BAB II

LANDASAN TEORI

Untuk mendukung pembuatan laporan Tugas Akhir ini, maka perlu dikemukakan hal-hal atau teori-teori yang berkaitan dengan lingkup pembahasan sebagai landasan dalam pembuatan laporan ini.

2.1 Konsep Dasar Sistem

A. Sistem

1. Definisi Sistem

Oktafianto & Muslihudin (2016), dalam bukunya yang berjudul Analisa dan Perancangan Sistem Informasi Menggunakan Model Terstruktur dan UML. “Sistem adalah sekumpulan komponen atau jaringan kerja dari prosedur-prosedur yang saling berkaitan dan saling bekerja sama membentuk suatu jaringan untuk mencapai sasaran atau tujuan tertentu.”

Menurut (Tyoso, 2016, p. 1) sistem adalah suatu kumpulan dari komponen komponen yang membentuk satu kesatuan.

Berdasarkan pengertian diatas dapat disimpulkan bahwa sistem adalah sebagian prosedur yang saling berkaitan dan saling terhubung untuk mencapai beberapa sasaran atau tujuan dengan maksud tertentu.

2. Pemograman Berorientasi Objek (OOP)

Menurut Andi dalam (Siddik & Sirait, 2019) Pemrograman Berorientasi Objek (OOP) adalah suatu cara baru dalam berpikir serta berlogika dalam menghadapi masalah-masalah yang akan dicoba atasi dengan bantuan komputer, dimana setiap objek adalah entitas tunggal yang memiliki kombinasi struktur data dan fungsi

tertentu. Sedangkan objek adalah orang, tempat, benda, kejadian, atau konsep-konsep yang ada di dunia nyata yang penting pada suatu aplikasi misalnya, objek sebuah benda seperti mesin, gedung, komputer, mobil dan sebagainya. Objek sebuah kejadian seperti pembayaran uang pendidikan, registrasi biodata siswa, membaca buku dan sebagainya. Dengan kata lain objek merupakan sesuatu yang dapat dilihat, disentuh, dirasakan, diraba, untuk mendapatkan manfaat.

B. Program

1. Pengertian Program

Menurut Raharjo dalam (Yulia, 2017) program adalah ”perangkat lunak (*software*) yang sebenarnya merupakan tuntunan instruksi yang ditulis dalam bentuk kode–kode menggunakan bahasa pemrograman tertentu dan telah dikompilasi dengan menggunakan compiler yang sesuai”.

Sedangkan menurut Kadir dalam (Fadallah & Rosyida, 2018) “Program adalah kumpulan instruksi yang digunakan untuk mengatur komputer agar melakukan suatu tindakan tertentu”. Jadi program adalah kumpulan instruksi yang ditulis dalam bentuk kode-kode menggunakan bahasa pemrograman tertentu yang digunakan untuk mengatur komputer dengan maksud untuk melakukan suatu tindakan tertentu.

2. Bahasa pemograman (Visual Basic.Net)

Menurut (Ruli, 2017) “Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasi lainnya dalam bentuk aplikasi *consol*e, aplikasi *Windows*, ataupun aplikasi *Web*. Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe”.

Disampaikan Ricyanto “Visual Basic.Net adalah generasi penerus Visual Basic 6 dari Microsoft. Dengan Visual Basic.Net, dapat membangun aplikasi *Windows*, *Web service* dan aplikasi web dengan *ASP.NET* secara cepat dan mudah. Aplikasi yang dibuat dengan Visual Basic.Net dibangun diatas *service common language runtime* sehingga memiliki keunggulan dari .*Net Framework*”. (Dharmawan, 2017).

Sehingga dapat disimpulkan Visual Basic. Net adalah sebuah bahasa pemograman yang beorientasi pada object untuk mendirikan dan mengembangkan aplikasi dengan bahasa pemograman yang Sederhana dan mudah di pahami.

C. Basis Data

Menurut Martin, dalam (Husda & Wangdra, 2016) “Basis data adalah suatu kumpulan data yang disimpan secara bersama-sama pada suatu media, tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data dengan cara-cara tertentu sehingga mudah untuk digunakan dan ditampilkan kembali, dapat digunakan untuk satu atau lebih program aplikasi secara optimal, data dapat disimpan tanpa mengalami ketergantungan pada program yang akan menggunakan, serta disimpan sedemikian rupa sehingga penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol. Sehingga basis data sendiri dapat disimpulkan sebagai :

1. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersamaan sedemikian rupa dan tanpa pengulangan, untuk memenuhi kebutuhan.
3. Kumpulan file/table/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

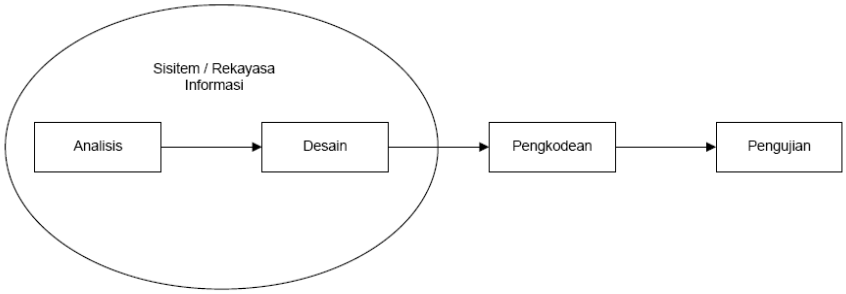
Menurut (Arizona, 2017) “MySQL adalah bahasa yang digunakan untuk mengelola data pada RDBMS”.

Sedangkan menurut (Risdiansyah, 2017) “MySQL merupakan *database server* yang bersifat multiuser dan *multi-threaded*. SQL adalah bahasa *database* standar yang memudahkan penyimpanan, pengubahan dan akses informasi. Pada MySQL dikenal istilah *database* dan tabel. Tabel adalah sebuah struktur data dua dimensi yang terdiri dari baris-baris record dan kolom”.

Dari pendapat-pendapat diatas dapat disimpulkan MySQL adalah sebuah perangkat lunak media penyimpanan data berupa SQL yang terdiri dari baris-baris record dan kolom.

D. Model Pengembangan Perangkat Lunak (Model *Waterfall*)

Pada pengembangan sistem ini, penulis menggunakan model *waterfall* (air terjun) yang mengacu pada ilmu rekayasa perangkat lunak, yaitu sebagai berikut.



Sumber : (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 29)

Gambar II.1 Model *Waterfall*

Menurut (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 28), Model *Software Development Life Cycle* (SDLC) air terjun *(waterfall)* sering juga di sebut model sekuensial linier *(sequential linear)* atau alur hidup klasik *(clasic life cycle).*

Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*). Adapun metode air terjun menurut Rosa dan Shalahuddin (2018:29) yaitu:

1. Analisis Kebutuhan Perangkat Lunak

Proses pengumpulan kebutuhan dilakukan secara insentif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. Desain

Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan Kode Program

Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

Pengujian fokus pada perangkat lunak secara dari segi *logic* dan fungsional serta memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung (*support*) atau Pemeliharaan (*maintenance*)

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.2. Teori Pendukung

A. Entity Relationship Diagram

1. Definisi *Entity Relationship Diagram* (ERD)

Fathansyah dalam (Fauzi et al., 2019) *Entity Relationship Diagram* (ERD) adalah model *entity relationship* yang berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing masing dilengkapi dengan atribut-atribut yang merepresentasikan seluruh fakta dari dunia nyata yang ditinjau sehingga dapat digambarkan dengan lebih sistematis.

Menurut (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 50) menyatakan bahwa "*Entity Relationship Diagram* **(**ERD) dikembangkan berdasar teori himpunan dalam bidang matematik, ERD digunakan untuk pemodelan basis data relasional, ERD digununakan untuk permodelan basis data relasional".

2. Komponen ERD

Berikut adalah simbol- simbol yang digunakan pada ERD dengan Notasi Chen:

Tabel II.1 Simbol ERD

No	Simbol	Keterangan
1	Entitas/ *Entity* nama_entitas	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data; benda yang memiliki data dan harus disimpan datanya adar dapat diakses oleh aplikasi komputer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel
2	Atribut Nama_atribut	*Field* atau kolom data yang butuh disimpan dalam suatu entitas
3	Atribut Kunci Primer Nama_Kunci Primer	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan; biasanya berupa id; kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama).
4	Antribut Multinilai/*multivalve* Nama_atribut	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.
5	Relasi Nama_relasi	Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja.
6	Asosiasi / association _____	Penghubung antara relasi dan entitas di mana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan *one to many* menghubungkan entitas A dan entitas B.

Sumber : (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 50)

3. Pengertian LRS (*Logical Record Structure*)

Menurut Friyadie dalam (Taufik, 2017) mengemukakan bahwa "sebelum *table* dibentuk dari *field* atau atribut entitas secara fisik atau level internal, maka harus dibuatkan suatu bentuk relational model yang dibuat secara *logic* atau level *external* dan konsep, dari pernyataan tersebut dibutuhkan yang disebut dengan *Logical Record Structure* (LRS)". Dalam pembuatan *Logical Record Structure* (LRS) terdapat tiga hal yang dapat mempengaruhi, yaitu : *one-to-one, one-to-many, many-to-many.*

B. Unified Modeling Language

1. Definisi *Unified Modeling Language* (UML)

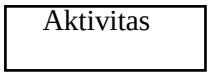
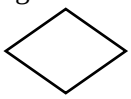
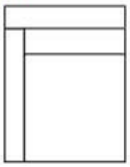
(A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 133), menjelaskan tentang pengertian UML sebagai berikut :

> UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML menyediakan serangkaian gambar dan diagram yang sangat baik. Beberapa diagram memfokuskan diri pada ketangguhan teori *objectoriented* dan sebagian lagi memfokuskan pada detail rancangan dan konstruksi. Semua dimaksudkan sebagai sarana komunikasi antar *team programmer* maupun dengan pengguna.

2. *Activity Diagram*

Menurut (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 161) "Diagram Aktivitas atau *Activity Diagram Workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak"

Tabel II.2 Simbol *Activity Diagram*

No	Simbol	Deskripsi
1	Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3	Percabangan/*Decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4	Penggabungan/*Join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5	Status Akhir 	Status akhir yang dilakukan oleh sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6	*Swimlane* 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Sumber : (A. S., Rosa. Shalahuddin & Sukamto, 2018)

3. *Use Case Diagram*

Menurut (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 155)“Diagram *use case* merupakan permodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *use case* mendeskripsikan sebuah imteraksi antara satu atau lebih *actor* dengan sistem informasi yang akan dibuat.

Tabel II.3 Simbol *Use Case Diagram*

Simbol	Deskripsi
Use Case Nama Use Case	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal frasa nama *use case*
Aktor/*Actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frasa nama aktor.
Asosiasi/ *Assosiasion* _____	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Include - - - - - - - - - -▶	Relasi *use case* dimana *use case* yang di *include* selalu membutuhkan *use case* utama yang di *include*, arah panah *include* dari *use case* utama menuju *use case* pendukung
Ekstensi / *Extend* ◀- - - - - - - - - -	Relasi *use case* tambahan kesebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
Generalisasi / *Generalitation* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya

Sumber : (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 156)

4. *Class Diagram*

Menurut (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 161) “Diagram Kelas atau *Class diagram* menggambarkan Struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem, kelas memiliki apa saja yang disebut atribut dan metode operasi.

Tabel II.4 Simbol *Class Diagram*

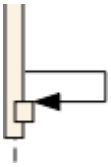
No	Simbol	Deskripsi
1	Kelas 	Kelas pada struktur sistem.
2	Antarmuka / *interface* 	Sama dengan konsep *interface* dalam pemograman berorientasi objek.
3	Asosiasi / a*ssociation* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4	Asosiasi berarah / *directed* a*ssociation* 	Relasi antar kelas denganmakna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
5	Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
6	Kebergantungan/ *dependensi* 	Relasi antar kelas dengan makna kebergantungan antar kelas.
7	Agregasi / *aggregation* 	Relasi antar kelas dengan makna semua-bagian *(whole-part).*

Sumber : (A. S., Rosa. Shalahuddin & Sukamto, 2018, p. 146)

5. *Sequence* Diagram

(Hendini, 2016) menyatakan bahwa Diagram Urutan (*Sequence Diagram*) *Sequence* Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *Sequence Diagram* yaitu:

Tabel II.5 Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan form *entry* dan form cetak
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek
	Message, simbol mengirim pesan antar class
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
	*Activation*, mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi

	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

Sumber : (Hendini, 2016)