BAB II

LANDASAN TEORI

2.1 Konsep Dasar Aplikasi

2.1.1 Pengertian Rancang Bangun

Rancang menurut Sutabri (2012, hal.26) adalah kegiatan yang memiliki tujuan untuk mendesain sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapi perusahaan yang diperoleh dari pemilihan alternatif sistem yang terbaik.

Bangun menurut Pressman (2010, hal.89) adalah kegiatan menciptakan sistem baru maupun mengganti atau memperbaiki sistem yang telah ada secara keseluruhan".

Rancang Bangun menurut Bambang (2013, hal.27) adalah proses pembangunan sistem untuk menciptakan sistem baru maupun mengganti atau memperbaiki sistem yang telah ada baik secara keseluruhan maupun hanya sebagian.

Berdasarkan pengertian diatas, peneliti menyimpulkan bahwa rancang bangun adalah penggambaran atau sketsa dari sistem untuk membuat sistem baru atau memperbaharui sistem sebelumnya.

2.1.2 Pengertian Aplikasi

Aplikasi menurut Jogiyanto (1999:12) adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian sehingga komputer dapat memproses input menjadi output.

2.1.3 Pengertian Masjid

Menurut Sidi Gazalba (1975), masjid secara harfiah adalah tempat sembahyang, tetapi dalam bahasa Arab berati tempat sujud, karena berasal dari kata sajadah, sebagai tempat sujud,

masjid memiliki makna lebih luas, bukan sekedar gedung, sebab dimanapun umat Islam bisa melaksanakan sujud atau penghambaan kepada Allah SWT.

Secara bahasa, kata masjid adalah tempat yang dipakai untuk bersujud. Kemudian maknanya meluas menjadi bangunan khusus yang dijadikan orang-orang untuk tempat berkumpul menunaikan shalat berjama'ah. Az-Zarkasyi berkata, "Manakala sujud adalah perbuatan yang paling mulia dalam shalat, disebabkan kedekatan hamba Allah kepada-Nya di dalam sujud, maka tempat melaksanakan shalat diambil dari kata sujud (yakni masjad = tempat sujud). Mereka tidak menyebutnya (tempat ruku') atau yang lainnya. Kemudian perkembangan berikutnya lafazh masjad berubah menjadi masjid, yang secara istilah berarti bengunan khusus yang disediakan untuk shalat lima waktu. Berbeda dengan tempat yang digunakan untuk shalat 'Id atau sejenisnya (seperti shalat Istisqa') yang dinamakan (mushallaa = lapangan terbuka yang digunakan untuk shalat 'Id atau sejenisnya). Hukum-hukum bagi masjid tidak dapat diterapkan pada mushalla.

"Kata "Masjid" berasal dari kata sajada-sujud yang berarti patuh, taat, serta tunduk penuh hormat, takzim. Sujud dalam syariat yaitu berlutut, meletakkan dahi kedua tangan ke tanah adalahn bentuk nyata dari arti kata tersebut. Oleh karena itu bangunan yang dibuat khusus untuk sholat disebut masjid yang artinya tempat untuk sujud" (Shihab, 1997:459). "Masjid sekurang-kurangnya mempunyai tiga tinjauan makna yaitu : Pertama, berkaitan dengan aspek individu adalah terciptanya manusia yang beriman. Kedua, berkaitan dengan aspek sosial adalah membentuk umat yang siap menjalankan kehidupan dalam berbagai situasi atau kondisi yang dihadapi dan mampu hidup bermasyarakat dalam arti yang luas, berbangsa dan bernegara. Yang terpenting dalam aspek ini adalah kepribadian (akhlak) sebagai basis dinamik bangunan sosial yang kokoh. *Ketiga*, berkaitan dengan aspek fisik-bangunan adalah sebagai pembuktian ketauhidan, kekokohan jalinan sosial yang memiliki sikap konstruktif dan produktif" (L.H.Hasibuan,2002:8-9).

2.2 Pengertian *Mobile*

2.2.1 Pengertian *Android*

Android menurut Nazruddin (2012) merupakan sistem operasi yang digunakan pada telepon pintar dan komputer tablet berbasis Linux yang terdiri dari sistem operasi, *middleware,* dan aplikasi utama. Seperti halnya Linux, *Android* juga menyediakan sebuah sumber terbuka atau biasa disebut *Open Source* yang dapat digunakan oleh para pengembang untuk membuat aplikasi mereka sendiri.

Sejarah singkat, *Android* Inc. adalah perusahaan yang pertama kali mengembangkan *Android* sebelum akhirnya dibeli oleh *Google* pada tahun 2005. Pada saat pertama kali dirilis sistem operasi *Android* pada tahun 2007 bersamaan itu didirikannya pula *Open Handset Alliance (OHA)* dari beberapa perusahaan piranti keras, piranti lunak, dan perusahaan telekomunikasi dengan tujuan untuk lebih mengembangkan *Android*.

Tujuan pertama kali dikembangkan sistem operasi *Android* adalah untuk perangkat kamera. Namun pasar untuk perangkat itu tidaklah terlalu besar, sehingga projek pengembangan *Android* dialihkan lebih banyak untuk telepon pintar atau *smartphone.* Sampai saat ini *Android* menjadi rajanya sistem operasi untuk telepon pintar dan komputer tablet, karena banyak sekali *vendor* yang mengembangkan produknya dengan menggunakan sistem operasi *Android*. Sistem operasi *Android* juga memiliki kode nama yang unik yang diberikan kepada setiap versi dari *Android*.

2.2.2 Flutter dan Pemrograman Dart

Menurut Budi Raharjo (2019:1) dalam bukunya yang berjudul Pemrograman *Android* dengan Flutter : "Flutter adalah *Software Development Kit* (SDK) buatan *Google* yang berfungsi untuk membuat aplikasi *mobile* menggunakan bahasa pemrograman Dart, baik untuk *Android* maupun *iOS* . Dengan Flutter, aplikasi *Android* dan *iOS* dapat dibuat menggunakan basis kode dan bahasa pemrogaman yang sama yaitu Dart, bahasa pemrograman yang juga diproduksi oleh *Google* pada tahun 2011."

Menurut Budi Raharjo (2019:23), Dart adalah sebuah bahasa pemrograman yang dikembangkan oleh *Google*, dirancang oleh Lars Bak dan Kasper Lund. Dart pertama kali dikenalkan pada 10 Oktober 2011. Versi 1.0 dari bahasa pemrograman ini baru dirilis pada bulan November 2013. Versi stabil terbaru adalah Dart 2.1, yang dirilis pada tanggal 15 November 2018. Dart dapat digunakan untuk membuat aplikasi *server* (berbentuk *commandline interface*), web, maupun *mobile* (*Android* dan *iOS*).

2.2.3 *Google Maps API*

Menurut Elian dkk. (2012), *Google Maps API* merupakan javasript *library* hasil pengembangan *Google Maps*, yaitu layanan aplikasi dan teknologi pemetaan berbasis web oleh *Google* yang bersifat gratis. Dengan *library* yang berbentuk *javascript* ini, dimungkinkan untuk memodifikasi peta yang ada di *Google Maps* sesuai dengan kebutuhan pengembangan.

2.2.4 Global Positioning System (GPS)

GPS atau *Global Positioning System*, merupakan sebuah alat atau sistem yang dapat digunakan untuk menginformasikan penggunanya dimana lokasinya berada (secara *global*) di permukaan bumi yang berbasiskan satelit. Data dikirim dari satelit berupa sinyal radio dengan data digital. Dimanapun pengguna tersebut berada, maka *GPS* bisa membantu menunjukan arah. Layanan *GPS* ini tersedia gratis.

Menurut (Winardi, 2006) GPS adalah sistem untuk menentukan letak di permukaan bumi dengan bantuan penyelarasan (*synchronization*) sinyal satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke Bumi. Sinyal ini diterima oleh alat penerima di permukaan, dan digunakan untuk menentukan letak, kecepatan, arah, dan waktu. Sistem yang serupa dengan GPS antara lain GLONASS Rusia, Galileo Uni Eropa, IRNSS India.

2.2.5 Haversine Formula

Formula Haversine menurut Yulianto, dkk (2018, hal.16) adalah persamaan penting dalam sistem navigasi, nantinya Formula Haversine akan menghasilkan jarak terpendek antara dua titik, misalnya pada bola yang diambil dari garis bujur (*longitude*) dan garis lintang (*latitude*). Formula ini pertama kali ditemukan oleh Jamez Andrew ditahun 1805, dan digunakan pertama kali oleh Josef de Mendoza y Ríos di tahun 1801. Istilah haversine ini sendiri diciptakan pada tahun 1835 oleh Prof. James Inman. Josef de Mendoza y Ríos menggunakan haversine pertama kali dalam penelitiannya tentang “Masalah Utama Astronomi Nautical“, Proc. Royal Soc, Dec 22. 1796. Haversine digunakan untuk menemukan jarak antar bintang.

Formula Haversine adalah persamaan yang digunakan dalam navigasi, yang memberikan jarak lingkaran besar antara dua titik pada permukaan bola (bumi) berdasarkan bujur dan lintang. Formula Haversine merupakan suatu metode untuk mengetahui jarak antar dua titik dengan memperhitungkan bahwa bumi bukanlah sebuah bidang datar namun adalah sebuah

bidang yang memilki derajat kelengkungan. Penggunaan rumus ini mengasumsikan pengabaian efek ellipsoidal, cukup akurat untuk sebagian besar perhitungan, juga pengabaian ketinggian bukit dan kedalaman lembah di permukaan bumi.

Berikut adalah rumus haversine:

Δlat = lat2- lat1

Δlong = long2- long1

a = sin2(Δlat/2) +

cos(lat1).cos(lat2).sin2(Δlong/2)

c = 2atan2(,)

d = R.c

Dimana :

R = jari-jari bumi sebesar 6371(km)

Δlat = besaran perubahan *latitude*

Δlong = besaran perubahan *longitude*

c = kalkulasi perpotongan sumbu

d = jarak (km)

1 derajat = 0.0174532925 radian

2.3 Peralatan Pendukung Sistem (*Tools System*)

2.3.1 UML (*Unified Modelling Language*)

UML (*Unified Modeling Language*) menurut Nugroho (2010, hal.6) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma (berorientasi objek).” Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

Berdasarkan pendapat yang dikemukakan Nugroho tersebut dapat ditarik kesimpulan bahwa UML adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk

menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis Objek (*Object Oriented programming*).

2.3.2 Model-Model Diagram UML

1. *Use Case* Diagram

Use Case diagram menurut Yasin (2012, hal.238) adalah gambar dari beberapa atau seluruh aktor dan *Use Case* dengan tujuan mengenali interaksi mereka dalam suatu sistem. *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 antara *actor* dengan sistem. *Use Case* menggambarkan kata kerja seperti *login* ke sistem, *maintenance user* dan sebagainya. Oleh karena itu, *Use Case* diagram dapat membantu menganalisa kebutuhan suatu sistem.

2. Skenario

Skenario menurut Yasin (2012, hal.238) adalah sebuah dokumentasi terhadap kebutuhan fungsional dari sebuah sistem. *Form* skenario merupakan penjelasan penulisan *Use Case* dari sudut pandang actor.

3. *Activity* Diagram

Activity Diagram menurut Fowler (2005, hal.163) adalah teknik untuk menggambarkan logika *procedural*, proses bisnis, dan jalur kerja. Dalam beberapa hal, *Activity* diagram memainkan peran mirip diagram alir, tetapi perbedaan prinsip antara notasi diagram alir adalah *Activity* diagram mendukung behavior parallel. Node pada sebuah *Activity* diagram disebut sebagai *action*, sehingga diagram tersebut menampilkan sebuah *Activity* yang tersusun dari *action*.

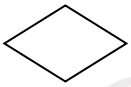
a. Simbol *Use Case* Diagram

Tabel 2.1 Simbol *Use Case* Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	*Actor* adalah pengguna sistem. *Actor* tidak terbatas hanya manusia saja, jika sebuah sistem berkomunikasi dengan aplikasi lain dan membutuhkan *input* atau memberikan *output*, maka aplikasi tersebut juga bisa dianggap sebagai *actor*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Association*	Asosiasi digunakan untuk menghubungkan *actor* dengan *Use Case*. Asosiasi digambarkan dengan sebuah garis yang menghubungkan antara *Actor* dengan *Use Case*.
4		*System Boundary*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
5		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
6	<<include>>	*Include*	Melakukan yang harus terpenuhi agar sebuah *event* dapat terjadi, dimana pada kondisi ini sebuah *Use Case* adalah bagian dari *Use Case* lainnya.
7	<<extend>>	*Extend*	Menspesifikasikan bahwa *Use Case* target memperluas perilaku dari *Use Case* sumber pada suatu titik yang diberikan.

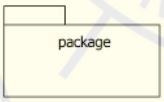
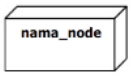
b. Simbol *Activity* Diagram

Tabel 2.2 Simbol *Activity* Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
2		*Initial Node*	Bagaimana objek dibentuk atau diawali.
3		*Actifity Final Node*	Bagaimana objek diakhiri
4		*Decission*	Pilihan untuk mengambil keputusan dan diakhiri kondisi
5		*Transition*	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya

4. *Deployment* Diagram

Menurut Sukamto dan Shalahuddin (2014:154) pada diagram *deployment* atau d*eployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal seperti sistem tambahan dan sistem *client/server*.

Simbol	Deskripsi
Package 	package merupakan sebuah bungkusan dari satu atau lebih *node*
Node 	biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika di dalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen
Kebergantungan / *dependency* 	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai
Link	relasi antar *node*

Gambar 2.1 Simbol *deployment* diagram

2.4 Metode Pengembangan Sistem

2.4.1 Waterfall

Menurut Pressman (2015:42), model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Nama model ini sebenarnya adalah *"Linear Sequential Model"*. Model ini sering disebut juga dengan *"classic life cycle"* atau metode waterfall. Model ini termasuk ke dalam model *generic* pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Fase-fase dalam *Waterfall Model* menurut referensi Pressman :

a. *Communication* (*Project Initiation & Requirements Gathering*)

Sebelum memulai pekerjaan yang bersifat teknis, sangat diperlukan adanya komunikasi dengan *customer* demi memahami dan mencapai tujuan yang ingin dicapai. Hasil dari komunikasi tersebut adalah inisialisasi proyek, seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendefinisikan fitur dan fungsi *software*. Pengumpulan data-data tambahan bisa juga diambil dari jurnal, artikel, dan internet.

b. *Planning* (*Estimating, Scheduling, Tracking*)

Tahap berikutnya adalah tahapan perencanaan yang menjelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko- resiko yang dapat terjadi, sumber daya yang diperlukan dalam membuat sistem, produk kerja yang ingin dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan *tracking* proses pengerjaan sistem.

c. *Modeling* (*Analysis & Design*)

Tahapan ini adalah tahap perancangan dan permodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur *software*, tampilan *interface*, dan algoritma program. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan.

d. *Construction* (*Code & Test*)

Tahapan *Construction* ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk/bahasa yang dapat dibaca oleh mesin. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem dan juga kode yang sudah dibuat. Tujuannya untuk menemukan kesalahan yang mungkin terjadi untuk nantinya diperbaiki.

e. *Deployment* (*Delivery, Support, Feedback*)

Tahapan *Deployment* merupakan tahapan implementasi *software* ke *customer*, pemeliharaan *software* secara berkala, perbaikan *software*, evaluasi *software*, dan pengembangan *software* berdasarkan umpan balik yang diberikan agar sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya. (Pressman,

2015:17)