BAB II

TINJAUAN PUSTAKA

2.1 Profil Kostanbandung.com

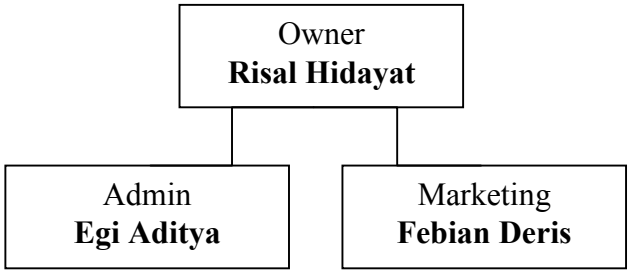
Kostanbandung.com adalah sebuah aplikasi berbasis *website* yang awalnya dibangun untuk memnuhi syarat tugas besar mata kuliah aplikasi dan teknologi mahasiswa UNIKOM jurusan teknik informatika angkatan 2007.

Tidak hanya sampai disana, aplikasi kostanbandung.com ini juga pernah dilibatkan dalam Pekan Ilmiah Mahasiswa Nasional (PIMNAS) XXIV mewakili kampus UNIKOM yang diselenggarakan oleh Direktorat Jenderal Pendidikan Tinggi (Ditjen *Dikti*) Kementerian Pendidikan Nasional walaupun harus gugur karena persaingan yang sangat ketat dari berbagai kampus di seluruh Indonesia.

Melihat respon yang baik dari pengguna Aplikasi Kostanbandung.com maka aplikasi ini perlu adanya pengembangan ke arah *mobile*. Dikarenakan penggunaan *smartphone* yang telah menjadi kebutuhan primer di masyarakat dan penggunaanya yang lebih fleksibel.

2.2 Struktur organisasi

Berikut adalah struktur organisasi dari Kostanbandung.com



Gambar II.1Struktur Organisasi

2.3 Aturan Bisnis dan Alur Pemesanan

Dalam aplikasi kostan bandung ini terdapat fasilitas *booking,* dimana pengguna dapat melakukan pemesanan kamar langsung pada sistem ini.

Berikut merupakan uraian mengenai alur pemesanan dan aturan bisnis yang digunakan pada sistem ini ;

1. Untuk mendapatkan kostan yang diharapkan dan supaya tidak kehabisan, maka pengunjung atau *member* pencari kost diharapkan *membooking* terlebih dahulu kostan yang sesuai dengan keiingannya.
2. Sistem *booking* dapat dilakukan melalui *website* ataupun *mobile* android, (fitur *booking* tersedia) atau dapat dilakukan dengan menghubungi secara langsung pihak pemilik kost (jika kontak pemilik kost tersedia).
3. *Booking* dinyatakan batal apabila tidak ada konfirmasi pembayaran yang dilakukan oleh pencari kostan terhadap pemilik kostan.

4. Pembatalan *booking* dilakukan secara otomatis oleh sistem dengan estimasi waktu yang ditawarkan hanya 24jam setelah proses *booking* dilakukan.
5. Transaksi dilakukan secara langsung antara pencari kostan dengan pemilik kostan.
6. Pihak pengelola *website* tidak ada campur tangan dalam masalah transaksi serta tidak meminta biaya apapun kepada pengguna dalam penggunaaan sistem ini.

2.4 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan aplikasi Kostanbandung.com.

2.4.1 Object Oriented Programming (OOP)

OOP merupakan terobosan baru dalam pemrograman setelah pemrograman prosedural, pemrograman modular dan pemrograman abstraksi data. Ada beberapa konsep utama OOP antara lain :

Pemrograman berorientasi-objek menekankan konsep berikut:

1. Class : kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing.

2. Objek : membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
3. Abstraksi : kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.
4. Enkapsulasi : memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.
5. Polimorfisme : melalui pengiriman pesan. tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan

menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelaspertama.

6. Inheritance - Mengatur polimorfisme dan enskapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada, objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut(terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bagian administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk

menyelesaikan suatu masalah dengan kolaborasi antar objek -objek yang ada karena setiap objek memiliki job description nya sendiri [1].

2.4.2 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Dikembangkan oleh Sun Microsystems dan diterbitkan tahun 1995. Java tidak boleh disalahpahami sebagai JavaScript. JavaScript adalah bahasa scripting yang digunakan oleh web browser [2].

2.4.3 Sejarah perkembangan Java

Bahasa pemrograman Java pertama lahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang [2].

Pertemuan proyek berlangsung di sebuah gedung perkantoran Sand Hill Road di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program Java Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (touch screen), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (Star Seven) [2].

Setelah era Star Seven selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto [2].

Perusahaan baru ini bertambah maju, jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara 33 mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer [2].

Mereka menjadikan perambah (browser) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java [2].

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar San Jose Mercury News pada tanggal 23 Mei 1995 [2].

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape [2].

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah Java) [2].

2.4.3.1 Versi Awal

Versi awal Java ditahun 1996 sudah merupakan versi release sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

1. Java.lang: Peruntukan kelas elemen-elemen dasar.
2. Java.io: Peruntukan kelas input dan output, termasuk penggunaan berkas.
3. Java.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas kelas penanggalan.
4. Java.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
5. Java.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI)
6. Java.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

2.4.3.2 Kelebihan

1. Multiplatform. Kelebihan utama dari Java ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / bytecode) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows.

Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mrac OS dan Sun Solaris. Penyebanya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk menginterpretasikan bytecode tersebut.

2. OOP (Object Oriented Programming - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Java merupakan salah satu bahasa pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut Object. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan Java sebagai salah satu bahasa pemograman termudah, bahkan untuk fungsi fungsi yang advance seperti komunikasi antara komputer sekalipun.
3. Perpustakaan Kelas yang Lengkap, Java terkenal dengan kelengkapan library/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.
4. Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke

Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.

5. Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.4.3.3 Kekurangan

1. Masih ada beberapa hal yang tidak kompatibel antara platform satu dengan platform lain. Untuk J2SE, misalnya SWT-AWT bridge yang sampai sekarang tidak berfungsi pada Mac OS X.
2. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan bytecode yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di-reverseengineer.
3. Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang

menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.4.3.4 Tahap Kompilasi Java

1. Tulis / Ubah. Pemrogram menulis program dan menyimpannya di media dalam bentuk berkas '.java'.
2. Kompilasi. Pengkompilasi membentuk bytecodes dari program menjadi bentuk berkas '.class'.
3. Muat. Pemuat kelas memuat bytecodes ke memori.
4. Verifikasi. Peng-verifikasi memastikan bytecodes tidak mengganggu sistem keamanan Java.
5. Jalankan. Penerjemah menerjemahkan bytecodes ke bahasa mesin tidak bisa di pakai.

2.5 *Web service*

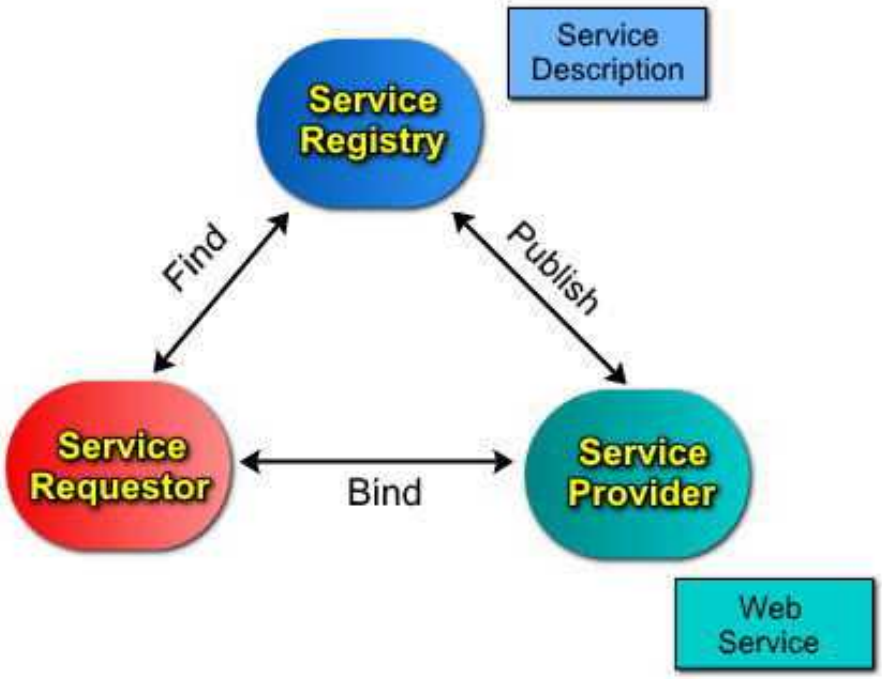
Web service adalah salah satu bentuk sistem perangkat lunak yang didesain untuk mendukung interaksi mesin-ke-mesin melalui jaringan. *Web service* memiliki *interface* yang dideskripsikan dalam format yang dapat dibaca oleh mesin. Sistem-sistem lainnya berinteraksi dengan *web service* menggunakan pesan SOAP yang umumnya dikirim melalui HTTP dalam bentuk XML [10].

Definisi diatas diberikan oleh *World Wide Web Consortium* (W3C) yang merupakan badan yang menciptakan dan mengembangkan standar *web service.* Tetapi secara umum, *web service* tidak terbatas hanya pada standar SOAP saja.

Salah satu pustaka yang mengulas lengkap tentang *web service* menyebutkan definisi yang lebih umum: *web service* adalah aplikasi yang diakses melalui *internet* menggunakan protokol standar *internet* dan menggunakan XML sebagai format pesannya.

2.5.1 Arsitektur *Web service*

Secara umum, arsitektur *web service* dapat dilihat pada gambar II.2.



Gambar II.2 Arsitektur Web Service [11].

Pada gambar diatas, ada tiga komponen yang membuat *web service* berjalan. Ketiga komponen itu adalah:

1. ***Service provide**r*, merupakan pemilik *Web service* yang berfungsi menyediakan kumpulan operasi dari *Web service*.
2. ***Service requesto**r*, merupakan aplikasi yang bertindak sebagai klien dari *Web service* yang mencari dan memulai interaksi terhadap layanan yang disediakan.
3. ***Service registry***, merupakan tempat dimana *Service provider* mempublikasikan layanannya. Pada arsitektur *Web service*, Service registry bersifat optional. Teknologi *web service* memungkinkan kita dapat menghubungkan berbagai jenis *software* yang memiliki *platform* dan sistem operasi yang berbeda.

2.5.2 Jenis-Jenis *Web service*

Jenis *web service* dapat dibagi menjadi dua, yaitu REST dan SOAP.

2.5.2.1 Representational State Transfer (REST)

REST adalah salah satu jenis *web service* yang menerapkan konsep perpindahan antar *state*. *State* disini dapat digambarkan seperti jika *browser* meminta suatu halaman *web*, maka *server* akan mengirimkan *state* halaman *web* yang sekarang ke *browser*. Bernavigasi melalui *link-link* yang disediakan sama halnya dengan mengganti *state* dari halaman *web*. Begitu pula REST bekerja, dengan bernavigasi melalui *link-link* HTTP untuk melakukan aktivitas tertentu, seakan-akan terjadi perpindahan *state* satu sama lain. Perintah HTTP yang bisa digunakan adalah fungsi *GET, POST, PUT* atau *DELETE*. Balasan yang dikirimkan adalah dalam bentuk XML sederhana tanpa ada protokol pemaketan

data, sehingga informasi yang diterima lebih mudah dibaca dan diparsing disisi *client*.

Dalam pengaplikasiannya, REST lebih banyak digunakan untuk *web service* yang berorientasi pada *resource*. Maksud orientasi pada *resource* adalah orientasi yang menyediakan *resource-resource* sebagai layanannya dan bukan kumpulan-kumpulan dari aktivitas yang mengolah *resource* itu. Alasan mengapa REST tidak digunakan dalam skripsi ini karena orientasi pada *resource*nya itu, sedangkan aplikasi lowongan pekerjaan membutuhkan pemanggilan metode yang bisa dikerjakan terhadap kumpulan *resource* data lowongan. Selain itu, karena standarnya yang kurang sehingga tidak begitu cocok diterapkan dalam aplikasi yang membutuhkan kerjasama antar aplikasi lain, dimana standar yang baik akan sangat berguna karena berbicara dalam satu bahasa yang sama. Beberapa contoh *web service* yang menggunakan REST adalah: Flickr API(*Application Program Interface*), YouTube API, Amazon API.

2.5.2.2 Simple Object Access Protokol (SOAP)

SOAP merupakan protokol standar yang ringan dan ditujukan untuk pertukaran informasi dalam desentralisasi sistem. Protokol ini melakukan pemanggilan metode dalam bahasa XML (*Extensible Markup Language*). Dengan demikian, SOAP adalah suatu mekanisme yang sederhana untuk melakukan pertukaran struktur dan tipe informasi dalam lingkungan yang tersebar dan terdistribusi menggunakan XML.

Secara garis besar, protocol SOAP dapat dibagi menjadi beberapa bagian berikut :

a. SOAP *Envelope*

Untuk mendefinisikan apa yang terdapat dalam pesan (*message*) yang dikirim, dan kepada siapa pesan tersebut akan digunakan. Elemen ini merupakan root dari pesan dan bersifat wajib.

b. SOAP *Header*

Elemen ini berisi informasi header dan bersifat opsional. Informasi yang ada pada *header* tidak ditampilkan di aplikasi.

c. SOAP *Body*

Elemen ini berisi panggilan dan merespon informasi. Elemen ini wajib.

d. SOAP *Encoding Rules*

Mendefinisikan mekanisme serialisasi yang dapat digunakan untuk pertukaran tipe data yang didefinisikan dari tiap aplikasi.

e. SOAP RPC *Representation*

Menyediakan suatu standar yang dapat digunakan untuk mewakili *remote procedure call* (RPC) dan data hasil kembaliannya.

f. SOAP *Binding*

Mendefinisikan standar yang digunakan utnuk pertukaran SOAP *Envelope* antara *peers* dengan protokol yang umum untuk transportasi.

Sintaks dari protokol SOAP adalah sebagai berikut :

```
<?xml version="1.0"?>
<soap:Envelope
      xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
      soap:encodingStyle="http://      www.w3.org/2001/12/soap-
envelope">
<soap:Header>
```

```
  . . .
  . . .
</soap:Header>
<soap:Body>
  . . .
  . . .
      <soap:Fault>
        . . .
        . . .
      </soap:Fault>
</soap:Body>
</soap:Envelope>
```

2.5.3 Web service Definition Language (WSDL)

WSDL merupakan suatu dokumen berbasis XML yang berfungsi untuk menjelaskan fungsi *public* dan metode-metode yang tersedia dalam *web service*. Selain itu, WSDL juga mengatur *end-point* pada pesan-pesan yang berisi dokumen maupun informasi prosedur.

Dokumen WSDL sanagat kompleks, dan memungkinkan untuk mendeskripsikan *end-point* serta pesan yang ada pada protokol jaringan. Salah satu bagian dokumen WSDL yang cukup penting dalam melakukan interaksi dengan SOAP, HTTP, serta MIME adalah *binding*.

SOAP dapat berjalan sendiri tanpa bantuan WSDL, tetapi hal ini akan menyulitkan pengembang di sisi *client* karena tidak tahu harus mengharapkan operasi dan bentuk data apa. Karena itu WSDL diciptakan sebagai pendefinisi sebuah *web service*. Dalam tugas akhir ini, *web service* yang dibangun akan dijabarkan menggunakan WSDL.

Elemen-elemen yang terdapat dalam WSDL adalah sebagai berikut :

1. *Elemen Types*

   Elemen Types menyertakan definisi tipe data yang sesuai untuk pertukaran *message*. Untuk meningkatkan operasi dari dalam dan dukungan netral *platform*, WSDL sering menggunakan XSD sebagai peraturan sistem dan memperlakukannya sebagai tipe sistem pada umumnya.

2. *Elemen Message*

   Message terdiri dari satu atau beberapa bagian, dan masing-masing bagian digabung dengan beberapa tipe sistem menggunakan atribut.

3. *Elemen Port Type*

   Elemen ini menjelaskan tentang operasi dan abstrak *message* yang terlibat di dalamnya. Atribut "*name*" dari elemen ini menyediakan nama unik di antara semua jenis port yang didefinisikan pada penutup dokumen WSDL.

4. *Elemen Binding*

   Binding berfungsi untuk menyatukan format *message* dan detail protokol untuk operasi. Umumnya ada beberapa nomor *binding* untuk menentukan tipe port.

5. *Elemen Port*

   Elemen ini bukanlah elemen *port type*, namun subelemen dalam elemen *service*. Fungsinya untuk mendefinisikan *end-point* individu melalui alamat tunggal khusus untuk *binding*.

6. *Elemen service*

   Kelompok *services* digunakan untuk mengatur hubungan yang terjadi antara port. Dengan demikian, elemen-elemen port akan diletakkan dalam *group services*.

Sintaks dari dokumen WSDL adakah sebagai berikut :

```
<definitions>
<types>
   definition of types……
</types>
<message>
   definition of message…….
</message>
<portType>
   definition of port……...
</portType>
<binding>
   definition of bind…….
</binding>
</definitions>
```

2.5.4 UDDI (*Uniform Description, Discovery, and Integration*)

Dalam penerapannya, *web service* memerlukan sumber dari semua informasi tentang *SOAP message*, cara membuat respons, mengumumkan *service* yang ada, dan sebagainya. Dengan demikian, aplikasi dapat menggunakannya untuk mencari WSDL. Untuk memenuhi hal tersebut, diperlukan UDDI yang menyediakan *service* untuk mencari informasi yang diperlukan.

UDDI memungkinkan suatu perusahaan untuk mengumumkan detail dari *web service* yang tersedia, biasanya melalui kata kunci atau deskripsi. Dengan

adanya UDDI tersebut, kita dapat mengetahui (*discover*) dimana *web service* perusahaan berada.

2.5.5 Konsep Dasar Basis Data

Basis data (*Database*) dapat dibayangkan sebagai sebuah lemari arsip yang ditempatkan secara berurutan untuk memudahkan dalam pengambilan kembali data tersebut. Basis data menunjukkan suatu kumpulan data yang dipakai dalam suatu lingkungan perusahaan atau instansi. Penerapan basis data dalam system informasi disebut sebagai sistem basis data (*database system*) [3].

2.5.5.1 Pengertian Basis Data

Basis data terdiri dari kata basis dan data, Basis Data diartikan sebagai gudang atau tempat bersarang dan Data yang berarti representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, hewan, peristiwa, konsep dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya.

Jadi dapat disimpulkan bahwa Basis Data merupakan kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (*redudancy*) yang tidak perlu, untuk memenuhi berbagai kebutuhan. Atau bisa diartikan sebagai kumpulan file atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elekronis [3].

2.5.5.2 Perancangan Basis Data

Perancangan basis data dibutuhkan agar diperoleh suatu sistem yang sesuai dengan apa yang diinginkan, dimana dapat melalui tahapan berikut :

a. *Entity Relationship Diagram*

ERD atau biasa disebut dengan diagram E-R secara grafis menggambarkan isi sebuah *database.* Diagram ini memiliki dua komponen utama yaitu entity dan relasi. Untuk melambangkan fungsi diatas maka digunakan simbol-simbol yang bisa dilihat pada daftar simbol.

b. Normalisasi

Normalisasi merupakan tahapan perancangan dalam membangun basis data relasional yang tidak secara langsung berkaitan dengan model data tetapi menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal. Adapun langkah-langkah untuk menghasilkan struktur tabel yang normal diantaranya yaitu :

1. Bentuk *Unnormallized*

Bentuk *unnormal* adalah bentuk tabel dengan mencantumkan semua field data yang ada.

2. Bentuk Normal Pertama / 1 NF (*First Normal Form*)

Bentuk normal pertama terpenuhi jika sebuah tabel tidak memiliki atribut bernilai banyak (*Multivalued Atribute*) atau lebih dari satu atribut dengan domain nilai yang sama.

3. Bentuk Normal Kedua / 2 NF (*Second Normal Form*)

Bentuk normal kedua terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam *key primer* memiliki ketergantungan fungsional pada *key primer* secara utuh.

4. Bentuk Normal Ketiga / 3 NF (*Third Normal Form*)

Bentuk normal ketiga terpenuhi jika dan hanya jika semua atribut bukan kunci memiliki dependensi transitif terhadap kunci primer.

5. *Boyce-Codd Normal Form* (BCNF)

Suatu tabel dikatakan berada pada BCNF jika dan hanya jika semua ketergantungan fungsional dengan notasi X→Y, maka X harus merupakan *superkey* pada tabel tersebut.

6. Bentuk Normal Keempat / 4 NF (*Fourth Normal Form*)

Suatu tabel dikatakan berada pada normal keempat jika tidak mengandung dua atribut atau lebih yang bernilai banyak.

7. Bentuk Normal Kelima / 5 NF (*Fifth Normal Form*)

Bentuk normal kelima berkaitan dengan ketergantungan relasi antar table (*join dependency*).

8. Relasi Tabel

Relasi tabel menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Dari adanya relasi tabel tersebut terdapat kardinalitas relasi yang menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Terdapat empat kemungkinan kombinasi kardinalitas yaitu satu ke satu (*one to one*), satu ke

banyak (*one to many*), banyak ke satu (*many to one*) dan banyak ke banyak (*many to many*) [3].

2.5.6 Pengenalan MySQL

MySQL adalah salah satu jenis database *server* yang sangat terkenal. Kepopulerannya disebabkan MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses databasenya. MySQL termasuk jenis RDBMS *(Relational Database Management System)*. Pada MySQL, sebuah database mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom [2].

Untuk mengelola database MySQL ada beberapa cara yaitu melalui *prompt* DOS *(tool command line)* dan dapat juga menggunakan program *utility* seperti:

1. PHP MyAdmin
2. MySQLGUI
3. MySQL Manager Java Based
4. MySQL Administrator for windows.

2.5.7 Wamp Server

Wamp Server adalah sebuah manager service yang akan menginstalApache, PHP5, database MySQL, PHPmyadmin dan SQLitemanager di komputer anda. WampServer 2.0e dahulu bernama WAMP5. Dengan WampServer anda tidak akan perlu ribet ketika harus menginstal Apache, PHP dan MySQL di Windows. Apache, PHP dan MySQL merupakan perpaduan yang

komplit dalam membuat sebuah website namun terkadang instalasi ketiga program ini sering menjadi kendala bagi pengguna awam. Bagi yang ingin belajar membuat website dengan memanfaatkan Apache, PHP dan MySQL atau ingin menguji website di komputernya sebelum diupload maka diperlukan sebuah environment yang serupa yang membuat pengguna seolah-olah sedang dalam environment 'live'. Environment itu bisa kita ciptakan dengan menginstal WampServer.

2.5.8 Android

Android merupakan suatu *software stack* untuk *mobile device*. Di dalamnya terdapat sistem operasi, *middleware*, dan *key application*. Aplikasi pada platform ini dikembangkan dalam bahasa pemrograman java [4].

Android sendiri memiliki banyak fitur diantaranya adalah:

- Merupakan sebuah **A*pplication Framework*** sehingga *programmer* dapat menggunakan beberapa fungsi yang telah disediakan.
- **Dalvik *virtual machine***. Tiap aplikasi dalam Android memiliki *instance virtual machine* yang dapat bekerja secara efisien dalam lingkungan memoriyang terbatas.
- ***Integrated browser*.** Web browser berbasis WebKit *engine* terdapat pada browser *default* Android atau pun dapat diintegrasikan dengan aplikasi lain.
- ***Optimized graphics***. *Library* grafis 2D yang kaya dan 3D berbasis OpenGL ES 1.0 yang mendukung akselerasi hardware.

- **SQLite**. Basis data relasional yang ringan namun sangat *powerful.*
- ***Media Support***. Mendukung berbagai format audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM *Telephony***. Mendukung fungsi komunikasi GSM.
- **Bluetooth,EDGE,3G** dan **WiFi**. Mendukung komunikasi pada jaringan (tergantung hardware).
- **Kamera**, **GPS**, **kompas** dan ***accelerometer***. Mendukung berbagai fitur yang disediakan oleh *hardware*.
- **Kakas pengembangan yang lengkap**. Termasuk *device emulator, tools* untuk *debugging*, *profiling* memori dan performa, *plugin* untuk Eclipse IDE.

2.5.9 Arsitektur Android

Android terdiri dari beberapa *stack software* yang terdiri dari: *Applications, Application Framework, Libraries, Android Runtime* dan *Kernel Linux.* Arsitekturlengkap platform ini dapat dilihat pada Gambar II.1.



Gambar II.2 Arsitektur platform android

2.5.9.1 Aplication

Application merupakan program yang langsung berhubungan dengan *user*. Baik program yang merupakan bawaan dari Android sendiri maupun program yang dibuat oleh *developer* menggunakan bahasa pemrograman java. Contoh program bawaan dari platform Android sendiri adalah *email client*, program SMS, *calendar*, *maps*, *web browser*, *contact* dan sebagainya [4].

2.5.9.2 Aplication Framework

Lapisan ini berisi sekumpulan API yang dapat digunakan oleh *programmer* maupun *core application* dari Android. Lapisan ini dirancang untuk memudahkan penggunaan komponen dari Android sendiri. Aplikasi manapun dalam Android dapat berbagi fungsi sehingga aplikasi lain dapat memanfaatkannya [4].

Aplikasi pada Android disusun atas beberapa komponen:

- Sekumpulan *Views*. Digunakan untuk mengatur tampilan pada aplikasi. Contohnya adalah *lists, grids, text box, button*, bahkan *embeddable webbrowser.*
- *Content providers.* Komponen yang mengatur agar aplikasi dapat mengakses *resources* dari aplikasi lain (seperti *Contacts*), atau berbagi data dengan aplikasi lain.
- *Resource Manager*. Menyediakan akses ke pada *resource non-code* seperti *localized string*, grafik dan *file layout*.
- *Notification Manager*. Memungkinkan agar suatu aplikasi dapat menampilkan peringatan yang dapat di kostumasi pada *status bar.*
- *Activity Manager*. Mengatur siklus aplikasi dan navigasi antar aplikasi yang sedang berjalan.

2.5.9.3 Libraries

Android mendukung beberapa *library* C/C++ yang digunakan pada berbagai komponen Android. Kemampuan ini dapat diakses oleh *developer* melalui Android *application framework*. Beberapa *library* diantaranya adalah:

- *System* C *library.* Implementasi *library* C standar (libc).
- *Media Libraries*. Mendukung berbagai format multimedia (termasuk MPEG4,H.264, MP3, AAC, AMR, JPG, PNG).
- *Surface Manager*. Mengatur akses ke subsistem *display*.
- LibWebCore. *Engine web browser* modern.

- SGL. *Engine* grafis 2D.
- 3D *Library*. Implementasi OpenGL ES 1.0 yang mendukung akselerasi hardware.
- FreeType. *Rendering* untuk *bitmap* dan *vector font.*
- SQLite. Basis data relasional yang kecil namun sangat ampuh.

2.5.9.4 Android Runtime

Tiap aplikasi pada Android memiliki proses-nya masing-masing. Tiap aplikasi tersebut memiliki *instans* dari Dalvik *virtual machine* (VM). Dalvik *virtual machine* dirancang agar suatu *device* dapat menjalankan beberapa VM secara efisien. Dalvik VM *mengeksekusi* file dengan format Dalvik *Executable format* (.dex) yang dirancang untuk meminimalkan *memory footprint* . Dalvik VM berbasis *register*, dan dapat menjalankan kelas-kelas yang dikompilasi dengan bahasa pemrograman java dan ditransformasikan menjadi format .dex. Dalvik VM sendiri bergantung pada Kernel Linux untuk fungsi dasarnya, seperti *threading* dan manajemen memori secara *low-level* [4].

2.5.9.5 Linux Kernel

Android menggunakan Kernel Linux versi 2.6 sebagai sistem utama. Fungsi kernel yang digunakan antara lain untuk keamanan, manajemen memori, manajemen proses, manajemen jaringan dan *driver model*. Kernel juga berfungsi sebagai *layer* abstrak antara *hardware* dan lapisan lainnya pada *software stack* [4].

2.5.10 Anatomi Aplikasi Android

Terdapat empat *building blocks* pada aplikasi Android yaitu:

- *Activity*
- *Intent Receiver*
- *Service*
- *Content Provider*

Tidak semua aplikasi membutuhkan keempat blok ini, tetapi suatu aplikasi dibuat menggunakan kombinasi beberapa blok ini. Setelah memutuskan blok komponen yang akan digunakan, blok tersebut didaftarkan pada suatu *file* yang disebut dengan AndroidManifest.xml. *File* XML ini digunakan untuk menyatakan komponen apa saja yang dibutuhkan oleh sebuah aplikasi dan kemampuan serta kebutuhan aplikasi tersebut [4].

2.5.10.1 Activity

Activity paling umum diantara keempat *building block*. *Activity* biasanya suatu tampilan (*screen*) dalam suatu aplikasi. Tiap *activity* diimplementasikan sebagai satu kelas yang meng-*extend* base class *Activity*. Kelas yang dibuat akan menampilkan *user interface* yang terdiri dari beberapa *views* dan akan merespon suatu *event*. Kebanyakan, aplikasi terdiri dari beberapa *screen*. Contohnya aplikasi pengiriman pesan akan memiliki *screen* untuk mendaftarkan pesan yang masuk, menulis pesan dan *screen* lainnya untuk melakukan pengaturan. Tiap *screen* pada contoh ini akan diimplementasi sebagai suatu *activity*. Aplikasi dapat berpindah ke *screen* lain dengan cara memulai suatu *activity* baru. Pada suatu kasus tertentu,

activity akan mengembalikan nilai kepada *activity* sebelumnya. Contohnya *activity* untuk memilih foto akan mengembalikan foto yang dipilih pada *activity* yang memanggilnya. Pada saat *screen* baru terbuka, screen sebelumnya akan dihentikan dan *state* nya akan disimpan pada *history stack*. *User* dapat melakukan navigasi ke *activity* sebelumnya melalui *history* tersebut. *Screen* akan di hapus dari *history stack* apabila sistem memutuskan bahwa *screen* tersebut tidak layak untuk berada pada memori. Android menyimpan *history stack* untuk setiap aplikasi yang dijalankan dari tampilan awal (*home screen*) [4].

2.5.10.2 Intent dan Intent Filter

Android menggunakan kelas khusus bernama *intent* untuk berpindah dari *screen* ke *screen* lainnya. Sebuah *intent* mendeskripsikan apa yang ingin aplikasi penuhi. Bagian struktur data dari *intent* yang paling penting adalah *action* dan *data* dari aksi yang dilakukan. Nilai *action* yang biasanya adalah MAIN, VIEW, PICK, EDIT dan sebagainya. Data diekspresikan sebagai URI. Sebagai contoh, untuk melihat informasi suatu *contact* seseorang, kita membuat *intent* dengan action VIEW dan data sebagai URI yang merepresentasikan orang tersebut.

Terdapat suatu kelas yang dinamakan IntentFilter. Apabila *intent* adalah suatu permintaan untuk melakukan sesuatu, IntentFilter adalah deskripsi yang dapat dilakukan oleh *intent activity*. *Activity* yang dapat menampilkan informasi *contact* dari seseorang akan mempublikasikan IntentFilter yang menyatakan bahwa ia dapat menangani *action* VIEW apabila dipakai pada data yang

merepresentasikan seseorang. Suatu aplikasi mempublikasikan IntentFilter pada *file* AndroidManifest.xml [4].

Navigasi dari suatu *screen* ke *screen* lainnya dipenuhi dengan cara me-*resolve intent*. Untuk melakukan navigasi ke depan, *activity* memanggil startActivity(myIntent). Sistem akan mencari IntentFilter untuk semua aplikasi dan memilih *activity* yang IntentFilter-nya yang paling mendekati dengan myIntent. *activity* baru akan mendapat informasi tentang *intent* yang menyebabkan *activity* tersebut dijalankan. Proses pencarian *intent* terjadi sewaktu runtime pada saat startActivity dipanggil. Hal tersebut membawa keuntungan sebagai berikut:

- *Activity* dapat melakukan *reuse functionality* dari komponen lain dengan cara melakukan permintaan dalam bentuk *intent*.
- *Activity* dapat di *replace* kapan pun dengan *activity* yang ekuivalen.

2.5.10.3 Intent Recivier

IntentReceiver dapat digunakan pada saat kita menginginkan kode kita dijalankan akibat reaksi dari *external event* tertentu. Contohnya, pada saat telepon berbunyi, atau pada saat ada data pada jaringan, atau pada suatu waktu tertentu. IntentReceiver tidak menampilkan *user interface*, meskipun ia dapat menggunakan Notification Manager untuk memperingatkan *user* apabila sesuatu terjadi. *IntentReceiver* didaftarkan pada *file* AndroidManifest.xml, tetapi secara manual juga bisa ditambahkan dengan fungsi Context.registerReceiver(). Suatu aplikasi tidak harus berjalan pada *IntentReceiver*nya agar bisa dipanggil, tetapi

sistem yang akan menjalankan aplikasi tersebut. Apabila diperlukan, pada saat IntentReceiver mendapat *trigger*, aplikasi juga dapat mengirimkan *intent broadcast* kepada aplikasi lainnya dengan fungsi Context.broadcastIntent() [4].

2.5.10.4 Service

Service adalah sebuah kode yang hidup terus-menerus dan berjalan tanpa memiliki tampilan. Contoh yang sederhana adalah suatu aplikasi *media player* yang sedang memainkan lagu dari *play list.* Pada aplikasi *media player*, kemungkinan terdapat satu atau lebih activity yang memperbolehkan *user* memilih lagu dan memainkannya. Tetapi musik yang sedang diputar tidak boleh ditangani oleh activity karena *user* akan menginginkan musik tetap diputar meskipun ia berpindah ke *screen* baru. Pada kasus ini activity *media player* dapat menjalankan *service* menggunakan fungsi Context.startService() untuk menjalankan *service* di *background* yang memutar musik. Sistem akan tetap memutar musik sampai selesai. Selain itu *activity* dapat terhubung dengan *service* menggunakan fungsi Context.bindService(). Pada saat terhubung, aplikasi dapat berkomunikasi melalui *interface* yang diberikan oleh *service* yang bersangkutan. Pada contoh *media player*, *interface service* akan memberikan fungsi *pause*, *rewind* dan sebagainya [4].

2.5.10.5 Content Provider

Content provider digunakan untuk pertukaran data antara aplikasi. Data yang dipertukarkan bersifat *shared*. *Content provider* adalah sebuah kelas yang

berisi *method* standar yang memungkinkan aplikasi lain dapat menyimpan dan mengambil suatu data yang ditangani oleh *content provider* [4].

2.5.11 Siklus Hidup Aplikasi Android

Pada banyak kasus, tiap aplikasi pada Android masing-masing memiliki Linux proses. Proses ini diciptakan untuk aplikasi tersebut pada saat kode program tersebut akan dieksekusi, akan terus berjalan sampai tidak dibutuhkan lagi dan sistem memerlukan memori untuk aplikasi lain. Konsep dasar program Android adalah suatu proses pada aplikasi tidak langsung di atur oleh aplikasi tersebut. Proses tersebut ditentukan oleh sistem melalui kombinasi : informasi aplikasi bagi sistem, seberapa pentingnya aplikasi tersebut bagi *user* dan ketersediaan memori. Contoh kasus *life-cycle bug* adalah *IntentReceiver* yang menciptakan *thread* pada saat menerima *intent* pada *method* onReceiveIntent() dan selesai dari fungsi tersebut. Setelah fungsi tersebut selesai, sistem menganggap bahwa *IntentReceiver* tersebut menjadi tidak aktif sehingga ia memiliki proses yang tidak dibutuhkan (kecuali terdapat komponen aplikasi lain di dalamnya). Oleh karena itu, sistem dapat menghapus proses tersebut kapanpun apabila dibutuhkan. Hal tersebut akan menghapus semua *thread* yang berjalan pada proses tersebut. Solusi masalah ini adalah memakai *Service* dari IntentReceiver, sehingga sistem mengetahui bahwa ada suatu yang aktif dalam proses tersebut [4].

Sebagai *programmer*, kita harus mengetahui bagaimana cara yang tepat untuk merancang aplikasi. Kesalahan pada perancangan akan membuat suatu

aplikasi akan dihentikan oleh sistem pada saat yang tidak diinginkan. Untuk menentukan proses mana yang akan dihapus pada situasi *low memory*, Android memiliki "*importancehierarchy*" berdasarkan komponen dan *state* yang sedang berjalan. Urutan berdasarkan kepentingan tersebut adalah:

1. ***Foreground process*.** Adalah suatu proses yang dibutuhkan untuk berinteraksi dengan *user* saat ini. Berbagai komponen suatu aplikasi dapat mengakibatkan proses di dalamnya menjadi *foreground proses*. Suatu proses dikatakan *foreground* apabila memenuhi kondisi sebagai berikut:
   - Menjalankan Activity pada *top screen* yang sedang berinteraksi dengan *user* (setelah *method* onResume() dipanggil).
   - Mempunyai IntentReceiver yang sedang berjalan (method IntentReceiver.onReceiveIntent() sedang dieksekusi).
   - Memilik service yang fungsi *callback*-nya (Service.onCreate(), Service.onStart() atau Service.onDestroy()) sedang dieksekusi.
   - Hanya akan terdapat beberapa proses seperti itu dalam sistem, proses ini hanya akan dihapus apabila sistem tidak memiliki pilihan lain. Contohnya apabila memori tinggal sedikit sehingga proses *foreground* tidak bisa lagi berjalan. Hal ini diperlukan untuk menjaga *user interface* tetap interaktif dengan *user*.
2. ***Visible process*.** Adalah proses yang memegang Activity yang dapat dilihat *user* pada layar tetapi tidak *foreground* (pada saat *method* onPause() dipanggil). Hal ini dapat terjadi, sebagai contoh *activity foreground* muncul dengan dialog sehingga *activity* sebelumnya terlihat di belakangnya. Proses

ini sangat penting dan tidak akan di hapus kecuali untuk menjaga aplikasi *foreground* tetap berjalan.

3. ***Service process*.** Adalah proses yang memegang Service yang dimulai dengan method StartService().Meskipun proses ini secara langsung tidak dapat dilihat oleh *user*, biasanya proses ini menjalankan perintah dari *user* (seperti memutar lagu, men-*download* data dari jaringan dan lain-lain). Sistem akan tetap membiarkan proses ini berjalan selama proses *foreground* dan *visible* masih bisa berjalan.
4. ***Background process***. Adalah proses yang memegang Activity yang saat ini tidak dapat dilihat oleh *user* (method onStop() sedang dipanggil). Proses ini tidak secara langsung mempengaruhi *user*. Apabila *activity life cycle* telah dipenuhi dengan benar, sistem dapat menghapus proses ini sewaktu-waktu untuk memberikan memori tambahan. Biasanya banyak ditemukan proses semacam ini, sehingga proses tersebut disimpan dalam LRU (*Least RecentlyUsed*) untuk menjaga agar proses yang baru saja dilihat oleh *user* akan dihapus paling akhir pada kondisi *low memory*.
5. ***Empty Process*.** Adalah proses yang tidak memegang suatu komponen yang sedang aktif. Alasan mengapa membiarkan proses semacam ini adalah untuk melakukan *cache* sehingga proses *startup* menjadi lebih cepat apabila aplikasi tersebut akan dijalankan kembali. Sistem seringkali menghapus proses semacam ini untuk mencapai keseimbangan dari *system resource* antara cached processes dan kernel caches.

2.5.12 Tools Implementasi

Tools implementasi merupakan peralatan-peralatan yang dibutuhkan dalam membangun perangkat lunak. Adapun tools implementasi yang digunakan untuk membuat aplikasi pemesanan makanan pada restoran adalah JDK sebagai perangkat pembangun Java, Eclipse IDE sebagai lingkungan pengembangan perangkat lunak, dan SQL sebagai sistem manajemen basis data [4].

2.5.12.1 JDK

JDK (Java Development Kit) adalah sebuah perangkat peralatan yang digunakan untuk membangun perangkat lunak dengan menggunakan bahasa pemrograman Java. JDK berjalan diatas sebuah virtual machine yang dinamakan JVM (Java Virtual Machine). Dokumentasi JDK berisi spesifikasi API, deskripsi fitur, panduan pengembang, referensi halaman untuk perkakas JDK dan utilitas, demo, dan link ke informasi terkait. Dokumentasi ini juga tersedia dalam bundel yand dapat diunduh dan diinstal pada computer [4].

2.5.12.2 Eclipse IDE

Eclipse adalah sebuat komunitas proyek open source yang difokuskan pada membangun sebuah platform pengembangan yang dapat diperluas, runtime dan kerangka aplikasi untuk membangun, menyebarkan dan mengelola perangkat lunak di seluruh daur hidup perangkat lunak. Komunitas Eclipse memiliki lebih dari 60 proyek open source.

Proyek-proyek ini secara konsep terbagi menjadi 7 kategori :

1. Enterprise Development
2. Embedded and Device Development
3. Rich Client Platform
4. Rich Internet Applications
5. Application Frameworks
6. Application Lifecycle Management (ALM)
7. Service Oriented Architecture (SOA)

Komunitas Eclipse juga didukung oleh ekosistem besar dan dinamis dari penyedia solusi teknologi informasi utama, start-up inovatif, universitas dan lembaga penelitian, dan individu yang memperluas, mendukung, dan melengkapi platform Eclipse [4].

2.5.13 UML

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat

(*tool*) desain berorientasi objek dari IBM.UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem [5].

2.5.13.1 Sejarah UML

Sampai era tahun 1990 puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dan sebagainya. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan kelompok/perusahaan lain yang menggunakan metodologi yang berlainan [5].

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek.

Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* [5].

2.5.13.2 Diagram UML

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case Diagram*untuk memodelkan proses bisnis.
2. *Conceptual Diagram*untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. *Sequence Diagram*untuk memodelkan pengiriman pesan (*message*) antar *objects*
4. *Collaboration Diagram*untuk memodelkan interaksi antar*objects*.
5. *State Diagram*untuk memodelkan perilaku*objects*di dalam sistem.
6. *Activity Diagram*untuk memodelkan perilaku*Use Cases*dan*objects*di dalam *system.*
7. *Class Diagram*untuk memodelkan struktur kelas.
8. *Object Diagram*untuk memodelkan struktur*object*.
9. *Component Diagram*untuk memodelkan komponen*object*.
10. *Deployment Diagram*untuk memodelkan distribusi aplikasi.

Berikut akan dijelaskan empat macam diagram yang paling sering digunakan dalam pembangunan aplikasi berorientasi object, yaitu*use case diagram, sequence diagram, collaboration diagram, dan class diagram* [5].

1. *UseCase* Diagram

Usecase diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Usecase* diagram terdiri atas diagram untuk *usecase* dan *actor*. Actor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. *Use case* merepresentasikan operasi-operasi yang dilakukan oleh actor [5].

Use case digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *usecase* [5].

2. *SequenceDiagram*

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi [5].

3. *CollaborationDiagram*

Collaborationdiagram dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *sequencediagram* yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, *collaboration* diagram lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh objek [5].

4. *ClassDiagram*

Classdiagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam

sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan [5].