BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian tentang sistem informasi administrasi keuangan sekolah sebelumnya telah dilakukan dengan judul "Aplikasi Sistem Administrasi Keuangan Sekolah Pada Madrasah Ibtidaiyah Unggulan Ar-Ridho Tajurhalang". Pengolahan data administrasi keuangan Madrasah Unggulan Ar-ridho masih menggunakan cara manual sehingga dalam pelaksanaannya kurang optimal. Beberapa kendala yang timbul sebagai akibat dalam pengolahan data secara manual tersebut adalah pembuatan atau penyajian dan tidak dapat dilakukan dengan cepat, kemungkinan kesalahan proses pengolahan data dapat terjadi, sehingga mengakibatkan tidak terkontrolnya data pembayaran spp, ujian, dan pembayaran daftar ulang. Aplikasi ini dibuat untuk mempermudah petugas tata usaha dalam pengolahan data administrasi keuangan sekolah. Aplikasi sistem ini dibangun menggunakan bahasa pemograman Java dan menggunakan metode *grounded research* [3].

Penelitian lain telah dilakukan dengan judul "Perancangan Sistem Aplikasi Administrasi Keuangan Sekolah Pada SMK Adi Luhur 2 Jakarta". Proses pencatatan dan pendataan pembayaran administrasi keuangan sekolah masih sangat manual, sehingga dapat terjadinya kesalahan-kesalahan dan juga kurangnya keamanan pada data serta pembuatan laporan yang membutuhkan waktu yang lama. Sistem ini dirancang dengan tujuan untuk menghasilkan sistem administrasi keuangan sekolah yang terkomputerisasi dengan aplikasi berbasis desktop menggunakan metode pengembangan *waterfall* dan bahasa pemograman Java [4].

Penelitian lain juga telah dilakukan dengan judul "Rancang Bangun Sistem Informasi Administrasi Monitoring Administrasi Keuangan Siswa Berbasis Web (Studi Kasus: SD Plus Mutiara Ilmu Pandaan)". SD Plus Mutiara Ilmu Pandaan memiliki permasalahan dalam kegiatan administrasi sekolah dengan cara pembayarannya dititipkan guru atau pihak yang tidak bersangkutan yang sebenarnya tidak boleh dilakukan dalam kegiatan administrasi, serta pencatatan dilakukan secara manual yaitu ditulis dalam kartu pembayaran, buku besar dan menggunakan *Microsoft Excel*, dimana penelusuran data membutuhkan waktu lama dan ketidakakuratan data karena kesalahan pencatatan dan bukti rekap hilang. Sistem ini dirancang dan dibangun

berbasis *web* dan aplikasi pendukung berupa *mobile* untuk memudahkan pembayaran SPP, monitoring transaksi administrasi setiap harinya, serta monitoring cicilan atau tunggakan administrasi. Sistem ini dibangun dengan bahasa pemograman *PHP* dan *Android WebView* dan pengembangan sistem yang digunakan yaitu *waterfall* [5].

Penelitian yang akan dilakukan penulis tentu akan berbeda dengan yang ada sebelumnya. Dalam penelitian ini penulis akan berfokus pada pengelolaan pemasukan data keuangan yang diperoleh dari pembayaran biaya sekolah dengan berbasis *web* dan menggunakan metode pengembangan sistem yaitu *prototype* serta akan dilengkapi dengan fitur *payment gateway* yang merupakan teknologi yang digunakan untuk mempermudah melakukan proses transaksi pembayaran secara *online.* Selain itu juga akan menampilkan profil dan informasi sekolah. Sistem ini akan dibangun menggunakan bahasa pemograman *PHP* dan *framework laravel,* serta dengan tujuan untuk mempermudah dalam pengelolaan pemasukan data keuangan sekolah.

2.2 Landasan Teori

2.1.1. Sistem Informasi

Sistem informasi adalah suatu sistem yang terdiri dari kumpulan komponen sistem, yaitu *software, hardware,* dan *brainware* yang mengolah data menjadi sebuah *output* berupa informasi yang berguna untuk mencapai tujuan tertentu dalam suatu organisasi [6]. Sistem informasi adalah suatu sistem pada suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasional, bersifat manajerial, dan kegiatan strategi dari suatu organisasi serta menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan [7].

2.1.2. Sistem Informasi Keuangan

Sistem informasi keuangan adalah sistem yang dirancang untuk menyediakan informasi mengenai arus uang bagi para pemakai di seluruh organisasi yang digunakan untuk memecahkan masalah-masalah keuangan [8]. Sistem informasi keuangan juga merupakan sistem yang memberikan informasi kepada orang atau kelompok baik di dalam perusahaan atau organisasi maupun di luar perusahaan atau organisasi mengenai pengelolaan keuangan [9].

2.1.3. Rekayasa Perangkat Lunak

Rekayasa perangkat lunak merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang dapat bekerja secara efisien menggunakan mesin [10]. Rekayasa perangkat lunak adalah suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak mulai dari tahap awal yaitu analisan kebutuhan pengguna, desain, pengodean, pengujian sampai dengan pemeliharaan sistem setelah dikembangkan [11]. Berikut ini metode dan *tools* pembangunan rekayasa perangkat lunak yang digunakan :

A. Metode Pengembangan Sistem

Metode yang digunakan untuk pengembangan sistem adalah model *prototype. Prototype* adalah versi awal dari sistem perangkat lunak yang digunakan untuk mendemostrasikan konsep-konsep, percobaan rancangan, dan menemukan lebih banyak masalah dan solusi yang memungkinkan. Sistem dengan model *prototype* memperbolehkan pengguna untuk mengetahui bagaimana sistem berjalan dengan baik. Model *prototype* mengharuskan pengembang perangkat lunak membuat sebuah *mockup* berupa model aplikasi, sangat cocok pada kondisi di mana pengguna tidak bisa menyajikan informasi secara jelas mengenai kebutuhan yang sesuai dengan keinginannya [12]. Adapun penjelasan dari tahapan metode *prototype*, yaitu [13] :

1. *Communication*
   Developer dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diinginkan dan gambaran bagian-bagian yang akan dibutuhkan.
2. *Quick Plan*
   Perancangan dilakukan cepat dan mewakili semua aspek *software* yang diketahui, spesifikasi untuk pengembangan berdasarkan kebutuhan sistem dan rancangan ini menjadi dasar pembuatan *prototype.*
3. *Modeling Quick Design*
   Pemodelan rancangan cepat berfokus pada representasi aspek *software* yang bisa dilihat *user.* Representasi model sistem yang akan dikembangkan seperti proses dengan perancangan menggunakan *Unified Modeling Language* (UML). Dalam tahap ini *prototype* dibangun dengan sistem rancangan sementara kemudian di evaluasi oleh klien apakah sudah sesuai dengan yang diinginkan atau masih perlu di evaluasi kembali. Setelah

sistem dianggap sesuai dengan apa yang diharapkan klien, langkah berikutnya adalah pembuatan aplikasi atau pengodingan dari rancangan sistem yang dibuat.

4. *Constuction of Prototype*
   Tahapan ini merupakan tahapan dimana semua rencana dan perancangan yang telah dilakukan diimplementasikan ke dalam bahasa pemograman.
5. *Deployment, Delivery & Feedback*
   Pada tahap ini keseluruhan sistem akan diuji untuk memastikan sistem berfungsi sebagaimana mestinya. Kemudian tahapan ini juga dibutuhkan untuk mendapatkan *feedback* dari klien sebagai hasil evaluasi dari tahapan sebelumnya dan implementasi dari sistem yang dikembangkan.

B. Metode Pengujian Sistem

Pengujian sistem memiliki tujuan untuk mengetahui performa dari sistem itu sendiri. Sehingga didapatkan sebuah hasil yang jelas apakah aplikasi sudah sesuai dengan yang direncanakan atau masih ada fungsionalitas sistem yang belum berfungsi [14]. *Black box testing* adalah pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dank ode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *black box testing* harus dibuat dengan kasus benar dan kasus salah [15].

C. Tools / Alat Bantu Penelitian

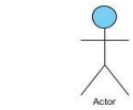
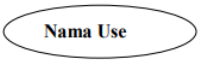
1. *Unified Modeling Language* (UML)

Unified Modeling Language atau UML adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek. Pemodelan sesungguhnya digunakan untuk penyederhanaan permasalahan yang kompleks menjadi lebih mudah dipahami [15]. UML (*Unified Modeling Language*) memiliki diagram-diagram yang digunakan dalam pembuatan aplikasi berorientasi objek, diantaranya [16] :

(a) ***Use Case Diagram***

Use Case Diagram merupakan diagram yang harus dibuat pertama kali saat pemodelan perangkat lunak berorientasi objek dilakukan. Dibawah ini adalah simbol yang digunakan untuk membuat *use case diagram*, antara lain :

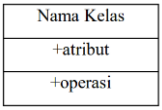
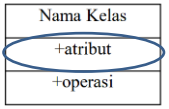
Tabel 2.1 Simbol-simbol *Use Case*

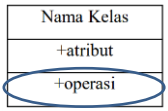
No	Simbol	Nama	Keterangan
1		*Actor*	Merupakan pengguna dari sistem. Penamaan aktor menggunakan kata benda
2		*Use Case*	Merupakan pekerjaan yang dilakukan oleh aktor. Penamaan *use case* dengan kata kerja
3		*Association* / Asosiasi	Hubungan antara aktor dengan *use case*
4		*Include*	Hubungan antara *use case* dengan *use case*. *Include* menyatakan bahwa sebelum pekerjaan dilakukan harus mengerjakan pekerjaan lain terlebih dahulu
5		*Extends*	Hubungan antara *use case* dengan *use case*. *Extends* menyatakan bahwa jika pekerjaan yang dilakukan tidak sesuai atau terdapat kondisi khusus, maka lakukan pekerjaan itu

(b) ***Class Diagram***

Class Diagram dibuat setelah *use case diagram* dibuat. Pada diagram ini harus menjelaskan hubungan apa saja yang terjadi antara suatu objek dengan objek lainnya sehingga terbentuklah suatu sistem aplikasi. Dibawah ini adalah simbol yang digunakan untuk membuat *class diagram*, yaitu :

Tabel 2.2 Simbol-simbol *Class Diagram*

No	Simbol	Nama	Keterangan
1		*Class*	Kelas mempresentasikan orang, tempat, atau objek lain yang diperlukan sistem untuk menyimpan informasi. Nama kelas biasanya ditulis dengan huruf tebal dan terletak di bagian kotak atas. Atribut kelas terletak di bagian kotak tengah dan operasi di kotak bawah. Meskipun demikian, kelas tidak secara spesifik menunjukkan operasi yang tersedia untuk semua kelas secara *eksplisit.*
2		Atribut	Atribut merupakan properti yang menggambarkan keadaan suatu objek. Dapat diturunkan dari atribut lain, ditampilkan dengan

			menempatkan garis miring sebelum nama atribut.
3		Operasi	Operasi merupakan representasi dari tindakan atau fungsi yang dapat dijalankan oleh sebuah kelas dalam pemograman.
4		*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity*

Relasi antar kelas mempunyai keterangan yang disebut dengan *multiplicity.*

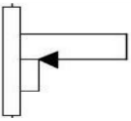
Tabel 2.3 *Multiplicity Class Diagram*

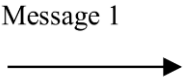
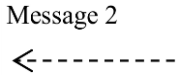
No	*Multiplicity*	Keterangan
1	1	Satu dan hanya satu
2	0..*	Boleh tidak ada atau 1 atau lebih
3	1..*	1 atau lebih
4	0..1	Boleh tidak ada, maksimal 1
5	n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimal 4

(c) ***Sequence Diagram***

Sequence Diagram adalah diagram yang dibuat untuk mengetahui alur dari interaksi antar objek. Isi dari *sequence diagram* harus sama dengan *use case* dan *class diagram.* Berikut adalah simbol dari *sequence diagram :*

Tabel 2.4 Simbol-simbol *Sequence Diagram*

No	Simbol	Nama	Keterangan
1		Actor	Entitas yang terlibat dalam skenario dan berinteraksi dengan sistem
2		*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
3		*Boundary Class*	Berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem
4		*Control Class*	Bertanggungjawab terhadap kelas-kelas terhadap objek yang berisi logika.
5		*Recursive*	Menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
6		Aktivasi	Menunjukkan masa hidup dari objek

7		*Lifeline*	Garis titik-titik yang terhubung dengan objek dan sepanjang *lifeline* terdapat *activation*
8		Pesan	Interaksi antara satu objek dengan objek lainnya. Objek dapat mengirimkan pesan ke objek lain. Interkasi antar objek ditunjukkan pada bagian operasi pada *class diagram*
9		*Return*	Pesan kembalian dari komunikasi antar objek

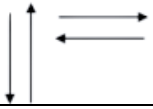
2. *Flowchart*

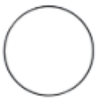
Flowchart merupakan sebuah diagram dengan simbol-simbol grafis yang menyatakan aliran algoritma atau proses yang menampilkan langkah-langkah yang disimbolkan dalam bentuk kotak, beserta urutan dengan menghubungkan masing-masing langkah tersebut menggunakan tanda panah. Diagram ini bisa memberi solusi langkah demi langkah untuk penyelesaian masalah yang ada di dalam proses atau algoritma tersebut [17]. Berikut akan dijelaskan mengenai simbol-simbol *flowchart* yang dibagi ke dalam 3 kategori, diantaranya [18] :

(a) Simbol Arus (*Flow Direction Symbols)*

Biasanya simbol yang termasuk ke dalam kategori ini digunakan sebagai simbol penghubung. Beberapa simbol yang termasuk ke dalam kategori ini, yaitu :

Tabel 2.5 Simbol Arus

No	Simbol	Nama	Keterangan
1		*Flow Direction / Connecting Li*ne	Berfungsi untuk menghubungkan simbol yang satu dengan yang

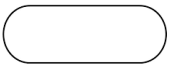
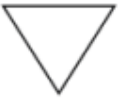
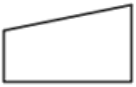
			lainnya, menyatakan arus suatu proses
2		*Communication Link*	Berfungsi untuk transmisi data dari satu lokasi ke lokasi lain
3		*Connector*	Digunakan untuk menyatakan sambungan dari proses yang satu ke proses berikutnya di halaman yang sama
4		*Offline Connector*	Digunakan untuk menyatakan sambungan dari proses yang satu ke proses berikutnya di halaman yang berbeda

(b) Simbol Proses (*Processing Symbols*)

Sesuai dengan namanya, simbol proses digunakan untuk menyatakan simbol yang berkaitan dengan serangkaian proses yang dilakukan. Berikut beberapa simbol yang termasuk ke dalam bagian proses, yaitu :

Tabel 2.6 Simbol Proses

No	Simbol	Nama	Keterangan
1		*Processing*	Digunakan untuk menunjukkan pengolahan yang akan dilakukan dalam komputer
2		*Manual Operation*	Digunakan untuk menunjukkan pengolahan yang tidak dilakukan oleh komputer

3		*Decision*	Digunakan untuk memilih proses yang akan dilakukan berdasarkan kondisi tertentu
4		*Predefined Process*	Digunakan untuk mempersiapkan penyimpanan yang sedang/akan digunakan dengan memberikan harga awal
5		Terminal	Digunakan untuk memulai atau mengakhiri program
6		*Offline Storage*	Berfungsi untuk menunjukkan bahwa data akan disimpan ke media tertentu
7		*Manual Input Symbol*	Digunakan untuk menginputkan data secara manual dengan *keyboard*

(c) Simbol I/O (*Input/Output*)

Simbol yang termasuk ke dalam bagian *input-output* berkaitan dengan masukan dan keluaran. Berikut beberapa simbol yang termasuk, yaitu :

Tabel 2.7 Simbol *Input / Output*

No	Simbol	Nama	Keterangan
1		*Input / Output*	Digunakan untuk menyatakan *input* dan *output* tanpa melihat

			jenisnya
2		*Punched Card*	Digunakan untuk menyatakan masukan dan keluaran yang berasal dari *card*
3		*Disk Storage*	Digunakan untuk menyatakan masukan dan keluaran yang berasal dari *disk*
4		*Magnetic Tape*	Digunakan untuk menyatakan masukan dan keluaran yang berasal dari pita magnetis
5		*Document*	Digunakan untuk menyatakan masukan dan keluaran yang berasal dari dokumen
6		*Display*	Digunakan untuk menyatakan masukan dan keluaran melalui layar monitor

2.1.4. Pemograman Berorientasi Objek

Pemograman Berorientasi Objek atau *Object Oriented Programming* (OOP) merupakan pemograman yang berorientasikan kepada objek, di mana semua data dan fungsi dibungkus dalam *class-class* atau *object-object.* Setiap *object* dapat menerima pesan, memroses data, mengirim, menyimpan, dan memanipulasi data. Beberapa *object* berinteraksi dengan saling memberikan informasi satu terhadap yang lainnya.

Masing-masing *object* harus berisikan informasi mengenai dirinya sendiri dan dapat dihubungkan dengan *object* lain. Penggunaan pemograman berorientasi objek sangat banyak sekali, misalnya java, php, perl, c#, dan lainnya. Dalam konsep pemograman berorientasi objek dikenal beberapa istilah umum, yaitu [19]:

a. Kelas (*class*)
Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama mungkin diciptakan oleh kelas tersebut. Sebuah kelas mempunyai sifaf (atribut), kelakuan (operasi/*method*), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan ke kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

b. Objek (*object*)
Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi dan mempunyai operasi yang dapat diterapkan.

c. Metode (*method*)
Method adalah fungsi atau prosedur yang dibuat oleh seorang *programmer* didalam suatu *class.* Pada sebuah *method* didalam sebuah kelas juga memiliki izin akses seperti atribut, yaitu *private, public,* dan *protected.* Sebuah kelas boleh memiliki lebih dari satu *method* dengan nama yang sama asalkan memiliki parameter masukan yang berbeda sehingga *compiler* atau *interpreter* dapat mengenali *method* mana yang dipanggil.

d. Atribut (*attribute*)
Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya. Atribut sebaiknya bersifat privat untuk menjaga enkasulapsi.

e. Enkapsulasi (*encapsulation*)
Pembungkusan atribut data dan layanan (operasioperasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

f. Pewarisan (*inheritance*)
Mekanisme yang memungkin suatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya

g. Polimorpisme (*polymorphism*)
Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

2.1.5. *Database*

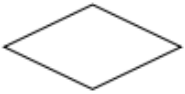
Database adalah sekumpulan informasi yang diatur supaya mudah dicari. Dalam arti umum *database* adalah sekumpulan data yang diproses dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan tepat [20]. *Database* adalah kumpulan dari tabel-tabel yang berisi data-data yang saling berkaitan [21].

A. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) merupakan diagram yang digambarkan dalam bentuk grafis dalam pembuatan *database* yang menghubungkan hubungan antar data yang satu dengan data yang lain. Fungsi ERD adalah untuk membantu pembuatan *database* dan memberikan gambaran terhadap kerja *database* yang akan dibuat [22]. Simbol ERD dapat dilihat pada Tabel 2.8.

Tabel 2.8 Simbol *Entity Relationship Diagram*

No	Simbol	Nama	Keterangan
1		*Entity*	Simbol yang menyatakan himpunan entitas ini bisa berupa suatu elemen lingkungan, sumber daya, atau transaksi yang begitu pentingnya bagi perusahaan sehingga didokumentasikan dengan data
2		*Attribute*	Simbol terminal ini untuk menunjukkan namanama attribut yang ada pada suatu *entity*

3		*Primary Key, Attribute*	Simbol attribut yang digaris bawahi, berfungsi sebagai key (kunci) diantara nama-nama atribut yang ada pada suatu *entity*
4		*Relationship*	Simbol ini menyatakan relasi ini digunakan untuk menunjukkan hubungan yang ada antara entiti yang satu dengan entiti yang lainnya
5		*Link*	Simbol berupa garis ini digunakan sebagai penghubung antara himpunan relasi dengan himpunan entitas dan himpunan entitas dengan atributnya

B. *Query*

1. DDL (*Data Definition Language*)

DDL (*Data Definition Language*) merupakan suatu perintah yang digunakan untuk menciptakan struktur data atau untuk membangun *database.* DDL mempunyai tugas untuk membuat untuk objek SQL dan menyimpan definisinya dalam tabel. Contoh objek tersebut yaitu *table*, *view*, dan *index*. DDL mempunyai fungsi untuk melakukan perubahan struktur tabel, mengubah dan nama tabel. Berikut perintah-perintah yang ada dalam golongan DDL [23] :

(a) *Create*, digunakan untuk membuat *database*, tabel, dan objek lain dalam *database*. Dibawah ini contoh *query* membuat *database* dan tabel :

CREATE DATABASE keuangan;

(b) *Alter*, digunakan untuk memodifikasi tabel, seperti mengubah nama tabel, *field*, dan menambah nama *field.* Contoh *query* untuk menambah *field* pada tabel di *database* :

ALTER TABLE users ADD alamat VARCHAR (40);

(c) *Drop*, digunakan untuk menghapus *database*, tabel, dan objek lain dalam *database*. Contoh *query* untuk menghapus *database*: *DROP DATABASE keuangan;*

2. DML (*Data Manipulation Language*)

DML (*Data Manipulation Language*) merupakan *database* yang digunakan untuk melakukan modifikasi dan pengambilan data pada suatu *database*. Pengolahan atau modifikasi ini meliputi:

(a) *Insert*, digunakan untuk melakukan penambahan data baru ke dalam sebuah tabel, contohnya:
INSERT INTO users (‘id’,’username’,’password’) VALUES (‘1’,’amanda’MD5(‘12345’));

(b) *Select*, digunakan untuk pengambilan data, contohnya: mengambil semua data dari sebuah tabel.
*SELECT * FROM users WHERE username = ‘amanda’;*

(c) *Update*, digunakan untuk melakukan perubahan data, contohnya:
UPDATE users SET username =’amanda’ WHERE id = ‘1’;

(d) *Delete*, digunakan untuk melakukan penghapusan data, contohnya:
DELETE FROM users WHERE username = ‘amanda’;

2.1.6. Administrasi Keuangan

Administrasi keuangan dapat diartikan sebagai suatu langkah-langkah yang dilakukan oleh organisasi untuk mengelola keuangannya [24]. Administrasi keuangan merupakan pengelolaan yang meliputi seluruh aktifitas yang berkaitan dengan keuangan untuk mencapai tujuan sebuah organisasi ataupun perusahaan tertentu. Sebagai proses pengelolaan yang melibatkan semua kegiatan yang berhubungan dengan keuangan, pembuatan laporan, dan pencapaian tujuan untuk kepentingan bersama [6].