

BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 . Tinjauan Pustaka

Pembuatan aplikasi mengenai sistem informasi pengarsipan dalam skripsi yang berjudul “Perancangan Sistem Informasi Arsip Surat Menyurat di Universitas U’Budiyah Indonesia Menggunakan *Php* dan *MySQL*”. Skripsi tersebut bertujuan untuk memperbaiki sistem sebelumnya yang masih manual dan semua data surat masuk dan keluar diarsipkan pada satu tempat dan laporannya masih disimpan pada sebuah buku besar. Perancangan sistem tersebut menggunakan metode SSAD (*Structured Systems Analysis and Design*). Sedangkan pembuatan aplikasi tersebut menggunakan *MySQL* sebagai *database* dan *Php* sebagai media pembuatan tampilan aplikasi (Junidar 2012).

Penelitian yang berjudul “Pembuatan Sistem Informasi Administrasi Berbasis Desktop Pada Bimbingan Belajar Citra Bagus Grup Sleman”. Penelitian tersebut bertujuan untuk menyederhanakan dan mempercepat pelaksanaan administrasi kegiatan dan untuk memfasilitasi presentasi dari pengolahan data dan mempercepat informasi ketika diperlukan. Pembuatan aplikasi tersebut menggunakan *Netbeans* dan *MySQL* sebagai *databasenya*. (Ana Wati Ndarbeni, 2013).

Penelitian yang berjudul “Aplikasi Pendataan Medical Check UP (MCU) Pada RST Ciremai Cirebon”. Bertujuan untuk membantu MCU dalam melakukan penyimpanan data ke dalam basis data, melakukan pendataan untuk pemeriksaan fisik dan laboratorium, dan menyajikan data dalam bentuk grafik. Perancangan sistem tersebut menggunakan metode *Prototyping*. Sedangkan pembuatan aplikasi menggunakan bahasa pemrograman *Java* dan *MySQL* sebagai basis data. (Juliyanti, Ika,2014).

Berdasarkan penelitian terdahulu yang sudah dijelaskan diatas,dapat disimpulkan bahwa terdapat persamaan diantara kegitanya yaitu sebagai berikut:

1. Membuat aplikasi untuk mempermudah pendataan di instansi masing-masing.
2. Pengoptimalan informasi yang di perlukan pada masing-masing instansi.
3. Menggunakan *MySQL* sebagai *databasenya*.

Sedangkan terdapat perbedaan yang dapat disimpulkan oleh penulis adalah:

1. Penulis menggunakan Bahasa pemrograman C# sedangkan penelitian sebelumnya menggunakan Bahasa pemrograman java dan php.
2. Penulis menambahkan Adanya fitur *Print* laporan pada aplikasi yang akan dibuat sedangkan pada penelitian sebelumnya tidak terdapat adanya fitur tersebut.

2.2. Landasan Teori

2.2.1. Pengertian Sistem

Secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau komponen yang terorganisir dan saling berinteraksi satu sama lainnya. Sistem adalah kelompok dari dua atau lebih komponen atau subsistem yang berhubungan yang berfungsi dan dengan tujuan yang sama. (James A, Hall, 2007:6). Sistem terdiri dari unsur-unsur seperti masukan (*input*) pengolahan (*processing*), serta keluaran (*output*). (Scoot,1996).

2.2.2. Pengertian Informasi

Informasi adalah data yang sudah diolah, dibentuk, atau dimanipulasi sesuai dengan keperluan tertentu. (Drs Zulfikar Amsyah, MLS, 2003:2). Informasi berarti data yang sudah

dibentuk menjadi sesuatu yang memiliki arti dan berguna bagi manusia. (Kenneth C. Laudon dan Jane P. Laudon, 2008:16).

2.2.3. Pengertian Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi, dan menyediakan pihak luar tertentu dengan laporan – laporan yang diperlukan. (Jogiyanto, 2005:11).

Jadi dapat disimpulkan bahwa sistem informasi adalah suatu alat yang membantu dalam menyediakan informasi bagi penerimanya dan untuk membantu dalam pengambilan keputusan bagi manajemen didalam operasi perusahaan sehari-hari dan informasi yang layak untuk pihak luar perusahaan.

2.2.4. Pengertian Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna.

Aplikasi adalah penerapan, penggunaan atau penambahan. Dari pengertian diatas dapat disimpulkan bahwa aplikasi adalah berbagai bentuk pekerjaan atau tugas-tugas tertentu seperti penerapan, penggunaan dan penambahan data.(Anisyah, 2000:3).

2.2.5. Pengertian Aplikasi Dekstop

Desktop application atau aplikasi desktop adalah suatu aplikasi yang dapat berjalan sendiri atau independen tanpa menggunakan *browser* atau koneksi internet disuatu komputer otonom. (Dew Omenn, 2013).

Aplikasi berbasis *desktop* merupakan aplikasi yang dijalankan pada masing-masing komputer atau klien. Aplikasi berbasis *desktop* harus diinstall terlebih dahulu ke dalam komputer agar dapat digunakan. Berdasarkan pengertian diatas penulis menyimpulkan bahwa aplikasi *desktop* adalah aplikasi yang berjalan pada komputer yang dapat digunakan secara langsung ketika kode program selesai dikompilasi. (Rafypr101, 2013).

2.2.6. Siklus Sistem Informasi

Dalam pengembangan sebuah sistem informasi memerlukan proses dan urutan yang standar, yaitu Analisis, Desain, Implementasi, Pemeliharaan. Proses-proses tersebut dituangkan dalam satu metode yang dikenal dengan nama *System Develoment Life Cycle (SDLC)*. *SDLC* merupakan metodologi umum dalam pengembanganya sistem yang memadai kemajuan usaha analisis dan desain. (Youssef Bassil, 2012)

SDLC meliputi fase-fase sebgai berikut:

1. *Analysis*
2. *Design*
3. *Implementation*
4. *Testing*
5. *Maintenance*

2.3. Teknologi Pengembangan Aplikasi


2.3.1. *Unified Modelling Language (UML)*



Unified Modelling Language (UML) adalah sebuah bahasa permodelan visual yang dirancang khusus untuk pengembangan dan analisis sistem berorientasi objek dan desain. UML pertama kali dikembangkan oleh Grady Booch, Jim Rumbaugh, dan Ivars Jacobson pada pertengahan tahun 1990. (Keng Siau and Qing Cao, 2001:26).

a. *Use Case Diagram*


Use Case Diagram adalah sebuah diagram yang menunjukkan hubungan antara *actors* dan *use cases*. Digunakan untuk analisis dan desain sebuah system. (Scott W. Ambler, 2005:33). Simbol-simbol yang digunakan dalam use case diagram dapat dilihat pada Tabel 2.1. dan Tabel 2.2.

Tabel 2. 1 *Use Case Diagram*

NO	Gambar	Nama Gambar	Keterangan
1.		<i>Use Case</i>	Merupakan fungsionalitas yang disediakan sistem sebagai unit yang bertukar pesan dengan <i>actor</i> .

2.		<i>Actor</i>	Merupakan <i>abstraction</i> dari orang yang mengaktifkan fungsi dari target sistem dan merupakan orang yang berinteraksi dengan <i>use case</i> .
3.		<i>Association</i>	Digambarkan dengan garis tanpa panah yang mengindikasikan siapa yang berinteraksi secara langsung dengan sistem.

Tabel 2. 2 Simbol-simbol dalam *Use Case* Diagram (lanjutan)


NO	Gambar	Nama Gambar	Keterangan
4.		<i>Generalization</i>	Mengindikasikan siapa yang berinteraksi secara pasif dengan sistem.




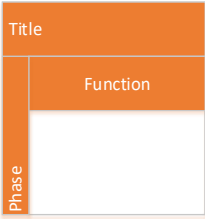
5.		<i>Include</i>	Mengidentifikasi hubungan antar dua <i>use case</i> dimana satu <i>usecase</i> memanggil <i>usecase</i> yang lain.
6.		<i>Extend</i>	Merupakan perluasan dari <i>use case</i> jika kondisi atau syarat terpenuhi.

b. *Activity Diagram*

Activity Diagram adalah suatu diagram yang menggambarkan konsep aliran data atau control, aksi terstruktur serta dirancang dengan baik dalam suatu sistem. (Conrad Bock, 2003:45). Simbol-simbol yang digunakan dalam activity diagram dapat dilihat pada Tabel 2.2.

Tabel 2. 3 Simbol–simbol dalam *Activity Diagram*

NO	Gambar	Nama Gambar	Keterangan
1.		<i>Start Point</i>	Merupakan awal dalam aktifitas

2.		<i>End Point</i>	Merupakan akhir dalam aktifitas.
3.		<i>Activities</i>	Menggambarkan suatu proses atau kegiatan bisnis
4.		<i>Decision Point</i>	Menggambarkan pilihan untuk pengambilan keputusan dalam aktifitas.
5.		<i>Swimlane</i>	Digunakan untuk pembagian <i>activity diagram</i> yang menunjukkan siapa yang melakukan aktifitas.

c. *Class Diagram*

Class Diagram adalah sebuah diagram yang menunjukkan hubungan antar class yang didalamnya terdapat atribut dan fungsi dari suatu objek. (Scott W.Ambler, 2005:47)

Class diagram mempunyai tiga relasi dalam penggunaannya, yaitu:

1. Assosiation

Assosiation adalah sebuah hubungan yang menunjukkan adanya interaksi antar class. Hubungan ini dapat ditunjukkan dengan garis dengan mata panah terbuka di ujungnya yang mengindikasikan adanya aliran pesan dalam satu arah.

2. Generalization

Generalization adalah sebuah hubungan antar class yang bersifat dari khusus ke umum.

3. Constraint

Constraint adalah sebuah hubungan yang digunakan dalam sistem untuk memberi Batasan pada sistem sehingga didapat aspek yang tidak fungsional.

2.3.2. Basis Data

Basis data merupakan salah satu teknologi yang harus dimiliki sebuah perusahaan, institusi, ataupun organisasi adalah teknologi yang dapat memproses data. Secara konsep basis data atau *database* adalah kumpulan dari data-data yang membentuk suatu berkas (*file*) yang saling berhubungan (*relation*) dengan tata cara tertentu untuk membentuk data baru atau informasi. Kumpulan dari data yang saling berhubungan (relasi) antara satu dengan lainnya yang diorganisasikan berdasarkan skema atau struktur tertentu (Supriyanto, 2005). Untuk mengelola *database* diperlukan suatu perangkat lunak yang disebut DBMS (*Data Base Management System*). DBMS merupakan suatu sistem perangkat lunak yang memungkinkan pengguna untuk membuat, memelihara, mengontrol, dan mengakses *database*. Dengan DBMS, pengguna dapat mengotrol dan memanipulasi data yang ada.

2.3.3. *Entity Relationship Diagram* (ERD)

Entity Relationship diagram adalah sebuah alat dalam suatu lingkup database yang digunakan dalam menganalisis dan mendesain data. ERD telah berhasil dibuktikan kehandalannya dalam hal menghubungkan data field, variables, maupun database. (Ronald Gege Allan, 2005:51).

Beberapa kelebihan ERD menurut *Business Intelligence Journal*: Ronald Gege Allan (2005:51) dibanding diagram sejenis seperti use case diagram dan sequence diagram adalah:

1. Mudah untuk dioperasikan dalam hal mencari data field yang dibutuhkan.
2. Mudah dalam hal melakukan Analisa terhadap data.
3. Kemampuan dalam hal menghilangkan redudansi data.

2.3.4. *SQL Server*

SQL merupakan kependekan dari *Structured Query Language* (Bahasa *Query* Terstruktur). SQL lebih dekat dengan DML dari pada DDL. Namun tidak berarti SQL tidak menyediakan perintah DDL. SQL lebih menekankan pada aspek pencarian dari dalam tabel. Aspek pencarian ini sedemikian penting karena di sinilah sebenarnya inti dari segala upaya kita melakukan pengelolaan data. Data dalam basis data diorganisasi sedemikian rupa dengan tujuan untuk memudahkan pencarian di kemudian hari.

Menurut Thomas Connolly dan Carolyn Begg (2010, p185-p186), sebagai sebuah bahasa, SQL telah distandarisasi dan mengalami beberapa perubahan atau penyempurnaan. SQL muncul pertama kali pada tahun 1970 dengan nama *SEQUEL*. Standarisasi yang pertama dibuat pada tahun 1986 oleh ANSI (*American National Standards Institute*) dan ISO (*International Standard Organization*), yang disebut SQL-86. Pada tahun 1989 SQL-86 diperbaharui menjadi SQL-89. Standar terakhir yang dibuat adalah pada tahun 2008 yaitu SQL-2008.

Microsoft SQL Server ialah salah satu produk *Relational Database Management System* (RDBMS) popular saat ini. Fungsi utamanya ialah sebagai *server* dari sebuah *database* yang mengatur semua proses penyimpanan data dan transaksi suatu aplikasi.

Dalam DBMS seperti MS SQL Server biasanya tersedia paket bahasa yang digunakan untuk mengorganisasi basis data yang ada, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

Data Definition Language (DDL)

Data Definition Language (DDL) adalah salah satu paket bahasa DBMS yang berguna untuk melakukan spesifikasi terhadap skema basis data. Hasil kompilasi dari DDL adalah satu set tabel yang disimpan dalam file khusus yang disebut *Data Directory* atau *Dictionary*. Secara umum perintah-perintah dalam DDL berhubungan dengan operasi-operasi dasar seperti membuat basis data baru, menghapus basis data, membuat tabel baru, menghapus tabel, membuat indeks, mengubah struktur tabel. Contoh perintah DDL misalnya, *Create Table, Create Index, Alter dan Drop Database*

Data Manipulation Language (DML)

Data Manipulation Language (DML) adalah satu paket DBMS yang memperbolehkan pemakai untuk mengakses atau memanipulasi data sebagaimana yang telah diorganisasikan sebelumnya dalam model data yang tepat.

Bahasa Pemrograman C#

C# (C Sharp) adalah sebuah Bahasa pemrograman berbasis objek yang didukung oleh *Microsoft .Net Framework*. *Microsoft .NET Framework* adalah perantara agar aplikasi dengan bahasa pemrograman yang didukung dapat berkomunikasi dengan sistem operasi yang digunakan oleh komputer. Selain itu, *.NET Framework* juga memungkinkan *c#* untuk berkomunikasi dengan

bahasa pemrograman lainnya yang juga didukung oleh *.NET Framework* seperti *VB .NET*, *F#* atau *C++*.

Dengan kata lain, aplikasi yang kita buat dapat menggunakan komponen-komponen lain yang dibuat dengan menggunakan *VB.NET*, *J#*, atau *C++*.

2.3.5. Bahasa Pemrograman C#

Menurut P. J. Deitel, et al (2010, p42-43). “*Visual C# (a new language based on C++ and Java that was developed expressly for the .NET platform)*”, yang terjemahannya adalah: *Visual C#* (bahasa baru berdasarkan *C++* dan *Java* yang dikembangkan tegas untuk *.NET platform*).

Menurut Paul Deitel, et al (2014, p47), “*In 2000, Microsoft announced the C# programming language. C# has roots in the C, C++ and Java programming language. It has similar capabilities to Java and is appropriate for the most demanding application developments tasks, especially for building today’s large-scale enterprise applications, and web-based, mobile and cloud-based apps. The latest versions of Visual Basic have capabilities comparable to those of C#*”, yang terjemahannya adalah: pada tahun 2000, *Microsoft* mengumumkan bahasa pemrograman *C#*. *C#* memiliki akar dalam bahasa pemrograman *C*, *C++* dan *Java*. Ia memiliki kemampuan yang mirip dengan *Java* dan sesuai untuk tugas-tugas pengembangan aplikasi yang paling menuntut, terutama untuk membangun skala besar aplikasi perusahaan saat ini dan aplikasi *mobile* dan berbasis *cloud* dan berbasis *web*. Versi terbaru dari *Visual Basic* memiliki kemampuan yang sebanding dengan *C#*.

Berdasarkan pendapat para ahli diatas, dapat disimpulkan bahwa *C#* adalah sebuah aplikasi bahasa pemrograman baru berdasarkan *C++* dan *Java* yang terdapat dalam *Visual Basic*.

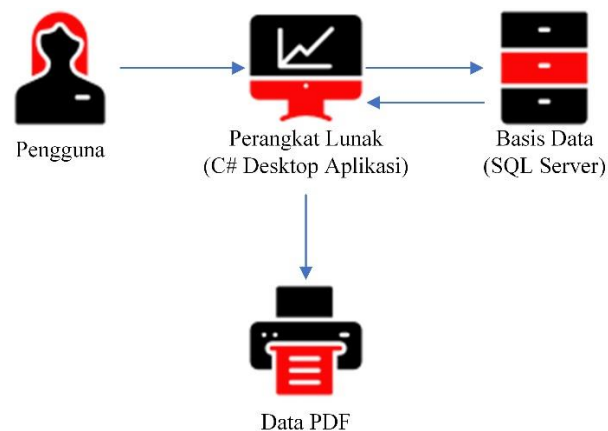
2.3.6. Visual Studio

Menurut Whitten dan Bentley (2012, p634). “*Most GUI based application development environments, such as Microsoft’s Visual Studio, can be easily used to construct nonfunctional prototypes of user interface screens*”, yang terjemahannya: banyak lingkungan pengembangan aplikasi berbasis GUI seperti *Microsoft’s Visual Studio*, dapat dengan mudah digunakan untuk membantu *prototype nonfunctional* dari layer antar muka pengguna.

Berdasarkan penjelasan diatas, maka dapat disimpulkan bahwa *Visual Studio* merupakan pengembangan aplikasi yang berbasis GUI yang membantu *prototype nonfunctional* dari *user interface*.

2.3.8. Arsitektur Perangkat Lunak

Arsitektur perangkat lunak merupakan struktur sebuah sistem yang meliputi elemen perangkat lunak, sifat (*property*) yang tampak dari elemen itu, serta relasi di antara elemen-elemen tersebut (*Bass et al dalam Krafzig et al, 2004*). Sifat yang tampak misalnya fungsi apa saja yang disediakan oleh elemen, bagaimana kinerjanya, bagaimana penanganan kesalahannya, sumber daya apa saja yang digunakan.



Gambar 2. 1 Arsitektur Perangkat Lunak

2.3.7. Black Box Testing

Dalam pengujian perangkat lunak ada dua yaitu *white box testing* dan *black box testing*. Dari kedua metode itu, pada skripsi dipilih menggunakan *black box testing* karena dianggap lebih tepat dibanding *white box testing*. Perangkat lunak memerlukan seperangkat tes untuk pencarian kesalahan fungsi-fungsi dalam aplikasi sehingga dalam hal ini *black box testing* lebih sesuai. Pengujian ini digunakan untuk mengetahui apakah fungsi-fungsi dalam perangkat lunak sudah sesuai dengan yang diharapkan.

Menurut Roger S. Pressman (2010), *black box testing* berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineer* untuk memperoleh *input* yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program. *Black box testing* berusaha untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang tidak benar atau fungsi yang hilang.
2. Kesalahan antarmuka.
3. Kesalahan dalam struktur data atau akses *database eksternal*.
4. Kesalahan kinerja.
5. Kesalahan inisialisasi dan pemutusan kesalahan.

