BAB II
DASAR TEORI

BAB II
DASAR TEORI

2.1 Landasan Teori

Landasan teori berisi hal-hal atau teori-teori yang berkaitan dengan permasalahan dan ruang lingkup permasalahan sebagai landasan dalam pembuatan laporan ini.

2.1.1. Sistem Informasi Akademik

Sistem informasi berfungsi sebagai koneksi atau hubungan antara manusia, peristiwa, data, jaringan komputer, dan teknologi yang bekerja secara terpadu untuk mendukung dan meningkatkan operasi sehari-hari suatu perusahaan. Untuk dianggap berkualitas, sistem informasi dalam lingkup perusahaan harus mampu membantu pengguna dalam mengambil keputusan manajerial. [5].

Sistem informasi terdiri dari beberapa komponen yang saling terkait, termasuk komponen *input*, komponen *model*, komponen *output*, komponen teknologi, komponen *hardware*, komponen *software*, komponen basis data, dan komponen kontrol. Semua komponen tersebut berinteraksi satu sama lain dan membentuk satu kesatuan yang bertujuan mencapai sasaran yang diinginkan[6].

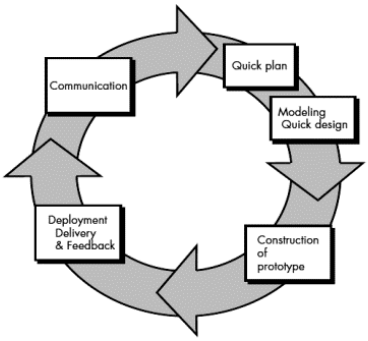
Sistem informasi akademik adalah sistem yang dikembangkan untuk mempermudah berbagai aspek, salah satunya adalah pengelolaan nilai[2]. Pada sistem informasi akademik ini, data yang akan diambil berupa data akademik yang berada di Politeknik Negeri Cilacap, seperti data nilai dan data mahasiswa

2.1.2. *Website*

Website atau situs merupakan kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara dan atau gabungan dari semuanya, baik bersifat statis maupun dinamis yang membentuk suatu rangkaian bangunan yang saling terkait, yang masing- masing dihubungkan dengan jaring-jaringan halaman[7]. *Website* dapat diartikan dalam singkat yaitu sekumpulan halaman informasi yang dapat diakses melalui internet[8]. *Website* dapat digunakan pada aplikasi *browser* anda contohnya *google chrome, mozilla firefox, safari,* dan aplikasi *browser* bawaan dari perangkat yang sudah disediakan

2.2 Metode Pengembangan Sistem

Metode *prototyping* menurut Hasmoro[9] adalah suatu sistem potensial yang menyediakan pengembang dan pengguna dengan suatu analisis tentang suatu sistem yang akan digunakan sebagaimana mestinya. Metode *prototyping* merupakan metode yang dapat menghasilkan sistem sebagai perantara antara *developer* dan pengguna karena dapat memudahkan pengguna untuk mendapatkan sistem yang diinginkan. Berikut tahapan metode *prototyping* yang dapat dilihat pada **Gambar 2.1**[4]:



Gambar 2. 1 Model *Prototyping*

1. *Communication*

   Pada tahap ini, dilakukan komunikasi antara pelanggan dan sistem untuk mengidentifikasi kebutuhan pelanggan. Komunikasi ini melibatkan wawancara langsung dengan pelanggan untuk memahami keluhan dan permasalahan yang dihadapi.

2. *Quick Plan*

   Pada tahap ini, dibuat sebuah *flowchart* yang berisi seluruh perencanaan dasar pembuatan sistem, sebagai panduan dalam pengembangan selanjutnya.

3. *Modelling Quick Design*

   Pada tahap ini, rancangan sistem dapat dilihat dari segi pengguna dalam pembuatan rancangan ini digambarkan dalam *use case* diagram. Rancangan ini akan memperlihatkan interaksi antara pengguna dan sistem yang akan dikembangkan.

4. *Construction of Prototype*

   Pada tahap ini, merupakan aktifitas pebuatan sistem yang akan dikembangkan.

5. *Deployment Delivery & Feedback*

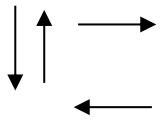
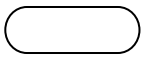
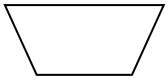
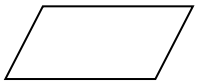
   Pada tahap ini, dilakukan pengujian sistem untuk mengevaluasi kecocokan sistem dengan kebutuhan pengguna dan mencari kelemahan dan kekurangan dalam sistem. Kemudian, umpan balik dari pengguna akan dijadikan masukan untuk memperbaiki spesifikasi kebutuhan sistem. Iterasi terjadi pada saat proses perbaikan prototipe untuk memastikan sistem yang dihasilkan dapat memenuhi kebutuhan pengguna dengan baik.

2.3 *Tools*/Alat Bantu Penelitian

2.3.1. Flowchart

Flowchart atau diagram alir merupakan suatu cara untuk menggambarkan algoritma atau prosedur yang digunakan dalam menyelesaikan suatu masalah secara berurutan. *Flowchart* menggunakan simbol-simbol yang mewakili proses-proses tertentu. Berikut uraian simbol-simbol yang digunakan dalam membangun sebuah *Flowchart* yang dapat dilihat pada **Tabel 2.1**[10]:

Tabel 2. 1 Simbol *Flowchart*

No	Simbol	Keterangan
1	 ***Flow Direction Symbol***	Berfungsi untuk menghubungkan simbol yang satu dengan yang lainnya, menyatakan arus suatu proses.
2	 ***Terminator Symbol***	Digunakan untuk memulai atau mengakhiri program.
3	 ***Processing Symbol***	Digunakan untuk menunjukan pengolahan yang akan dilakukan dalam komputer.
4	 ***Manual Operation Symbol***	Digunakan untuk menunjukan pengolahan yang tidak dilakukan oleh komputer.
5	 ***Decision Symbol***	Digunakan untuk memilih proses yang akan dilakukan berdasarkan kondisi tertentu.
6	 ***Input-Output Symbol***	Digunakan untuk meng*input*kan data secara manual dengan *keyboard*.
7	 ***Document Symbol***	Digunakan untuk menyatakan masukan dan keluaran yang berasal dari dokumen.

2.3.2. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah salah satu teknik pemodelan yang digunakan dalam proses desain dan pengembangan perangkat lunak yang menggunakan pendekatan berorientasi objek. UML adalah sebuah standar penulisan kerangka kerja yang berisi proses bisnis dan penulisan kelas-kelas menggunakan bahasa spesifik. Salah satu bentuk dari *Unified Modeling Language* (UML) antara lain *Use Case Diagram*[11]:

1. *Use Case Diagram*

Use case diagram sering disebut sebagai diagram perilaku yang menggambarkan serangkaian tindakan (*use case*) yang harus atau dapat dilakukan oleh suatu sistem atau subjek, bekerja sama dengan satu atau lebih pengguna eksternal sistem (aktor)[12]. Berikut adalah simbol-simbol yang ada pada *use case* diagram yang dapat dilihat pada **Tabel 2.2:**

Tabel 2. 2 Simbol *Use Case Diagram*

No	Simbol	Keterangan
1	 ***Use Case***	Simbol untuk deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
2	 ***Actor***	Simbol yang menspesifikasikan himpunan peran yang pengguana mainkan ketika berinteraksi dengan *use case.*
3	***Association***	Simbol yang menghubungkan antara objek satu dengan objek lainnya.
4	 <<***include***>>	Simbol yang menspesifikan bawa *use case* sumber secara eksplisit.
5	 <<***extended*** >>	Simbol yang menspesifikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6	 ***Generalization***	Simbol yang hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek induk *(ancestor).*
7	 ***System***	Yaitu simbol yang menyatakan *input* berasal dari dokumen dalam bentuk kertas atau *output* dicetak ke kertas.
8	 ***Collaborattion***	Simbol interaksi aturan-aturan elemen lain yang berkerja sama untuk menyediakan perilaku yang

		lebih besar dari jumlah dan elemennya (sinergi).
9	 ***Note***	Simbol elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

2.3.3. Laragon

Laragon adalah *software* yang memiliki banyak sistem operasi sebagai *server* lokal. *Laragon* terdapat beberapa layanan, alat dan fitur yang terdiri *web server, PHP server,* dan *Laravel*. *Laragon* terdapat fitur ekstensi jika ingin membuat *website* menggunakan *Framework Laravel* atau *CodeIgniter* yang dapat memudahkan programmer saat membuat *website*[13].

2.3.4. Pemrograman Berorientasi Objek (*OOP*)

Pemrograman Beriorentasi Objek (*OOP*) adalah sebuah pendekatan dalam membangun perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek. Objek-objek tersebut mengandung data dan operasi yang dapat dilakukan terhadapnya. *OOP* merupakan sebuah paradigma atau teknik pemrograman yang berfokus pada objek sebagai unit dasar dalam membuat aplikasi[14].

Metode berorientasi objek sekarang banyak digunakan dalam pengembangan atau pembuatan aplikasi. Metode ini banyak digunakan karena aplikasi yang dibuat sekarang memiliki struktur yang semakin kompleks. Pemrograman Berorientasi Objek (*OOP*) memiliki beberapa manfaat, yaitu meningkatkan produktivitas, mempercepat pengembangan, mempermudah pengembangan, menjaga konsistensi, dan dapat meningkatkan kualitas perangkat lunak.

2.4 Rekayasa *Web*

Rekayasa *web* adalah subdisiplin dalam rekayasa perangkat lunak yang memberikan metodologi untuk merancang, mengembangkan, memelihara, dan menyebarluaskan aplikasi *web*. Penekanan pada disiplin rekayasa *web* ini menunjukkan kebutuhan akan kesuksesan pengembangan aplikasi dan sistem berbasis *web*. Dalam rekayasa *web*, digunakan pendekatan ilmiah, rekayasa, dan sistematis serta prinsip-prinsip manajemen guna mencapai keberhasilan dalam pengembangan, penyebaran, dan pemeliharaan aplikasi dan sistem *web* berkualitas tinggi. Rekayasa *web* membantu pengembang sistem untuk tetap terkendali, mengurangi risiko, dan meningkatkan kualitas, keberlanjutan, serta skalabilitas aplikasi *web*. Tujuan utama dari rekayasa *web* adalah mencapai kesuksesan dan mengelola kompleksitas serta keragaman dalam

pengembangan aplikasi *web*. Sebagai konsekuensi, kegagalan dalam rekayasa *web* dapat memiliki dampak yang serius[15].

2.4.1. *Laravel*

Laravel adalah *Framework PHP opensource* yang dikembangkan oleh Tylor Otwell yang berlisensikan oleh *MIT*. *Laravel* ini dapat memudahkan programmer dengan kode-kodenya yang sederhana, aman dan efisien[16]. *Laravel* juga menjadi salah satu bahasa pemrograman yang dinamis. Dasar Bahasa yang digunakan pada *Laravel* ini seperti pada arti diatas yaitu menggunakan *PHP*.

2.4.2. Basis Data (*Database*)

Basis data (*Database*) adalah sebuah wadah di mana data yang saling terkait dikumpulkan dalam suatu perusahaan atau organisasi. Tujuannya adalah untuk memudahkan dan mempercepat akses serta penggunaan ulang data tersebut[17]. Basis data berfungsi sebagai tempat penyimpanan data yang besar dan dapat digunakan secara bersamaan oleh banyak pengguna.

2.4.3. DBMS (*Database Management System*)

DBMS (*Database Management System*) adalah perangkat lunak khusus yang bertanggung jawab dalam pengelolaan dan penggunaan *database*. DBMS ini berperan penting dalam mengatur bagaimana data diorganisasi, disimpan, diubah, dan diambil kembali. Dalam penggunaan *database*, pengguna tidak berinteraksi langsung dengan data, melainkan melalui DBMS. DBMS menyediakan antarmuka yang memungkinkan pengguna untuk mengakses, memanipulasi, dan mengelola data dengan cara yang ditentukan. DBMS juga menjaga keamanan dan integritas data, serta memastikan konsistensi dan efisiensi operasi *database*. Dengan penggunaan DBMS yang tepat, pengguna dapat mengoptimalkan penggunaan *database*, meningkatkan produktivitas, dan memastikan data tersedia dan akurat untuk kebutuhan bisnis atau organisasi[18]. DBMS memberikan berbagai fasilitas yang meliputi[19]:

2.4.3.1. DDL (*Data Definition Language*)

DDL (*Data Definiton Language*) terdiri dari sejumlah perintah yang umumnya digunakan dalam *database* untuk mengatur skema dan subskema *database*. Dalam DDL terdapat perintah-perintah seperti:

a. *Create* : Digunakan untuk membuat, termasuk membuat *database* dan tabel baru.
b. *Alter* : Digunakan untuk mengubah struktur tabel yang telah dibuat.
c. *Drop* : Digunakan untuk menghapus *database* dan tabel.

2.4.3.2. DML (*Data Manipulation Language*)

DML (*Data Manipulation Language*) merujuk pada sekumpulan perintah yang memungkinkan pengguna untuk mengakses dan memanipulasi data yang telah diorganisir sebelumnya sesuai dengan model data yang telah ditentukan. DML digunakan untuk melakukan manipulasi pada *database* yang telah didefinisikan menggunakan DDL. Dalam DML terdapat perintah-perintah seperti:

a. *Insert* : Digunakan untuk memasukan data ke dalam tabel.

b. *Select* : Digunakan untuk mengambil atau menampilkan data dari suatu tabel.

c. *Update* : Digunakan untuk memperbaharui data lama menjadi data terbaru.

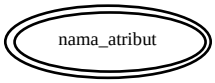
d. *Delete* : Digunakan untuk menghapus data dari tabel.

2.4.4. ERD (*Entity Relationship Diagram*)

ERD (*Entity Relationalship Diagram*) adalah sebuah alat yang digunakan untuk melakukan pemodelan struktur data dengan menggambarkan secara abstrak (konseptual) entitas dan hubungan antar entitas (*relationship*). ERD berperan penting dalam memodelkan struktur data dan menggambarkan hubungan antara data dalam sebuah sistem. Untuk merepresentasikan entitas dan hubungan tersebut, ERD menggunakan berbagai notasi dan simbol yang ditentukan[20]. Adapun simbol-simbol yang terdapat pada ERD dapat dilihat pada **Tabel 2.3**.

Tabel 2. 3 Simbol ERD

No	Simbol	Keterangan
1	nama_entitas ***Entity***	Suatu objek yang utuh dari independen terhadap suatu objek lain dalam lingkup masalah yang ditinjau (memiliki fungsi relevan terhadap sistem)
2	nama_entitas ***Week Entity***	Suatu *entity set* yang keberadaannya lemah (tidak diperlukan jika *entity* kuatnya tidak ada)
2	nama_atribut ***Atribut***	Pendeskripsian karakteristik dari entitas.
3	nama_atribut primer **Primary Key Atribute**	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan, biasanya berupa id.

4	 ***Multivalue***	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki lebih dari satu.
5	 ***Relation***	Deskripsi hubungan antar *entity* dari kategori yang berbeda atau sama.
6	 ***Association***	Penghubung antara relasi dan entitas dimana 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* yang menghubungkan entitas Adan entitas B.

~Halaman Ini Sengaja Dikosongkan~