BAB II

LANDASAN TEORI

2.1 *Inventory Management*

Inventory management menunjukkan kerangka kerja yang luas dalam mengelola inventaris. Teknik *inventory management* lebih berguna dalam menentukan tingkat persediaan yang optimal dan menemukan jawaban untuk masalah *safety stock* dan *lead time*. *Inventory Management* telah sangat berkembang untuk memenuhi tantangan yang terus meningkat di sebagian besar entitas perusahaan[11].

Inventory Management adalah proses bisnis yang bertanggung jawab untuk mengembangkan dan mengelola inventaris, apakah persediaan itu bahan baku, setengah jadi bahan ataupun barang jadi, sehingga persediaan yang memadai harus selalu tersedia dan perusahaan harus memastikan biayanya kelebihan atau kekurangan stok selalu rendah[12].

Inventory Management adalah meminimalkan investasi dalam persediaan namun tetap konsisten dengan penyediaan tingkat pelayanan yang diminta. Dalam manajemen persediaan terdapat 2 hal keputusan persediaan yang perlu diperhatikan,keputusan persediaan yang bersifat umum merupakan keputusan yang menjadi tugas utama dalam penentuan persediaan baik secara kualitatif maupun kuantitatif[10].

Oleh karena itu data dari hasil pengiriman batubara akan terintegrasi ke dalam aplikasi ini sehingga akan lebih mudah untuk mengawasi pengiriman dan mengawasi *inventory* batubara dalam rangka pengelolaan inventarisasi/ *inventory management* batubara perusahaan.

2.2 *System Development Life Cycle*

System Development Life Cycle adalah siklus yang dimanfaatkan oleh para *developer* dalam mengambangkan sistemnya untuk membantu mereka menciptakan sistem secara efisien.

Pada skripsi ini akan digunakan metode *Prototyping*, metode ini sendiri merupakan metode yang dapat membantu pengambang untuk mengembangkan aplikasi tahap per tahap dan menghasilkan hasil yang lebih sesuai dengan keinginan user. Alasan digunakannya metode *Prototyping* karena beberapa kelebihan berikut:

- Lebih mudah menentukan *requirement* sistem.
- Waktu pengembangan dapat dipersingkat.
- *User* dapat terlibat langsung dalam pengambangan sistem sehingga hasil akhir akan lebih sesuai dengan keinginan *user*.

metode ini cocok digunakan oleh karena pada skripsi ini aplikasi yang dibuat harus nyaman digunakan oleh *user* dilapangan.

2.2.1 Metode *Prototyping*

Aplikasi yang akan dibuat pada penelitian akan menggunakan metode *prototyping* untuk pembangunan sistem, alasan penggunaan metode ini ada beberapa oleh karena aplikasi yang digunakan ini akan dibuat khusus hanya untuk perusahaan ini secara internal maka penggunaan *metode prototyping* ini sangatlah cocok, karena metode ini sangat melibatkan klien atau calon *user* dalam pembuatan sistem sehingga tampilan dan fitur-fitur yang ada pada aplikasi akan cocok dengan keinginan klien.

Metode *Prototyping* melihatkan proses pembuatan desain awal sebuah aplikasi, dan akan dikembangkan secara bertahap. Dengan metode *prototyping*, *developer* akan melibatkan calon *user* pada setiap tahap selama proses pengembangan aplikasi sehingga mendapatkan hasil yang sangat memuaskan bagi *user*, dibuatnya sebuah *prototyping* bagi *developer* untuk mengumpulkan informasi sebanyak mungkin dari calon *user*, *prototype* menggambarkan versi awal dari sistem untuk kelanjutan sistem sesungguhnya yang lebih besar[13].

Prototyping dapat digunakan untuk pengambangan sebuah projek skala kecila ataupun besar dengan harapan selama proses pengambangan aplikasi dapat berjalan dengan teratur dan sesuai keinginan *user*. Keterlibatan *user* pada setiap langkah pengembangan aplikasi, sehingga ketika *prototype* terbentuk akan menguntungkan seluruh pihak yang terlibat, bagi pimpinan, pengguna sendiri serta pengembang sistem.

Manfaat lainnya dari penggunaan *prototyping* adalah :

1. Menampung semua masukan dari *user* sehingga aplikasi yang akan tercipta nantinya akan sesuai keinginan *user*.
2. *user* akan dapat lebih mudah menerima penerapan sistem atau aplikasi baru karena *user* telah melihat proses pengembangan aplikasi tersebut dari awal sampai akhir.
3. *Prototype* termasuk fleksibel karena akan mengikuti saran dan masukan *user*, serta kemajuan tahap demi tahap dapat diikuti langsung oleh pengguna.
4. Penghematan *resource* dan waktu dalam menghasilkan produk akhir yang lebih baik dan tepat sasaran bagi pengguna.

2.2.2 Waterfall

Waterfall model adalah [14] adalah strategi pengembangan perangkat lunak terapan pertama, menyerupai desain yang digunakan di industri lain. Strategi ini memungkinkan proyek untuk dipecah menjadi beberapa fase tetap, dengan setiap fase membutuhkan analisis dan bekerja dari fase sebelumnya:

- *Requirement*

  Menganalisa kebutuhan bisnis dan melakukan dokumentasi atas semua fitur.

- *Design*

memilih semua teknologi yang diperlukan dan merencanakan infrastruktur dan interaksi perangkat lunak lengkap

- *Coding*

  menyelesaikan semua masalah, mengoptimalkan solusi dan mengimplementasikan setiap komponen yang dijelaskan dalam fase *requirement*, menggunakan diagram dan cetak biru dari fase *design*

- *Testing*

  Melakukan *testing* dari semua fitur dan komponen yang telah diimplementasikan dan menyelesaikan semua masalah yang muncul selama proses *testing*.

- *Operation*

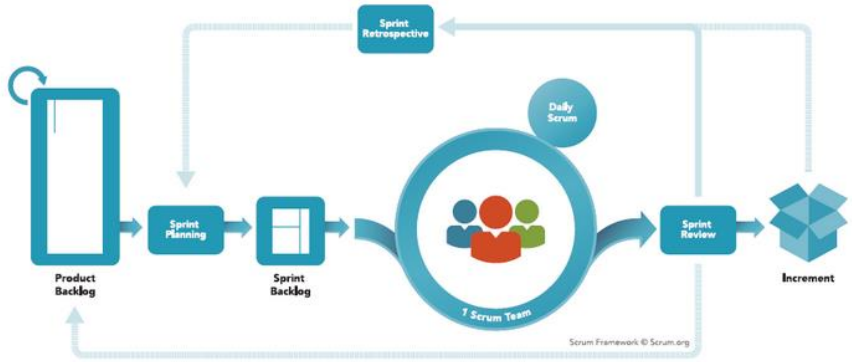
  Melakukan peluncuran ke *production environment*

Model *Waterfall* mengasumsikan bahwa setelah persyaratan awal ditetapkan dan setiap tujuan telah dibersihkan dari ambiguitas, ada jalan yang tidak terhalang yang akan diikuti oleh tim pengembangan untuk menyelesaikan proyek. Namun dalam kebanyakan kasus kehidupan nyata, ini tidak benar, karena pelanggan dapat mengubah pendapat mereka terhadap fitur yang berbeda, dalam hal ini beberapa, jika tidak semua, fase harus dievaluasi ulang. Ini menarik biaya tambahan dan waktu yang

dihabiskan untuk berbagai bagian proyek, yang pada gilirannya dapat menurunkan kepuasan pelanggan. Ini adalah cacat paling jelas dari model *waterfall*, tetapi itu tidak berarti, strategi ini tidak boleh diterapkan.

2.2.3 Scrum (agile)

Scrum [15] adalah kerangka kerja untuk mengembangkan proyek yang kompleks dan mengorganisir pekerjaan, berdasarkan serangkaian nilai (keberanian, fokus, komitmen, rasa hormat, keterbukaan), prinsip dan praktik yang memberikan landasan bagi semua anggota. Kerangka kerja Scrum menawarkan fleksibilitas dan keserbagunaan luar biasa ketika menghadapi kendala yang terus berubah, baik finansial maupun teknologi dan kunci keberhasilannya selalu menargetkan tugas-tugas dengan prioritas tertinggi, masing-masing melalui proses 7 langkah:



Gambar 2. 1 Proses Scrum[15]

1. *Product Backlog*

Product backlog bertujuan untuk proses dilakukannya rapat antar semua divisi dan pihak yang terlibat untuk menentukan hal apa

yang memiliki prioritas yang tertinggi dan menentukan hal tersebut dari yang paling tinggi sampai yang paling rendah.

2. *Sprint Backlog*

Sprint Backlog merupakan kumpulan beberapa *item* dari hasil *Product Backlog* yang dipilih untuk Sprint, dan termasuk dengan rencana bagaimana untuk mengeksekusi *sprint* ini agar dapat berjalan dengan baik dan sesuai target.

3. *Sprint Planning*

Sprint planning merupakan tahap yang paling krusial dalam memulai satu *sprint*. Pada tahap ini semua divisi yang terlibat akan menjabarkan pekerjaan apa saja yang mereka lakukan dan menentukan *deadline*-nya.

4. *Daily Scrum*

Tahap ini akan dilakukan berulang kali hingga selesai. Hal yang dilakukan pada tahap ini adalah setiap harinya setiap divisi akan berkumpul untuk melaporkan apa saja yang mereka akan lakukan hari ini dan membahasnya agar target yang telah ditentukan dapat tercapai.

5. *Sprint Review*

Sprint review adalah saat semua anggota tim mendemonstrasikan apa yang telah mereka selesai kerjakan dan

akan di *review* untuk menentukan apa yang telah dikerjakan tersebut sesuai dengan harapan, jadi setiap tahap selesai dengan sempurna.

6. Sprint Retrospective

Sprint retrospective dilakukan di akhir setiap sprint. Pada tahap ini semua anggota akan membahas bagaimana *sprint* yang telah dilakukan tanpa membahas sisi teknis dari proyek yang telah dikerjakan, dan dari tersebut untuk kedepannya *sprint* dapat lebih disesuaikan dengan kondisi lapangan.

7. Evaluation

Terakhir, akan dilakukan evaluasi dari *sprint* yang telah dilakukan dengan menggunakan beberapa pertanyaan seperti beberapa hal seperti :

- *Went well* --> apa saja yang berjalan dengan baik?
- *Not well* --> apa saja yang tidak berjalan dengan baik?
- *Improvements* --> apa saja yang perlu diperbaiki di sprint berikutnya?

2.3 UML

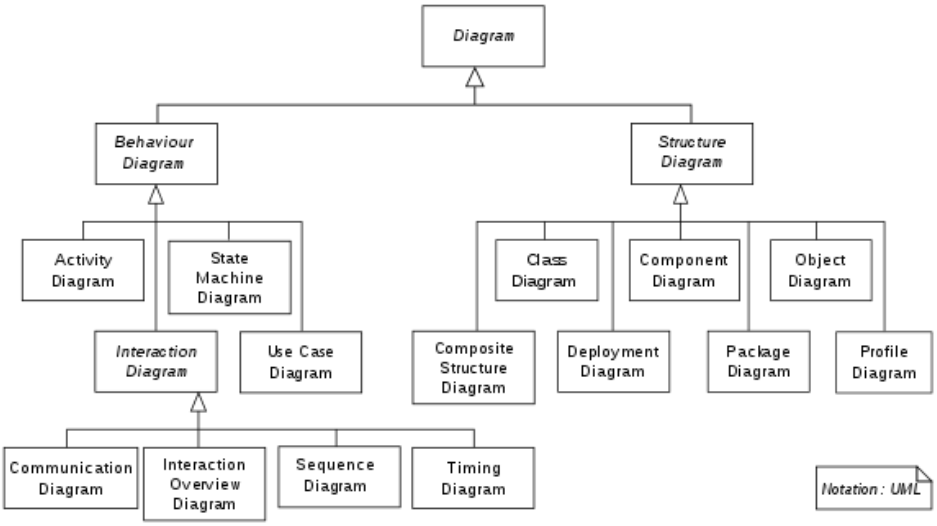
Unified Modelling Language (UML)[16] menyediakan pemodelan visual yang dapat membantu *developer* untuk menggambarkan *blue print* sebuah sistem yang akan dibuat menjadi lebih jelas. UML menggunakan

beberapa jenis diagram untuk mengkomunikasi hal-hal tersebut. Diagram-diagram tersebut digunakan untuk :

1. Mengkomunikasikan ide
2. Melahirkan ide-ide baru dan peluang-peluang baru
3. Menguji ide dan membuat prediksi
4. Memahami struktur dan relasi-relasinya.

Fungsi dari penggunaan UML sendiri adalah:

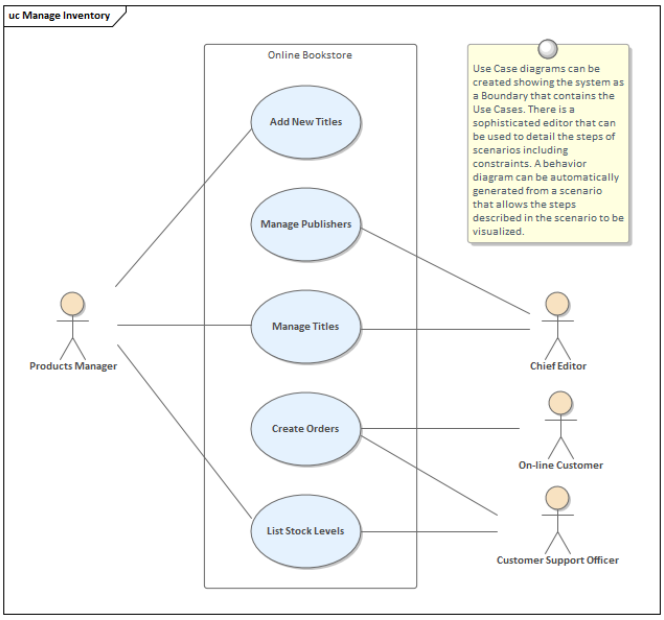
1. Menggambarkan batasan dan fungsi-fungsi sistem secara umum.
2. Menggambarkan proses bisnis yang dilakukan secara umum, penggambaran ini dibuat dengan interaction diagrams.
3. Untuk menggambarkan representasi struktur statik sebuah sistem dalam bentuk class diagrams
4. Membuat *state behavior* ”yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan state transition diagrams
5. Untuk menyatakan arsitektur implementasi fisik menggunakan component and development diagram.



Gambar 2. 2 Diagram yang terdapat dalam UML [16]

2.3.1 Use Case Diagram

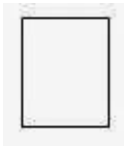
Use case model [16] terdiri dari *actors* dan *use cases*, *actors* adalah *environment* dari sistem yang dimodelkan dan *use cases* dari sistem ini. *actors* adalah objek yang berinteraksi dengan sistem target. *Use* cases adalah layanan dan fungsionalitas yang digunakan oleh *actors*, oleh karena itu sebuah *use cases* harus sesuai dengan kebutuhan atau *requirement* dari *actors*. Diagram *use case* mengidentifikasi harapan objek eksternal untuk sistem dan fitur spesifik sistem. Namun, diagram tersebut tidak merujuk pada rincian implementasi sistem. *Use cases* dapat dihubungkan dengan *relationship*. Ada tiga hubungan dasar di antara *use cases*: generalisasi dan dua stereotip standar, yaitu «*include*» dan «*extend*». Hubungan antara *use cases* adalah alat penting untuk mengelola kompleksitas skenario. Gambar di bawah merupakan contoh dari *use case* diagram:



Gambar 2. 3 Contoh *use case* diagram[16]

Tabel 2. 1 Notasi *use case* diagram [16]

Notasi *use case* diagram	Nama	Fungsi
	Aktor	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan use case.
	*Use case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
	Association	Berfungsi untuk menghubungkan antara objek satu dengan objek lainnya.

Notasi *use case* diagram	Nama	Fungsi
	*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

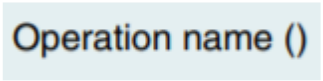
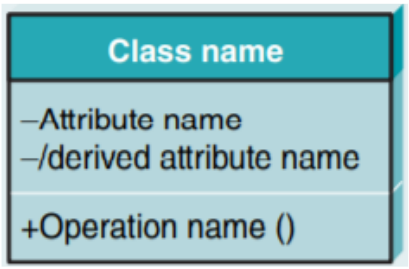
2.3.2 Class Diagram

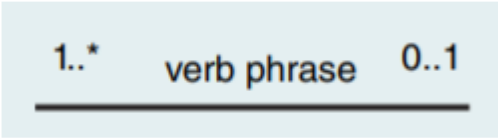
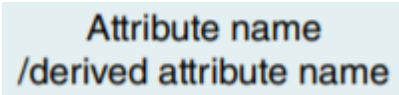
Class diagram merupakan penjelasan proses database dalam suatu program, Diagram ini [16] digunakan untuk menggambarkan tampilan statis aplikasi, konstituen utama adalah *classes* dan *relationship*. *classes* adalah deskripsi konsep, dan mungkin memiliki atribut dan operasi yang terkait dengannya. *classes* direpresentasikan dengan bentuk persegi panjang. *relationship* antara dua *classes* dijelaskan dengan garis. *Inheritance relationship* mengindikasikan bahwa sebuah atribut dan operasi dari 1 *class* (”*superclass*”) diwariskan oleh *class* lain (“*subclass*”),tanpa perlu secara eksplisit merepresentasikan dari *subclass* itu sendiri.

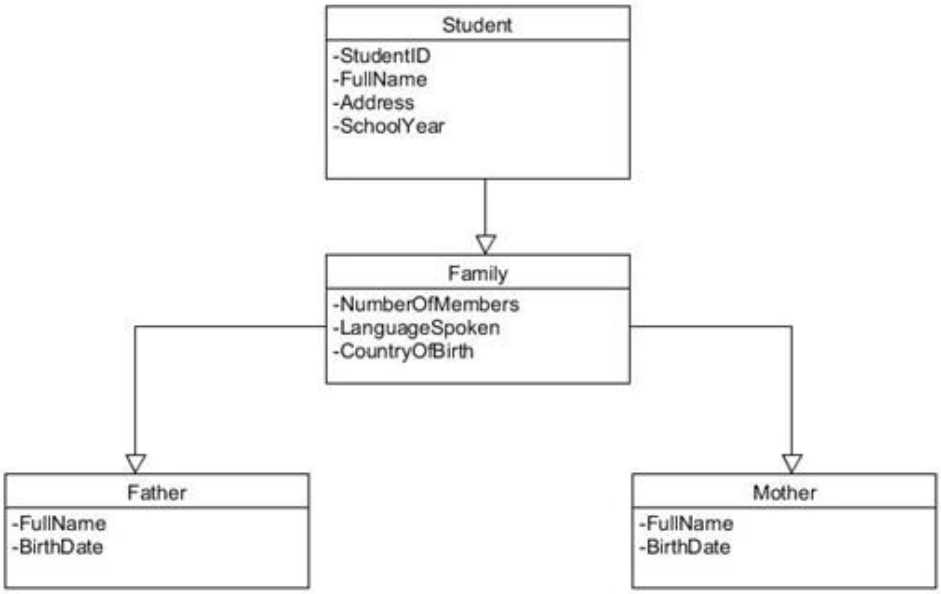
Relasi yang digunakan dalam penggambaran kelas diagram dapat dilihat dalam tabel berikut:

Tabel 2. 2 Notasi *Class* Diagram[16]

Simbol	Keterangan
Association	Asosiasi menggambarkan kelas yang memiliki atribut berupa kelas lain, atau kelas yang harus mengetahui eksistensi kelas lain. Asosiasi biasanya disertai dengan *multiplicity*.

Simbol	Keterangan
Directed Association 	Asosiasi dengan makna kelas yang satu digunakan oleh kelas yang lain. Asosiasi berarah juga biasanya disertai dengan *multiplicity.*
Agregation 	Hubungan yang menyatakan bahwa suatu kelas menjadi atribut bagi kelas lain
Composition 	Bentuk khusus dari agregasi dimana kelas yang menjadi bagian diciptakan setelah kelas menjadi *whole* dibuat.
Realization 	Hubungan antar kelas dimana sebuah kelas memiliki keharusan untuk mengikuti aturan yang ditetapkan kelas lainnya.
	*Method* merupakan sebuah *actions* atau *functions* dari sebuah *class* yang dapat dijalankan. *Method* dapat diklasifikasikan sebagai *constructor*, *query*, atau *update operation*.
	*Class* dapat berupa seseorang, tempat, atau sesuatu barang dimana informasi tersebut harus diambil dan disimpan oleh sistem. *Class* juga mempunyai daftar atribut di bagian tengah dan daftar operasi di bagian bawah kotak tersebut.

Simbol	Keterangan
	*Association* merupakan sebuah hubungan antara banyak classes atau dengan sebuah class itu sendiri.
	*Attribute* digunakan sebagai properti untuk menggambarkan keadaan dari suatu objek. *Attribute* juga dapat diturunkan dari *attribute* lain, ditunjukan dengan menempatkan garis miring di depan nama *attribute* yang dituju.



Gambar 2. 4 Contoh Class Diagram[16]

2.3.3 Activity Diagram

UML Activity Diagram [17] secara umum menggambarkan tentang aktifitas yang terjadi pada sistem. Dari pertama sampai akhir, diagram ini menunjukkan langkah – langkah dalam proses kerja sistem yang dibuat., mulai dari urutan kegiatan atau tindakan dari proses bisnis dalam suatu organisasi atau di antara organisasi sampai ke detail suatu algoritma. Proses penyempurnaan bertahap dilakukan diagram aktivitas semakin kompleks. Untuk menjamin konsistensi perilaku dan kebenaran dalam penyempurnaan, diagram aktivitas harus didekomposisi sesuai dengan strategi membagi dan menaklukkan strategy.

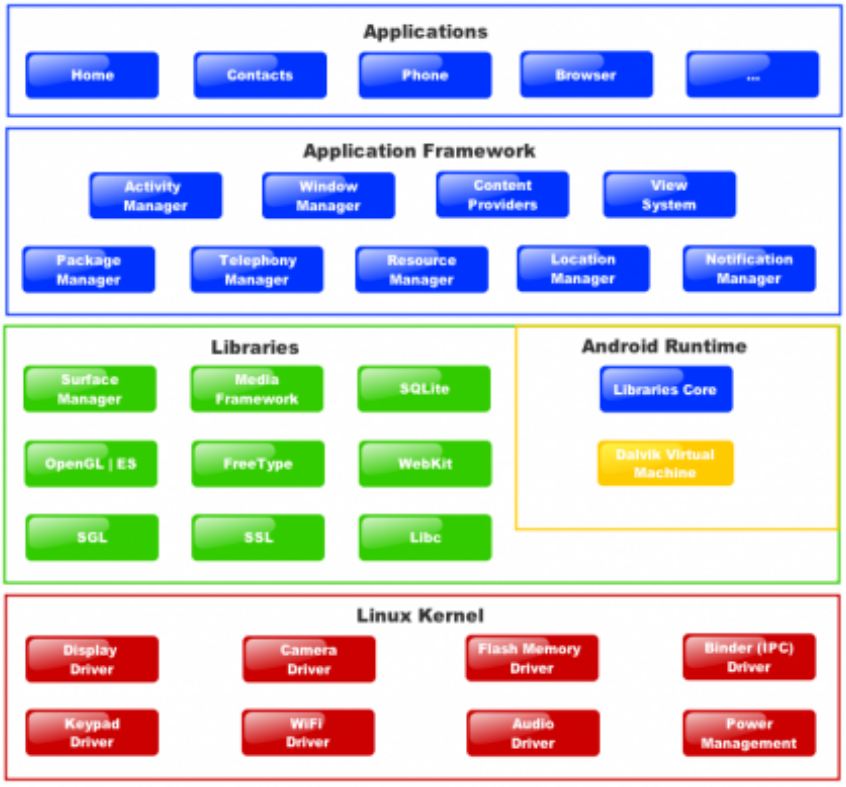
Tabel 2. 3 Notasi *activity* diagram[16]

Simbol	Keterangan
	Titik awal
	Titik akhir
	*Actitvity*
	Pilihan untuk mengambil keputusan

2.4 Struktur aplikasi android

Aplikasi yang dibuat adalah aplikasi *mobile* berbasis android karena aplikasi ini akan diimplementasikan di ponsel pintar (*smartphone)* yang mudah digunakan di lapangan dan semua pengguna seperti *foreman* maupun *driver* / karyawan pengemudi umumnya menggunakan *smartphone* dengan sistem Android.

Android adalah *platform software* dan *OS* (*operation system*) untuk perangkat seluler, berdasarkan kernel Linux, dan dikembangkan oleh Google. Bahasa pemrograman yang digunakan adalah Java, mengambangkan perangkat melalui *library* Java yang dikembangkan Google. Android merupakan sebuah *open source software* yang dapat diunduh secara bebas dan gratis untuk perangkat seluler yang mencakup sistem operasi, *middleware*, dan aplikasi utama berbasis Linux dan Java. Google merilis kode Android sebagai sumber terbuka di bawah Lisensi Apache. Pertama-tama pada *developer* akan mengembangkan aplikasi mereka menggunakan bahasa pemrograman java, dan kemudian dapat dimasukkan ke dalam *playstore* yang dikelola oleh Google sendiri[18]. Struktur aplikasi pada android dapat digambarkan pada gambar di bawah.



Gambar 2. 5 Struktur Aplikasi Android[9]

Penjelasan secara garis besar dari setiap bagian struktur aplikasi android tersebut menurut [8] :

a) *Application dan Widgets*

Bagian paling atas yang berhubungan langsung dengan aplikasi yang ter-*install* pada perangkat *mobile*.

b) *Application Frameworks*

Layer ini tempat para *developer* mengambangkan sistem yang akan digunakan oleh android seperti fitur sms, notifikasi, dan lain-lain.

c) *Libraries*

Merupakan *layer* untuk para *developer* mengakses *libraries* apa saja yang ada pada sistem Android.

d) *Android Run Time*

Layer yang menjalankan sistem android dan menggunakan sistem proses Linux..

e) Linux Kernel

Merupakan *layer* inti dari sistem android yang berisi hal-hal penting yang mengatur *processing, memory*, dan lain-lain.

2.5 IDE

Integrated Development Environment tool digunakan untuk berbagai macam hal seperti untuk pengembangan sistem, penulisan kode, dan *compiling*. Alasan penggunaan IDE pada penelitian pun karena penggunaan IDE memiliki kelebihan sebagai berikut:

- Memiliki fitur *autocomplete* dalam proses *coding.*
- Import yang terorganisasi .
- Memudahkan untuk menemukan *error* pada *coding*-an.
- Membantu dalam menemukan sebuah dokumentasi.

- Menjaga tampilan file, kesalahan / peringatan / konsol / tes unit dll dan kode sumber semua pada layar pada saat yang sama dengan mudah.
- Kemudahan menjalankan *unit test* pada *window* yang sama.
- *Debugging* yang terintegrasi.
- *source control* yang terintegrasi.

2.5.1 Android Studio

Android Studio adalah *IDE* baru (Lingkungan Pengembangan Terpadu) yang dibuat oleh Google untuk android *Developers* secara gratis. Android Studio mencakup banyak alat untuk membantu *developers* mengembangkan aplikasi. Seperti fitur *SDK* emulator yang dapat digunakan untuk mengimplementasikan beberapa fitur perangkat android seperti (meluncurkan *Google Browser*, terhubung ke internet, mengirim pesan, melihat informasi dasar perangkat (ruang memori, dan keamanan)) fitur-fitur ini juga bisa diimplementasikan menggunakan perangkat virtual tanpa menggunakan perangkat asli karena fitur yang tersedia dari *android studio*[19]

2.5.2 Eclipse

Eclipse adalah *integrated development environment* (*IDE*) yang dimanfaatkan untuk mengambangkan *software* yang menggunakan bahasa pemrograman Java, C++, dan lain-lain. *Platform Eclipse* yang menyediakan pondasi untuk *Eclipse IDE* memiliki beberapa *plug-in* dan dirancang agar dapat diperluas menggunakan *plug-in* dari luar. Dikembangkan

menggunakan bahasa pemrograman Java. *Eclipse* dapat digunakan untuk mengembangkan sebuah aplikasi menggunakan bahasa pemrograman apapun selama *plug-in*-nya tersedia.

Java Development Tools (JDT) menyediakan *plug-in* yang memungkinkan *Eclipse* untuk digunakan sebagai *Java IDE* Contohnya ada ada *plug-in* yang memungkinkan untuk menggunakan Eclipse sebagai *Java IDE* menggunakan sebuah *plug-in* yang disediakan oleh *Java Development Tools (JDT),* selain itu terdapat juga *PyDev* memungkinkan Eclipse untuk digunakan sebagai *IDE Python*, dan juga ada *C / C ++ Development Tools* (CDT) *plug-in* yang dapat digunakan agat *Eclipse* dapat digunakan untuk mengembangkan aplikasi menggunakan bahasa pemrograman C / C ++. Selain itu terdapat juga *Eclipse Scala plug-in* yang dapat digunakan untuk memanfaatkan Eclipse untuk mengembangkan aplikasi *Scala* dan *PHPeclipse* adalah *plug-in* untuk *eclipse* yang menyediakan alat pengembangan lengkap untuk PHP. [20]

2.6 SDK

SDK merupakan sebuah akronim untuk “Software Development Kit”, SDK merupakan sekumpulan *tools* yang digunakan untuk menciptakan dan mengembangkan sebuah aplikasi. SDK merupakan sebuah *tools* yang digunakan oleh para *developer* yang membutuhkan sebuah *tools* untuk menyediakan antarmuka yang unik dan terstandar kepada pengguna akhir dari suatu aplikasi SDK juga memiliki serangkaian fungsi yang kuat, seperti penanganan kesalahan, kinerja yang konsisten, *reusable code*, yang

menghilangkan kerumitan bagi pengembang perangkat lunak. SDK memastikan bahwa setiap API yang disediakan diimplementasikan dengan benar, dapat dikaitkan dengan penerapan API.

2.6.1 Android SDK

Android *Software Development Kit* (SDK) digunakan oleh para *developer* untuk mengembangkan sistem android dan pada *kit* ini memiliki berbagai macam *tools* seperti emulator, *libraries*, *debugger*, dan lain-lain.

2.7 *Programming Language*

Programming language yang akan digunakan pada rancang bangun aplikasi yang dibuat adalah menggunakan bahasa pemrograman *Dart*, jika pada *android development* umumnya menggunakan bahasa pemrograman Java atau Kotlin, namun penggunaan bahasa pemrograman *Dart* dikarenakan pada pengembangan aplikasi ini menggunakan *framework* flutter dari *google* yang menggunakan *Dart* sebagai bahasa pemrogramannya.

2.7.1 *Dart*

Bahasa pemrograman *Dart* merupakan bahasa pemrograman yang dikembangkan oleh 2(dua) orang *programmer* yang bernama Lars Bak dan Kasper Lund. Dart ini dibuat dengan tujuan untuk menciptakan bahasa pemrograman yang mudah dipelajari dan disebarkan.

Dart digunakan untuk pengembangan aplikasi pada berbagai macam platform seperti *mobile application, web, server*¸ dan perangkat teknologi lainnya yang mengarah ke *IOT (Internet of Things).*

Dart dapat digunakan untuk mengembangkan aplikasi dari *codebase* tunggal menjadi aplikasi Android maupun IOS. *Dart* juga memiliki beberapa kelebihan seperti berikut :

1. Fleksibilitas bahasa pemrograman ini mendukung untuk pengambangan aplikasi baik di Android maupin di IOS
2. memiliki ketersediaan *Software development kit (SDK)* yang lengkap dengan ragam macam seperangkat *tools*.

2.8 Framework

Framework merupakan sebuah kerangka kerja yang digunakan untuk mengambangkan sebuah *website* atau *wesbite development* maupun *mobile*.

Framework dibuat dengan tujuan untuk memudahkan para *developer* mengembangkan aplikasi lebih cepat serta tersusun dan terstruktur. Penggunaan *framework* dapat sangat membantu mereka oleh karena hanya perlu menyusun komponen-komponen pemrograman yang sudah ada. Sehingga para *developer* dan *programmer* tidak perlu melakukan koding-an hal *basic* yang sama secara berulang.

2.8.1 Flutter

Framework Flutter merupakan SDK untuk *mobile development* yang dikembangkan oleh Google sendiri. Sama seperti React Native, *Framework* dapat digunakan untuk pengembanagan *device* iOS dan Android.

Semua kode pada Flutter di *Compile* dalam kode *Native* nya (*Android NDK, LLVM, AOT-compiled*) tanpa ada *Interpreter* pada prosesnya sehingga mempercepat proses *compile*.

Dalam menggunakan Flutter akan membantu cepatnya proses *development* karena Flutter memiliki sebuah fitur bernama *hot reload*. *Hot reload* bekerja dengan cara menginjeksi kode program yang mengalami perubahan ke dalam *Dart Virtual Machine*. Setelah itu Flutter akan secara otomatis akan meng-*update* dan menyusun kembali semua komponen yang ada secara langsung. Beberapa kelebihan yang dimiliki Flutter:

1. Penulisan kode yang lebih cepat dengan bantuan fitur *hot reload* dan *hot restart*
2. Satu kode untuk 2 platform sekaligus yaitu android dan IOS.
3. membuat tampilan desain aplikasi menjadi lebih mudah dengan bantuan *widget* dari Flutter.

2.9 *Database*

Database merupakan sebuah kumpulan data yang dikelola berdasarkan ketentuan. Melalui pengelolaan tersebut hal ini dapat

mempermudah *developer* dalam memperoleh, menyimpan dan membuang informasi.

2.9.1 NoSql

NoSQL adalah singkatan dari *Not Only SQL*. *Database management* system ini bersifat tanpa relasi (non-relational). Artinya, NoSQL bisa mengelola database dengan skema yang fleksibel dan tidak membutuhkan *query* yang kompleks.

Dengan pendekatan ini, NoSQL mempunyai skalabilitas tinggi untuk dapat berkembang sesuai dengan kebutuhan data yang ada. Tak heran, jenis database *NoSQL* ini dianggap paling cocok untuk mengolah *big data* yang sangat cepat pertumbuhan datanya.

Database noSQL adalah jenis *Database* yang tidak memiliki struktural atau semi struktural dan *database* ini juga tidak memiliki relasi seperti *MySql*. Tujuan utama untuk menggunakan jenis *database* ini adalah untuk pengembangan sistem yang memiliki data fleksibel, biasanya seperti *real time application*.

Database NoSQL memberikan berbagai model data, seperti nilai kunci, dokumen, grafik, yang dioptimalkan untuk kinerja dan skala.

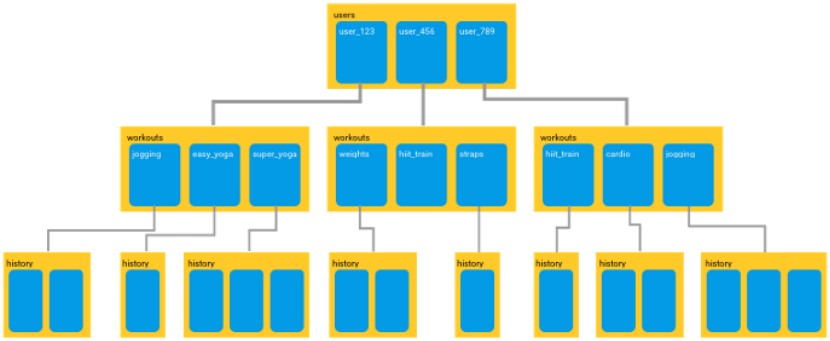
Database No Sql memiliki sifat yang fleksibel, *scaleable,* kinerja tinggi sehingga cocok digunakan untuk pengembangan banyak sistem modern. Berikut beberapa kelebihan dari *Database* NoSQL:

- **Fleksibilitas:** *Database* NoSQL umumnya menyediakan skema fleksibel yang memungkinkan pengembangan yang lebih cepat dan lebih berulang. Dengan skema *Database* yang biasanya menggunakan kumpulan *collection* dan *documents*.
- **Skalabilitas:** *Database* NoSQL umumnya didesain untuk meningkatkan skala dengan menggunakan klaster perangkat keras yang terdistribusi alih-alih meningkatkan skala dengan menambah server yang mahal dan *robust*. Beberapa penyedia layanan *cloud* menangani aktivitas di balik operasi ini sebagai layanan yang dikelola sepenuhnya.
- **Kinerja tinggi:** *Database* NoSQL dioptimalkan untuk model data spesifik dan pola akses yang memberikan kinerja yang lebih.
- **Fungsionalitas tinggi:** *Database* NoSQL menyediakan *API* yang dikembangkan secara khusus untuk bekerja dengan maksimal dengan berbagai macam bentuk data.

2.9.2 Firebase

Firebase adalah suatu layanan dari Google untuk memberikan kemudahan bahkan mempermudah para developer aplikasi dalam mengembangkan aplikasinya. Firebase alias *BaaS (Backend as a Service)* merupakan solusi yang ditawarkan oleh Google untuk mempercepat pekerjaan *developer*.

Firebase merupakan *NoSql Database* oleh karena itu Firebase tidak menggunakan struktur *Database* dengan tabel-tabel seperti *relational Database*.struktur dari sistem Firebase sendiri dapat dilihat pada gambar di bawah ini.



Gambar 2. 6 Skema Firebase [9]

Alasan penggunaan Firebase pun oleh karena Firebase ini cocok digunakan aplikasi ini yang akan membantu proses pembuatan fitur *map* / peta dan melacak lokasi pengguna lebih mudah, kelebihan lain menggunakan Firebase-pun ada sebagai berikut:

- Tersedia banyak fitur yang tersedia karena di *support* oleh google seperti *in app messaging, cloud messaging, google analytics*, dan lain-lain
- *Multi-Platform*
- *Real-Time Database*

2.10 Penelitian Terdahulu

Tabel 2. 4 Penelitian Terdahulu

No	Nama Jurnal,Vol,No	Authors	Hasil Penelitian	Kontribusi di penelitian ini
1	Perancangan Aplikasi Berbasis Android untuk Manajemen Proyek Reparasi Kapal (2017)	Marlen Jenri Hutapea, Triwilaswandio Wuruk Pribadi, dan Imam Baihaqi	Pada aplikasi ini terdapat dua otoritas yang memiliki fungsi berbeda, yaitu admin dan user.Sistem dalam android dapat menunjukkan bahwa seorang PM terintegrasi dengan pihak lain dalam proyek. Pengecekan terhadap kemajuan pekerjaan dan mutu proyek dapat dilakukan dengan menggunakan aplikasi ini	Referensi merancang aplikasi *mobile* yang nyaman digunakan dalam lapangan.

No	Nama Jurnal,Vol,No	Authors	Hasil Penelitian	Kontribusi di penelitian ini
2	Aplikasi Tracking Pada Sistem Pengiriman PT. Hijau Murni Alami(2016)	Salim, Arabella Margaret	Sistem *tracking* ini dapat membantu perusahaan PT. Hijau Murni Alami dalam menyelesaikan masalah pengirimannya. Karena dengan menerapkan sistem tracking yang tidak menggunakan third party ini memungkinkan pembeli untuk melihat secara langsung posisi barang mereka pada map dengan GPS yang disediakan aplikasi tracking tersebut.	Referensi fitur tracking pada aplikasi.
3	Rancang Bangun Aplikasi Pencatatan Keuangan Pribadi dengan Menggunakan Metode Gamifikasi Berbasis Android (2019)	Juwono, Firdayanti	Aplikasi pencatatan keuangan pribadi dengan metode gamifikasi menggunakan Marczewski Gamification Framework telah berhasil dirancang dan dibangun. game mechanics yang mencakup points, levels, leaderboards, badges, onboarding, dan challenges.	Referensi membuat aplikasi pencatatan sebuah pengeluaran *platform* android.