BAB II
LANDASAN TEORI

2.1 Aplikasi

(Juansyah, 2015) Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju.

2.2 Pengertian *Round-Robbin*

(Pangera & Ariyus, 2010) Konsep dasar dari algoritma ini adalah dengan menggunakan t*ime sharing*. Pada dasarnya algoritma ini sama dengan *first come first served* (FCFS), hanya saja bersifat *preemptive*. Setiap proses mendapatkan waktu CPU Yang disebut dengan waktu quantum (quantum time) untuk membatasi waktu proses, biasanya 1-100 milidetik. Setelah waktu habis, proses ditunda dan ditambahkan pada *ready queue*.

Jika suatu proses memiliki CPU burst lebih kecil dibandingkan dengan waktu quantum, maka proses tersebut akan melepaskan CPU jika telah selesai bekerja, sehingga CPU dapat segera digunakan oleh proses selanjutnya. Sebaliknya, jika suatu proses memiliki CPU burst yang lebih besar dibandingkan dengan waktu quantum, maka proses tersebut akan dihentikan sementara jika sudah mencapai waktu quantum, dan selanjutnya mengantri kembali pada posisi ekor dan ready queue. CPU kemudian menjalankan proses berikutnya.

Jika terdapat *n* proses pada *ready queue* dan dan waktu quantum *q*, maka setiap proses mendapatkan 1/n dari waktu CPU paling banyak q unit waktu pada sekali penjadwalan CPU. Tidak ada proses yang menunggu lebih dari (*n-1*) q unit waktu. Performansi algoritma round robin dapat dijelaskan sebagai berikut, jika q besar maka yang digunakan adalah algortima FIFO, tetapi jika q kecil maka sering terjadi context switch.

$$\lambda = \sum W/n \quad \text{..... (1)}$$

Keterangan:

λ = waktu tunggu rata-rata

W = waktu tunggu tiap proses

n = jumlah proses antrian

2.3 Android Studio

(Juansyah, 2015) Mendefinisikan Android Studio adalah IDE (Integrated Development Environment) resmi untuk pengembangan aplikasi Android dan bersifat *open source* atau gratis. Android Studio diluncurkan sebagai pengganti Eclipse. Sebagai IDE resmi untuk mengembangkan aplikasi Android.

2.4 Java

(Wardhani & Yaqin, 2013) Java adalah sebuah bahasa pemrograman yang populer dikalangan para akademisi dan praktisi komputer. Java pertama kali dikembangkan untuk memenuhi kebutuhan akan sebuah bahasa komputer yang ditulis satu kali dan dapat dijalankan dibanyak sistem komputer berbeda tanpa perubahan kode berarti. Pada umumnya, para pakar pemrograman berpendapat bahwa bahasa Java memiliki konsep yang konsisten dengan teori pe1mrograman objek dan aman untuk digunakan.

2.5 SQLite

(Pambudi, 2013) Android memiliki fasilitas untuk membuat database yang dikenal dengan SQLite. Di dalam Android, SQLite termasuk dalam Anroid runtime, sehingga setiap versi dari android dapat membuat database dengan SQLite. Untuk mengunakan SQLite dalam pembuatan database pada Android tidak tersedia secara otomatis, melainkan harus created data base sendiri, mendefinisikan table, index, serta datanya. Untuk membuat dan membuka database menggunakan libraries, yaitu:

Import android database sqlite, SQLite Open Helper, libraries tersebut menyediakan tiga metode yaitu:

1. Constructor, menyediakan representasi versi dari database skema yang kita gunakan.
2. Oncreate(). Menyediakan SQLite Database object yang kita butuhkan dalam definisi table dan inisialisasi data.
3. OnUpgrade(). Menyediakan fasilitas convert database dari skema database versi lama ke skema database versi yang baru.

2.6 Android SDK Manager dan AVD

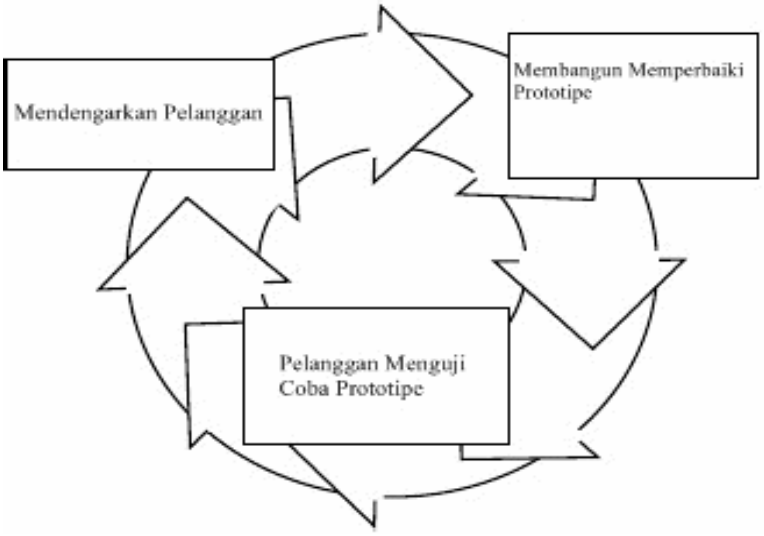
(Pambudi, 2013) *Android SDK Manager* adalah sebuah alat pengembangan perangkat lunak yang berguna untuk mengembangkan dan membuat aplikasi untuk *platform Android*. Didalam *Android SDK* terdapat *project* sample dengan kode sumber, alat-alat untuk membuat aplikasi, sebuah emulator Android(AVD), dan *library* yang dibutuhkan untuk membangun aplikasi *Android*. Aplikasi yang ditulis pada bahasa *Java* dan berjalan di *Dalvik*, mesin virtual yang dirancang khusus untuk penggunaan *embedded* yang berjalan diatas kernel Linux.

2.7 Metode Pengembangan Perangka Lunak

2.7.1 Metode Prototype

(Nurajizah, 2015) menjelaskan dasar dari pemikiran ini adalah membuat prototipe secepat mungkin, bahkan dalam waktu semalam, lalu memperoleh umpan balik dari pengguna yang akan memungkinkan prototype tersebut diperbaiki kembali dengan sangat cepat. Semua rancangan diagaram atau model yang dibuat tidak diharuskan telah sempurna dan final dalam pendekatan prototype. Tujuan utama dari penyiapan rancangan adalah sebagai alat bantu dalam memberi gambaran sistem seperti materi dan menu yang perlu dimasukkan dalam prototype yang akan dikembangkan. Setelah rancangan terbentuk, dilanjutkan dengan mulai mengembangkan prototype. Metode prototype sesuai untuk menjelaskan kebutuhan pengguna secara lebih rinci karena pengguna sering mengalami kesulitan dalam penyampaian kebutuhannya secara detail tanpa melihat gambaran yang jelas.

Untuk mengantisipasi agar proyek dapat berjalan sesuai dengan rencana, target waktu, dan biaya diawal, maka sebaiknya spesifikasi kebutuhan sistem harus sudah disepakati terlebih dahulu oleh pengembang dengan pengguna dalam hal ini klien. Adapun tahapan-tahapannya metode prototype adalah sebagai berikut:



Gambar 2.1 Model Prototype

(Widyaningtyas, 2014) tahapan untuk suatu prototyping yaitu :

a. Mengidentifikasi kebutuhan pengguna Pengembangan mewawancarai pengguna untuk mendapatkan ide mengenai apa yang diminta dari sistem.

b. Mengembangkan *prototype* Pengembangan mempergunakan satu alat *prototyping* atau lebih untuk membuat *prototype*. Contoh dari alat-alat *prototyping* adalah generator aplikasi *terintegrasi* dan *toolkit prototyping*. Generator aplikasi terintegrasi adalah sistem peranti lunak siap pakai yang mampu membuat seluruh fitur yang diinginkan dari sistem baru-mu, laporan, tampilan, basis data, dan seterusnya.*Toolkit prototyping* meliputi sistem-sistem peranti lunak terpisah seperti s*preadsheet* elektronik atau sistem manajemen basis data, yang masing-masing mampu membuat sebagian dari fitur-fitur sistem yang diinginkan.

c. Menentukan apakah *prototype* dapat diterima Pengembangan mendemonstrasikan *prototype* kepada para pengguna untuk mengetahui apakah telah memberikan hasil yang memuaskan. Jika ya, langkah 4 akan diambil; dan jika tidak, *prototype* direvisi dengan mengulang kemblai langkah 1, 2, dan 3 dengan pemahanan yang lebih baik mengenai kebutuhan pengguna.

d. Menggunakan *Prototype* menjadi sistem produksi.

2.8 *Unified Modeling Language* (UML)

(Destiana, 2014) UML adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Adapun beberapa jenis diagram pada UML yang dapat membantu perancangan sistem adalah sebagai berikut:

a. *Use Case Diagram*

   Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem dan bukan "bagaimana". Sebuah *use case* mempresentasikan sebuah interaksi antar actor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja dan sebagainya. Seorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

b. *Activity Diagram*

   Activity diagram menggambarkan berbagai air aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state* diagram khusus, dimana

sebagian besar *state* adalah *action* dan sebagain besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing).*

c. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code,* baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time* maupun *run time.* Umumnya komponen terbentuk dari beberapa *class* dan/atau *package,* tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface,* yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

d. *Deployment Diagram*

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server dan hal-hal lain yang bersifat fisikal.