BAB 2
LANDASAN TEORI

2.1 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan aplikasi *Penstagram.*

2.1.1 M-Commerce

M-commerce didefinisikan sebagai "semua kegiatan yang berhubungan dengan (potensial) transaksi komersial yang dilakukan melalui jaringan komunikasi dengan antarmuka perangkat nirkabel" [4]. Tranasaksi *m-commerce,* melibatkan perpindahan kepemilikan atau hak untuk menggunakan barang dan jasa, yang dimulai dan / atau diselesaikan dengan menggunakan akses mobile ke dimediasi jaringan komputer dengan bantuan perangkat elektronik. *M-commerce* merupakan bagian dari *E-commerce*, yang spesifik pada proses transaksi yang dilakukan secara elektronik melalui jaringan *internet* dan *seluler*. Selain dilakukan secara remote melalui jaringan seluler, transaksi *m-commerce* juga dapat dilakukan secara lokal server dengan menggunakan koneksi *Bluetooth, InfraRed* atau *RFID*.

2.1.2 Karakteristik M-Commerce

M-commerce memiliki karakteristik yang tidak dimiliki teknologi pendahulunya. Karakteristik M-Commerce adalah sebagai berikut [5]:

1. *Ubiquity,* aplikasi *M-commerce* dapat digunakan dimana pun pengguna berada dan dalam situasi apa pun. Dengan kata lain layanan *M-commerce* tersedia kapan pun dan dimana pun dibutuhkan.
2. *Reachability,* melalui perangkat handset pengguna dapat bertransaksi dan berkomunikasi dengan pihak lain yang tidak terjangkau karena perbedaan ruang dan waktu.

3.*Localization, M-commerce* memberikan layanan aplikasi informasi lokasi yang dibutuhkan oleh pengguna berdasarkan lokasi fisik penggunanya

4.*Personalization*, aplikasi *M-commerce* hanya dapat digunakan oleh pengguna yang memiliki handset karena adanya perbedaan penomoran atau nomor unik yang dikeluarkan oleh penyelenggara telekomunikasi.

5.*Dissemination,* handset memungkinkan pengguna menerima dan menyebarluaskan informasi secara cepat dan kini (saat itu juga).

2.1.3 Web Service

Web service adalah salah satu bentuk sistem perangkat lunak yang didesain untuk mendukung interaksi mesin-ke-mesin melalui jaringan.*Web service* memiliki *interface* yang dideskripsikan dalam format yang dapat dibaca oleh mesin. Sistem-sistem lainnya berinteraksi dengan *web service* menggunakan pesan *SOAP* yang umumnya dikirim melalui *HTTP* dalam bentuk *XML*. [7]

Definisi diatas diberikan oleh *World Wide Web Consortium(W3C)* yang merupakan badan yang menciptakan dan mengembangkan standar *web service*. Tetapi secara umum, *web service* tidak terbatas hanya pada standar *SOAP* saja. Salah satu pustaka yang mengulas lengkap tentang *web service* menyebutkan definisi yang lebih umum: Semua *web service* setidaknya menggunakan *HTTP* dan format pertukaran data standar berupa *XML*, *JSON*, atau media lain. Selain itu, *web service* menggunakan *HTTP* dalam dua cara yang berbeda yaitu sebagai protokol standar untuk menentukan perilaku standar pelayanan serta sebagai media transportasi untuk menyampaikan data.

2.1.4 JSON

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generated*) oleh komputer [6] . *JSON* terbuat dari dua struktur yaitu:

1. 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.
2. Daftar nilai terurutkan (*an* Donasi*ed list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*Sequence*).

JSON menggunakan bentuk sebagai berikut:

1. Objek

Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

3. Nilai

Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.

4. String

String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.

5. Angka

Format oktal dan heksadesimal tidak digunakan.

2.1.5 Instagram

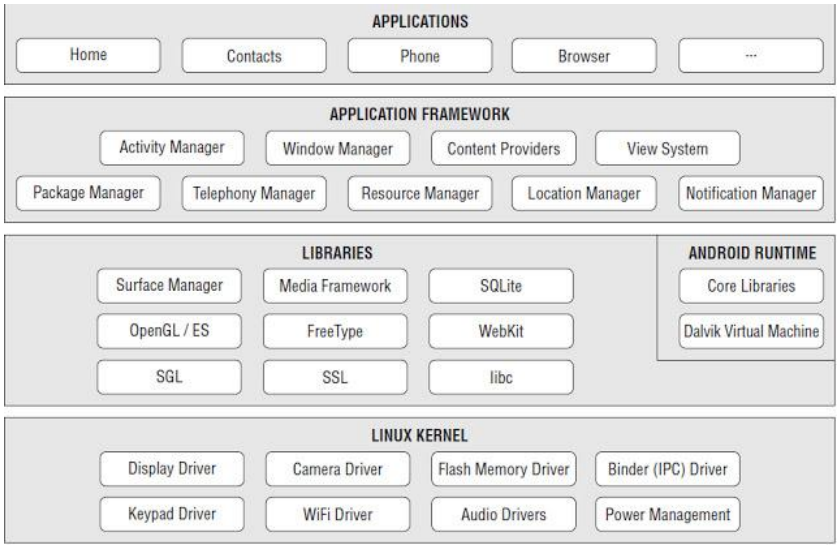
Instagram adalah sebuah aplikasi berbagi foto yang memungkinkan pengguna mengambil foto, menerapkan filter digital, dan membagikannya ke berbagai layanan jejaring sosial, termasuk milik *Instagram* sendiri [7]. Satu fitur yang unik di *Instagram* adalah memotong foto menjadi bentuk persegi, sehingga terlihat seperti hasil kamera Kodak *Instamatic* dan *Polaroid*. Hal ini berbeda dengan rasio aspek 4:3 yang umum digunakan oleh kamera pada peralatan bergerak.

Instagram dapat digunakan di *iPhone, iPad atau iPod Touch* versi apapun dengan sistem operasi *iOS* 3.1.2 atau yang terbaru dan telepon kamera *android* apapun dengan sistem operasi 2.2 (Froyo) atau yang terbaru. Aplikasi ini tersebar melalui *Apple App Store* dan *Google Play*.

2.1.6 Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux yang mencakup sistem operasi. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia [8].

Android memiliki arsitektur sistem bisa dilihat pada gambar 2-1 :



Gambar 2-1 Arsitektur android

Fitur-fitur yang tersedia pada platform Android saat ini antara lain :

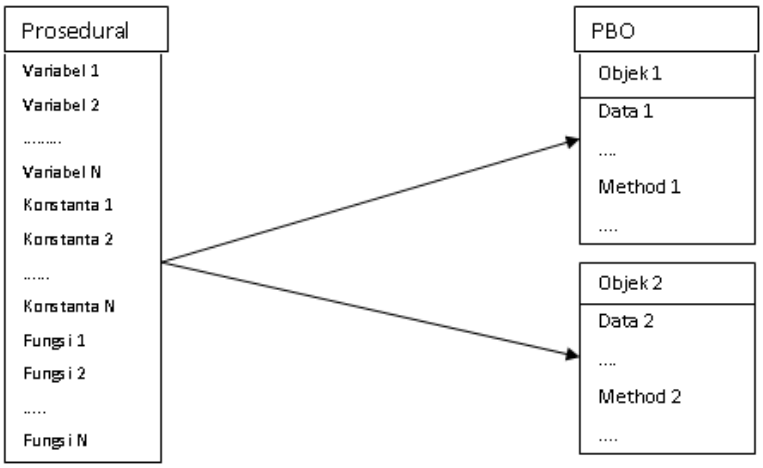
1. *Framework* aplikasi yang mendukung penggantian komponen dan *reusable*.
2. Mesin *virtual Dalvik* berjalan diatas *Linux kernel* dan dioptimalkan untuk perangkat *mobile*.
3. *Integrated browser* berdasarkan *open source engine WebKit*.
4. *Grafis* yang dioptimalkan dan didukung oleh *library grafis 2D* yang terkostumisasi, *grafis 3D* berdasarkan spesifikasi *openGL ES 1,0* (Opsional akselerasi hardware).
5. *SQLite* untuk penyimpanan data
6. Media Support yang mendukung audio, video dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM Telephony (tergantung hardware).
7. *Bluetooth, EDGE, 3D* dan *WiFi* (tergantung hardware).
8. Kamera, GPS, kompas, dan accelerometer (tergantung hardware).
9. Lingkungan *Development* yang lengkap dan kaya termasuk perangkat emulator, *tools* untuk *debugging*, profil dan kinerja memori, dan plugin untuk *IDE eclipse*.

2.1.7 Pemrograman Berorientasi Objek (*Object Oriented Programing*)

Pemrograman berorientasi objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan titik kulminasi dalam menemukan cara yang efektif dalam membangun sistem dan menajsi metode yang paling banyak dipakai oleh para pengembang perangakat lunak saat ini. Orientasi objek merupakan teknik pemodelan sistem riil yang berbasis objek.Inti dari konsep ini adalah objek yang merupakan model dari sistem nyata.

2.1.8 Perbedaan Pemrograman Berorieantasi Objek dengan Prosedural

Pada model pemrograman tradisional atau procedural (*Process-Oriented Model*), semua data dan kode digabung menjadi satu bagian dalam satu program. Untuk program-program sederhana yang membutuhkan hanya beberapa buah baris kode, pengguanaan model ini tentu tidak begitu menjadi masalah. Permasalah baru akan timbul pada saat program tersebut berkembang kearah yang lebih besar dan kompleks, yang mungkin akan membutuhkan ratusan atau bahkan ribuan kode. Alasan inilah yang menjadi benih diciptakannya model pemrograman baru, yaitu model pemrograman berorientasi objek (*Object-Oriented Mode*l). Pada model ini, komponen-komponen penyusun program secara konseptual akan dipecah menjadi bagian-bagian tersendiri yang disebut dengan objek. Berikut ini gambar yang mengilustrasikan perbedaan yang terdapat antara pemrograman prosedural dan pemrograman berorientasi objek.



Gambar 2-2 Pemograman prosedural dan berorientasi objek

Seperti yang dapat anda lihat pada gambar di atas, dalam pemrograman berorientasi objek, setiap objek akan memiliki data (sifat, berupa variabel maupun konstanta) dan *method* (perillaku atau kemampuan melakukan sesuatu, berupa fungsi). Jadi, objek dapat di definisikan sebagai sesuatu entitas yang memiliki data dan *method*. Sebagai contoh, manusia adalah objek yang memiliki data-data (nama, jenis kelamin, tinggi badan, berat badan, dan sebagainya) dan juga *method*(misalnya : cara bicara, cara berjalan, cara marah, dan sebagainya).

2.1.9 Karakteristik Pemrograman Berorientasi Objek

Adapun karakteristik yang dimiliki pemrograman berorientasi objek yaitu :

a. *Objects*

   Objek adalah entitas yang memiliki atribut, karakter (*behaviour*) dan kadang kala disertai kondisi (state).

b. *Class*

   Kelas adalah penggambaran satu objek yang memiliki atribut dan behavior yang sama. Kelas mirip dengan tipe data pada pemrograman non-objek, tapi lebih komperehensif karena terdapat struktur sekaligus karakteristiknya.Kita dapat membentuk kelas baru yang spesifik dari kelas *general*-nya.Kelas dan objek merupakan jantung dari pemrograman berorientasi objek.

c. *Abstraction*

Abstraksi adalah pengabstrakan atau penyembunyian kerumitan dari suatu proses. Sebagai contoh, orang tidak perlu berpikir bahwa mobil adalah kumpulan dari puluhan atau ribuan bagian-bagian kecil yaitu mesin, kabel, baut, roda, dan sebagainya. Orang hanya perlu berpikir bahwa mobil adalah sebuah objek yang telah memiliki perilaku spesifik, yang dapat digunakan sebagai alat treansportasi, sehingga orang hanya perlu menggunakannya tanpa harus mengetahui kerumitan proses yang terdapat pada mobil tersebut.

d. *Encapsulation*

Ciri penting lainnya dari OOP adalah *encapsulation*. *Encapsulation* adalah sebuah proses dimana tidak ada akses langsung ke data yang diberikan,bahkan tersembunyi. Jika seseorang ingin mendapat data, maka ia harus berinteraksi dengan objek perantara yang bertanggung jawab atas data tersebut.

e. *Polymorphism*

Polymorphism atau polimorfisme adalah kemampuan dua buah objek berbeda untuk merespon pesan permintaan yang sama dalam suatu cara yang unik. Contohnya, fungsi cetak akan menghasilkan keluaran berbeda ketika dijalankan pada mesin pencetak dan layar monitor. Dalam OOP, penerapan tipe *polymorphism* melalui proses yang disebut *overloading*. Hal tersebut mengizinkan pengimplementasian metode yang berbeda pada sebuah objek yang mempunyai nama sama.

f. *Inheritance*

Penggunaan *inheritance* dalam OOP untuk mengklasifikasikan objek dalam program sesuai karakteristik umum dan fungsinya. Hal ini sama dengan konsep pewarisan sifat pada manusia. Jika manusia memiliki keturunan, maka beberapa sifat dari orang tuanya terkadang menurun pada anaknya. Analogi tersebut kemudian berlaku pada OOP dalam *inheritance* ini. Konsep ini kemudian menjadikan pemrograman lebih mudah karena memungkinkan pemrogram untuk mengkombinasikan karakteristik umum kedalam objek orang tua (*superclass*) dan mewariskan karakteristik ini ke objek anaknya (*subclass*).

2.1.10 Unified Modeling Language (UML)

UML singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standar. (*Chonoles*, 2003 :bab 1) mengatakan sebagai bahasa, berarti UML memiliki sintaks dan semantik [9]. Ketika kita membuat model menggunakan konsep *UML* ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada.

1.Use Case

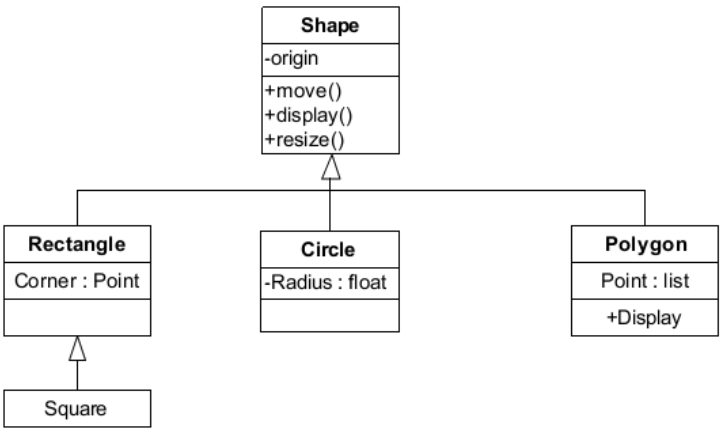
Menurut *(pilone,2005:bab 7.1)* Use Case menggambarkan fungsi tertentu dalam suatu sistem berupa komponen, kejadian atau kelas. Sedangkan *(Whitten, 2004 :258)* mengartikan use case sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario), baik terotomatisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal. contoh symbol usecase dapat dilihat pada gambar 2-3:



Gambar 2-3 simbol usecase

2.Class Diagram

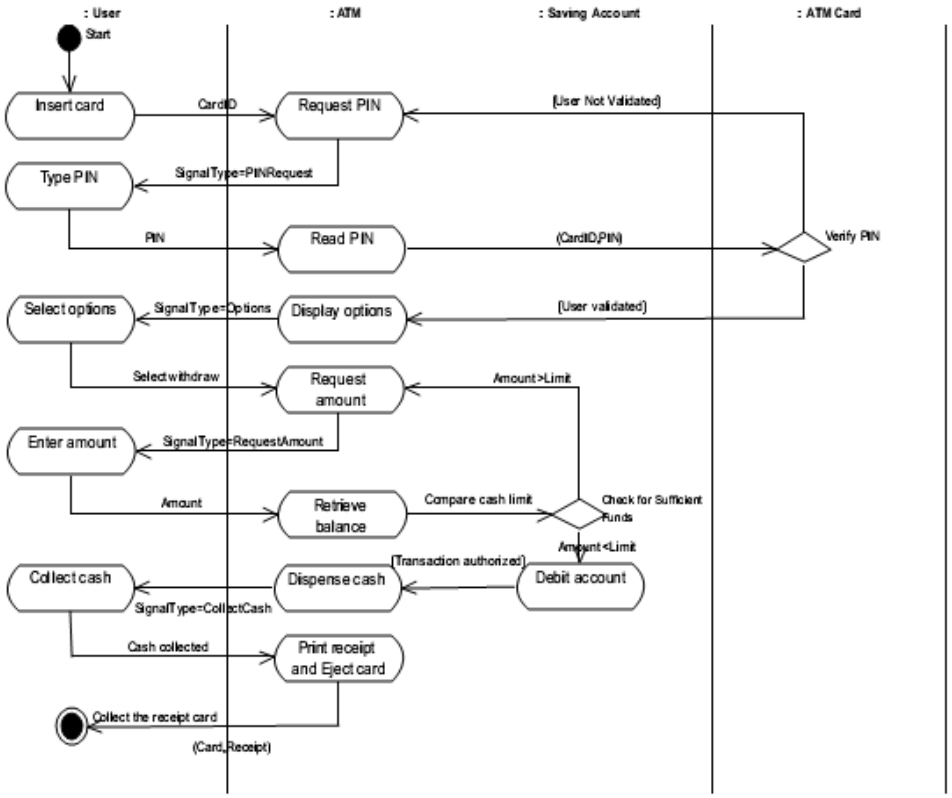
Diagram kelas adalah inti dari proses pemodelan objek. Baik *forward engineering* maupun reverse engineering memanfaatkan diagram ini. *Forward engineering* adalah proses perubahan model menjadi kode program sedangkan *reverse engineering* sebaliknya merubah kode program menjadi model. Contoh class diagram dapat dilihat pada gambar 2-4:



Gambar 2-4 diagram class

3.Activity Diagram

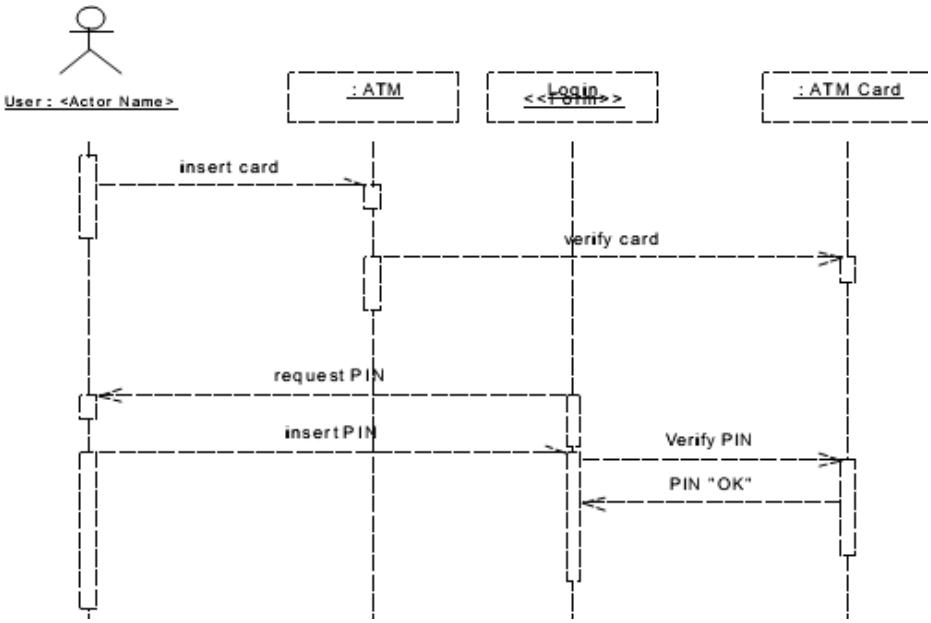
Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem itu dirakit.Diagram ini tidak hanya memodelkan software melainkan memodelkan model bisnis juga. Diagram aktivitas sistem dalam bentuk kumpulan aksi – aksi. Contoh activity diagram dapat dilihat pada gambar 2-5 :



Gambar 2-5 activity diagram

4.Sequence Diagram

Menggambarkan kolaborasi dinamis antara sejumlah objek dan untuk menunjukan rangkaian pesan yang dikirim antara objek juga interaksi, sesuatu yang terjadi pada titik tertentu dalam eksekusi system. contoh sequence diagram dapat dilihat pada gambar 2-6:



Gambar 2-6 sequence diagram