BAB II

LANDASAN TEORI

2.1. Tinjauan Jurnal

Pada aplikasi perhitungan contaner ini untuk memperkuat penelitian penulis, penulis mengambil dua jurnal referensi yang dijadikan teori dasar tentang container

menurut rasyidah (2016:2):

PT. Masaji Tatanan Container merupakan anak perusahaan dari Samudera Indonesia Group yang bergerak dibidang depo container (penyimpanan container yang biasa digunakan untuk kegiatan ekspor impor) . Container yang disimpan adalah container kosong milik perusahaan container. Dalam kegiatannya, PT. Masaji Tatanan Container (MTCon) sudah mengaplikasikan penggunaan barcode dalam proses layanan. Barcode tersebut berada pada surat pengambilan container (bon muat) yang dimiliki oleh customer. Surat tersebut nantinya akan digunakan untuk proses pengambilan container (gate out).

2.2. Konsep Dasar Program

Aplikasi dengan hasil yang optimal berasal dari sebuah pembuatan program yang dibangun secara bertahap dan terstruktur dalam prosesnya pembangunannya.

Menurut Irawan (2009:137) ”program adalah perangkat lunak yang terdiri dari sekumpulan kode, bisa dieksekusi dan dijalankan melalui sistem operasi”. Adapun tahapan-tahapan secara umum dalam pembangunan program antara lain :

1. Definisi Masalah

Tahapan ini bertujuan untuk merumuskan dari permasalahan yang ada, sehingga akan didapatkan usulan-usulan yang dapat menyelesaikan permasalahan tersebut.

2. Analisa Kebutuhan

   Tahapan ini bertujuan untuk menentukan masukan-masukan dan keluarannya, mengetahui kebutuhan-kebutuhan penggunanya, menentukan spesifikasi minimum dan spesifikasi yang disarankan dari perangkat yang akan digunakan sebagai penerapan aplikasi tersebut.

3. Desain Algoritma

   Untuk menentukan urutan langkah-langkah komputasi yang tepat yang nantinya akan dieksekusi oleh komputer untuk memproses masukan-masukan mejadi keluaran yang diinginkan.

4. Pengkodean

   Tahapan penulisan pada teknologi pemrograman yang dipilih dengan mengikuti dari desain algortima yang sudah dibuat sebelumnya.

5. Testing

   Tahapan yang bertujuan untuk mengetahui kesalahan-kesalahan yang mungkin masih ada dalam aplikasi yang sedang dibangun, yang kemudian dilakukan koreksi sampai aplikasi dibebaskan dari kesalahan.

6. Dokumentasi

   Menyimpan semua informasi mengenai cara mengoperasikan aplikasi tersebut, prosedur-prosedur yang bekerja dan bagaimana aplikasi tersebut berjalan.

7. Pemeliharaan

Pemeliharaan digunakan untuk memantau aktivitas dari analisis sistem pada saat perangkat lunak telah dipergunakan oleh pemakai.

2.2.1. Java

Menurut Supardi (2010:1) mengatakan "Java merupakan bahasa pemprograman yang dikembangkan dari bahasa pemprograman C++. Sehingga bahasa pemprograman ini seperti bahasa C++ ".

Bahasa Java merupakan bahasa pemrograman berorientasi objek yang dibuat berdasarkan kemampuan-kemampuan terbaik bahasa pemrograman objek sebelumnya.

Beberapa karakteristik bahasa pemrograman Java antara lain :

1. Sederhana (*simple*)

   Menggunakan sintaks mirip dengan C++ namun telah banyak diperbaiki. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection*.

2. Berorientasi Objek (*object oriented*)

   Memodelkan dunia nyata kedalam objek. Membuat program dapat dibuat secara modular dan dapat digunakan kembali.

3. Terdistribusi (*Distributed*)

   Dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries networking* yang terintegrasi pada Java.

4. *Interpreted*

   Menggunakan *Java Virtual Machine*, sehingga *source code* Java yang telah dikompilasi menjadi Java *bytecodes* dapat dijalankan di platform yang berbeda

5. *Robust*

Mampu mendeteksi error secara lebih teliti dibandingkan dengan bahasa pemrograman lain.

6. *Secure*

   Memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7. *Architecture Neutra*l

   Program cukup mempunyai satu buah versi yang dapat dijalankan pada platform berbeda dengan *Java Virtual Machine*.

8. *Portable*

   Source code maupun program Java dapat dengan mudah dibawa ke *platform* yang berbeda-beda tanpa harus di kompilasi ulang.

9. *Performance*

   Performance Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan *Inprise*.

10. *Multithreaded*

    Kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.

11. *Dynamic*

    Perubahan pada suatu *class* dengan menambahkan *properties* ataupun *method* dapat dilakukan tanpa mengganggu program yang menggunakan *class* tersebut.

2.2.2. Android

Menurut Safaat (2011:1) mengungkapkan bahwa "*Android* adalah sebuah sistem informasi untuk perangkat *mobile* berbasis linux yang mencakup sistem informasi, *middleware* dan aplikasi". Secara umum Android adalah platform yang

terbuka (*Open Source*) bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh berbagai piranti bergerak.

Perangkatberbasis*android*hanyamempunyaisatulayar*foreground*. Saatmenghidupkan*android*, yang pertama kali terlihatadalah*home*, kemudianbilamemulaisebuahaplikasi, maka*User Interface* (UI) aplikasitersebutakanmenumpukdiataslayarsebelumnya (*home*). Semua proses tersebutdirekam di *application stack*olehsistem*activity manager*. Menekantombol back akanmembuatkitakembalikehalamansebelumya.

Pada proses pengembanganaplikasiberbasis*android*adabeberapakomponenaplikasi yang sangatharusdiperhatikan, antaralain :

1. *Activity*

   Fungsi *activity* adalah untuk mewakili satu buah *user interface*.

2. *Service*

   Service tidak memiliki *user interface*, namun berjalan di belakang layar.

3. *Intens*

   Intens adalah mekanisme untuk menggambarkan sebuah *action* secara detail seperti bagai mana cara mengambil sebuah photo.

4. *Content Providers*

   Menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu*activity*, seperti aplikasi berbasis peta.

5. *Resource*

   Resource digunakan untuk menyimpan*file-file non-coding* yang diperlukan pada sebuah aplikasi.

2.2.3. Eclipse

Menurut Satyaputra dan Aritonang (2012:12) “Eclipse merupakan sebuah IDE yang gratis dan *open source* atau yang dapat dikembangkan dan digunakan untuk membangun sebuah program komputer dan dapat dijalankan di semua *platform*”

Eclipse yang diluncurkan oleh IBM pada tanggal 5 November 2001 telah mengeluarkan beberapa versi sejak awal peluncurannya. Berikut adalah versi Eclipse sejak awal peluncurannya:

Tabel II.1. Versi Eclipse

KodePeluncuran	**TanggalPeluncuran**	**Versi**	**Nama Proyek**
Eclipse 3.0	21 Juni 2004	3.0	-
Eclipse 3.1	28 Juni 2005	3.1	-
Callisto	30 Juni 2006	3.2	Callisto*projects*
Europa	29 Juni 2007	3.3	Europa *projects*
Ganymede	25 Juni 2008	3.4	Ganymede *projects*
Galileo	24 Juni 2009	3.5	Galileo *projects*
Helios	23 Juni 2010	3.6	Helios *projects*
Indigo	22 Juni 2011	3.7	Indigo *projects*
Juno	27 Juni 2012	3.8 dan 4.2	Juno *projects*
Kepler	Juni 2013	Dalamrencana	Kepler *projects*

Sumber: SatyaputradanAritonang (2012:14)

2.2.4. Android SDK

Menurut Safaat (2011:15) menjelaskan bahwa “Android SDK adalah tool API(Applicaramming Interface) yang diperlukan untuk memulai mengembangkan aplikasi pada platform android menggunakan bahasa pemprograman java”. Untuk source SDK android ini dapat dilihat dan diunduh langsung dari situs resmi penembang android di http//:www.developer.android.com.

2.3. Metode Algoritma

Dalam perancangan aplikasi ini, algoritma yang digunaka adalah algoritma searching yaitu sesuai dengan source code aplikasi yaitu

```
/** Class filter untukmelakukan filter (pencarian) */
      privateclass BarangFilter extends Filter{

            @Override
            protected FilterResults performFiltering(CharSequence
constraint) {
                  List<Contaner> filteredData = new
ArrayList<Contaner>();
                  FilterResults result = new FilterResults();
                  String filterString =
constraint.toString().toLowerCase();
                  for(Contaner brg: list){
                        if(brg.getNoContaner()
.toLowerCase().contains(filterString) || brg.getNamaPT()
.toLowerCase().contains(filterString)){
                              filteredData.add(brg);
                        }
                  }
                  result.count = filteredData.size();
                  result.values =  filteredData;
                  return result;
            }

            @Override
            protectedvoid publishResults(CharSequence constraint,
FilterResults results) {
                  filterd = (List<Contaner>) results.values;
                  notifyDataSetChanged();
            }
```

2.4. Pengujian Sistem

Dalam melakukan pengujian pada program , penulis menggunakan*black box testing*dan *white box testing.*

Menurut Pressman (2010:597) "pengujian kotak hitam, juga disebut pengujian perilaku, berfokus pada persyaratan fungsional perangkat lunak". Dengan demikian, teknik pengujian kotak hitam memungkinkan perakayasa perangkat lunak untuk membuat beberapa kumpulan kondisi masukan yang sepenuhnya akan melakukan semua kebutuhan fungsional untuk suatu program. Pengujian kotak hitam bukan teknik alternatif untuk teknik kotak putih. Sebaliknya ini merupakan pendekatan pelengkap yang mungkin dilakukan untuk mengungkap kelas kesalahan yang berbeda dari metode kotak putih.

Menurut Pressman (2010:597) pengujian kotak hitam berupaya untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang salah atau hilang
2. Kesalahan antarmuka
3. Kesalahan dalam struktur data atau akses basis data eksternal
4. Kesalahan perilaku atau kinerja
5. Kesalahan inisialisasi dan penghentian.

Tidak seperti pengujian kotak putih yang dilakukan pada awal proses pengujian, pengujian kotak hitam cenderung diterapkan selama tahap-tahap pengujian selanjutnya. Karena pengujian kotak hitam sengaja mengabaikan struktur kendali, perhatian difokuskan pada ranahinformasi.

Menurut Efendi (2011:50) menjelaskan bahwa " Pengujian *White Box* merupakan pengujian yang dititik beratkan kepada prosedur cara program tertentu bekerja dengan memberikan suatu kondisi". White Box Testing adalah pengujian dengan melihat ke dalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak. Jika ada modul yang menghasilkan output yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris-baris program, variabel, dan parameter yang terlibat pada unit tersebut akan dicek satu persatu dan diperbaiki, kemudian di-*compile* ulang.

2.5. Peralatan Pendukung

Dalampembuatan aplikasi *android* ini diperlukan berbagai macam peralatan pendukung yang dapat membantu terbentuknya suatu aplikasi mulai dari logika hingga konsep pemrogramannya.

2.5.1 UML (*Unifield Modeling Language*)

Menurut Sukamto dan Shalahuddin (2013:133) menjelaskan bahwa "UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin divisualkan",

Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman yang berbasis objek yaitu *Unifield Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk mengimplmentasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk

pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

Secara analogi jika dengan bahasa yang kita gunakan sehari-hari belum tentu penyampaian bahasa dengan puisi adalah hal yang salah. Sistem informasi bukanlah ilmu pasti, maka jika ada banyak perbedaan dan implementasi didalam bidang sistem informasi merupakan hal yang sangat wajar.

1. *Class Diagram*

Menurut Sukamto dan Shalahuddin (2013:141) menjelaskan bahwa " Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem".

Kelas memiliki apa yang disebut atribut dan metode atau operasi.

a. Artibut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

2. *Use Case Diagram*

Menurut Sukamto dan Shalahuddin (2013:155) menjelaskan bahwa *"Use Case* atau diagram *use case* merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat".*Use case* mendeskripsikan sebuah intraksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat, *use case* digunakan untuk mengetahui fungsi apa saja ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami, ada dua hal utama pada *use case* yaitu pendefinisian atau apa saja yang disebut aktor dan *use case.*

a. Aktor merupakan orang, proses atau sistem lain yang berintraksi dengan sistem informasi ayang akan dibuat diluar sistem sistem informasi yang akan dibuat itu sendiri jadi walaupun jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

3. Activity Diagram

Menurut Sukamto dan Shalahuddin (2013:161) menjelaskan bahwa "Diagram aktivitas atau *activity diagram* menggambarkan *work flow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis",

Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor jadi aktivitas yang dapat dilakukan sistem.

4. Sequence Diagram

Menurut Sukamto dan Shalahuddin (2011:137) menjelaskan bahwa " Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek".

Oleh karena itu untuk menggambarkan diagram sekuen makan harus diketahui objek-objek yang terlibat dalan *use case* beserta metode-metode yang dimiliki kelas yang diimplementasikan menjadi objek itu.Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang didefiniskan maka diagram sekuen yang harus dibuat juga semakin banyak.