BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Manfaat tentang penelitian ini telah diterapkan oleh Sri Murni, Raja Sabaruddin. Pada tahun 2018 dengan judul Pemanfaatan *QR Code* Dalam Pengembangan Sistem Informasi Kehadiran Siswa Berbasis *Web*, dibangun dengan tujuan untuk mengembangkan sistem informasi kehadiran siswa menggunakan *QR Code.* Menggunakan metode SDLC(*System Development Life Circle*) dengan model *waterfall*[1].

Penelitian tentang sistem presensi menggunakan *QR Code* sebelumnya telah dilakukan oleh Qurotul Aini , Untung Rahardja, Anggy Fatillah pada tahun 2018 dengan judul Penerapan *QR Code* sebagai Media Pelayanan Untuk Absensi pada *Website* berbasis *PHP Native*, dibangun untuk absensi kehadiran asisten lab supaya lebih efektif dan tidak dapat dimanipulasi. Nama sistem itu diberi nama Pensil (Penilaian Asisten Lab). Dalam sistem ini mereka menerapkan absensi menggunakan *QR Code*, proses absensi yang berjalan jika menggunakan Pensil hanya tinggal *create generate QR Code* pada sebuah *website* pensil.raharja.ac.id lalu tinggal *scan QR Code* pada perangkat yang telah di sediakan[2].

Penelitian selanjutnya juga menggunakan jurnal ilmiah sebagai referensi dari Reymon Rotikan pada tahun 2016 dengan judul Sistem Informasi Absensi Berbasis *Web* Untuk Kegiatan Konferensi, dibangun untuk pelaksanaan pada konferensi ilmiah di Universitas Klabat. Kegiatan pengambilan absen oleh para peserta masih dilakukan secara manual. Begitu juga dengan pembuatan laporan yang nantinya harus di rekap ke komputer menggunakan *Microsoft Office Excel*, lalu baru bisa menyusun laporan. Untuk pengembangan sistem yang digunakan disini adalah model *spiral* yang memungkinkan pengembangan sistem secara sistematis dan iterative untuk setiap fiturnya[3].

Dan untuk referensi lainnya ada dari Elin Herlina dan Taufik Hidayatulloh dengan judul Penerapan *QR Code* Untuk Sistem Absensi Siswa SMP berbasis *Web* di di SMP Negeri 11 Kota Sukabumi., kendala pada sistem sebelumnya yaitu terletak pada pengumpulan data hadir siswa, yang bagaimana sistem absensi bisa membantu proses pencatatan

data hadir dan memberikan laporan harian, bulanan, dan tahunan, maka sistem ini dibangun untuk memperbaiki dan mengembangkan sistem pengumpulan data hadir siswa yang sebelumnya. Dengan bahasa pemrograman *Ruby, Framework Ruby On Rails, Database PostgreSQL, Code Editor* yaitu *Sublime Text 3*[4].

Penerapan Sistem Presensi ini juga telah diterapkan oleh Ken Rio Agizki, Regiolina Hayami dan Harun Mukhtar dengan judul Penerapan *Quick Respons* (QR) *Code* berbasis *Web* di Puskesmas Payung Sekaki Pekanbaru. Data absensi pada puskesmas masih menggunakan kertas sebagai arsip data presensi pegawai, sehingga sering terjadinya kehilangan data dan data ganda yang membuat kekeliruan pada bagian kepegawaian. Dan juga timbul kecurangan dalam melakukan presensi dengan cara penitipan presensi. Sehingga sistem absensi menggunakan *QR Code* ini dibuat untuk mempermudah dalam melakukan absensi dan juga meminimalisir kecurangan dalam melakukan absensi. Untuk metode yagn digunakan yaitu model *waterfall*, untuk bahasa pemrograman PHP, *native* dan disisipi oleh *text Javascript,* CSS, *Bootstrap* dan lain-lain[5].

Sistem presensi ini diharapkan dapat digunakan, dengan pengembangan sistem metode SDLC *(System Development Life Circle)* dengan model *waterfall* berbasis *website*. Sistem ini direncanakan akan dibangun menggunakan bahasa pemrograman PHP *(Hypertext Preprocessor)* dan HTML *(HyperText Markup Language), Framework CodeIgniter 4*, dan implementasi database menggunakan *MySQL*. Dengan ini sistem presensi yang akan dikerjakan memiliki persamaan yaitu mengenai sistem presensi berbasis *QR Code*, sedangkan untuk perbedaannya terletak pada topik yang diangkat yaitu tentang sistem presensi pada himpunan mahasiswa.

2.2 Landasan Teori

2.2.1 Rekayasa *Web*

Rekayasa *web* adalah proses yang diunakan untuk menciptakan aplikasi *web* yang berkualitas tinggi[6]. Rekayasa *web* adalah subdisiplin dari rekayasa perangkat lunak yang membantu menyediakan metodologi untuk merancang, mengembangkan, memelihara, dan melibatkan aplikasi *web*. Rekayasa *web* membantu para pengembang sistem di bawah *control*, memperkecil risiko–risiko yang akan terjadi

dan meningkatkan kualitas, dapat dipelihara, dan memiliki skalabilitas aplikasi *web*.

2.2.2 Pemrograman Berorientasi Obyek

Object Oriented Programming adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap 107 bagian dari suatu permasalahan adalah objek, objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi[4]. PBO juga merupakan sebuah paradigma pemograman yang beroritasikan kepada objek. Semua data dan fungsi didalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Berbeda dengan Pemrograman Terstruktur atau Pemrograman Berorientasi Prosedur atau *Procedural Oriented Programming* (POP), yang dimana setiap objek menerima pesan/data, memprosesnya, dan mengirimnya ke objek lain.

Beberapa kelebihan Pemrograman Berorientasi Objek :

1. *Reusable*, artinya kode objek yang diimplementasikan dapat digunakan kembali pada program aplikasi lainnya.
2. *Extensible*, artinya objek yang sudah dibuat dapat kita ubah lagi implementasi fungsi-fungsinya sesuai dengan keinginan.
3. *Maintenable,* artinya objek yang kita buat dapat dengan mudah dirawat (maintain/manage).
4. *Extendable,* artinya objek yang sudah dibuat dapat kita kembangkan lagi menjadi objek yang lebih besar/kompleks.

Pemrograman berorientasi objek fokus ke manipulasi objek. Mengapa seorang *programmer* harus mempelajari PBO, bahkan seorang *programmer* yang tidak pernah bekerja dengan PBO pun tetap harus mempelajarinya juga. Hal ini dikarenakan suatu hari nanti semua bahasa pemograman akan menambahkan kemampuan PBO pada bahasanya. Konsep dasar umum yang biasanya terdapat pada semua bahasa pemograman yang mendukung PBO adalah sebagai berikut :

1. *Object*

   Merupakan sesuatu yang dapat dianalogikan dengan benda, orang, tempat, kejadian maupun konsep-konsep yang ada dalam dunia nyata yang digunakan pada perangkat lunak atau sistem informasi. Dalam bahasa teoritis PBO , *object* berfungsi membungkus data dan fungsi bersama menjadi satu unit dalam sebuah komputer. *Object*

merupakan dasar dari modularitas dan struktur dalam sebuah program komputer yang berorientasi objek.

2. *Class*
   Definisi *class* yaitu template untuk membuat objek. *Class* merupakan prototipe atau *blue print* yang mendifinisikan variable-variabel dan method-method secara umum. Objek merupakan hasil instansiasi dari suatu *class.* Proses pembentukan objek dari suatu kelas disebut sebagai *instantiation.* Dalam bahasa teoritis PBO, *class* merupakan kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu.
3. *Attribute*
   Merupakan data yang membedakan antara objek satu dengan yang lainnya. Dalam *class, attribute* sering disebut sebagai *variable*. *Atribute* dibedakan menjadi dua jenis yaitu *Instance Variable* dan *Class Variable*. *Instance Variable* adalah atribut intuk tiap objek dari kelas yang sama. Tiap objek memunyai dan menyimpan nilai atributnya sendiri. Jadi, tiap objek dari *class* yang sama boleh mmpunyai nilai yang sama atau berbeda. *Class Variable* adalah atribut untuk semua objek yang dibuat dari *class* yang sama. Semua objek mempunyai nilai atribut yang sama. Jadi semua objek dari *class* yang sama mempunyai hanya satu nilai yang *value*nya sama.
4. *Method*
   Behavior atau tingkah laku merupakan hal-hal yang bisa dilakukan oleh objek dari suatu *class*. *Behavior* dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke objek lain untuk melakukan suatu tugas. Dalam *class, behavior* disebut juga sebagai *methods. Methods* sendiri adalah serangkaian statements dalam suatu *class* yang meng*handle* suatu *task* tertentu. Cara objek berkomunikasi denga objek yang lain adalah dengan menggunakan *method*.
5. *Encapsulation* (pembungkusan)
   Salah satu ciri penting PBO adalah *Encapsulation*. Definisi *encapsulation* secara teoritis merupakan pembungkusan *variable* dan *method* dalam sebuah objek yang terlindungi serta menyediakan *interface* untuk mengakses *variable* tersebut. Variabel *method* yang dimiliki oleh suatu objek bisa ditentukan hak aksesnya.
6. *Inheritance* (pewarisan)
   Inheritance merupakan pewarisan *atribut* dan *method* dari sebuah *class* ke *class* lainnya. *Class* yang mewarisi disebut *superclass* dan

class yang diwarisi disebut *subclass. Subclass* bisa berlaku sebagai *superclass* bagi *class* lainnya, disebut sebgai *multilevel inheritance*. Prinsip dasar *inheritance* yaitu persamaan-persamaan yang dimiliki oleh beberapa *class* dapat digabungkan dalam sebuah *class* induk sehingga setiap kelas yang diturunkannya memuat hal-hal yang spesifik untuk *class* yang bersangkutan. Secara singkat bisa diartikan, teknik yang menyatakan anak dari *object* akan mewarisi data/atribut dan metode dari induknya langsung.

7. *Polimorfisme*
   Merupakan kemampuan suatu *object* untuk mempunyai lebih dari satu bentuk. *Polimorfisme* tidak tergantung pada pemanggilan subrutin. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada *object* tertentu dimana pesan tersebut dikirim. *Polimorfisme* bisa juga diartikan sebagai aksi yang sama yang dapat dilakukan terhadap beberapa objek. *Polimorfisme* berarti bahwa operasi yang sama mungkin mempunyai perbedaan dalam *class* yang berbeda.

2.2.3 Basis Data

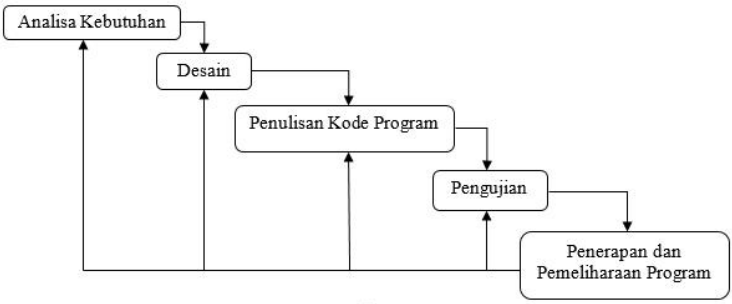
Basis data adalah sebuah objek yang komplek untuk menyimpan informasi yang terstruktur, yang diorganisir, dan disimpan dalam suatu cara yang memungkinkan informasi diakses secara cepat[7]. Definisi data disini dibedakan dari program aplikasi, yang umumnya sama dengan pendekatan pengembangan modern perangkat lunak, dimana definisi internal dan eksternal dari sebuah objek dipisahkan. Salah satu keuntungan dari pendekatan tersebut adalah abstraksi data dimana kita dapat mengubah definisi internal dari sebuah objek tanpa mempengaruhi pengguna dari objek jika definisi eksternal objek tersebut tidak berubah. Sistem *database* memiliki empat komponen penting, yaitu :

1. Perangkat Keras (*Hardware*)
   Merupakan perangkat keras yang dibutuhkan dalam pengelolaan database, berupa komputer beserta seluruh kelengkapan yang dibutuhkan, seperti prosesor, memori, *harddisk* sebagai media penyimpanan datanya, dan lain sebagainya.
2. Data
   Merupakan informasi yang disimpan dalam suatu struktur tertentu yang terintegrasi.

3. Perangkat Lunak (*Software*)
   Merupakan perangkat lunak yang digunakan untuk melakukan pengolahan data. Perangkat lunak ini sering disebut sebagai *Database Management System* (DBMS).
4. Pengguna (*User)*
   Merupakan orang yang menggunakan data yang tersimpan data dan terkelola. User dapat berupa seorang yang mengelola database tersebut, yang disebut dengan *database administrator* (dba), bisa juga *end user* yang mengambil hasil dari pengolahaan database melalui bahasa *query*. *User* juga dapat seorang *programer* yang membangun aplikasi yang terhubung ke *database* dengan menggunakan bahasa perograman.

2.2.4 SDLC (*System Development Life Cycle*)

SDLC (*System Development Life Cycle*) model *waterfall* (air terjun) merupakan siklus hidup pengembangan sistem. Model ini menyediakan pendekatan alur hidup perangkat lunak secara terurut dimulai dari analisa, desain, pengkodean, pengujian dan tahap pendukung[1].



Gambar 2. 1 Metode *Waterfall* menurut Rosa dan Shalahudin

Keterangan dari gambar 2.1 diantaranya sebagai berikut :

1) Analisa Kebutuhan
   Proses pengumpulan kebutuhan dilakukan dengan teliti untuk menspesifikasikan kebutuhan perangkat lunak supaya mudah untuk dipahami, perangkat lunak seperti apa yang dibutuhkan oleh pengguna. Spesifikasi kebutuhan perangkat lunak di tahap ini

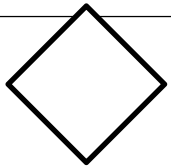
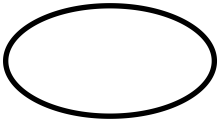
menjadi acuan untuk menerjemahkan ke dalam bahasa pemrograman.

2) Desain Sistem
   Desain perangkat lunak merupakan proses multi langkah yang berfokus pada desain pembuatan program perangkat lunak, representasi antarmuka dan prosedur pengkodean. Tahap ini penulis mencoba merancang antarmuka pemakai sistem dimana *website* yang akan dibuat sesuai prosedur yang sudah dianalisa, seperti merancang halaman utama website, merancang *database* dan *tools* apa saja yang akan dibutuhkan selama proses pembuatan.
3) Penulisan Kode Program
   Desain harus di translasikan ke dalam program perangkat lunak. Hasilnya merupakan program komputer yang sesuai dengan desain yang telah dibuat pada tahap desain.
4) Pengujian
   Pengujian berfokus pada perangkat lunak dari *segi logic* dan fungsional dan juga memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk mengurangi adanya kesalahan dan memastikan hasil yang diinginkan sesuai. Pengujian yang dilakukan menggunakan metode *black box testing* yaitu dengan mengamati hasil eksekusi data uji dan fungsional perangkat lunak khususnya pada *input* dan *output website.*
5) Penerapan dan Pemeliharaan Program
   Kemungkinan adanya perubahan pada perangkat lunak saat dikirim ke pengguna biasa terjadi karena, kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Pada tahap ini bisa mengulangi proses pengembangan mulai dari analisis spesisfikasi untuk perubahan perangkat lunak yang sudah ada, namun tidak membuat perangkat lunak baru.

2.2.5 ERD (Entity Relationship Diagram)

ERD adalah model konseptual yang mendeskripsikan hubungan antara penyimpanan. ERD digunakan untuk memodelkan struktur data dan hubungan antar data. Dengan ERD, model dapat diuji dengan mengabaikan proses yang dilakukan. ERD pertama kali dideskripsikan oleh Peter Chen yang dibuat sebagai bagian dari perangkat lunak CASE. Notasi yang digunakan dalam ERD dapat dilihat pada Tabel di bawah ini :

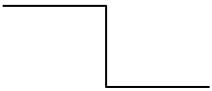
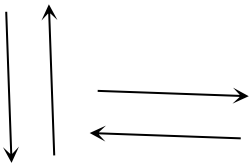
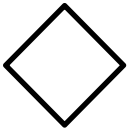
Tabel 2. 1 Simbol ERD

No.	Simbol	Nama Simbol dan Keterangan
1.		Entitas ialah suatu objek yang dapat dibedakan dengan obejek lainya. Entitas berfungsi untuk memberikan indentitas pada entitas yang memiliki table dan nama. Benda yang memiliki nama data harus harus disimpan datanya agar dapat diakses oleh aplikasi komputer, penamaan biasanya lebih ke nama benda dan belum merupakan nama table.
2.		Relasi ialah hubungan yang terjadi antara 1 entitas atau lebih yang tidak mempunyai fisik tetapi hanya sebagai konseptual. Dan berfungsi untuk mengetahui jenis hubungan yang ada antara 2 file.
3.		Atribut, ialah karakteristik dari entitas atau relasi yang menyediakan penjelasan detail tentang entitas atau relasi tersebut. Dan berfungsi untuk memperjelas atribut yang dimiliki oleh sebuah entitas.
4.		Penghubung antar relasi dan entitas yang kedua ujungnya memiliki *multiplicity*, kemungkinan pemakaian, kemungkinan jumlah maksimum keterhubungan anatara entitas yang lain disebut dengan kardinalitas.

2.2.6 Flowchart

Flowchart adalah bagian-bagian yang mempunyai arus untuk menggambarkan langkah – langkah penyelesaiann suatu masalah. Simbol – simbol *flowchart* dapat dilihat di Tabel 2. 2 sebagai berikut :

Tabel 2. 2 Simbol *Flowchart*

No.	Simbol	Nama Simbol dan Keterangan
1.		Terminal, untuk memulai dan mengakhiri suatu program.
2.		*Communication link*, untuk menyatakan bahwa adaya transsi suatu data/ informasi dri suatu lokasi ke lokasi lainya.
3.		Arus/Flow, menghunbungkan antara *symbol* satu dengan *symbol* lainnya yang berfungsi untuk menyatakan jalanya arus suatu proses.
4.		*Decision*/logika, memilih proses yang berdasarkan kondisi yang ada dengan menghasilkan dua kemungkinan jawaban, ya/tidak.
5.		Manual, untuk menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual.)
6.		*Keying Operation*, untuk meyatakan segala jenis operasi yang diproses dengan menggunakan suatu mesin

		yang mempunyai *keyboard*.
7.		*Document,* untuk mencetak laporan ke *printer*.
8.		*Display,* untuk menyatakan peralatan *output* yang digunakan berupa *layer* (video, komputer.)
9.		Input – Output, untuk menyatakan proses *input* dan *output* tanpa tergantung dengan jenis peralatanya.