaan yang dapat dikelola sendiri ataupun dengan adanya bantuan dari pihak ketiga [11]. *Private Cloud* memiliki tiga tujuan utama yaitu sebagai berikut[9]:

1. Hemat Biaya

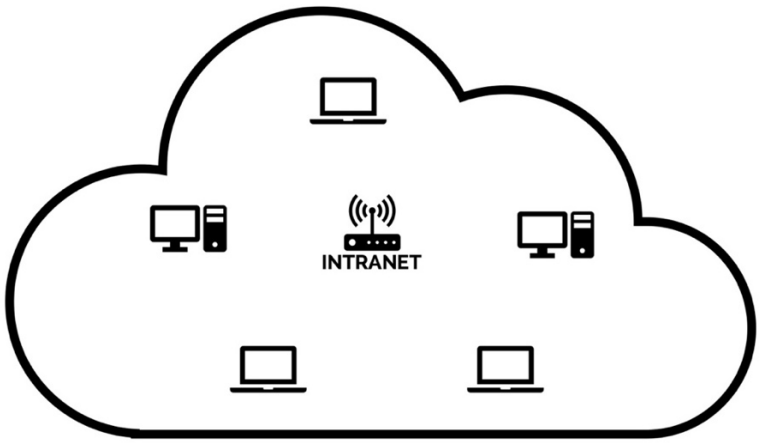
   Hanya dibutuhkan sarana sebuah komputer atau lebih yang memiliki fasilitas jaringan intranet (LAN/*Local Area Network*), sehingga tidak perlu lagi mengeluarkan biaya lebih untuk penyediaan akses internet.

2. Privasi

   Karena menggunakan jaringan local yang ada di dalam lingkungan gedung/kantor maka penggunaan relatif lebih aman, karena akses data, informasi, dan *file – file* penting hanya oleh internal perusahaan yang bersangkutan saja.

3. Latar belakang pengguna

   Umumnya pengguna merupakan anggota dari internal perusahaan. Sehingga akan lebih mudah untuk menyediakan sebuah lingkungan *deployment Private Cloud* untuk pengguna secara terbatas (*private*) pada lingkungan dan pengguna dari perusahaan saja.

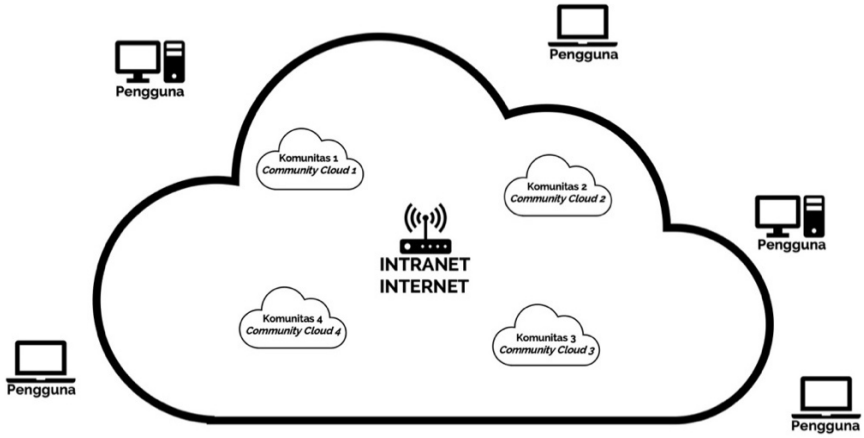


Gambar 2.7. Bagan *Private Cloud*

2.1.3.2. *Community Cloud*

Community Cloud merupakan model pengembangan yang dibangun untuk organisasi – organisasi yang bertujuan dalam mendukung komunitas tertentu yang memiliki tujuan, visi, dan misi yang sama [11]. *Community Cloud* memiliki beberapa tujuan yatiru sebagai berikut [9] :

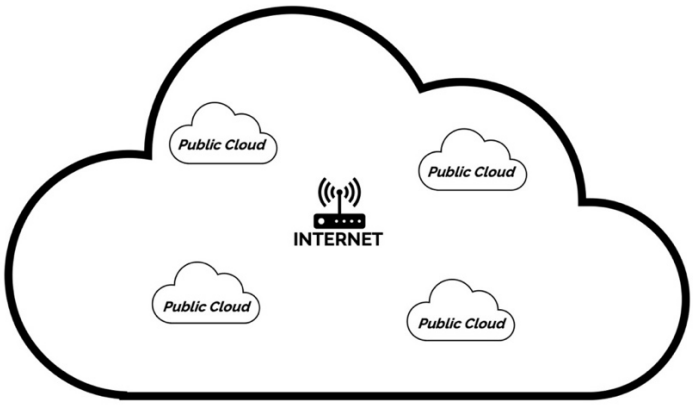
1. Untuk memudahkan komunitas di dalam berbagi data antar anggota.
2. Menyatukan komunitas yang memiliki kepentingan yang sama ke dalam bentuk layanan *Cloud Computing*.
3. Sebagai upaya dari komunitas untuk bersama-sama menyediakan layanan *Cloud* baik untuk komunitas itu sendiri maupun public di luar komunitas (masyarakat umum).



Gambar 2.8. Bagan *Community Cloud*

2.1.3.3. *Public Cloud*

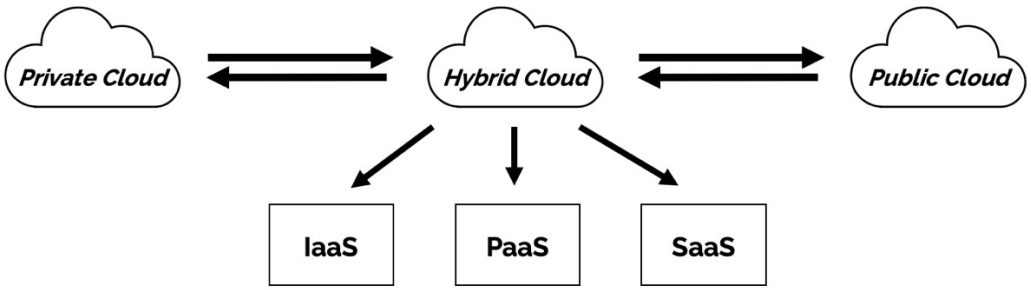
Public Cloud merupakan model pengembangan yang disediakan untuk umum atau untuk industri skala besar, dimiliki oleh perusahaan yang menjual layanan *cloud.* Sehingga layanan, data, dan informasi di dalamnya dapat digunakan dan dibagikan dengan mudah ke seluruh pengguna [11].



Gambar 2.9. Bagan *Public Cloud*

2.1.3.4. *Hybrid Cloud*

Hybrid Cloud merupakan model pengembangan yang menggabungkan dua atau lebih awan (*private, community,* dan *public*) yang masih memiliki entitas unik namun terikat bersama-sama oleh standar teknologi atau kepemilikan data dan portabilitas aplikasi [11]. Tujuan dari *Hybrid Cloud* adalah untuk memudahkan di dalam manajemen keamanan dan manajemen data. Oleh karena itu, saat ini hingga ke depan nanti model pengembangan *Hybrid Cloud* inilah yang akan banyak dipilih dan digunakan [9].



Gambar 2.10. Bagan *Hybrid Cloud*

2.1.4. Komponen Pada *Cloud Computing*

Cloud Computing memiliki enam komponen mencakup *Cloud Clients, Cloud Services, Cloud Applications, Cloud Platform, Cloud Storage,* dan *Cloud Infrastructure* [11].

2.1.4.1. *Cloud Clients*

Cloud Service adalah seperangkat komputer ataupun *software* yang didesain secara khusus untuk penggunaan layanan berbasis *cloud computing*. Contohnya :

1. *Mobile* : Android, IoS, Windows Mobile dan lain – lain
2. *Thin Client* : Windows Terminal Service, CherryPal dan lain – lain
3. *Thick Client* : Internet Explorer, FireFox, Chrome, dan lain - lain

2.1.4.2. *Cloud Services*

Cloud Applications adalah produk, layanan dan solusi yang dipakai dan disampaikan secara real-time melalui media Internet. Contoh yang paling popular adalah web service.

1. Identitas : OpenID, OAuth, dan lain - lain.
2. *Integration* : Amazon Simple Queue Service.
3. *Payments* : PayPal, Google Checkout.
4. *Mapping* : Google Maps, Yahoo! Maps.

2.1.4.3. *Cloud Applications*

Cloud Applications memanfaatkan cloud computing dalam hal arsitektur software. Sehingga user tidak perlu menginstal dan menjalankan aplikasi dengan menggunakan komputer. Contohnya :

1. *Peer-to-peer* : BitTorrent, SETI, dan lain-lain.
2. *Web Application* : Facebook, Instagram dan lain - lain.
3. SaaS : Google Apps, SalesForce.Com, dan lain-lain.

2.1.4.4. *Cloud Platform*

Cloud Platform merupakan layanan berupa platform komputasi yang berisi hardware dan software2 infrasktruktur. Biasanya mempunyai aplikasi bisnis tertentu dan menggunakan layanan PaaS sebagai infrastruktur aplikasi bisnisnya. Contohnya :

1. *Web Application Frameworks* : Python Django, Ruby on Rails, .NET
2. *Web Hosting*
3. *Propietary* : Force.Com

2.1.4.5. *Cloud Storage*

Cloud Storage melibatkan proses penyampaian penyimpanan data sebagai sebuah layanan. Contohnya :

1. *Database* : Google Big Table, Amazon SimpleDB.
2. *Network Attached Storage* : Nirvanix CloudNAS, MobileMe iDisk.

2.1.4.6. *Cloud Infrastructure*

Cloud Infrastructure merupakan penyampaian infrastruktur komputasi sebagai sebuah layanan. Contohnya :

1. *Grid Computing* : Sun Grid.
2. *Full Virtualization* : GoGrid, Skytap.
3. *Compute* : Amazon Elastic Compute Cloud.

2.2. LINE

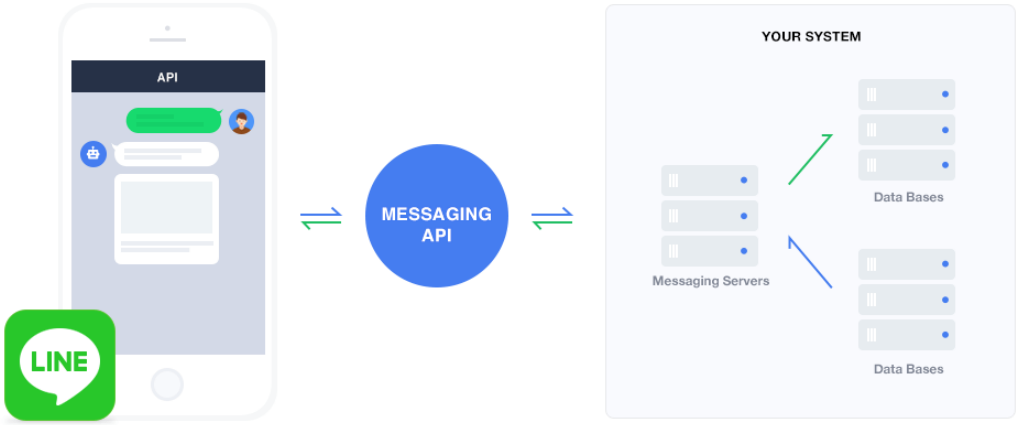
LINE pertama kali dikenalkan oleh Lee Hae Jin pada Juni 2011, kini LINE dikembangkan oleh perusahaan Jepang yakni HNN Corporation. LINE merupakan aplikasi pesan instan yang membantu penggunanya dalam kegiatan berkomunikasi dengan mengirimkan pesan teks, pesan suara, pesan gambar, pesan audio, dan lain - lain. Bukan hanya dikenal sebagai aplikasi pesan instan, LINE dikenal sebagai platform jejaring sosial, dikarenakan pengguna LINE dapat saling berbagi aktivitasnya melalui fitur *timeline*. LINE juga merupakan *smart portal* dengan beragam solusi terkait konten, hiburan dan bisnis. Aplikasi ini telah tersedia dalam bentuk *standalone apps* untuk berbagai macam sistem operasi diantaranya Windows, Mac, Android dan IOS.

Revie Sylviana, *Strategic Partnership Director* LINE Indonesia mengatakan jika pengguna LINE di Indonesia berjumlah lebih dari 90 juta pengguna aktif [3].

2.2.1. LINE *Messaging* API

LINE@ memberikan fasilitas untuk membuat akun bisnis yang dapat digunakan untuk mengirim pesan ke *customer* dan juga berkomunikasi langsung dengan pelanggan. Komunikasi tersebut bisa berbentuk *chat* langsung maupun dengan menggunakan pesan *auto reply* dan *keyword reply*. Messaging API ini memungkinkan LINE@ untuk dapat membuat respon yang dapat disesuaikan dengan kebutuhan khusus yang tak tertangani fitur *auto reply* dan *keyword reply* standar.

Melalui penggunaan *Messaging* API, LINE@ dapat berkirim informasi antara server kita dengan aplikasi LINE pengguna melalui platform LINE [15].



Gambar 2.11. Alur Kerja LINE *Messaging* API

Ada beberapa fitur yang dapat dimanfaatkan pada *Messaging* API, di antaranya [15]:

1. ***Push Message***

   Fitur ini memungkinkan bot dapat mengirimkan pesan tertuju langsung kepada pengguna LINE yang sudah berteman dengan bot.

2. ***Reply Message***

   Fitur ini memungkinkan bot dapat membalas pesan yang dikirim oleh pengguna. Interaksi yang dilakukan dapat dikirimkan ke alamat *URL webhook* server kita.

3. ***Imagemap Message***

   Fitur ini memungkinkan bot dapat mengirim gambar dengan teks dan tautan didalamnya. Imagemap dapat digunakan untuk mengirim konten seperti kupon, promo spesial, berita pengumuman, dan posting blog.

4. ***Template Message***

   Fitur ini memungkinkan bot dapat mengirimkan pesan dengan bentuk dan format yang beragam, seperti gambar, teks, opsi pilihan dan tombol.

2.3. Chatbot

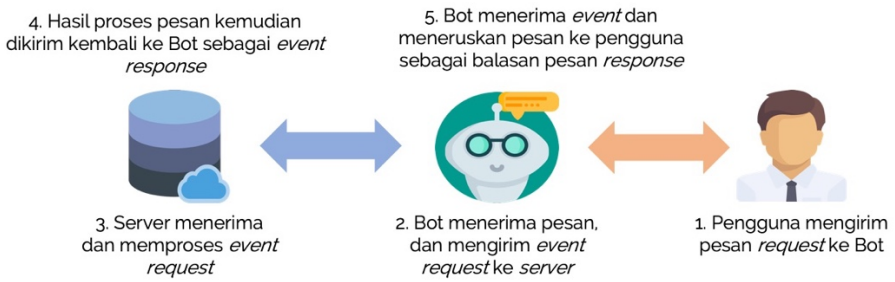
Chatbot atau "robot obrolan" merupakan sebuah mekanisme komunikasi dengan teknologi canggih melalui antarmuka obrolan. Program ini membantu dalam melakukan percakapan dengan pengguna [16].

Chatbot juga merupakan QA system atau question-answering system, yaitu memberikan kemampuan pada sebuah mesin (komputer) untuk menginterpretasikan bahasa alami untuk melakukan dialog dengan pengguna hampir seperti dialog antara dua orang manusia dalam bahasa sehari-hari [17].

Secara teknis, chatbot hanyalah sebuah program komputer yang meniru percakapan dengan orang-orang yang dirancang dengan menggunakan teknologi kecerdasan buatan. Layanan yang difasilitasi oleh chatbot dapat dibuat khusus untuk memberikan tindakan - tindakan tertentu yang dikirim melalui sebuah aplikasi pesan instan. Tetapi di luar perangkat lunak komputer, chatbot juga merupakan teknologi masa kini yang dimanfaatkan untuk berhubungan dengan pelanggan. Chatbot memberikan cara yang nyaman dan interaktif untuk berkomunikasi dengan pelanggan dimana pun dan kapan pun. Chatbot dapat mengurangi beban staf manusia yang ada untuk memenuhi tuntutan tersebut secara efisien dan hemat biaya [18].

2.4. Metode *Webhook*

Metode *webhook* adalah salah satu metode dalam pembuatan suatu bot menggunakan hosting sebagai *server* untuk memproses semua *event* yang diberikan oleh bot. Pada metode *webhook* penggunaan protokol jaringan diwajibkan berupa HTTPS. Metode ini memungkinkan pengguna lain dapat menggunakan bot tanpa harus membuka *file* skrip, namun *webhook* akan bekerja secara *server-side* dan akan melakukan tugasnya ketika mendapat sebuah *trigger*. Alur kerja metode *webhook* dapat dilihat pada Gambar 2.12.



Gambar 2.12. Alur kerja metode *webhook*

2.5. Metode *Forward Chaining*

Algoritma *forward-chaining* adalah satu dari dua metode utama *reasoning* (pemikiran) ketika menggunakan *inference engine* (mesin pengambil keputusan) dan bisa secara logis dideskripsikan sebagai aplikasi pengulangan dari modus ponens (satu set aturan inferensi dan argumen yang valid). Lawan dari *forward-chaining* adalah *backward-chaining* [19].

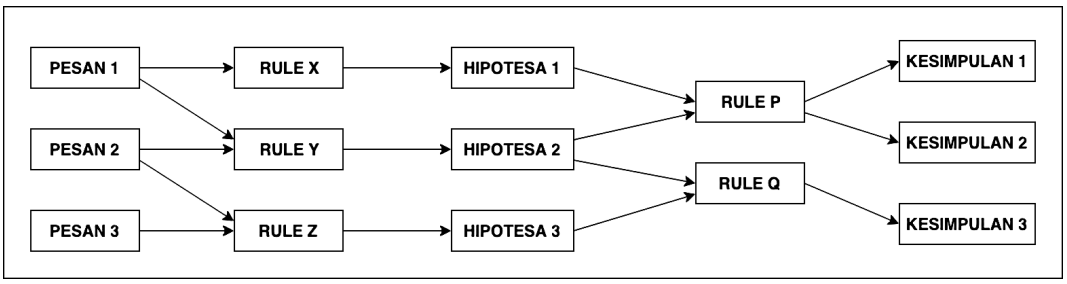
Forward chaining disebut juga penalaran dari bawah ke atas karena penalaran dari fakta pada level bawah menuju konklusi pada level atas didasarkan pada fakta. Penalaran dari bawah ke atas dalam suatu sistem pakar dapat disamakan untuk pemrograman konvensional dari bawah ke atas. Fakta merupakan satuan dasar dari paradigma berbasis pengetahuan karena mereka tidak dapat diuraikan ke dalam satuan paling kecil yang mempunyai makna [6].

Forward-chaining adalah contoh konsep umum dari pemikiran yang dikendalikan oleh data (data-*driven*) yaitu, pemikiran yang mana fokus perhatiannya dimulai dari data yang diketahui. *Forward-chaining* bisa digunakan didalam agen untuk menghasilkan kesimpulan dari persepsi-persepsi yang datang, seringkali tanpa query yang spesifik [19].

Adapun langkah-langkah dalam membangun algoritma ini yaitu :

1) Pendefinisian Masalah.
   Tahap ini meliputi pemilihan domain masalah dan akuisisi pengetahuan.
2) Pendefinisian Data Input.
   Sistem *forward chaining* memerlukan data awal untuk memulai inferensi.
3) Pendefinisian Struktur Pengendalian Data.
   Aplikasi yang kompleks memerlukan premis tambahan untuk membantu mengendalikan

Alur kerja dari *forward-chaining* dapat dilihat pada Gambar 2.13.



Gambar 2.13. Alur kerja metode *forward-chaining*

2.6. Metode *Jaro-Winkler Distance*

Jaro-Winkler *distance* merupakan varian dari Jaro *distance* metrik yaitu sebuah algoritma untuk mengukur kesamaan antara dua string, biasanya algoritma ini digunakan di dalam pendeteksian duplikat. Semakin tinggi Jaro-Winkler distance untuk dua string, semakin mirip dengan string tersebut. Jaro-Winkler distance terbaik dan cocok untuk digunakan dalam perbandingan string singkat seperti nama orang. Skor normalnya seperti (0) menandakan tidak ada kesamaan, dan (1) adalah sama persis [20].

Algoritma Jaro-Winkler *distance* memiliki kompleksitas waktu *quadratic runtime complexity* yang sangat efektif pada string pendek dan dapat bekerja lebih cepat dari algoritma edit *distance*. Dasar dari algoritma ini memiliki tiga bagian:

1. Menghitung panjang string.
2. Menemukan jumlah karakter yang sama di dalam dua string.
3. Menemukan jumlah transposisi.

Pada algoritma *Jaro-Winkler Distance* digunakan rumus untuk menghitung jarak (d_j) antara dua string yaitu S_1dan S_2 adalah :

$$d_j = \frac{1}{3}\ x\ (\frac{m}{S_1} + \frac{m}{S_2} + \frac{m-t}{m})$$

Dimana :

m = Jumlah karakter yang sama persis

$|S_1|$ = Panjang string 1

$|S_2|$ = Panjang string 2

t = Jumlah transposisi

Maka untuk menghitung *Jaro-Winkler Distance* (d_w) adalah :

$$d_w = d_j + (l\ x\ p\ (1 - d_j))$$

Dimana :

d_j = *Jaro distance* untuk string S_1 dan string S_2

l = Panjang *prefix* (panjang karakter yang sama sebelum ditemukan ketidaksamaan) nilai maksimum 4 karakter

p = Konstanta *scaling factor* (Nilai standar untuk konstanta ini menurut *Winkler* adalah p = 0.1)

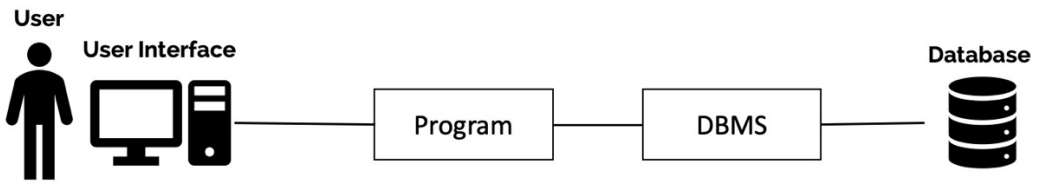
2.7. Basis Data

Basis data atau *Database* merupakan sebuah kumpulan data yang terintegrasi dan diatur sedemikian rupa sehingga data tersebut dapat dimanipulasi, diambil dan dicari secara cepat. Basis data memiliki beberapa model diantaranya model relasional. Dalam model relasional, tabel – tabel yang terdapat dalam suatu basis data idealnya harus saling berelasi [21].

2.7.1. *Database Management System* (DBMS)

Database Management System (DBMS) adalah kumpulan program yang digunakan untuk mendefinisikan, mengatur, dan memproses *database*. DBMS merupakan alat atau *tool* yang berperan untuk membangun struktur tersebut. Dalam satu DBMS dapat memiliki lebih dari satu *database*. DBMS juga sering disebut sebagai *server database* [21].

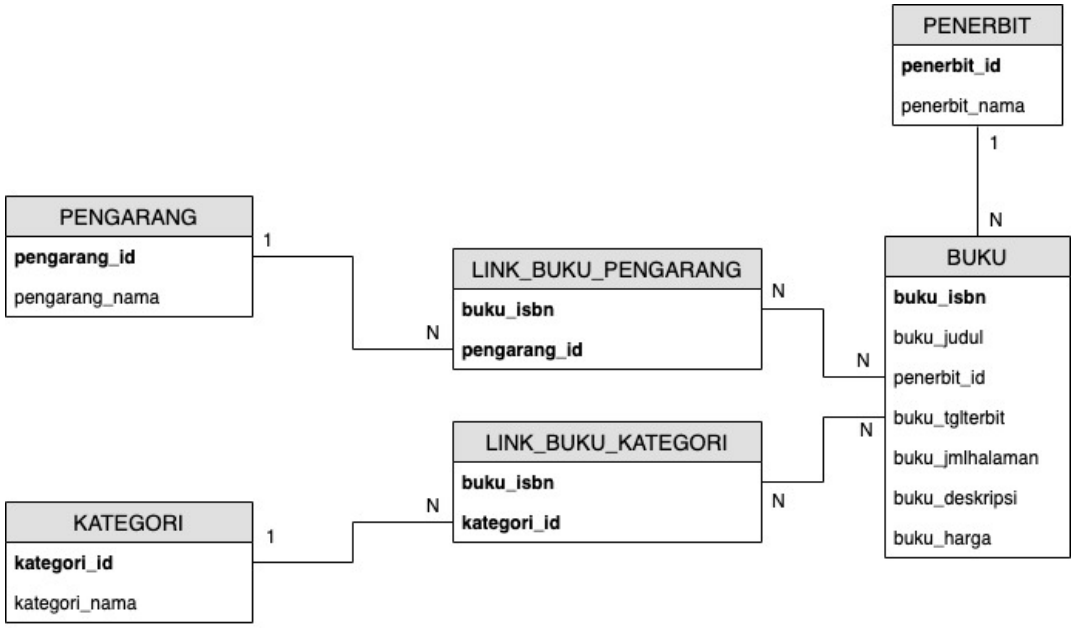
Untuk dapat memahami perbedaan antara *database* dan DBMS dapat dilihat pada Gambar 2.14.



Gambar 2.14. Perbedaan *database* dan DBMS

2.7.2. *Entity Relationship Diagram* (ERD)

Entity Relalationship Diagram (ERD) merupakan salah satu alat bantu (berupa gambar) dalam model *database* relasional yang berguna untuk menjelaskan hubungan atau relasi antar tabel yang terdapat di dalam *database.* Dalam ERD juga dapat melihat daftar kolom yang menyusun masing-masing tabel [21]. Untuk dapat memahami ERD dapat dilihat pada Gambar 2.15.



Gambar 2.15. Contoh *Entity Relationship Diagram*

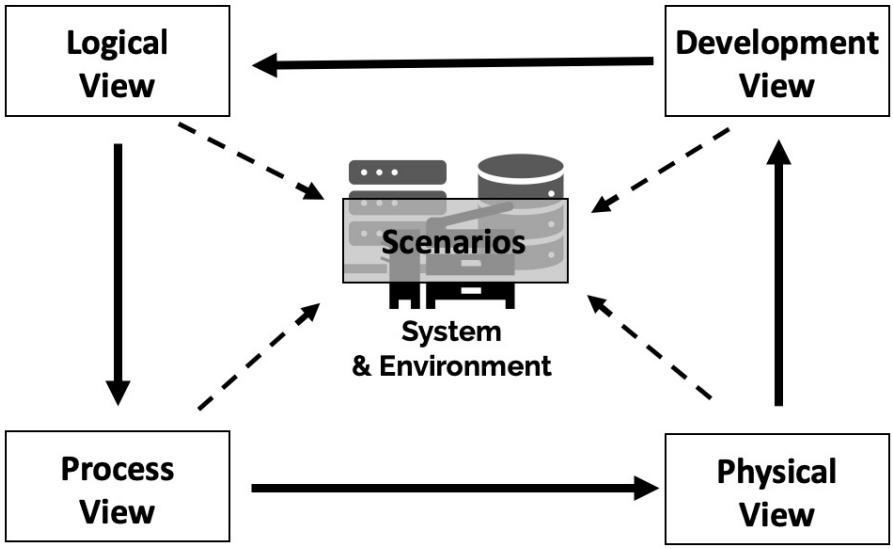
2.8. *Object Oriented Analysis Desain*

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memerika requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelaskelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau subsistem [22].

2.8.1. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah salah saru alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain [23].

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahwa struktur sebuah sistem dideskripsikan dalam 5 *view* dimana salah satu diantaranya *scenario*. *Scenario* ini memegang peran khusus untuk mengintegrasikan *content* ke *view* yang lain [23]. Model 4+1 *view* dapat dilihat pada Gambar 2.16.



Gambar 2.16. Model 4+1 *view*

2.8.2. *Use Case Diagram*

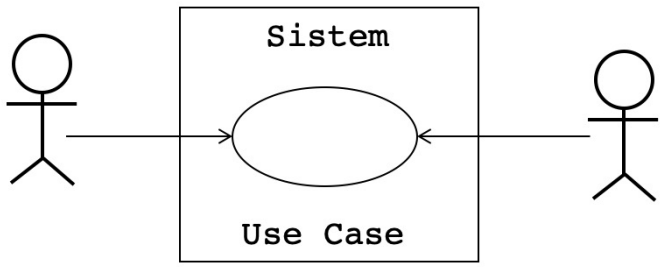
Use Case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use Case* bekerja dengan cara mendeskripsikan tipikal interaksi antara pengguna sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai.

Use Case diagram digunakan untuk menggambarkan analisis kebutuhan dari sistem dari level atas melalui fungsionalitas dari sistem dan interaksi diantara para aktor. Aktor adalah sesuatu yang berinteraksi dengan sistem.

Secara umum, tujuan dari *use case diagram* adalah sebagai berikut [23]:

1. Digunakan untuk mengumpulkan kebutuhan dari sebuah sistem
2. Untuk mendapatkan pandangan dari luar sistem
3. Untuk mengidentifikasi factor yang mempengaruhi sistem baik internal maupun eksternal
4. Untuk menunjukkan interaksi dari para actor dari sistem

Ilustrasi dari *actor, use case* dan *boundary* dapat dilihat pada Gambar 2.17.



Gambar 2.17. *Use Case Model*

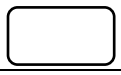
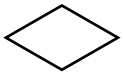
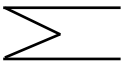
2.8.3. *Activity Diagram*

Activity Diagram adalah bagian penting dari UML yang menggambarkan aspek dinamis dari sistem. Logika procedural, proses bisnis, dan aliran kerja suatu bisnis bias dengan mudah dideskripsikan dalam *activity diagram*. *Activity diagram* memiliki peran seperti halnya *flowchart*, akan tetapi perbedaannya adalah *activity diagram* bisa mendukung perilaku pararel.

Tujuan dari *activity diagram* adalah untuk menangkap tingkah laku dinamis dari sistem dengan cara menunjukkan aliran pesan dari satu aktifitas ke aktifitas lainnya. Secara umum tujuan *activity diagram* adalah sebagai berikut [23]:

1. Menggambarkan aliran aktivitas dari sistem
2. Menggambarkan urutan aktifitas dari satu aktifitas ke aktifitas lainnya
3. Menggambarkan pararelisme, percabangan dan aliran konkuren dari sistem

Tabel 2.1. Simbol - simbol yang sering dipakai pada *activity diagram*

Simbol	Keterangan
	Titik Awal
	Titik Akhir
	Activity
	Pilihan untuk pengambilan keputusan (percabangan)
	Fork ; digunakan untuk menunjukan kegiatan yang dilakukan secara pararel atau untuk menggabungkan dua kegiatan pararel menjadi satu
	Rake ; menunjukan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (*Flow Final*)

2.8.4. *Class Diagram*

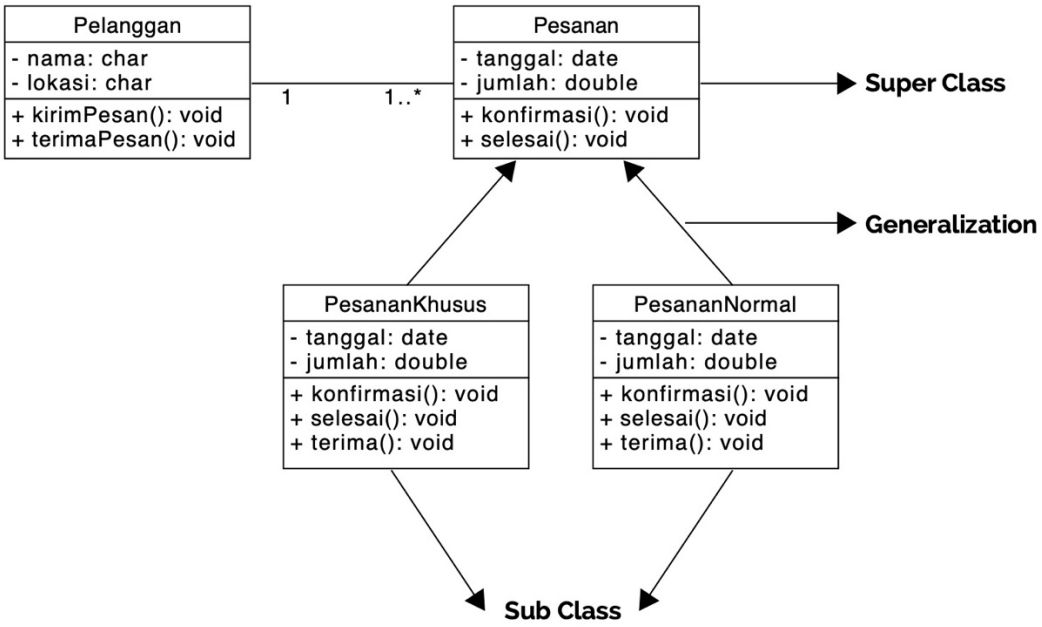
Class Diagram adalah diagram statis yang mewakili pandangan statis dari suatu aplikasi. *Class diagram* tidak hanya digunakan untuk memvisualisasikan, menggambarkan, dan mendokumentasikan berbagai aspek sistem tetapi juga untuk membangun kode eksekusi (*executetable code*) dari aplikasi perangkat lunak. *Class diagram* menunjukan koleksi kelas, antarmuka, asosiasi, kolaborasi, dan *constraint*. *Class diagram* juga dikenal sebagai diagram struktural.

Tujuan dari *class diagram* adalah untuk memodelkan pandangan statis suatu aplikasi. Secara lebih rinci tujuannya adalah sebagai berikut [23]:

1. Analisis dan desain pandangan statis aplikasi
2. Menjelaskan tanggung jawab suatu sistem

3. Basis untuk diagram komponen dan penyebaran (*deployment*)
4. *Forward and reverse engineering*

Contoh *class diagram* dapat dilihat pada Gambar 2.18.

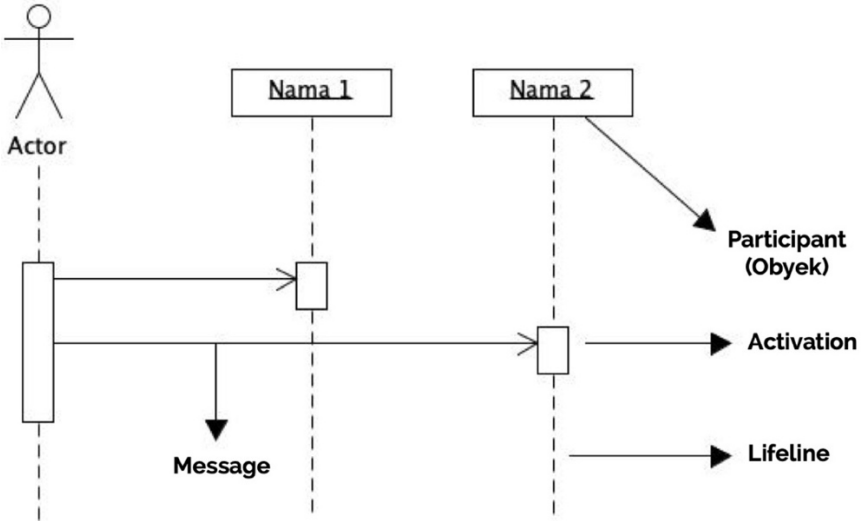


Gambar 2.18. Contoh *Class Diagram* Sistem Pemesanan

2.8.5. *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Diagram ini menunjukan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *use case*.

Komponen utama *sequence diagram* terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progress vertical [23]. Simbol – symbol yang ada pada *sequence diagram* dapat dilihat pada Gambar 2.19.



Gambar 2.19. Simbol-simbol yang ada pada *sequence diagram*

2.9. Metode Pengujian

Metode pengujian merupakan metode – metode yang digunakan dalam pengujian perangkat lunak yang dibangun. Pengujian dilakukan untuk memberikan evaluasi atas perangkat lunak yang dibangun, apakah telah sesuai dengan kebutuhan yang menjadi masalah atau belum sesuai.

Betikut merupakan beberapa metode pengujian yang dilakukan pada penelitian ini.

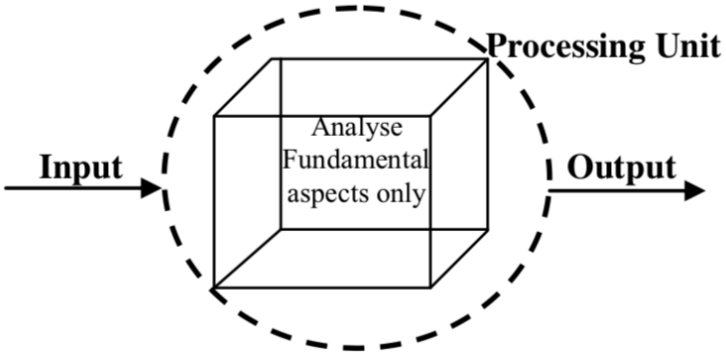
2.9.1. Pengujian *Alpha*

Pengujian *alpha* merupakan salah satu strategi pengujian perangkat lunak yang paling umum digunakan dalam pengambang perangkat lunak. Pengujian ini dilakukan pada sisi pengembang perangkat lunak. Pengujian *alpha* merupakan pengujian akhir sebelum nantinya perangkat lunak di publikasi ataupun digunakan oleh pengguna luas. Pengujian ini bertujuan agar ketika perangkat lunak digunakan oleh pengguna dapat meminimalisir kesalahan pada perangkat lunak [24].

2.9.1.1. Metode *Black Box*

Metode *Black Box* adalah teknik pengujian tanpa memiliki pengetahuan tentang kerja internal dari aplikasi. Hanya mengkaji aspek fundamental dari sistem

dan tidak memiliki atau relevansi kecil dengan struktur logis internal sistem. Saat melakukan uji kotak hitam, tester harus mengetahui arsitektur sistem dan tidak akan memiliki akses ke kode sumber [25].



Gambar 2.20. Ilustrasi Metode *Black Box*

Metode *Black Box* memiliki keuntungan dan kekurangan diantaranya [25] :

A. Keuntungan
   1. Efisien diterapkan pada segmentasi kode yang besar.
   2. Persepsi yang harus dimiliki *tester* sederhana.
   3. Perspektif pengguna dipisahkan dari perspektif pengembang (*programmer* dan *tester* independen satu sama lain).
   4. Pengembangan kasus uji relatif cepat.

B. Kekurangan
   1. Hanya sejumlah skenario yang dilakukan dan dipilih dalam pengujian. Akibatnya, cakupan pada pengujian terbatas.
   2. Tanpa spesifikasi yang jelas sehingga kasus uji sulit untuk dirancang.
   3. Pengujian tidak efisien.

2.9.2. Pengujian *Beta*

Pengujian *Beta* dikenal sebagai pengujian sisi lapangan, di mana pengujian ini dilakukan pada pengguna yang berperan sebagai target pengguna perangkat lunak. Tujuan pengujian *beta* adalah untuk menempatkan aplikasi kepada pengguna nyata di luar tim pengembang perangkat lunak. Hal ini untuk menemukan

kekurangan atau masalah dari perspektif pengguna sebelum perangkat lunak benar – benar di publikasi secara luas [26].

2.9.2.1. Skala Likert

Skala likert adalah skala pengukuran yang dikembangkan oleh Likert. Skala likert mempunyai empat atau lebih butir-butir pernyataan yang di kombinasikan sehingga membentuk sebuah skor/nilai yang merepresentasikan sifat individu, misalkan pengetahuan, sikap, dan perilaku. Dalam proses analisis data, komposit skor, biasanya jumlah atau rataan, dari semua butir pertanyaan dapat digunakan.

Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam kuesioner, dan skala yang paling banyak digunakan dalam riset berupa survei. Nama skala ini diambil dari nama Rensis Likert, yang menerbitkan suatu laporan yang menjelaskan penggunaannya. Sewaktu menanggapi pertanyaan dalam skala likert responden menentukan tingkat persetujuan mereka terhadap suatu pernyataan dengan memilih salah satu dari pilihan yang tersedia. Biasanya disediakan lima pilihan skala dengan format seperti [27]:

1. Sangat setuju
2. Setuju
3. Netral
4. Tidak Setuju
5. Sangat Tidak Setuju

Skala Likert kerap digunakan sebagai skala penilaian karena memberi nilai terhadap sesuatu.

Untuk keperluan analisis kuantitatif, skala jawaban pada skala likert dapat diberi skor misalnya :

1. Sangat Setuju (SS) diberi skor 5
2. Setuju (ST) diberi skor 4
3. Ragu-ragu (RG) diberi skor 3
4. Tidak Setuju (TS) diberi skor 2
5. Sangat Tidak Setuju (STS) skor 1

2.10. Perangkat Lunak Pendukung

Perangkat lunak pendukung merupakan perangkat yang berupa aplikasi, bahasa pemograman dan lainnya. Perangkat lunak ini digunakan sebagai pendukung dalam pembangunan perangkat lunak dalam penelitian ini.

Berikut merupakan beberapa perangkat lunak pendukung yang diperlukan pada penelitian ini.

2.10.1. *Hypertext Markup Language* (HTML)

Hypertext Markup Language (HTML) adalah bahasa yang digunakan dalam menulis halaman web. HTML merupakan pengembangan dari standar pemformatan dokumen teks, yaitu *Standard Generalized Markup Language* (SGML). HTML pada dasarnya merupakan dokumen ASCII atau teks biasa, yang dirancang untuk tidak tergantung pada suatu sistem operasi tertentu. Kegunaan Bahasa ini ialah untuk memanipulasi *browser* sehingga dapat menampilkan informasi yang dapat dibaca oleh pengguna komputer [28].

Kode html dibuat dalam file teks biasa yang disimpan dengan ekstensi "*.htm" atau "*.html". 26 kode HTML terdiri dari tag-tag yang memiliki fungsi yang unik. Tag berarti penanda item baik yang akan ditampilkan oleh web browser maupun tidak. Biasanya tag ini ditulis berpasangan dan mengapit item yang akan dijelaskan oleh tag tersebut. Tag tidak *case sensitive*, jadi bisa menggunakan huruf kapital maupun huruf kecil atau keduanya dan akan menghasilkan output yang sama. Dokumen HTML dibagi menjadi tiga bagian utama yaitu tag HTML, Head, dan Body. Untuk memulai bekerja dengan HTML bisa menggunakan teks editor apa saja, seperti Notepad, Notepad++, Sublime-text, Adobe Dreamweaver, dan lainnya [29].

2.10.2. *Cascading Syle Sheet* (CSS)

Cascading Style Sheet (CSS) adalah suatu bahasa *stylesheet* yang digunakan untuk mengatur tampilan suatu *website*, baik tata letaknya, jenis huruf, warna, dan semua yang berhubungan dengan tampilan. Pada umumnya CSS digunakan untuk memformat halaman web yang ditulis dengan HTML atau XHTML [28].

Kode css itu sendiri dapat kita sisipkan langsung pada file html, dan bisa juga kita tuliskan pada file terpisah yang berekstensi "*.css". Pada dasarnya tidak ada ketentuan dalam penulisan kode *stylesheet* di file html ataupun terpisah di file css. Namun bila memiliki banyak file html yang harus di kelola, maka menuliskan kode *stylesheet* pada file css terpisah merupakan pilihan yang lebih baik dalam menghindari penulisan kode yang sama berulang kali dan membuat kode *stylesheet* menjadi lebih *reusable* [30].

2.10.3. *Javascript*

Javascript adalah bahasa *script* berdasar pada objek yang memperbolehkan pemakai untuk mengendalikan banyak aspek interaksi pemakai pada suatu dokumen HTML. Di mana objek tersebut dapat berupa suatu *window*, *frame,* URL, dokumen, *form*, *button* atau item yang lain. Yang mana item tersebut memiliki sebuah property yang saling berhubungan dengannya, dan masing-masing memiliki nama, lokasi, warna nilai, dan atribut lain [28].

Kode *javascript* dapat diselipkan pada dokumen html "*.htm" atau dapat pula disimpan pada file terpisah dengan ekstensi "*.js" lalu dipanggil dari dokumen html dengan tujuan agar kode *javascript* dapat lebih *reusable*. Kelebihan menggunakan *client side scripting* adalah dapat digunakan untuk membangun sebuah tampilan yang dinamis dan lebih atraktif. Namun, kekurangannya adalah karena merupakan *client side scripting*, maka kode program dapat di lihat oleh browser pengguna, hal ini dapat menimbulkan celah keamanan jika salah dalam penggunaannya [31].

2.10.4. *Hypertext Preprocessor* (PHP)

PHP atau kependekan dari *Hypertext Preprocessor* adalah salah satu bahasa pemrograman *open source* yang sangat cocok atau dikhususkan untuk pengembangan web dan dapat ditanamkan pada sebuah skripsi HTML. Bahasa PHP dapat dikatakan menggambarkan beberapa bahasa pemrograman seperti C, Java, dan Perl serta mudah untuk dipelajari. PHP merupakan bahasa *scripting server – side*, dimana pemrosesan datanya dilakukan pada sisi server. Sederhananya, web

server yang akan melakukan penerjemahan skrip program, kemudian hasilnya akan dikirim kepada *client* yang melakukan permintaan [32].

2.10.5. MySQL

MySQL merupakan *software* RDBMS (atau *server database*) yang dapat mengelola *database* dengan sangat cepat, dapat menampung data dalam jumlah sangat besar, dapat diakses oleh banyak pengguna (*multi-user*), dan dapat melakukan suatu proses secara sinkron atau berbarengan (*multi-threaded*) [21].

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*Database Management System*/DBMS) yang *multi-thread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL [33].

2.10.6. *Model View Controller* (MVC)

Model view controller (MVC) merupakan salah satu contoh dari *Architectural Design Pattern*. Pada saat ini banyak *framework* yang didedikasikan pemrograman berorentasi objek [34]. Beberapa keuntungan adalah:

1. Penggunaan ulang komponen-komponen antarmuka (*user interface reusable component*).
2. Kemampuan untuk mengembangkan aplikasi dengan antarmuka pengguna secara terpisah.
3. Kemampuan untuk melakukan pewarisan (*inheritance*) dari berbagai bagian yang berbeda pada suatu hierarki kelas.
4. Kemampuan untuk mendefinisikan kelas-kelas pengaturan tampilan (*control style*) yang menyediakan fitur-fitur umum secara terpisah dengan fitur-fitur yang akan ditampilkan oleh aplikasi yang dikembangkan.

2.10.7. *Framework*

Framework adalah kerangka kerja yang memudahkan *programmer* akan lebih mudah melakukan perubahan (*customize/maintenance*) terhadap aplikasinya dan dapat memakainya kembali untuk aplikasi lain yang sejenis. *Framework* biasanya sudah memiliki aplikasi implementasinya, maka beda antara pola desain dan *framework* adalah pola desain merupakan sebuah konsep sedangkan *framework* bias merupakan aplikasi hasil implementasi dari pola desain atau aplikasi kerangka kerja yang tidak berasal dari implementasi sebuah konsep [34].

2.10.8. CodeIgniter

CodeIgniter merupakan *Application Development Framework* yang dibuat pertama kali oleh Rick Ellis seorang CEO Ellislab, Inc. *CodeIgniter* dirancang menggunakan bahasa pemograman PHP dengan menerapkan pendekatan *model-view-controller* (MVC) pada base skripnya. *CodeIgniter* memiliki *base code* yang terstruktur sehingga dapat membantu seorang *developer* dalam merancang dan *maintenance* sebuah *website*. *CodeIgniter* menyediakan banyak *library* yang dapat digunakan langsung untuk mempercepat dalam pembangunan *website* seperti *query builder, session management, email, database, form and data validation, file uploading class, security and xss filtering, image manipulating, pagination,* dan *library* lainnya [35].

2.10.9. Bootstrap

Bootstrap adalah kerangka kerja *front-end open source* yang dikelola oleh Twitter untuk mengembangkan situs web dan aplikasi web yang responsif. Artinya, tampilan web yang dibuat oleh bootstrap akan menyesuaikan ukuran layar dari browser yang kita gunakan baik di desktop, tablet ataupun mobile device. Bootstrap di dalamnya sudah termasuk kode HTML, CSS, dan JavaScript yang dapat membantu membangun komponen antarmuka pengguna dengan lebih cepat dan mudah.

Sistem *grid* pada Bootstrap membantu dalam membangun tata letak, dan komponen antarmuka dengan presisi 12 kolom yang responsif. Bootstrap memiliki banyak komponen yang dapat digunakan dan dikembangkan yang *prestyled* dan dilengkapi plugin jQuery khusus, seperti *button*, *alert*, *dropdown*, *modal*, *tab tooltip*, *pagination*, *slider*, *badge*, ikon, dan banyak lagi [36].

2.10.10. Metode Klasifikasi Tingkatan

Metode yang digunakan untuk memperoleh optimalisasi *hardware server* salah satunya adalah klasifikasi tingkatan. Klasifikasi tingkatan dapat berbeda – beda sesuai dengan kebutuhan aplikasi yang dibangun. Adapun variable yang digunakan dalam metode ini diantaranya [37]:

1. Jumlah pengguna yang akan aktif dalam aplikasi
2. Jumlah pengguna yang berpotensi mengakses aplikasi secara bersamaan
3. Jumlah pengguna yang gagal melakukan operasi pada aplikasi
4. Data atau informasi apa yang akan diproses

Perhitungan pada metode ini dilakukan dengan 4 langkah diantaranya :

1. Menghitung rasio aktifitas bisnis yang dilakukan pengguna

   $$rasioAktifitas = 100\% + \left(\frac{userFailed}{userAccess}\ x\ 100\%\right)$$

2. Menghitung optimalisasi pengguna

   $$optimalisasiPengguna = userNumber\ x\ 100\%\ x\ userNumber$$

   $$aktifitasPuncak = maxUser\ x\ Rasio\ x\ maxUser$$

3. Menghitung rasio optimalisasi

   $$rasioOptimalisasi = \frac{optimalisasiPengguna}{aktifitasPuncak}$$

4. Melihat tabel klasifikasi berdasarkan hasil rasio optimalisasi

   Tabel klasifikasi RAM terlihat pada Tabel 2.2.

Tabel 2.2. Klasifikasi RAM

Ukuran RAM (GB)	Rasio Optimalisasi
2	< 10%
4	> 10%
8	> 40%
16	> 50%

Tabel klasifikasi CPU terlihat pada Tabel 2.3.

Tabel 2.3. Klasifikasi CPU

Clock Rate CPU	Rasio Optimalisasi
450 MHz	< 10%
1,2 GHz	> 10%
2,4 GHz	> 40%
3,10 GHz	> 50%

Tabel klasifikasi *disk* terlihat pada Tabel 2.4.

Tabel 2.4. Klasifikasi *Disk*

Size Min Disk	Rasio Optimalisasi
10 GB	< 10%
25 GB	> 10%
50 GB	> 40%
100 GB	> 50%