BAB 2
TINJAUAN PUSTAKA

2.1 *Food Combining*

Food Combining sudah diperkenalkan oleh bangsa Esseni di palestina 2000 tahun silam dan pada awal abad ke-20 *food combining* dipopulerkan oleh dokter William Howard Hay.[3], [4]

2.1.1 Pengertian *Food Combining*

Food Combining merupakan metode untuk mengatur pola makan untuk mendapatkan energi dan asupan gizi yang seimbang untuk kebutuhan aktifitas sehati-hari. *Food combining* mengatur pola makan mulai dari waktu atau kapan makanan dimakan dan kombinasi makanan apa saja yang baik. Dengan *food combining* dapat mencapai kondisi ideal tubuh di mana seluruh fungsi berjalan sempurna. Indikator tercapainya keseimbangan tubuh dengan melihat asam dan basa tubuh atau kondisi pH netral (7,35-7,45). Untuk menghindari kerusakan organ tubuh dan datangnya berbagai penyakit, dibutuhkan keseimbangan tubuh.

2.1.2 Pola *Food Combining*

Terdapat tiga pola makan pada *food combining*, diantaranya:[5]

1. Apa yang kita makan

Ada beberapa unsur makanan yang diperkenalkan oleh *food combining*, diantaranya:

a. Pati

Pati dikenal juga sebagai karbohidrat yang sering disalah artikan sebagai penyebab masalah kesehatan terutama kelebihan berat badan dan diabetes. Pati yang baik adalah yang masih memiliki zat-zat gizi alamiah dan minim proses. Misalnya, yang masih mengandung vitamin, serat, enzim, mineral, dan substansi penting lain yang bisa dimanfaatkan oleh tubuh secara maksimal.

b. Protein

Protein memiliki kandungan asam amino yang lengkap. Asam amino adalah untur utama pembentukan sel, bahan utama pembangunan dan perbaikan jaringan

tubuh, hormon, enzim, dan masih banyak lagi yang lainnya. Ada beberapa jenis protein yang dapat dikonsumsi diantaranya protein dari hewani dan nabati. Protein nabati dalam bentuk daging-dagingan, susu dan telur. Sedangkan protein nabati dalam bentuk kacang-kacangan dan polong-polongan. Selain protein hewani dan nabati, sayur dan buah pun menyumbang protein dalam bentuk asam amino sederhana yang lebih mudah diserap oleh tubuh.

c. Sayuran

Sayuran adalah unsur makanan yang sangat berguna sebagai pembentuk sifat basa. Dengan mengkonsumsi sayuran dengan benar, sayuran dapat menetralkan pH dan menciptakan kondisi homeostatis bagi tubuh. Sayuran mengandung banyak karbohidrat, serat, vitamin dan mineral.

d. Buah

Buah adalah salah satu kelompok makanan penyumbang air, enzim, karbohidrat, serat, vitamin dan mineral. Seperti halnya sayuran, dengan mengkonsumsi buah dengan benar dapat menghasilkan sifat basa pada tubuh. Buah juga dapat mempermudah tubuh mencapai kondisi homestatis.

2. Waktu makan

Food combining mengacu pada *ritme biologis* dalam mengatur waktu dan jenis makanan yang sesuai dengan kebutuhan tubuh. *Ritme biologis* sebagai berikut:

a. 12:00-20:00 Waktu Cerna

Pada fase ini, sistem pencernaan aktif menerima makanan sehingga dapat leluasa untuk mengkonsumsi makanan.

b. 20:00-04:00 Waktu Penyerapan

Pada fase ini, tubuh secara maksimal memanfaatkan makanan yang dimakan pada waktu sebelumnya. Pada fase ini penyerapan zat gizi, sirkulasi zat-zat berguna yang di proses dari makanan, pergantian sel, perbaikan jaringan, dan sebagainya dilakukan.

c. 04:00-12:00 Waktu Pembersihan

Pada pase ini, sisa metabolisme serta tumpukan makanan pada usus besar dikonsentrasikan tubuh untuk dibuang. Dibutuhkan banyak energi dialokasikan

tubuh untuk membersihkan sisa-sisa kotoran. Menyibukan sistem pencernaah dengan makan makanan berat dapat mengganggu alokasi energi.

Berdasarkan *ritme biologis*, maka pembagian waktu makan *food combining* adalah sebagai berikut:

a. *Jenifer Time* (04:00-06:00)

Jenifer Time dilakukan pada saat bangun tidur atau idealnya antara pukul 04:00-06:00. *Jenifer* adalah air hangat yang dicampur dengan perasan jeruk nipis atau lemon. Takaran air *jenifer* sebatas kenyang saja dan tidak kekenyangan. Tambahan air jeruk nipis atau lemon cukup 3 sendok makan per gelas. *Jenifer* diminum seteguk demi seteguk, didiamkan sesaat dalam mulut hingga bercampur dengan air ludah baru ditelan bukan langsung dihabiskan sekaligus.

b. Sarapan Buah (06:00-12:00)

Sarapan buah bisa berupa buah potong atau jus buah. Pemanfaatan buah segar sebagai bahan baku makanan untuk sarapan. Sifat buah yang mudah dicerna, ringan, tetapi memberikan asupan energi yang signifikan. Konsumsi sarapan yang sesuai dengan kebutuhan tubuh di fase ini bukan sesuai kemauan. Pilih substaansi makanan yang sehat dan tepat.

Konsumsi buah dalam keadaan perut kosong atau beri jarang 15-20 menit sebelum makan. Sesudah makan sebaiknya tidak menyantap buah baik buah potong atau jus buah. Beri waktu sekitar 4-5jam kemudian untuk mengkonsumsi buah. Buah potong atau jus buah dikonsumsi perlahan, dikunyah dengan baik pastikan bercampur dengan air liur. Saat perut kenyang hentikan makan. Hindari mengkonsumsi buah beralkohol seperti durian, nangka dan cempedak

c. Makan Siang (12:00-16:00) dan Makan Malam (18:00-20:00)

Makanan dibagi ke dalam 3 unsur dasar: pati, protein dan sayuran. Perpaduan unsur-unsur makanan tersebut adalah hal yang paling utama dari metode *food combining*.

1) Protein Hewani dan Pati

Protein hewani dan pati(karbohidrat) akan menghasilkan masalah bagi pencernaah. Masing-masing unsur makanan tersebut memerlukan enzim yang berbeda untuk nantinya diloah tubuh. Karbohidrat dicerna oleh enzime cerna

amilase (terdapat di air liur) sedangkan protein hewani dicerna oleh enzime pepsin (bekerja begitu makanan memasuki alat cerna di dalam tubuh). Kedua enzim ini tidak dapat bekerja sama karena enzim amilase akan berhenti bekerja sehingga menghasilkan karbohidrat yang belum terurai sempurna sepanjang proses pencernaan.

Selain kedua enzim itu tidak dapat bekerja sama, waktu cerna atau terurainya unsur-unsur makanan memiliki waktu yang berbeda. Paduan kedua unsur makanan tersebut dapat menyebabkan endapan sisa makanan yang tidak terurai dengan baik. Endapat ini akan menumpuk dan sulit dikeluarkan sehingga mengundang bakteri serta parasit yang akan mengganggu kesehatan tubuh. Sumber protein yang paling merusak atau mengundang penyakit diataranya : hidangan ayam pada restoran cepat saji atau daging sapi dalam bentuk sosis atau burger.

Gejala pendek yang dapat ditimbulkan oleh perpaduan unsur makanan ini diantaranya: mudah pusing, cepat lelah, dan daya tahan tubuh menurun. Dalam kasus yang lebih parah, dapat menyebabkan penyakit yang serius dan parah. Kondisi ini juga dapat membuat pH tubuh tidak berapa pada kondisi netral dan cenderung pH tubuh menjadi asam.

2) Protein dan Sayuran

Protein dan sayuran adalah kombinasi idela dan saling melengkapi satu sama lain. Protein hewani pembentuk asam dan sayur-sayuran segar sebagai pembentuk basa menjadi kombinasi ideal karena mengkonsumsi keduanya dapat meminimalisir pengaruh buruk protein hewani terhadap tubuh. Mengkonsumsi sayuran segar dapat membantu sistem pencernaan dalam memproses protein hewani. Sayuran yang tinggi akan nilai patinya seperti kentang, talas, ubi, jagung dan sejenisnya bukanlah jenis sayuran yang dianjurkan untuk dikombinasikan dengan protein hewani. Sayuran yang memerlukan proses memasak yang lama seperti sayur lodeh, gulai pakis, dan sup tomat tergolong sulit memberikan efek positif untuk membantu mengurai protein hewani.

3) Protein Nabati dan Pati

Protein nabati tergolong protein yang netral, terutama dalam bentuk pasca-permentasi seperti tempe karena ringan pada saat proses cernanya. Kandungan pada protein nabati pun tidak memberatkan. Karena itu, protein nabati tidak tergolong pada kombinasi ideal bila dikombinasikan dengan pati.

4) Pati dan Sayuran

Seperti halnya dengan kombinasi protein dan pati, kombinasi pati dengan sayuran saling melengkapi dan ideal. Serat pada sayuran dapat sedikit mengatasi efek buruk pati yang ditimbulkan. Kombinasi pati dan sayuran harus dalam perbandingan sama. Sebagai contoh, bagilah piring menjadi tiga bagian seperti sepertiga untuk pati, sepertiga untuk sayuran segar dan sisanya untuk menu pelengkap yang tidak bertentangan dengan panduan *food combining*.

d. Kudapan Sore (16:00-17:30)

Kudapan sore dilakukan jika perut merasa lapan dan waktu makan malam belum tiba. Kudapan sore dapat dilakukan atau tidak tergantuk pada kondisi perut. Apabila makan siang berupa protein hewani + sayuran, maka kudapan sorenya adalah hidangan yang netral seperti sayuran (jus sayuran atau salad sayuran). Apabila makan siang berupa protein mudah cerna kudapan sore bisa kudapan pati (bubur kacang hijau atau pisang rebus).

2.1.3 Aturan *Food Combining*

Ada beberapa aturan dalam *food combining*, diantaranya:

1. Minum air hangat perasan jeruk nipis atau lemon sesaat setelah bangun tidur.
2. Hindari menggabungkan pati dan protein.
3. Konsumsi melon dan semangka saat perut kosong.
4. Dalam satu minggu *food combining* tidak harus dilakukan setiap hari. *Food combining* bisa dilakukan selama 5 hari dan 2 hari bebas dari *food combining*, biasa di sebut cheating. Sebaiknya lakukan cheating pada waktu makan siang dan malam, sebagiknya sarapan buah tetap dilakukan.
5. Tidak ada porsi makan dalam *Food Combining*, porsi makan ditentukan oleh tubuh masing-masing. Saat makan sudah kenyang berhenti, karena masing-masing orang memiliki porsi makan yang berbeda-beda.

6. Pada setiap kombinasi menu makanan (makan siang dan makan malam) sayuran harus lebih banyak dari pada protein atau pati. Misal 70% sayuran dan 30% protein atau pati.
7. komposisi menu dalam satu hari sebagai berikut:
   a. 1 menu protein + sayuran,
   b. 1 menu pati + sayuran,
   c. 1 menu buah-buahan.

   Atau,
   a. 2 menu pati + sayuran,
   b. 1 menu buah-buahan.

   Atau,
   a. 1 menu protein + sayuran
   b. 2 menu buah-buahan

   Atau,
   a. 1 menu pati + sayuran
   b. 2 menu buah-buahan.

2.1.4 Contoh *Food Combining*

Terdapat dua jenis contoh menu yaitu menu dalam keadaan ideal dan kurang ideal. Menu dalam keadaan ideal yaitu hidangan rumahan atau memiliki cukup dana untuk membeli beragam bahan makanan yang sehat dan berkualitas. Dan menu dalam keadaan kurang ideal adalah keadaan yang tidak memiliki dapur di tempat tinggal, menu hasil dari membeli di tempat sederhana atau hanya mampu membeli bahan makanan terbatas. Berikut adalah contoh *food combining* dengan menu ideal dan kurang ideal.[5]

1. Menu ideal

Tabel 2.1 Contoh Food Combining dengan Menu Ideal

Waktu	Keterangan	Menu
05:00	Bangun Pagi	Minum segelas air hangat dengan perasan air jeruk lemon
07:00	Sarapan	Jus buatan sendiri tanpa gula

09:00	Kudapan pagi	Buah potong segar
12:00	Makan siang	Menu protein: Ayam goreng, taoge goreng, lalapan sayur segar, sambal
16.00	Kudapan sore	Lumpia isi sayur, segelas susu kedelai ditambah satu sendok madu asli
20:00	Makan malam	Nasi merah, tumis brokoli, tahu-tempe goreng, lalapan sayur segar, sambal

2. Menu kurnag ideal

Tabel 2.2 Contoh Food Combining dengan Menu Kurang Ideal

Waktu	Keterangan	Menu
05:00	Bangun Pagi	Minum segelas air hangat dengan perasan air jeruk lemon/nipis
07:00	Sarapan	Buah potong
09:00	Kudapan pagi	Buah potong segar rujak kaki lima
12:00	Makan siang	Nasi, semur tahu, tempe goreng, irisan 1 buah timun segar
16.00	Kudapan sore	Satu porsi siomay terdiri dari tahu, kentang dan kol rebus, sedikit bumbu kacang
20:00	Makan malam	Ketoprak atau gado-gado tanpa telur

2.2 Android

Android adalah sistem operasi berbasis Linux yang dimodifikasi untuk perangkat bergerak (mobile devices) yang terdiri dari sistem operasi, middleware, dan aplikasi-aplikasi utama. Awalnya, Android dikembangkan oleh oleh Android Inc. Perusahaan ini kemudian dibeli oleh google pada tahun 2005. Sistem operasi Android kemudian diluncurkan bersamaan dengan dibentuknya organisasi Open Handset Alliance tahun 2007. Selain Google, nama-nama besar juga ikut serta dalam Open Handset Alliance, antara lain Motorola, Samsung, LG, Sony Ericsson, T-Mobile, Vodafone, Toshiba dan Intel.[6]

Sumber Gambar : https://www.freepik.com/free-vector/android-boot-logo_683826.htm

Gambar 2.1 Logo Android

1. Android SDK (Software Development Kit)

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi-netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan .

2. ADT (*Android Development Tools*)

Android Develpoment Tools adalah plugin yang di desain untuk IDE Eclipse yang memberikan kemudahan dalam mengembangkan aplikasi android dengan menggunakan IDE Eclipse. Dengan menggunakan ADT untuk Eclipse akan memudahkan dalam membuat aplikasi *project Android*, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya. Mengembangkan aplikasi Android dengan menggunakan ADT di Eclipse sangat dianjurkan dan sangat mudah untuk memulai mengembangkan aplikasi Android.

3. Arsitektur Android

Secara garis besar Arsitektur Android dapat di jelaskan dan di gambarkan sebagai berikut :

a. *Application and Widgets*

Application and Widgets adalah layer yang berhubungan dengan aplikasi-aplikasi inti yang berjalan pada Android OS. Seperti klien email, program SMS, kalender, browser, peta, kontak, dan lain-lain. Semua aplikasi ini dibuat dengan menggunakan bahasa Java.

b. *Applications Frameworks*

Applications Framework merupakan layer dimana para pembuat aplikasi menggunakan komponen-komponen yang ada di sini untuk membuat aplikasi mereka. Beberapa contoh komponen yang termasuk di dalam Applications Framework adalah sebagai berikut:

1) Views
2) Content Provider
3) Resource Manager
4) Notification Manager
5) Activity Manager

c. *Libraries*

Libraries merupakan layer tempat fitur-fitur android berada. Pada umumnya *libraries* diakses untuk menjalankan aplikasi. Beberapa *library* yang terdapat pada android diantaranya adalah *libraries* Media untuk memutar media video atau audio, *libraries* untuk menjalankan tampilan, *libraries* Graphic, *libraries* SQLite untuk dukungan database, dan masih banyak *library* lainnya.

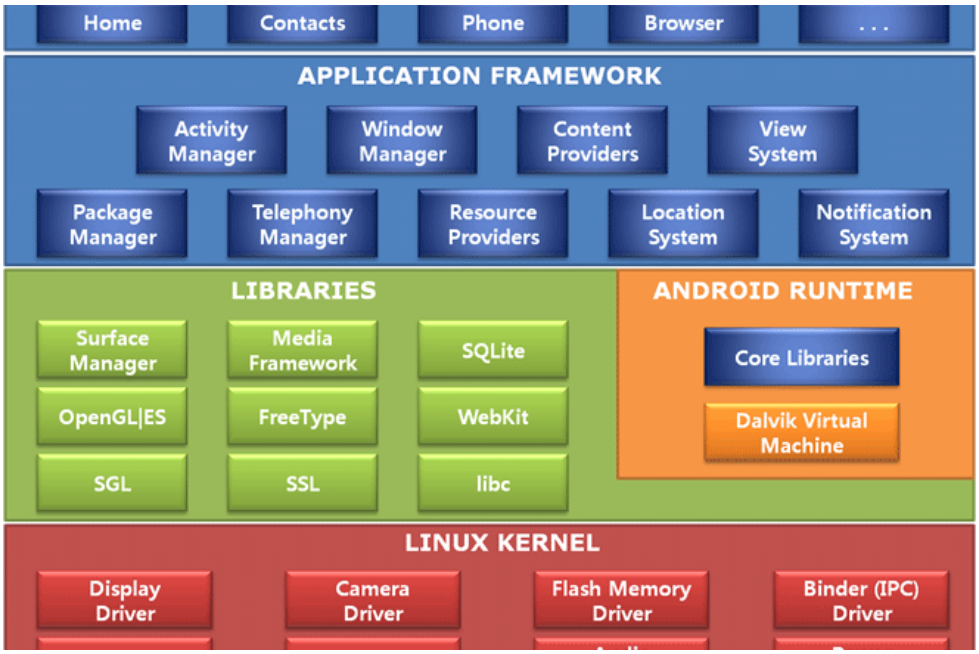
d. *Android Run Time*

Android *RunTime* merupakan layer yang membuat aplikasi android bisa dijalankan. Android RunTime dibagi menjadi dua bagian yaitu:

1) Core Libraries : berfungsi untuk menerjemahkan bahasa Java/C
2) Dalvik Virtual Machine : sebuah mesin virtual berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi pada Android secara efisien.

e. *Linux Kernel*

Linux Kernel adalah *layer* dimana inti dari *operating system* dari Android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem-sistem opersi Android lainnya.



Sumber Gambar : https://www.inguh.com/arsitektur-sistem-operasi-android/

Gambar 2.2 Arsitektur Android

4. Fundamental Aplikasi

Aplikasi Android ditulis dalam bahasa pemrograman Java. Kode Java dikompilasi bersama dengan data file *resource* yang dibutuhkan oleh aplikasi, dimana prosesnya di-*package* oleh *tools* yang dinamakan "*apt tools*" ke dalam paket Android sehingga menghasilkan file dengan ekstensi apk. Ada empat jenis komponen pada aplikasi Android yaitu :

a. *Activities*

b. Service

c. Broadcast Receiver

d. Content Provider

5. Versi Android

Android telah mengalami sejumlah pembaruan sejak pertama kali dirilis . Rata-rata, versi terbaru dari Android dirilis setiap 6 bulan. Tabel 1 menunjukkan beberapa jenis Android dan nama kodenya. Penamaan kode menggunakan nama makanan dan huruf depannya urut sesuai abjad.[7]

Tabel 2.3 API Android

Versi	Tanggal Rilis	Kode
1.0	23 September 2008	API Level 1
1.1	9 Februari 2009	API Level 2
1.5	30 April 2009	Cupcake (API Level 3)
1.6	15 September 2009	Donut (API Level 4)
2.0	26 Oktober 2009	Eclair (API Level 5)
2.0.1	3 Desember 2009	Eclair (API Level 6)
2.1	12 Januari 2010	Eclair (API Level 7)
2.2-2.2.3	20 Mei 2010	Froyo (API Level 8)
2.3–2.3.2	6 Desember 2010	Gingerbread (API Level 9)
2.3.3-2.3.7	9 Februari 2011	Gingerbread (API Level 10)
3.0	22 Februari 2011	Honeycomb (API Level 11)
3.1	10 Mei 2011	Honeycomb (API Level 12)
3.2	15 juli 2011	Honeycomb (API Level 13)
4.0–4.0.2	19 Oktober 2011	Ice Cream Sandwich (API Level 14)
4.0.3–4.0.4	16 Desember 2011	Ice Cream Sandwich (API Level 15)
4.1	9 Juli 2012	Jelly Bean (API Level 16)
4.2	13 November 2012	Jelly Bean (API Level 17)
4.3	24 Juli 2013	Jelly Bean (API Level 18)
4.4	31 Oktober 2013	Kitkat (API Level 19)
5.0	25 Juni 2015	Lollipop (API Level 21)
6.0	5 oktober 2015	Marshmallow (API Level 23)
7.0	22 Agustus, 2016	Nougat (API Level 24)
7.1-7.1.2	4 Oktober, 2016	Nougat (API Level 25)
8.0	21 Agustus 2017	Oreo (API Level 26)
9.0	6 Agustus 2018	Pie (API Level 28)

2.3 Android Studio

Android Studio adalah IDE resmi untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Di atas IntelliJ yang kuat *code editor* dan pengembang alat, Android Studio menawarkan lebih banyak fitur yang meningkatkan produktivitas Anda ketika membangun aplikasi Android, seperti :[8]

a. Sebuah *fleksibel* berbasis *Gradle* membangun sistem
b. Membangun varian dan beberapa generasi file APK
c. Kode template untuk membantu Anda membangun fitur aplikasi umum
d. Sebuah *layout editor* kaya dengan dukungan untuk *drag* dan *drop tema editing*
e. Alat *Lint* untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya
f. Kode menyusut dengan ProGuard dan sumber daya menyusut dengan Gradle
g. Built-in dukungan untuk Google *Cloud Platform*, sehingga mudah untuk mengintegrasikan Google *Cloud Messaging* dan App *Engine*.

1. Struktur Proyek

Setiap proyek di Android Studio berisi satu atau lebih modul dengan file kode sumber dan file sumber daya. Berbagai jenis modul meliputi :

a. Android app modules
b. Test modules
c. Library modules
d. App Engine modules

Secara *default*, Android Studio menampilkan file proyek Anda dalam tampilan proyek Android. Pandangan ini diselenggarakan oleh modul untuk menyediakan akses cepat ke file kunci sumber proyek Anda. Semua file membangun terlihat di tingkat atas di bawah *Script Gradle* dan setiap modul aplikasi mengandung tiga unsur berikut :

1. Manifests : file Manifest.
2. Java : Sumber file kode.
3. Res : file Sumber Daya.

Struktur proyek Android pada disk berbeda dari representasi pipih ini. Untuk melihat struktur file yang sebenarnya dari proyek, pilih Proyek dari Proyek *drop-*

down. Anda juga dapat menyesuaikan tampilan dari file proyek untuk fokus pada aspek-aspek tertentu dari pengembangan aplikasi Anda. Misalnya, memilih Masalah tampilan proyek Anda menampilkan link ke file sumber yang berisi setiap diakui coding dan sintaksis kesalahan, seperti hilang elemen XML tag penutup dalam *file layout*.

2. Versi Android Studio

   Versi Android Studio sebagai berikut:[9]

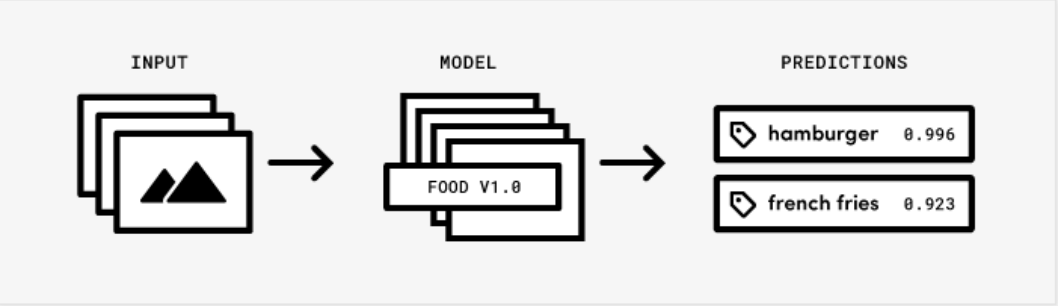
Tabel 2.4 Versi Android Studio

Versi	Tanggal Rilis
Android Studio v0.1.x	Mei 2013
Android Studio v0.2.x	Juli 2013
Android Studio v0.3.2	Oktober 2013
Android Studio v0.4.2	Januari 2014
Android Studio v0.4.6	Maret 2014
Android Studio v0.5.2	Mei 2014
Android Studio v0.8.0	Juni 2014
Android Studio v0.8.6	Agustus 2014
Android Studio v0.8.14	Oktober 2014
Android Studio v1.0	Desember 2014
Android Studio v1.0.1	Desember 2014
Android Studio v1.1.0	Februari 2015
Android Studio v1.2.0	April 2015
Android Studio v1.2.1	Mei 2015
Android Studio v1.2.2	Juni 2015
Android Studio v1.3.0	Juli 2015
Android Studio v1.3.1	Agustus 2015
Android Studio v1.3.2	Agustus 2015
Android Studio v1.4.0	September 2015
Android Studio v1.4.1	Oktober 2015
Android Studio v1.5.0	November 2015

Android Studio v1.5.1	Desember 2015
Android Studio v2.0.0	April 2016
Android Studio v2.1.0	April 2016
Android Studio v2.1.1	Mei 2016
Android Studio v2.1.2	Juni 2016
Android Studio v2.1.3	Agustus 2016
Android Studio v2.2	September 2016
Android Studio v2.2.1	Oktober 2016
Android Studio v2.2.2	Oktober 2016
Android Studio v2.2.3	Desember 2016
Android Studio v2.3	Maret 2017
Android Studio v2.3.1	April 2017
Android Studio v2.3.2	April 2017
Android Studio v2.3.3	Juni 2017
Android Studio v3.0	Oktober 2017

2.4 Clarifai API

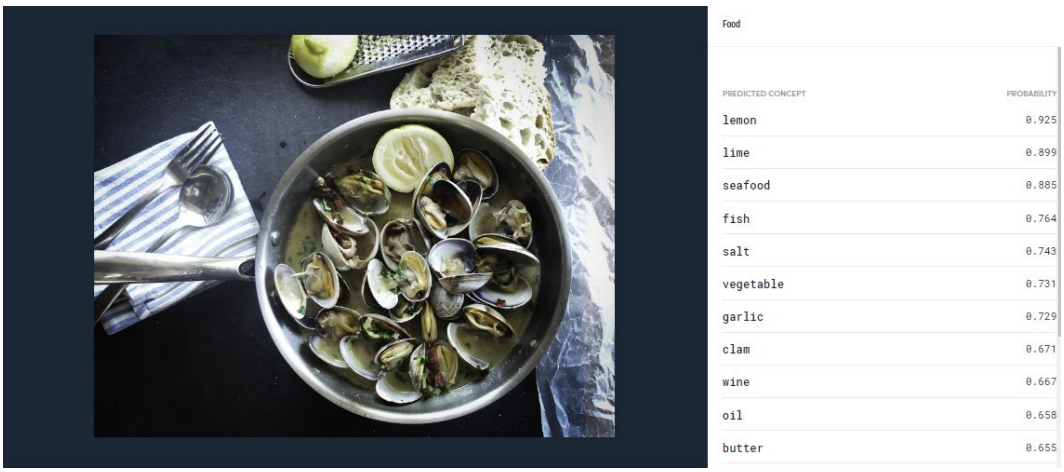
Clarifai adalah layanan untuk mengenali gambar dan video yang secara otomatis memberikan prediksi tentang apa yang ada di dalam gambar atau video beserta besaran probabilitasnya. [10] Konsep kerja dari clarifai dapat dilihat pada gambar 2.3.



Sumber Gambar : https://clarifai.com/developer/guide/

Gambar 2.3 Konsep Kerja Clarifai

Model yang digunakan pada penelitian ini adalah *food model*. *Food model* dapat mengenali lebih dari 1000 makanan dalam gambar dan dapat mengenali sampai ke level bahan makanan. Contohnya dapat dilihat pada 2.4

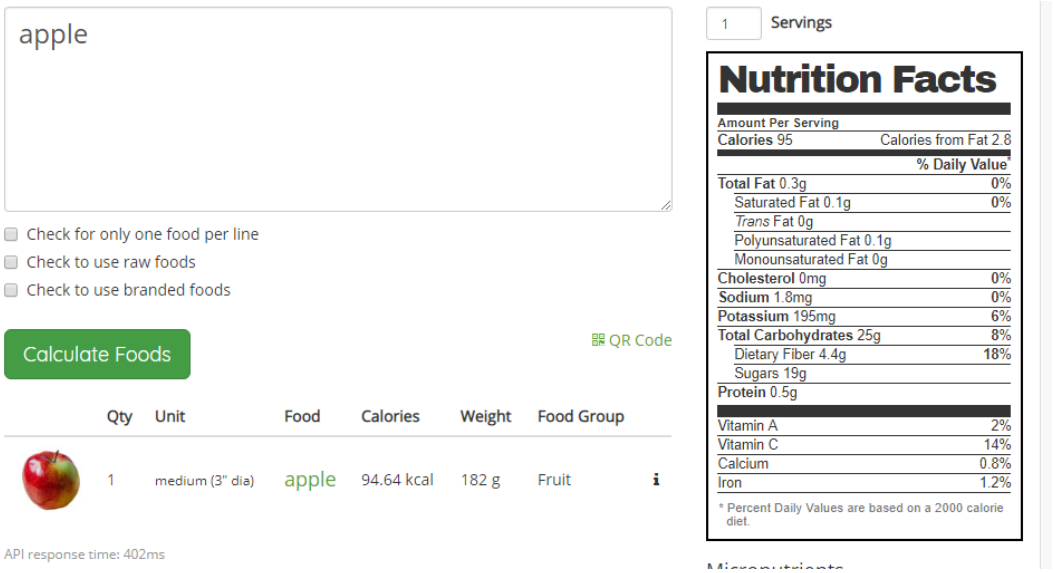


Gambar 2.4 Contoh *Food Model*

2.5 Nutritionix API

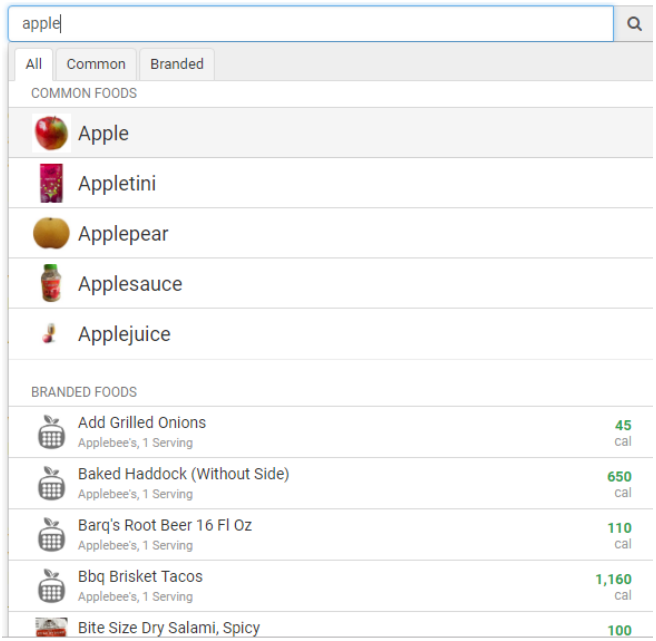
Nutritionix API adalah situs penyedia informasi nutrisi dari setiap bahan dan makanan. Ada 6 fitur yang terdapat pada nutritionix api, diantaranya: [11]

1 *Natural Language,* mengubah teks menjadi analisis nutrisi yang tepat dengan *state of the art*. Contoh dapat dilihat pada 2.5



Gambar 2.5 Contoh *Natural Language*

2 *Autocomplate Search,* dapat memprediksi inputan teks yang kita ketik pada kolom *search*. Contoh dapat dilihat pada 2.6



Gambar 2.6 Contoh *Autocomplate Search*

3 *Common Foods,* dimulai dengan mendaftarkan ke *United States Department of Agriculture* (USDA) dan berhasil memenuhi standar untuk pengumpulan makanan dan resep, selain dari USDA Nutritionix juga mengumpulkan ribuan jenis makanan dan resep internasional

4 *Branded Food,* memiliki database makanan terbesar dengan memiliki 568.000 bahan makanan yang dilengkapi barcode dan makanan yang berapa pada restaurant sebanyak 146.000.

5 *Dietitan Verified*, memiliki ahli diet untuk memverifikasi data dan prosedur API untuk memastikan keakuratan data nutrisi.

6 *Restaurant Geolocation,* terdapat 202.495 restoran yang dapat dimanfaatkan untuk melihat lokasi restoran berdasarkan data nutrisi yang tersedia.

2.6 Food2Fork API

Food2fork API adalah situs yang menyediakan resep dari berbagai dunia yang terus menambahkan basis data setiap harinya. Dengan memiliki fitur yang mudah digunakan oleh pengguna. Food2fork memiliki fitur pencarian berdasarkan bahan, dan sistem peringkat berdasarkan popularitas di sosial media. [12]Contoh menu dapat dilihat pada 2.7

Gambar 2.7 Contoh Resep pada Food2Fork

2.7 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di *Sun Microsystems* saat ini merupakan bagian dari *Oracle* dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam *p-code* (*bytecode*) dan dapat dijalankan pada berbagai Mesin *Virtual Java* (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula

dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web .[13]

Sumber Gambar : http://www.vectorsland.com/vector/java-eps-logo-99090.html

Gambar 2.8 Logo Java

1. Sejarah Perkembangan Java

Proyek Java dimulai pada tahun 1991, ketika sejumlah insinyur perusahaan Sun yang dimotori oleh James Gosling mempunyai keinginan untuk mendesain sebuah bahasa komputer kecil yang dapat dipergunakan untuk peralatan konsumen seperti kotak tombol saluran TV. Proyek ini kemudian diberi nama sandi Green.

Keharusan untuk membuat bahasa yang kecil, dan kode yang ketat mendorong mereka untuk menghidupkan kembali model yang pernah dicoba oleh bahasa UCSD Pascal, yaitu mendesain sebuah bahasa yang portable yang menghasilkan kode intermediate. Kode intermediate ini kemudian dapat digunakan pada banyak komputer yang interpreternya telah disesuaikan.

Karena orang-orang Sun memiliki latar belakang sebagai pemakai unix sehingga mereka lebih menggunakan C++ sebagai basis bahasa pemrograman mereka, maka mereka secara khusus mengembangkan bahasa yang berorientasi objek bukan berorientasi prosedur. Seperti yang dikatakan Gosling ”Secara keseluruhan, bahasa hanyalah sarana, bukan merupakan tujuan akhir”. Dan Gosling memutuskan menyebut bahasanya dengan nama “Oak” (diambil dari

nama pohon yang tumbuh tepat diluar jendela kantornya di Sun), tetapi kemudian nama Oak diubah menjadi java, karena nama Oak merupakan nama bahasa komputer yang sudah ada sebelumnya.

Pada tahun 1994 sebagian besar orang menggunakan mosaic, browser web yang tidak diperdagangkan yang berasal dari pusat Supercomputing Universitas Illinois pada tahun 1993.(Mosaic sebagian ditulis oleh Marc Andreessen dengan bayaran $6.85 per jam, sebagai mahasiswa yang melakukan studi praktek. Di kemudian hari ia meraih ketenaran sebagai salah seorang pendiri dan pemimpin teknologi di netscape)

Browser yang sesungguhnya dibangun oleh Patrick Naughton dan Jonathan Payne dan berkembang ke dalam browser HotJava yang kita miliki saat ini. Browser HotJava ditulis dalam Java untuk menunjukkan kemampuan Java. Tetapi para pembuat juga memiliki ide tentang suatu kekuatan yang saat ini disebut dengan applet, sehingga mereka membuat browser yang mampu penerjemahkan kode byte tingkat menengah. “Teknologi yang Terbukti” ini diperlihatkan pada SunWorld ‟95 pada tanggal 23 mei 1995, yang mengilhami keranjingan terhadap Java terus berlanjut.

2. Kelebihan Java

Kelebihan dari java adalah sebagai berikut :

a) **Multiplatform.** Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform*/sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin/*bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan.
b) **OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek),** OOP (Object Oriented Programming) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, nah objek itu sendiri merupakan gabungan dari beberapa objek

yang lebih kecil lagi. Saya ambil contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dll. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

c) **Perpustakaan Kelas Yang Lengkap**, Java terkenal dengan kelengkapan library/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

d) **Bergaya C++**, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.

e) **Pengumpulan Sampah Otomatis**, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

3. Kekurangan Java

Kekurangan dari java adalah sebagai berikut :

a) **Tulis sekali, jalankan di mana saja** - Masih ada beberapa hal yang tidak kompatibel antara platform satu dengan platform lain. Untuk J2SE, misalnya SWT-AWT bridge yang sampai sekarang tidak berfungsi pada Mac OS X.

b) **Mudah didekompilasi**. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan bytecode yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di-reverse-engineer.

Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.8 XML (*Extensible Markup Language*)

XML (*Extensible Markup Language*) adalah bahasa markup untuk keperluan umum yang disarankan oleh W3C untuk membuat dokumen markup keperluan pertukaran data antar sistem yang beraneka ragam. XML merupakan kelanjutan dari HTML (*HyperText Markup Language*) yang merupakan bahasa standar untuk melacak Internet.[14]

```
<?xml version="1.0" encoding="utf-8"?>
<katalog>
    <buku category="WEB">
        <judul bahasa="en">Learning XML</judul>
        <pengarang>Erik T. Ray</pengarang>
        <tahun>2003</tahun>
        <harga>39.95</harga>
    </buku>
    <buku>
        ......
        ......
    </buku>
</katalog>
```

Sumber Gambar : https://www.brebeshost.com/pengenalan-xml/amp/

Gambar 2.9 Contoh XLM

1. Pengenalan XML

XML di desain untuk mempu menyimpan data secara ringkas dan mudah diatur. Kata kunci utama XML adalah data (jamak dari datum) yang jika diolah bisa memberikan informasi. XML menyediakan suatu cara terstandarisasi namun bisa dimodifikasi untuk menggambarkan isi dari dokumen. Dengan sendirinya, XML dapat digunakan untuk menggambarkan sembarang *view database*, tetapi dengan suatu cara yang standar.

2. Tipe XML

XML memiliki tiga tipe file, yaitu :

a. XML, merupakan standar format dari struktur berkas (file).
b. XSL, merupakan standar untuk memodifikasi data yang diimpor atau diekspor.
c. XSD, merupakan standar yang mendefinisikan struktur database dalam XML.

2.9 JSON (*JavaScript Object Notation*)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. [15] Kelebihan-kelebihan dari JSON adalah:

a. Format Penulisan : Untuk merepresentasikan sebuah struktur data yang rumit dan berbentuk hirarkis penulisan JSON relatif lebih terstruktur dan mudah.
b. Ukuran karakter yang dibutuhkan JSON lebih kecil dibandingkan XML untuk data yang sama.
c. Proses parsing merupakan proses pengenalan token atau bagian-bagian kecil dalam rangkaian dokumen XML/JSON.

JSON terbuat dari dua struktur :

a. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.

b. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).


```
{"list_event":
  [
    {"eventID":"1",
    "judul":"Pengajian Akbar menyambut ramadhan",
    "tanggal":"2014-05-19",
    "jam":"20:00",
    "lokasi":"Masjid Kampus UGM"
    },
    { "eventID":"2",
    "judul":"Bedah buku kesesatan syiah",
    "tanggal":"2014-05-21",
    "jam":"09:00",
    "lokasi":"Masjid Kampus UII "
    }
  ]
}
```


Sumber Gambar : http://www.candra.web.id/penjelasan-struktur-data-json/

Gambar 2.10 Contoh JSON

2.10 API (*Application Programming Interface*)

API atau Application Programming Interface merupakan suatu dokumentasi yang terdiri dari interface, fungsi, kelas, struktur dan sebagainya untuk membangun sebuah perangkat lunak. Dengan adanya API ini, maka memudahkan programmer untuk "membongkar" suatu software untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya yang memungkinkan programmer menggunakan sistem function. Proses ini dikelola melalui operating system. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya dapat saling berhubungan dan berinteraksi. Bahasa pemrograman yang digunakan oleh Google Maps yang terdiri dari HTML, Javascript dan AJAX serta XML, memungkinkan untuk menampilkan peta Google Maps di website lain.

Keuntungan dengan menggunakan API adalah sebagai berikut:

a. Portabilitas.
b. *Developer* yang menggunakan API dapat menjalankan programnya dalam sistem operasi mana saja asalkan sudah terinstal API tersebut.
c. Lebih Mudah Dimengerti

API menggunakan bahasa yang lebih terstruktur dan mudah dimengerti daripada bahasa *system call*. Hal ini sangat penting dalam hal editing dan pengembangan.
Cara menggunakan API :

a. Dilakukan dengan mengimpor *package*/kelas.
b. Ada beberapa kelas bernama sama dipackage yang berbeda, yaitu:
1) Import salah satu dan gunakan nama lengkap untuk yang lain.
2) Gunakan nama lengkap semua kelas

Kebanyakan Sistem Operasi seperti Windows, menyediakan fasilitas API sehingga *programmer* dapat melakukan aktivitas programming dengan lebih konsisten. Meskipun API didesain untuk programer, namun API juga baik untuk user karena setidaknya dapat menjamin bahwa program tersebut memiliki interface yang sama, sehingga lebih mudah untuk dipelajari.

2.11 OOAD (*Object Oriented Analysis and Design*)

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. Pengertian "berorientasi objek" berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. 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 kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

1. OOA (*Object Oriented Analysis*)

OOA mempelajari permasalahan dengan menspesifikasikannya atau mengobservasi permasalahn tersebut dengan menggunakan metode berorientasi objek. Biasanya analisa sistem dimulai dengan adanya dokumen permintaan (requirement) yang diperoleh dari semua pihak yang berkepentingan. (Misal:

klien,developer, pakar, dan lain-lain). Dokumen permintaan memiliki 2 fungsi yaitu : memformulasikan kebutuhan klien dan membuat suatu daftar tugas. Analisis berorientasi obyek (OOA) melihat pada domain masalah, dengan tujuan untuk memproduksi sebuah model konseptual informasi yang ada di daerah yang sedang dianalisis. Model analisis tidak mempertimbangkan kendala-kendala pelaksanaan apapun yang mungkin ada, seperti konkurensi, distribusi, ketekunan, atau bagaimana sistem harus dibangun. Kendala pelaksanaan ditangani selama desain berorientasi objek (OOD).

Sumber-sumber untuk analisis dapat persyaratan tertulis pernyataan, dokumen visi yang formal, wawancara dengan stakeholder atau pihak yang berkepentingan lainnya. Sebuah sistem dapat dibagi menjadi beberapa domain, yang mewakili bisnis yang berbeda, teknologi, atau bidang yang diminati, masing-masing dianalisis secara terpisah.

Hasil analisis berorientasi objek adalah deskripsi dari apa sistem secara fungsional diperlukan untuk melakukan, dalam bentuk sebuah model konseptual. Itu biasanya akan disajikan sebagai seperangkat menggunakan kasus, satu atau lebih UML diagram kelas, dan sejumlah diagram interaksi. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan pelanggan.

UML (Unified Modeling Language) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (Object-Oriented). UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software. Unified Model Language (UML) adalah bahasa universal untuk memvisualisasikan grafis model yang tepat, menetapkan model yang tepat, lengkap, dan tidak ambigu untuk mengampil semua keputusan penting dalam analisis, desain dan implementasi. membangun model yang dapat dihubungkan langsung dengan bahasa pemrograman. mendokumentasikan semua

informasi yang dikumpulkan oleh tim sehingga memungkinkan untuk berbagi informasi.

2. OOD (*Object Oriented Design*)

Object Oriented Design adalah metode untuk mengarahkan arsitektur perangkat lunak yang didasarkan pada manipulasi objek-objek sistem atau subsistem. OOD mengubah model konseptual yang dihasilkan dalam analisis berorientasi objek, memperhitungkan kendala yang dipaksakan oleh arsitektur yang dipilih dan setiap non fungsional teknologi atau lingkungan kendala.

2.12 OOP (*Object Oriented Programming*)

Pemrograman berorientasi objek adalah metode implementasi dimana program diorganisasikan sebagai kumpulan objek yang bekerja sama, masing-masing objek merepresentasikan instan dari kelas, dan kelas-kelas itu anggota suatu hirarki kelas kelas yang disatukan lewat keterhubungan pewarisan. Tiga aspek penting dalam pemrograman berorientasi objek:

1. Menggunakan objek-objek bukan algoritma-algortima sebagai blok-blok bangunan logika dasar (hirarki "*part of"*).
2. Masing-masing objek adalah instal satu kelas.
3. Kelas-kelas saling berhubungan lewat keterhubungan pewarisan ("*is a"*).

Bahasa pemrograman berorientasi objek umumnya sama dalam mendukung konsep-konsep dasar pendekatan berorientasi objek berikut:

1. Objek, kombinasi data dan operasi
2. *Polimorphism* saat jalan
3. Pewarisan

Tujuan akhir dari mempelajari pemrograman berorientasi objek adalah kemampuan untuk membangun program menggunakan konsep berorientasi objek. Sifat utama dari konsep berorientasi objek adalah *reusable* atau guna ulang, yaitu penggunaan program menajdi lebih efektif karena tidak terjadi pengulangan penulisan kode program.

2.13 UML (*Unified Modelling Language*)

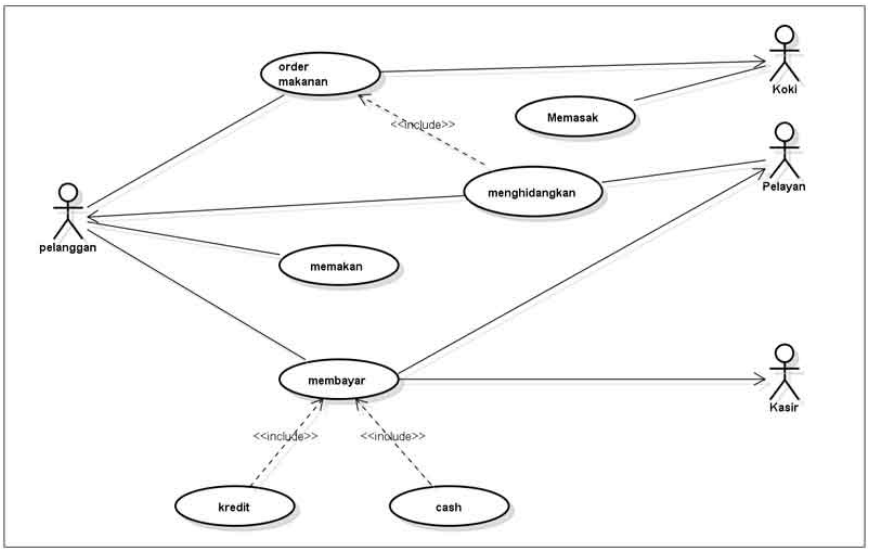
Unified Modelling Language (UML) adalah bahasa grafis untuk mendokumentasi, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemaduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh kakas-kakas yang diintegrasikan lewat XML (XMI). Standar UML dikelola oleh OMG (*Object Management Group*) . UML adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak-artifak dari sistem. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu : *Grady Booch* OOD (*Object Oriented Design*), James Rumbaugh OMT (*Object Modeling Technique*) dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*).[16]

1. *Use Case Diagram*

Use case Diagram merupakan salah satu diagram untuk memodelkan aspek perilakau sistem. Masing-masing diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. Diagram *use case* adalah penting untuk memvisualisasikan, menspesifikasikan, dan mendokumentasikan kebutuhan perilaku sistem. Diagram-diagram *use case* merupakan pusat pemodelan perilaku sistem, subsistem, dan kelas. Diagram *use case* digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Ada empat hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan use case :

a. Sistem yaitu sesuatu yang hendak kita bangun.
b. Relasi adalah relasi antara aktor dengan *use case*.
c. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
d. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut contoh *Use Case Diagram* dapat dilihat pada gambar 2.11



Sumber Gambar : http://www.pengertianku.net/2015/09/pengertian-uml-dan-jenis-jenisnya-serta-contoh-diagramnya.html

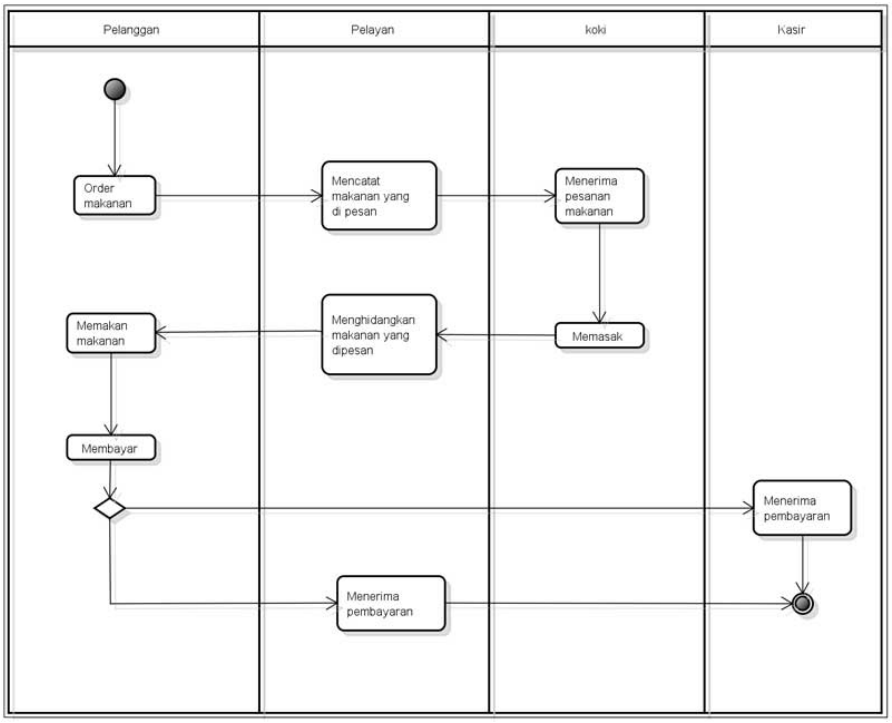
Gambar 2.11 Contoh *Use Case Diagram*

2. *Activity Diagram*

Diagram aktivitas adalah diagram *flowchart* yang diperluas yang menunjukan aliran kendali satu aktivitas ke aktivitas lain di sistem. Diagram aktivitas ini digunakan untuk memodelkan aspek dinamis sistem. Diagram aktivitas mendeskripsikan aksi-aksi dan hasilnya. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
d. Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut contoh *Activity Diagram* yang dapat dilihat pada gambar 2.12.



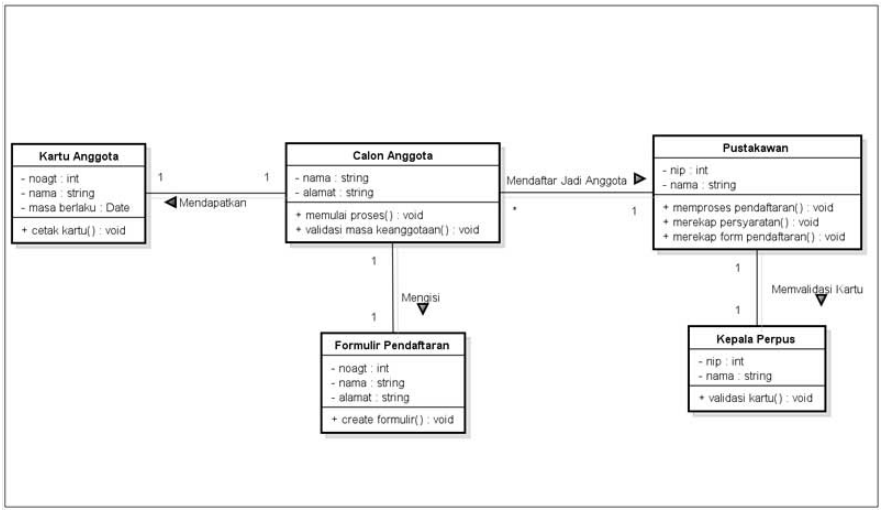
Sumber Gambar: http://www.pengertianku.net/2015/09/pengertian-uml-dan-jenis-jenisnya-serta-contoh-diagramnya.html

Gambar 2.12 Contoh *Activity Diagram*

3. *Class Diagram*

Class diagram merupakan diagram yang selalu ada di pemodelan system berorientasi objek. *Class* diagram menunjukkan hubungan antar class dalam system yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai satu tujuan. Kelas pada kelas diagram terdiri dari 3 bagian utama yaitu nama kelas, isi *property* dari kelas beserta metode yang ada pada kelas tersebut. Kelas juga memiliki jenis-jenis hubungan seperti asosiatif, dependensi, agregasi, komposisi, spesifikasi dan generalisasi. Hubungan ini digunakan untuk menggambarkan bagaimana hubungan dan interaksi yang terjadi antar kelas. Masing-masing komponen penyusun kelas memiliki hak akses seperti *public*, *private* dan *protected*.

Berikut contoh *Class Diagram* dapat dilihat pada gambar 2.13.



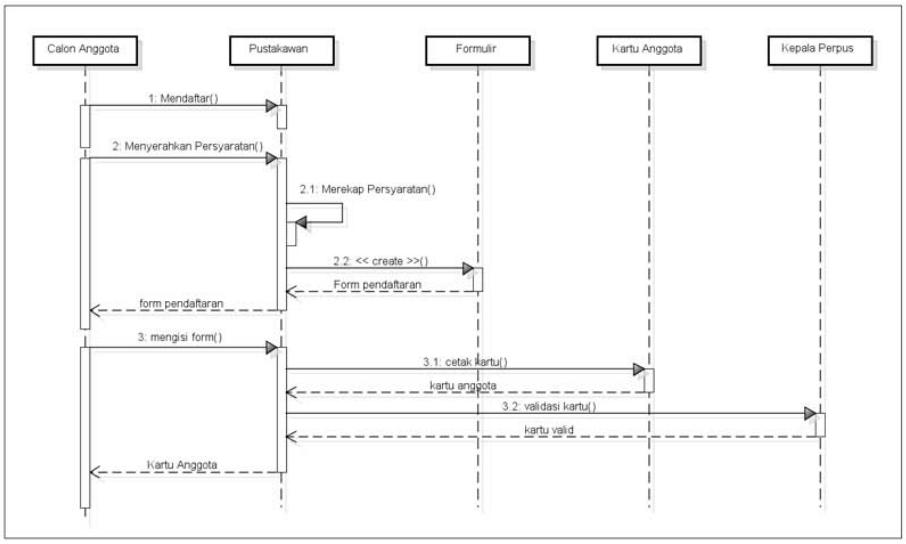
Sumber Gambar: http://www.pengertianku.net/2015/09/pengertian-uml-dan-jenis-jenisnya-serta-contoh-diagramnya.html

Gambar 2.13 Contoh Class Diagram

4. Sequence Diagram

Sequence Diagram menunjukan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada sisi basis keberurutan waktu dari pesan-pesan yang terjadi. Diagram sekuen menunjukan objek sebagai garis vertikal dan tiap kejadian sebagai panah horisontal dari objek pengirim ke objek penerima. Waktu berlalu dari atas ke bawah dengan lama waktu tidak relevan. Diagram ini hanya menunjukan barisan kejadian, bukan perwaktuan nyata. Kecuali untuk sistem waktu nyata yang mengharuskan konstrain barisan terjadi.

Berikut Contoh *Sequence Diagram* dapat dilihat pada gambar 2.14



Sumber Gambar: http://www.pengertianku.net/2015/09/pengertian-uml-dan-jenis-jenisnya-serta-contoh-diagramnya.html

Gambar 2.14 Contoh Sequence Diagram