BAB II

LANDASAN TEORI

2.1 Tinjauan Jurnal

Menurut Sembiring, Jhoni Pranata (2013:28) Dalam pembuatan Aplikasi Kamus pencarian dibutuhkan algoritma atau metode yang efektif, karena proses pencarian merupakan salah satu bagian yang penting dalam pemprosesan data. Algoritma dan metode yang diterapkan yaitu *Sequential Search*. Algoritma *Sequential search* merupakan algoritma pencarian *linier*, algoritma ini melakukan pencarian lebih cepat karena proses pencarian sudah dalam keadaan terurut.

Menurut Gunawan. (2016:122). Proses pencarian kata kamus berupa buku dapat memakan waktu yang cukup lama karena proses pencariannya secara manual. Sehingga perlu dibuat suatu aplikasi, salah satunya yaitu aplikasi kamus digital yang dapat mempermudah dalam pencarian kata dan tidak perlu mengeluarkan biaya tambahan untuk membeli kamus. Dalam pembuatan aplikasi kamus digital ini perlu metode yang efektif, karena dalam proses pencarian data merupakan suatu bagian yang penting.

2.2 Konsep Dasar Program

2.2.1. Java

Menurut Suyanto (2015:2), menyatakan bahwa:
Java diciptakan oleh suatu tim yang dipimpin oleh Patrick Naughton dan James Gosling dalam suatu proyek dari *sun microsystem* yang memiliki kode *green* dengan tujuan untuk menghasilkan bahasa komputer sederhana yang dapat dijalankan di peralatan sederhana dengan tidak terikat pada arsitekture tertentu, mulanya disebut oak, tetapi karena oak sendiri merupakan nama dari bahasa pemrograman komputer yang sudah ada, maka *sun* mengubahnya menjadi *java.sun* kemudian meluncurkan *browser* dari *java* yang disebut *hot java* yang mampu menjalankan *applet.* Setelah itu teknologi *java* diadopsi oleh *Netscape* yang memungkinkan program *java* dijalankan di *browser netscape* yang kemudian diikuti *Internet Explore*. Karena keunikan dan kelebihannya, teknologi *java* mulai menarik banyak *vendor* seperti *IBM, Symantec, Inprise,* dll. Sun merilis versi awal *java* secara resmi pada awal tahun 1996 yang kemudian terus berkembang hingga muncul *jdk 1.1* kemudian *jdk 1.2* yang mulai disebut sebagai versi *java2* karena banyak mengandung peningkatan dan perbaikan. Perubahan utama adalah swing yang merupakan teknologi *GUI (Graphical User*

Interface) yang mampu menghasilkan windows yang portable. Dan pada tahun 1998-1999 lahirlah teknologi *J2EE (Java 2 Enterprise Edition).*

Java juga memiliki banyak fitur-fitur yang menarik menurut Suyanto (2015:3). Antara lain sebagai berikut:

1. *Applet*

   Program *Java* yang dapat berjalan di atas *browser*, yang dapat membuat halaman *HTML* lebih dinamis dan menarik.

2. *Java Networking*

   Sekumpulan API (*Application Programming Interface*) yang menyediakan fungsi-fungsi untuk aplikasi – aplikasi jaringan, seperti penyediaan akses untuk TCP, UDP, IP *Adrress* dan URL. Tetapi Java *Networking* tidak menyediakan akses untuk ICMP dikarenakan alasan sekurity dan pada kondisi umum hanya administrator (*root*) yang bisa memanfaatkan *protokol ICMP*.

3. *Java Database Connectivity (JDBC)*

   JDBC menyediakan sekumpulan *API* yang dapat digunakan untuk mengakses *database* seperti *Oracle, MySQL, PostgreSQL, Microsoft SQL Server*.

4. *Java Security*

   Java Security menyediakan sekumpulan API untuk mengatur security dari *aplikasi Java* baik secara *high level* atau *low level*, seperti *public/private key management* dan *certificates*.

5. *Java Swing*

Java Swing menyediakan sekumpulan API untuk membangun aplikasi-aplikasi GUI (*Graphical User Interface*) dan model GUI yang diinginkan bisa bermacam–macam, bisa *model Java*, model *Motif CDE* atau model yang *independent* terhadap *platform* yang digunakan.

6. *Java RMI*

   Java RMI menyediakan sekumpulan API untuk membangun aplikasi – aplikasi Java yang mirip dengan model *RPC* (*Remote Procedure Call*) jadi *object-object Java* bisa di call secara *remote* pada jaringan komputer.

7. *Java 2D/3D*

   Java 2D/3D menyediakan sekumpulan API untuk membangun *grafik – grafik 2D/3D* yang menarik dan juga akses ke printer.

8. *Java Server Pages*

   Berkembang dari *Java Servlet* yang digunakan untuk menggantikan aplikasi–aplikasi CGI, JSP *(Java Server Pages*) yang mirip ASP dan PHP merupakan alternatif terbaik untuk solusi aplikasi *Internet*.

9. *JNI (Java Native Interfac*e*)*

   JNI menyediakan sekumpulan API yang digunakan untuk mengakses fungsi–fungsi pada *library* yang dibuat dengan bahasa *pemrograman* yang lain seperti *C,C++*, dan *Basic.*

10. *Java Sound*

    Java Sound menyediakan sekumpulan API untuk *manipulasi sound*.

11. *Java IDL + CORBA*

    Java IDL (*Interface Definition Language*) menyediakan dukungan Java untuk *implementasi* CORBA (*Common Object Request Broker*) yang

merupakan *model distributed-Object* untuk solusi aplikasi besar di dunia *networking.*

12. *Java Card*

Java Card utamanya digunakan untuk aplikasi – aplikasi pada *smart card*, yang sederhana wujudnya seperti *SIM Card* pada *handphone.*

13. *JTAPI* (*Java Telephony API*)

Java Telepony API menyediakan sekumpulan API untuk memanfaatkan *devices –devices telephone*, sehingga akan cocok untuk aplikasi– aplikasi *CTI* (*Computer Telephony Integration*) yang dibutuhkan seperti *ACD* (*Automatic Call Distribution*), PC-PBX dan lainnya.

2.2.2. JDK (*Java Development Kit*)

Menurut Khannedy (2011:1), mengemukakan bahwa: "*Java Development Kit* merupakan perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode *java* menjadi *bytecode* yang dapat dimengerti dan dapat dijalankan oleh *Java Runtime Envirotment*".

2.2.3. Android

Menurut Juhara (2016:1), menyatakan bahwa:
Android adalah s*istem Operasi* berbasis *linux* yang dimodifikasi untuk perangkat bergerak (*Mobile Devices)* yang terdiri dari *Sistem Operasi*, *middleware*, dan aplikasi-aplikasi utama. Awalnya, Android dikembangkan oleh Android *Inc*. Perusahaan ini kemudian dibeli oleh *google* pada tahun 2005. Sistem operasi Android kemudian diluncurkan bersamaan dibentuknya Organisasi *Open Hanset Alliance* pada tahun 2007. Selain *google*, beberapa nama-nama besar juga ikut serta dalam *open handset alliance* antara lain: *Motorola, Samsung, LG, Sony Ericsson, T-Mobile, Vodafone, Toshiba,* dan *Intel.*

2.2.4. *Android Development Tools* (ADT)

Menurut Muhammad Irfan Luthfi (2014:2) Android *Development Tools (*ADT*)* adalah plugin yang didesain untuk *IDE Eclipse* yang memberikan kita kemudahan dalam mengembangkan aplikasi android dengan menggunakan *IDE Eclipse*. Dengan menggunakan ADT untuk *Eclipse* akan

memudahkan kita dalam membuat aplikasi *project* android, membuat *GUI* aplikasi, dan menambakan komponen-kompenen yang lainnya, begitu juga kita dapat melakukan running aplikasi menggunakan *Android SDK* melalui *Eclipse*. Dengan ADT juga kita dapat melakukan pembuatan *package* android *(.apk)* yang digunakan untuk distribusi aplikasi android yang kita rancang.

2.2.5. Eclipse

Menurut Faridayonarisa (2012:44), *Eclipse* adalah sebuah *IDE (Integrated Development Environment)* untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform *(platform-independent)*, sifat dari *Eclipse* adalah :

a. *Multi-platform*: Target sistem operasi *Eclipse* adalah *Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X*.
b. *Mulit-language*: *Eclipse* dikembangkan dengan bahasa pemrograman *Java*, akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti *C/C++, Cobol, Python, Perl, PHP,* dan lain sebagainya.
c. *Multi-role*: Selain sebagai *IDE* untuk pengembangan aplikasi, *Eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

2.2.6. XML (*Extensible Markup Language*)

Menurut Wardhani (2016:1), Mengemukakkan bahwa:
XML (Extensible Markup Language) di manfaatkan dalam mendefinisikan dokumen dengan format standar dimana yang dapat dibaca dan di dukung oleh aplikasi-aplikasi xml yang kompatibel. Bahasa format xml bisa digunakan dengan halaman html, akan tetapi xml itu sendiri bukan bahasa markup. Sebaliknya, xml itu merupakan "*metabahasa*" yang dapat di pakai dalam membuat bahasa markup untuk aplikasi khusus. Sebagai contoh nya itu, dapat menggambarkan item yang bisa diakses di saat membutuhkan Halaman *web* Dimana pada dasarnya, xml ini dapat memungkinkan Anda untuk membuat database informasi tanpa memiliki database yang

sebenarnya. meskipun secara default hanya digunakan dalam aplikasi *web*, banyak program lainnya juga yang dapat menggunakan dokumen *xml*, misalnya kode sumber aplikasi Android.
Mungkin jelasnya pengertian dari *xml* (*Extensible Markup Language*) adalah bahasa markup untuk keperluan umum yang telah disarankan oleh *w3c* dalam hal membuat dokumen markup untuk kepentingan pertukaran data antar sistem yang beraneka ragam. Tepatnya *xml* yaitu kelanjutan dari *HTML* (*HyperText Markup Language*) dimana yang merupakan bahasa standar untuk melacak Internet. Jadi, untuk membaca bahasa markup ada kesinambungannya. Baca dan cari tahulah selengkap-lengkap nya
XML justru didesain untuk mempu menyimpan data secara lengkap, ringkas serta mudah dalam mengatur. Kata kunci utama dari *XML* ini adalah data (jamak dari datum) apabila jika diolah bisa memberikan informasi.
XML juga menyediakan suatu cara terstandarisasi namun dapat dimodifikasi untuk menggambarkan isi dari dokumen. Dengan sendirinya, *XML* dapat di gunakan dalam menggambarkan sembarang *view database*, akan tetapi hanya dengan suatu cara yang standar.

XML dan dalam pengelolaan konten menurut wardani (2016:1), mengemukakkan bahwa:

1. Banyak dari sekian situs-situs yang menggunakan *xml*. Ironisnya, begitu banyak para desainer dan juga pengembang konten tidak tahu bahwa mereka sedang menggunakan *xml* padahal xml ada disana. Hal ini bisa jadi karena umumnya ada cms atau dikenal dengan *sistem manajemen konten* yang berada di depan xml yang dalam membuatnya lebih mudah bagi si penulis konten.Kemudian untuk menulis konten mereka pun tidak perlu khawatir atau panik dengan menulis *html* atau pun saat mendesain halaman web nya bagi para si pengguna.
2. *XML* dan Dokumentasi begitu banyak perusahaan yang berpindah ke *xml* dimana dalam hal menulis dokumentasi *internal* mereka. Keunggulan dari *xml* pada bagian dokumentasi adalah bisa digunakan untuk menentukan bagaimana ciri-ciri umum dalam sebuah buku-buku, koran, majalah, cerita, iklan, dan lainnya. Kisah menarik tentang xml untuk dokumentasi ini adalah

bahwa *xml* begitu mudah dipahami bagi si pengguna pemula, baik dari dokumentasi yang sebenarnya, dan juga kode *xml* sekitarnya. xml mampu digunakan untuk semua macam dari dokumentasi, dan untuk penerbitan dalam bahan pemasaran.

XML ini mempunyai kemudahan perpindahan (*portabilitas*) yang jauh lebih bagus. Misalnya seperti halnya *html, xml* ini juga menggunakan elemen yang ditandai dengan tag pembuka *(diawali dengan '<' dan diakhiri dengan '>'),* tidak lupa tag penutup *(diawali dengan '</ 'diakhiri '>')* dan untuk atribut elemen (parameter yang dinyatakan dalam tag pembuka contohnya *<form name="isidata">)*. bedanya disini yaitu untuk html medefinisikan dari awal tag dan atribut yang dipakai didalamnya, nah sedangkan untuk xml kita bisa menggunakan tag dan atribut sesuai dengan kehendak atau keinginan kita.

2.3. Metode Algoritma

Menurut Munir (2011:397) mengemukakan bahwa Algoritma *Searching* yang dimaksud yaitu pencarian data di dalam larik (array). Sebuah algoritma pencarian dijelaskan secara luas adalah sebuah algoritma yang menerima sebuah masukan dan menghasilkan sebuah solusi untuk masalah tersebut. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan atau tidak ditemukan.

2.4. Pengujian *Aplikasi*.

Menurut Ayuliana (2009:1), mengemukakkan bahwa: Metode ujicoba *blackbox* memfokuskan pada keperluan *fungsional* dari *software*. Karna itu uji coba *blackbox* memungkinkan pengembang *software* untuk membuat himpunan kondisi *input* yang akan melatih seluruh syarat-syarat *fungsional* suatu *program*. Ujicoba *blackbox* bukan merupakan alternatif dari ujicoba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya.

Ujicoba *blackbox* berusaha untuk menemukan kesalahaan dalam beberapa kategori menurut Ayuliana (2009:1), diantaranya:

1. Fungsi-fungsi yang salah atau hilang.
2. Kesalahan *interface.*
3. Kesalahan dalam struktur data atau akses *database eksternal.*
4. Kesalahan performa.
5. Kesalahan *inisialisasi* dan *terminasi*.

Menurut Ayuliana (2009:1), Ujicoba didesain untuk dapat menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana *validitas* fungsionalnya diuji?
2. Jenis *input* seperti apa yang akan menghasilkan kasus uji yang baik ?
3. Apakah sistem secara khusus sensitif terhadap nilai input tertentu ?
4. Bagaimana batasan-batasan kelas data diisolasi?
5. Berapa rasio data dan jumlah data yang dapat ditoleransi oleh sistem?
6. Apa akibat yang akan timbul dari kombinasi spesifik data pada operasi sistem?

2.5 Peralatan Pendukung (*Tool System*)

Aplikasi kamus Bahasa Indonesia-Jawa ini dirancang menggunakan *UML* dan dibuat ke dalam diagram-diagram yaitu *Activity, Use case, Sequence, Class* dan *Deployment*.

> Menurut Prastuti Sulistyorini (2009:23) *Unified Modelling Language (UML)* adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. *UML* menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan *UML* dapat dibuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena *UML* juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti *C++, Java, atau VB. NET.*

Menurut Sulistyorini (2009:1), mengemukakkan bahwa: *Aplikasi mobile learning* ini dirancang menggunakan *uml* dan dibuat ke dalam *diagram-diagram* yaitu:

2.5.1. Activity Diagram

Menurut Prastuti Sulistyorini (2009:24) *Activity Diagram* bersifat dinamis. *Activity Diagram* adalah tipe khusus dari *diagram state* yang memperlihatkan aliran dari suatu aktifitas ke aktifitas lainnya dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi – fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek.

2.5.2. *Use Case Diagram*

Menurut Prastuti Sulistyorini (2009:24) *Use Case Diagram* bersifat statis. *Use Case Diagram* memperlihatkan himpunan *use case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.

2.5.3. *Sequence Diagram*

Menurut Prastuti Sulistyorini (2009:24) *Sequence Diagram* bersifat dinamis. Diagram *sequence* merupakan diagram interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu.

2.5.4. *Class Diagram*

Menurut Prastuti Sulistyorini (2009:24) Diagram kelas bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi serta relasi.

2.5.5 *Deployment Diagram*

Menurut Prastuti Sulistyorini (2009:24) *Deployment Diagram* bersifat statis. *Deployment Diagram* memperlihatkan konfigurasi saat aplikasi dijalankan (saat *run time*). Dengan ini memuat simpul–simpul *(node)* beserta komponen-komponen yang ada di dalamnya. *Deployment diagram* berhubungan erat dengan diagram kompoen dimana *deployment diagram* memuat satu atau lebih komponen–komponen. Diagram ini sangat berguna saat aplikasi berlaku sebagai aplikasi yang dijalankan pada ban.