IMPLEMENTASI FRAMEWORK MVC BERBASIS ANDROID

(Studi Kasus Aplikasi Kamus Teknologi Informasi)

TUGAS AKHIR

Oleh :

NIM		Nama
3310901030	:	Efrizal Hardiman
3310901049	:	Raihana

Disusun untuk memenuhi syarat kelulusan Program Diploma III

PROGRAM STUDI TEKNIK INFORMATIKA

POLITEKNIK NEGERI BATAM

BATAM

2012

LEMBAR PENGESAHAN

Batam, 28 Februari 2012

Pembimbing,

Mir'atul Khusna Mufida, S.ST

NIK. 109057

LEMBAR PERNYATAAN

Dengan ini, saya:

NIM : 3310901030

Nama : Efrizal Hardiman

adalah mahasiswa Teknik Informatika Politeknik Batam yang menyatakan bahwa tugas akhir dengan judul:

IMPLEMENTASI FRAMEWORK MVC BERBASIS ANDROID

(Studi Kasus Aplikasi Kamus Teknologi Informasi)

disusun dengan:

1. tidak melakukan plagiat terhadap naskah karya orang lain
2. tidak melakukan pemalsuan data
3. tidak menggunakan karya orang lain tanpa menyebut sumber asli atau tanpa ijin pemilik

Jika kemudian terbukti terjadi pelanggaran terhadap pernyataan di atas, maka saya bersedia menerima sanksi apapun termasuk pencabutan gelar akademik.

Lembar pernyataan ini juga memberikan hak kepada Politeknik Batam untuk mempergunakan, mendistribusikan ataupun memproduksi ulang seluruh hasil Tugas Akhir ini.

Batam, 28 Februari 2012

Efrizal Hardiman
3310901030

LEMBAR PERNYATAAN

Dengan ini, saya:

NIM : 3310901049

Nama : Raihana

adalah mahasiswa Teknik Informatika Politeknik Batam yang menyatakan bahwa tugas akhir dengan judul:

IMPLEMENTASI FRAMEWORK MVC BERBASIS ANDROID

(Studi Kasus Aplikasi Kamus Teknologi Informasi)

disusun dengan:

1. tidak melakukan plagiat terhadap naskah karya orang lain
2. tidak melakukan pemalsuan data
3. tidak menggunakan karya orang lain tanpa menyebut sumber asli atau tanpa ijin pemilik

Jika kemudian terbukti terjadi pelanggaran terhadap pernyataan di atas, maka saya bersedia menerima sanksi apapun termasuk pencabutan gelar akademik.

Lembar pernyataan ini juga memberikan hak kepada Politeknik Batam untuk mempergunakan, mendistribusikan ataupun memproduksi ulang seluruh hasil Tugas Akhir ini.

Batam, 28 Februari 2012

Raihana
3310901049

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT, berkat rahmat dan hidayat-Nya, penyusun dapat menyelesaikan Tugas Akhir yang berjudul Implementasi Framework MVC Berbasis Android (Studi Kasus Aplikasi Kamus Teknologi Informasi). Aplikasi ini bertujuan untuk mengimplementasikan framework MVC pada platform Android.

Dalam kesempatan ini, penyusun mengucapkan terima kasih kepada :

1. Bapak Dr.Priyono Eko Sanyoto, selaku Direktur Politeknik Batam,
2. Bapak Uuf Brajawidagda, MT selaku koordinator Tugas Akhir,
3. Ibu Mir'atul Khusna Mufida, S.ST selaku pembimbing Tugas Akhir,
4. Bapak/Ibu Dosen program studi Teknik Informatika yang telah memberikan kritik dan saran yang sangat bermanfaat bagi penulis,
5. Kedua orang tua tercinta yang memberikan dukungan baik moril maupun materil,
6. Teman-teman seperjuangan Teknik Informatika angkatan 2009 yang memberikan semangat.

Dalam penulisan ini, penyusun mengakui bahwa masih terdapat kekurangan-kekurangan dalam penyusunannya. Oleh karena itu, penyusun sangat mengharapkan bantuan dari berbagai pihak berupa kritik ataupun saran guna penyempurnaan selanjutnya. Akhir kata penyusun ucapkan terima kasih, semoga laporan ini dapat bermanfaat bagi pembaca yang ingin mengembangkan sebuah aplikasi yang serupa.

Batam, 28 Februari 2012

Penulis

DAFTAR ISI

Daftar Gambar

Daftar Tabel

Bab I Pendahuluan

I.1 Latar Belakang

Berkembangnya teknologi sekarang ini dapat dilihat dengan munculnya berbagai perangkat *mobile* seperti *handphone* yang memiliki fitur canggih yang biasa disebut *smartphone*, disamping itu ada lagi jenis perangkat portable lain yang bernama tablet pc, dan *laptop*. Salah satu *Operating Sistem* (OS) yang berkembang pesat dan hampir menyamai iOS adalah android. Android sendiri adalah OS yang berbasiskan kernel linux sehingga menerapkan *open source* seperti linux pada umumnya. Keunggulan Android sekarang ini sudah tidak diragukan lagi dalam dunia teknologi. Dibandingkan dengan OS mobile lain android sudah semakin jelas, ia bahkan mampu menandingi Symbian, sebagai OS *mobile* yang mendominasi OS *mobile* dunia juga *Blackberry* yang mendominasi pasar Indonesia saat ini. Hal ini dibuktikan dengan laporan riset teknologi Gartner persentase penjualan *smartphone* sebagai berikut :

Tabel I. 1 Penjualan Smartphone Berdasarkan Sistem Operasi

Operating System	2011		2010	
	Unit	Market Share	Unit	Market Share
Android	46,775.9	43.4	10,652.7	17.2
Symbian	23,853.2	22.1	25,386.8	40.9
iOS	19,628.8	10.2	8,743.0	14.1
Research In Motion	12,652.3	11.7	11,628.8	18.7
Bada	2,055.8	1.9	557.0	0.9
Microsoft	1,722.8	1.6	3,058.8	4.9
Others	1,050,6	11.0	2,010.9	3.2
Total	**107,740.4**	**100.06**	**2,058.1**	**100.0**

Dalam perjalanan karir android para pengembang *software mobile* melihat android sebagai sebuah peluang bisnis. Sehingga banyak aplikasi yang dikembangkan di android. Namun dalam pengembangan aplikasi android, para *developer* cenderung melakukan *coding* ulang terhadap aplikasi yang memiliki struktur *coding* yang sama sehingga banyak membuang waktu. Maka dibutuhkan sebuah framework yang dapat memudahkan pengembangan aplikasi. Kemudahan menggunakan

framework MVC salah satunya adalah memudahkan pembuatan *coding* yang bersifat repetitif dan *source code* secara otomatis akan mengikuti struktur *coding* yang ada di framework tersebut sehingga memudahkan manajemen *source code*. Agar framework MVC ini dapat di uji kemudahannya maka dilakukan implementasi dengan studi kasus aplikasi kamus istilah teknologi informasi.

Saat ini banyak perangkat lunak menyediakan berbagai layanan yang memberikan kemudahan bagi pengguna komputer dan *handphone.* Berkembangnya Teknologi Informasi saat ini, juga diikuti dengan terciptanya istilah-istilah baru yang terus terasa banyak jumlahnya. Pencarian definisi dari istilah-istilah teknologi informasi dapat dilakukan dengan cara pencarian melalui kamus non-*digital*.

Tugas Akhir ini membuat suatu sistem aplikasi kamus istilah teknologi informasi atau pencarian arti dari suatu istilah yang bekerja sebagaimana sebuah kamus. Aplikasi kamus istilah teknologi informasi ini dibuat untuk mempermudah masyarakat dalam mengetahui istilah-istilah di bidang teknologi informasi dengan pengimplementasian framework MVC berbasis Android.

I.2 Rumusan Masalah

Rumusan masalah dari Tugas Akhir ini adalah :

1. Bagaimana menerapkan framework MVC pada platform Android?
2. Bagaimana menerapkan framework MVC pada aplikasi kamus istilah teknologi informasi?

I.3 Batasan Masalah

Batasan masalah Tugas Akhir ini, yaitu kamus yang digunakan adalah kamus istilah teknologi computer dan informasi (telematika) yang disusun oleh Andino Maseleno.

I.4 Tujuan

Tujuan dari Tugas Akhir ini adalah :

1. Dapat mengimplementasikan framework MVC pada platform Android.
2. Dapat mengimplementasikan framework MVC untuk membuat aplikasi kamus istilah teknologi informasi.

I.5 Sistematika Penulisan

Bab I Pendahuluan

Berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan dan sistematika penulisan.

Bab II Tinjauan Pustaka

Berisi tentang teori-teori yang berhubungan dengan penelitian.

Bab III Analisis dan Perancangan

Berisi tentang batasan sistem, definisi sistem kamus, use case, analisis kelas, interaction sequence diagram, class diagram, algoritma dan struktur MVC.

Bab IV Implementasi

Berisi tentang implementasi dan pengujian aplikasi.

Bab V Kesimpulan dan Saran

Berisi tentang kesimpulan dan saran.

Bab II Tinjauan Pustaka

Tinjauan pustaka tugas akhir ini membahas tentang Tinjauan Pustaka Metode Ontologi Taksonomi pada Istilah Teknologi Informasi, Framework, MVC, Struktur MVC, Android, Eclipse, Pengertian dan Jenis Kamus, dan Aplikasi Kamus.

II.1 Penerapan Metode Ontologi Taksonomi pada Istilah Teknologi Informasi

Kamus istilah teknologi informasi yang ada pada saat ini umumnya hanya menyajikan arti dari istilah yang dicari. Kamus tersebut dapat dikembangkan dengan cara menambahkan fitur berupa pencarian istilah yang berkaitan dengan istilah yang dicari. Penambahan fitur ini diharapkan membantu pengguna memahami arti suatu istilah dengan cara mengetahui istilah lain yang berkaitan dengan istilah tersebut.

Pengembangan fitur tambahan ini memerlukan suatu metode untuk membentuk struktur istilah yang saling berkaitan. Metode yang digunakan adalah metode ontologi taksonomi. Metode ontologi taksonomi adalah suatu cabang ilmu komputer yang mempelajari hubungan antar entitas dalam bentuk general spesifik. Proses pencarian istilah yang berkaitan menggunakan algoritma *Breadth First Search* yang merupakan algoritma pencarian secara melebar. Metode dan algoritma tersebut dapat diterapkan untuk membuat aplikasi pembangun ontologi taksonomi.

Perancangan aplikasi pembangun ontologi taksonomi untuk istilah teknologi infomasi ini yaitu dalam bidang *hardware*. Aplikasi ini menyediakan fungsi pencarian istilah yang berkaitan dengan suatu istilah *hardware* baru beserta artinya yang dimasukkan pengguna. Pengguna juga dapat melihat arti istilah yang

sudah ada beserta istilah yang berkaitan dengan istilah teknologi informasi yang ingin di cari oleh pengguna.

Dalam perancangan implementasi framework MVC berbasis android dengan studi kasus aplikasi kamus istilah teknologi informasi ini mengacu pada tugas akhir mahasiswi Politeknik Negeri Batam Margareth Managae (3310801074) dengan judul Penerapan Metode Ontologi Taksonomi pada Istilah Teknologi Informasi dan dalam pengimplementasian istilah-istilah teknologi informasi diambil dari kamus istilah teknologi computer dan informasi (telematika) yang disusun oleh Andino Maseleno.

II.2 Framework

Framework adalah sekumpulan perintah atau fungsi dasar yang dapat membantu dalam menyelesaikan proses-proses yang lebih kompleks. Secara umum, framework menggunakan struktur MVC (Model View Controller). Gambaran umumnya sebagai berikut :

Input > Processing > Output = Controller > Model >View

Keuntungan menggunakan framework, yaitu :

- Struktur yang konsisten. Sangat berguna bila *developer* banyak dan *turnover* tinggi.
- Struktur merupakan*best practices*. Semua sudah ditempatkan di tempat yang paling sesuai.
- Dapat belajar tentang desain aplikasi yang baik.

II.3 MVC

Model View Controller atau MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (model) dari tampilan (view) dan cara bagaimana memprosesnya (controller). Dalam implementasinya kebanyakan framework dalam aplikasi website adalah berbasis arsitektur MVC.

MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna, dan bagian yang menjadi kontrol dalam sebuah aplikasi web.

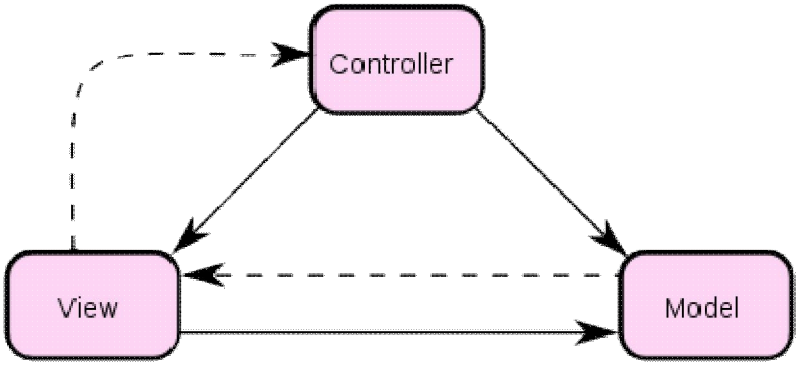
Dengan menggunakan metode MVC maka aplikasi akan lebih mudah untuk dirawat dan dikembangkan. Adapun bagian dari MVC, yaitu :

1. Model

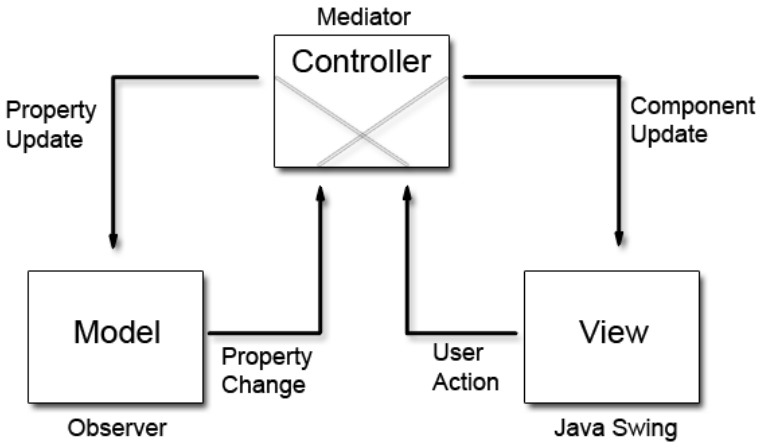
   Model mewakili struktur data. Biasanya berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaharuan data dan lain-lain. Untuk dapat melakukan koneksi ke database bisa menggunakan tambahan driver seperti hibernate, persistence, JPA, ADO, mysql, Ms. *Access*, oracle.
2. View

   View adalah bagian yang mengatur tampilan ke *user*. Biasa dikatakan berupa halaman *web* atau tampilan GUI desktop. Untuk desain tampilan bisa menggunakan Swing, AWT dan render kode HTML yang menampilkan halaman *web*.
3. Controller

   Controller merupakan bagian yang menjembatani model dan view. Controller berisi perintah-perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman web atau ke GUI desktop. Dalam hal ini fungsi yang biasa dipakai adalah struts, struts2, spring dan bisa juga menggunakan fungsi insert, update, delete.

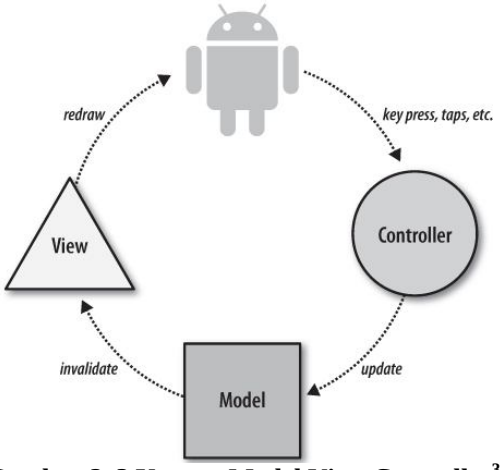


Gambar 2. 1 Konsep ModelViewController[1]

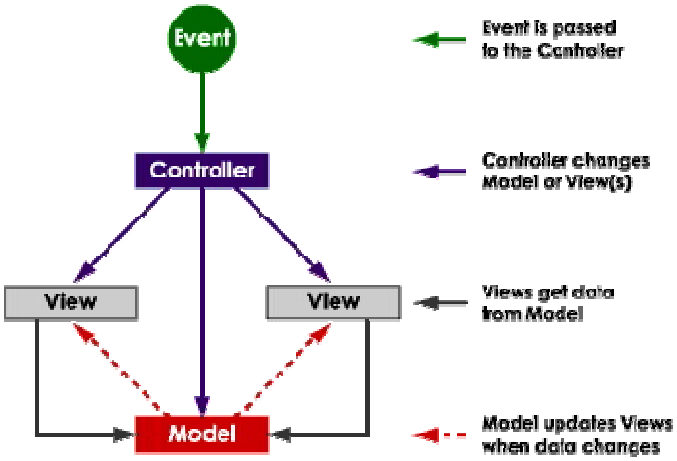


Gambar 2. 2 Konsep Model View Controller[2]

[1]http://www.ask.com/wiki/Model-view-controller
[2]http://www.codeproject.com/KB/tips/ModelViewController.aspx



Gambar 2. 3 Konsep Model View Controller[3]



Gambar 2. 4 Konsep Model View Controller[4]

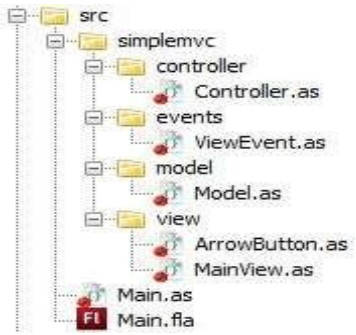
II.4 Struktur MVC

Pembuatan framework pada umumnya menggunakan struktur MVC. Model View Controller (MVC) adalah *meta-pattern*, artinya MVC bukan merupakan pattern yang berdiri sendiri tapi merupakan sekumpulan dari beberapa pattern. Pada prinsipnya, implementasi MVC membagi kode ke dalam tiga bagian, yaitu:

[3]http://androidapps.org.ua/i_sect13_d1e11121.html
[4]http://www.enode.com/x/markup/tutorial/mvc.html

1. Model sebagai sumber data
2. View sebagai representasi data dan *user interface*
3. Controller yang berfungsi sebagai otak atau *business logic* yang memproses user input dan meng-*update* Model dan View (jika diperlukan)

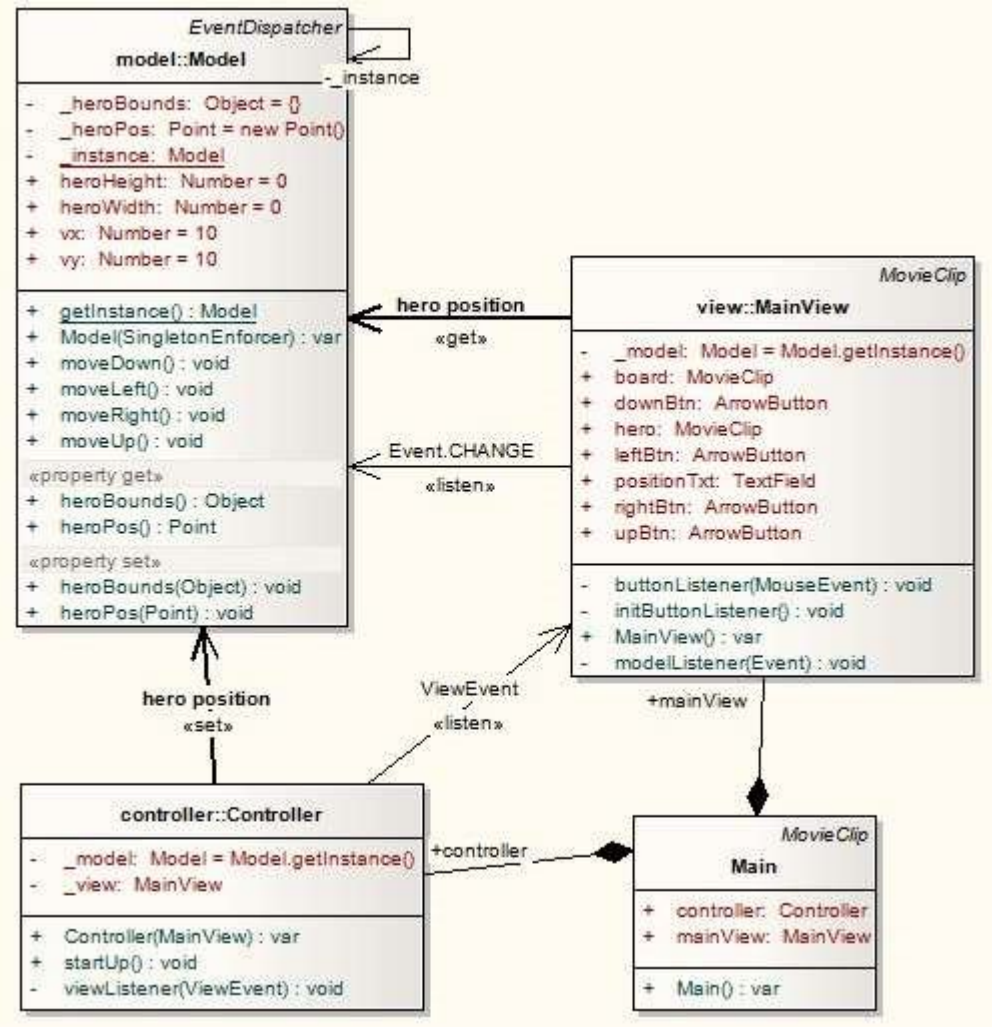
Keuntungan MVC yaitu *source code* lebih *maintainable* karena pengembang bisa mengubah salah satu bagian tanpa harus mengubah bagian yang lain. MVC juga mempermudah *debugging* karena pengembang bisa memperkirakan bagian mana yang bermasalah tanpa harus membongkar seluruh kode yang sudah dibuat. Jadi secara umum, keuntungan MVC jauh lebih besar daripada kerepotan yang ditimbulkannya. Pada gambar 2.5 merupakan Struktur dasar MVC pada direktori di *Flash Develop.*



Gambar 2. 5 Direktori Project di Flash Develop[5]

Pada Gambar 2.6 merupakan contoh struktur MVC sederhana untuk penulisan kode yang *maintainable* dengan memecah kode menjadi beberapa bagian berdasarkan fungsinya (*seperation of concerns*).

[5]http://masputih.com/2009/05/mvc-sederhana-untuk-pemula



Gambar 2. 6 Contoh Class Diagram[6]

II.4.1 View dan ViewEvent

Package view berisi class yang berhubungan dengan library simbol yaitu MainView dan ArrowButton. MainView bertugas menyiarkan ViewEvent jika salah satu tombol navigasi diklik. *Event* ini ditangkap oleh Controller yang kemudian meng-*update* Model. Selain itu MainView juga menampilkan posisi kotak kuning.

[6]http://masputih.com/2009/05/mvc-sederhana-untuk-pemula

Untuk meminimalkan *coupling* antara MainView dan Controller maka dibuatlah ViewEvent tanpa harus membuat controller langsung. Berikut *source code* View dan ViewEvent.

```
[js]
package simplemvc.view {

import flash.display.MovieClip;
import flash.events.Event;
import flash.events.MouseEvent;
import flash.geom.Point;
import flash.text.TextField;
import simplemvc.events.ViewEvent;
import simplemvc.model.Model;

/**
* …
* @author Anggie Bratadinata
*/
public class MainView extends MovieClip {

//timeline objects
public var hero:MovieClip;
public var board:MovieClip;
public var leftBtn:ArrowButton;
public var rightBtn:ArrowButton;
public var upBtn:ArrowButton;
public var downBtn:ArrowButton;
public var positionTxt:TextField;

private var _model:Model = Model.getInstance();

//———————————————— INIT

public function MainView() {

initButtonListener();

_model.addEventListener(Event.CHANGE, modelListener);

}

private function initButtonListener():void {
var i:int = 0;
while (i < numChildren) {
if (this.getChildAt(i) is ArrowButton) {
this.getChildAt(i).addEventListener(MouseEvent.CLICK, buttonLis-
tener);
}
i++;
}
```

```
}

private function buttonListener(e:MouseEvent):void {

switch(e.currentTarget) {
case leftBtn:
dispatchEvent(new ViewEvent(ViewEvent.MOVE_LEFT));
break;
case rightBtn:
dispatchEvent(new ViewEvent(ViewEvent.MOVE_RIGHT));
break;
case upBtn:
dispatchEvent(new ViewEvent(ViewEvent.MOVE_UP));
break;
case downBtn:
dispatchEvent(new ViewEvent(ViewEvent.MOVE_DOWN));
break;
}
}

private function modelListener(e:Event):void {

hero.x = _model.heroPos.x;
hero.y = _model.heroPos.y;

positionTxt.text = _model.heroPos.toString();

}

}

}

[/js]

[js]
package simplemvc.events {
import flash.events.Event;

/**
* …
* @author Anggie Bratadinata
*/
public class ViewEvent extends Event {

public static const MOVE_LEFT:String = “moveLeft”;
public static const MOVE_RIGHT:String = “moveRight”;
public static const MOVE_UP:String = “moveUp”;
public static const MOVE_DOWN:String = “moveDown”;

public function ViewEvent(type:String) {
super(type, true);
```

```
}

override public function clone():Event {
return new ViewEvent(type);
}

override public function toString():String {
return formatToString("ViewEvent", "type", "bubbles", "can-
celable", "eventPhase");
}

}

}

[/js]
```

II.4.2 Model

Model adalah bagian dimana data berada. Untuk data yang bersifat global, pengembang bisa mengimplementasikan *Singleton Pattern*. Singleton Class hanya bisa diinstansiasi satu kali selama aplikasi berjalan. Umumnya, referensi ke instance dari singleton diakses dengan memanggil static method getInstance().

Model yang pengembang gunakan untuk penyimpanan data berupa posisi kotak kuning. Pada saat data berubah, model akan mend-dispatch Event.change yang didengarkan oleh MainView yang kemudian meng-*update* posisi kotak kuning. Berikut merupakan *source code* dari Model.

```
[js]
package simplemvc.model {

import flash.events.Event;
import flash.events.EventDispatcher;
import flash.geom.Point;

/**
* …
* @author Anggie Bratadinata
*/
public class Model extends EventDispatcher {

public var heroHeight:Number = 0;
public var heroWidth:Number = 0;
```

```
//————————————————— HERO VELOCITY

public var vx:Number = 10;
public var vy:Number = 10;

//————————————————— HERO POSITION

private var _heroPos:Point = new Point();

public function get heroPos():Point { return _heroPos; }

public function set heroPos(value:Point):void {

if (value.x >= heroBounds.xMin && value.x <= heroBounds.xMax &&
value.y >= heroBounds.yMin && value.y <= heroBounds.yMax ) {

_heroPos = value;
dispatchEvent(new Event(Event.CHANGE));

}
}

public function moveDown():void {
heroPos = new Point(heroPos.x, heroPos.y + vy);
}

public function moveUp():void {
heroPos = new Point(heroPos.x, heroPos.y – vy);
}

public function moveLeft():void {
heroPos = new Point(heroPos.x – vx, heroPos.y);
}

public function moveRight():void {
heroPos = new Point(heroPos.x + vx, heroPos.y );
}

//————————————————— HERO BOUNDS

private var _heroBounds:Object = {};

public function get heroBounds():Object { return _heroBounds; }

public function set heroBounds(value:Object):void {
//trace(value);
_heroBounds.xMin = value.x;
_heroBounds.xMax = value.x + value.width – heroWidth;
_heroBounds.yMin = value.y;
_heroBounds.yMax = value.y + value.height – heroHeight;
```

```
}

//———————————————— INIT
private static var _instance:Model;
public function Model(enf:SingletonEnforcer) {}
public static function getInstance():Model {
if (_instance == null) _instance = new Model(new SingletonEnfor-
cer());
return _instance;
}

}

}
//INNER CLASS
class SingletonEnforcer { };

[/js]
```

II.4.3 Controller

Controller mendengarkan ViewEvent yang disiarkan oleh MainView dan meng-update model berdasarkan even tersebut. Berikut merupakan *source code* controller.

```
[js]
package simplemvc.controller {
import flash.events.Event;
import flash.geom.Point;
import simplemvc.model.Model;
import simplemvc.view.*;
import simplemvc.events.*;

/**
* …
* @author Anggie Bratadinata
*/
public class Controller {

private var _model:Model = Model.getInstance();
private var _view:MainView;

//———————————————— INIT

public function Controller(view:MainView) {

_view = view;
```

```
_model.heroHeight = _view.hero.height;
_model.heroWidth = _view.hero.width;
_model.heroBounds = _view.board.getBounds(_view);

_view.addEventListener(ViewEvent.MOVE_DOWN, viewListener);
_view.addEventListener(ViewEvent.MOVE_UP, viewListener);
_view.addEventListener(ViewEvent.MOVE_LEFT, viewListener);
_view.addEventListener(ViewEvent.MOVE_RIGHT, viewListener);

}

public function startUp():void {
_model.heroPos = new Point(0, 0);
}

private function viewListener(e:ViewEvent):void {

switch(e.type) {
case ViewEvent.MOVE_DOWN:
_model.moveDown();
break;
case ViewEvent.MOVE_LEFT:
_model.moveLeft();
break;
case ViewEvent.MOVE_RIGHT:
_model.moveRight();
break;
case ViewEvent.MOVE_UP:
_model.moveUp();
break;
}
}

}

}

[/js]
```

Berikut merupakan source code kelas utamanya.

```
[js]
package {

import flash.display.MovieClip;
import simplemvc.controller.Controller;
import simplemvc.view.MainView;

/**
* …
* @author Anggie Bratadinata
*/
```

```
public class Main extends MovieClip {

public var mainView:MainView;
public var controller:Controller;

public function Main() {
controller = new Controller(mainView);
controller.startUp();
}

}

}

[/js]
```

II.5 Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Dilain pihak, Google merilis kode-kode Android dibawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. Di dunia ini terdapat dua jenis distributor sistem operasi Andorid. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Service* (GSM) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD).

Regenerasi Android, yaitu :

- **Produk awal (2007-2008)**

  Sekitar september 2007 sebuah studi melaporkan bahwa Google mengajukan hak paten aplikasi telepon seluler. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android ARM Holdings, Atheros, Communications, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan

Vodafone Group Plc. Seiring pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka, Android, perangkat bergerak (*mobile*) yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru. Telepon pertama yang memakai sistem operasi Android adalah HTC Dream, yang dirilis pada 22 Oktober 2008. Pada penghujung tahun 2009 diperkirakan di dunia ini paling sedikit terdapat 18 jenis telepon seluler yang menggunakan Android.

- **Android versi 1.1**

  Pada 9 Maret 2009, Google merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

- **Androidversi 1.5 (Cupcake)**

  Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (*Software Development Kit*) dengan versi 1.5 (Cupcake). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke *Youtube* dan gambar ke *Picasa* langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung secara otomatis ke headset Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.

- **Android versi 1.6 (Donut)**

  Donut (versi 1.6) dirilispada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, camcorder dan galeri yang dintegrasikan; CDMA / EVDO, 802.1x, VPN, Gestures, dan Text-to-speech engine; kemampuan dial kontak; teknologi

text to change speech (tidak tersedia pada semua ponsel; pengadaan resolusi VWGA).

- **Android versi 2.0/2.1 (Eclair)**

  Pada 3 Desember 2009 kembali di luncurkan ponsel Android dengan versi 2.0/2.1 (Eclair), perubahan yang di lakukan adalah pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 MP, digital Zoom, dan Bluetooth 2.1.

- **Android versi 2.2 (Froyo : Frozen Yoghurt)**

  Pada 20 Mei 2010, Android versi 2.2 (Froyo) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebihcepat, intergrasi V8 JavaScript engine yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi Hotspot portabel, dan kemampuan *auto update* dalam aplikasi Android Market.

- **Android versi 2.3 (Gingerbread)**

  Pada 6 Desember 2010, Android versi 2.3 (Gingerbread) diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi copy paste, layar antarmuka (*User Interface*) didesain ulang, dukungan format video VP8 dan*Web*M, efek audio baru (*reverb, equalization, headphone virtualization*, dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC), dan dukungan jumlah kamera yang lebih dari satu.

- **Android versi 3.0/3.1 (Honeycomb)**

  Android Honeycomb dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. User Interface pada Honeycomb juga berbeda karena sudah didesain untuk tablet. Honeycomb juga mendukung multi prosesor dan juga akselerasi perangkat keras (*hardware*) untuk grafis. Tablet pertama yang dibuat dengan menjalankan

Honeycomb adalah *Motorola Xoom*. Perangkat tablet dengan *platform* Android 3.0 akan segera hadir di Indonesia. Perangkat tersebut bernama *Eee Pad Transformer* produksi dari Asus. Rencana masuk pasar Indonesia pada Mei 2011.

- **Android versi 4.0 (Ice Cream)**
  Android versi 4.0 akan dirilis akhir tahun 2011.

Fitur-fitur yang tersedia di Android adalah :

- Kerangka aplikasi : Memungkinkan penggunaan dan penghapusan komponen yang tersedia
- Dalvik mesin virtual : Mesin virtual dioptimalkan untuk perangkat mobile
- Grafik : Grafik di 2D dan gratis 3D berdasarkan pustaka OpenGL
- SQLite : Aplikasi penyimpanan data
- Mendukung media : Audio, video, dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM, Bluetooth, EDGE, 3G, dan WiFi (Hardware department)
- Kamera, Global Positioning Sistem (GPS), Kompas, dan Accelerometer (tergantung hardware)

Adapun android yang digunakan dalam tugas akhir ini adalah android versi SDK 11. Dan platform android yang digunakanadalah platform 2.3 (Gingerbread).

II.6 Eclipse

Eclipse adalah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan disemua platform (*platform-independent*). Berikut ini adalah sifat dari eclipse :

- **Multi-paltform**

  Target sistem operasi *eclipse* adalah *Microsoft Windows, Linux, Solaris, AIX, HP-UX* dan *Mac OS X*.
- **Multi-language**

  Eclipse dikembangkan dengan bahasa pemograman Java, akan tetapi eclipse mendukung pengembangan aplikasi berbasis bahasa pemograman lainnya, seperti C/C++, Cobol, Phyton, Perl, PHP, dan lain sebagainya.
- **Multi-role**

  Selain sebagai IDE untuk pengembangan aplikasi, eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembagan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembagan web dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemograman perangkat lunak ini. Selain itu, kelebihan dari eclipse yang membuatnya populer adalah kemampuan untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

Sejak versi 3.0, eclipse pada dasarnya merupakan sebuah kernel, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut komponen yang membentuk RCP, yaitu *Core platform*, OSGi, SWT (*Standard Widget Toolkit*), Jface, Eclipse *Workbench*.

Secara standar eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemograman Java.

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran eclipse secara rutin dan simultan yang dikenal dengan nama *Simultaneous Release*. Setiap versi peluncuran terdiri dari eclipse platform dan juga sejumlah proyek yang terlibat dalam proyek eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi eclipse dengan fitur-fitur dan versi terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah *deployment* dan *maintenance* untuk sistem *enterprise*, serta untuk kenyamanan. Peluncuran simultan dijadwalkan pada bulan juni setiap tahunnya. Adapun versi eclipse yang digunakan adalahversi 3.7 indigo.

II.7 Pengertian dan Jenis Kamus

Dalam Kamus Besar Bahasa Indonesia, kamus diartikan sebagai : "Buku acuan yang memuat kata dan ungkapan, biasanya disusun menurut abjad beserta penjelasan tentang makna dan pemakaiannya". Kamus biasanya dibagi berdasarkan penggunaan bahasa, ukuran dan kamus istimewa. Kamus berdasarkan penggunaan bahasa dapat dibedakan menjadi :

- **Kamus Ekabahasa**

  Kamus ini hanya menggunakan satu bahasa. Kata yang dijelaskan dan penjelasannya terdiri daripada bahasa yang sama. Contoh bagi kamus eka bahasa ialah Kamus Besar Bahasa Indonesia.

- **Kamus Dwibahasa**

  Kamus ini menggunakan dua bahasa, yakni kata daripada bahasa yang dikamuskan diberi padanan dengan menggunakan bahasa yang lain. Contohnya, Kamus Inggris-Indonesia dan Kamus Indonesia-Jepang.

- **Kamus Aneka Bahasa**

  Kamus ini sekurang-kurangnya menggunakan tiga bahasa atau lebih. Misalnya, kata Bahasa Melayu Bahasa Inggris dan Bahasa Mandarin secara serentak. Contoh bagi kamus aneka bahasa ialah Kamus Melayu-Cina-Inggris Pelangi susunan Yuen Boon Chan pada tahun 2004.

Kamus juga dibedakan berdasarkan ukurannya. Ukuran yang dimaksud adalah ketebalan dan luas permukaan kamus tersebut. Oleh karena itu kamus dapat dibedakan berdasarkan ukurannya sebagai berikut :

- **Kamus Mini**

  Dikenal sebagai kamus saku karena memiliki ukuran yang pas dimasukkan ke dalam saku. Tebalnya kurang daripada 2 cm.

- **Kamus Kecil**

  Merupakan kamus berukuran standar yang sering dijumpai yang biasa digunakan untuk proses belajar seperti Kamus Inggris-Indonesia oleh John M. Echols dan Hassan Shadily.

- **Kamus Besar**

  Kamus ini memuatkan penjelasan yang lengkap terhadap suatu kata. Sehingga biasanya berukuran besar dan tebal. Contohnya Kamus Besar Bahasa Indonesia.

Kamus istimewa dikatakan istimewa karena memiliki fungsi khusus (istimewa). Kamus istimewa memiliki beberapa contoh sebagai berikut :

- **Kamus Istilah**

  Kamus ini berisi istilah-istilah khusus dalam bidang tertentu yang berfungsinya untuk kegunaan ilmiah. Contohnya ialah Kamus Kedokteran, teknologi informasi, dan lain sebagainya.

- **Kamus Etimologi**

  Kamus yang menerangkan asal usul sesuatu perkataan dan maksud asalnya.

- **Kamus Tesaurus**

  Kamus yang menerangkan maksud sesuatu perkataan dengan memberikan kata-kata searti (sinonim) dan dapat juga kata-kata yang berlawanan arti (antonim). Kamus ini membantu para penulis untuk meragamkan penggunaan diksi. Contohnya, Tesaurus Bahasa Indonesia.

- **Kamus Peribahasa/Simpulan Bahasa**

  Kamus yang menerangkan maksud sesuatu peribahasa/simpulan bahasa. Selain digunakan sebagai rujukan, kamus ini juga sesuai dibaca dengan tujuan keindahan.
- **Kamus Kata Nama Khas**

  Kamus ini hanya menyimpan kata nama khas seperti nama tempat, nama tokoh, dan juga nama bagi institusi. Tujuannya adalah untuk menyediakan rujukan bagi nama-nama ini.
- **Kamus Terjemahan**

  Kamus yang menyedia kata yang searti dengan bahasa asing untuk satu bahasa sasaran. Kegunaannya adalah untuk membantu para penerjemah.
- **Kamus Kolokasi**

  Kamus yang menerangkan tentang padanan kata, contohnya kata 'terdiri' yang selalu berpadanan dengan 'dari' atau 'atas'.

Dalam tugas akhir ini mengacu pada kamus istilah dalam bidang teknologi informasi.

II.8 Aplikasi Kamus

Pengertian dari aplikasi adalah penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan. Dapat juga dinyatakan sebagai program komputer yang dibuat untuk membantu manusia dalam melaksanakan tugas tertentu. Aplikasi dapat dibagi menjadi dua bagian yaitu :

- **Aplikasi Spesialis**

  Yaitu program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
- **Aplikasi Paket**

  Suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu.

Dapat disimpulkan bahwa Aplikasi Kamus adalah program komputer dengan konsep kamus yang dirancang untuk menjalankan tugas menjelaskan tentang makna dan pemakaian kata atau istilah. Dalam pembuatan aplikasi ini mengacu pada Aplikasi Paket.

Bab III Analisis dan Perancangan

Pada bab analisis dan perancangan ini akan menjelaskan tentang Batasan Sistem, Definisi Sistem Kamus, Use Case, Analisis Kelas, Interaction Sequence Diagram, Class Diagram, Algoritma dan Struktur MVC.

III.1 Batasan Sistem

Dalam pembuatan tugas akhir ini memiliki batasan sistem. Adapun spesifikasi pada sistem ini, yaitu :

Tabel III. 1 Spesifikasi Batasan Sistem pada PC

Deskripsi	:	Spesifikasi
Sistem Operasi	:	Linux
Platform	:	Android 2.2 (Froyo)
IDE	:	Eclipse for Java Development
Plugin	:	ADT 12.0.0
Emulator	:	Android SDK 13

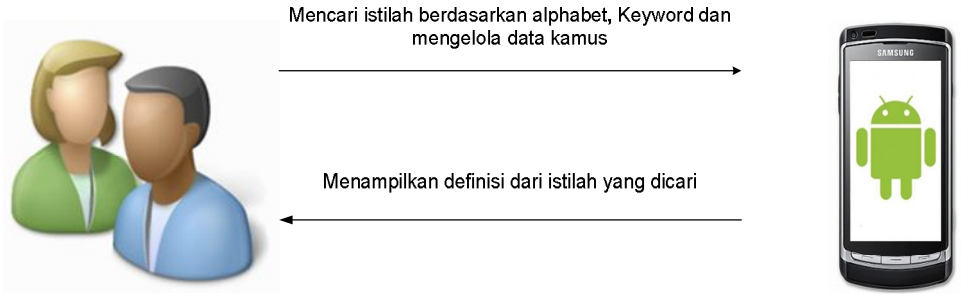
Tabel III. 2 Spesifikasi Batasan Sistem pada Mobile

Deskripsi	:	Spesifikasi
Sistem Operasi	:	Android
Version	:	Android 2.2 (Froyo)

III.2 Definisi Sistem Kamus

Tugas akhir ini dirancang dan dibuat dengan tujuan untuk memudahkan menterjemahkan istilah teknologi informasi yang ingin diketahui. Dimana dalam membuatnya menggunakan framework MVC dan dibuat dilingkungan *mobile* berbasis android.

III.2.1 Gambaran Kerja Sistem Aplikasi Kamus



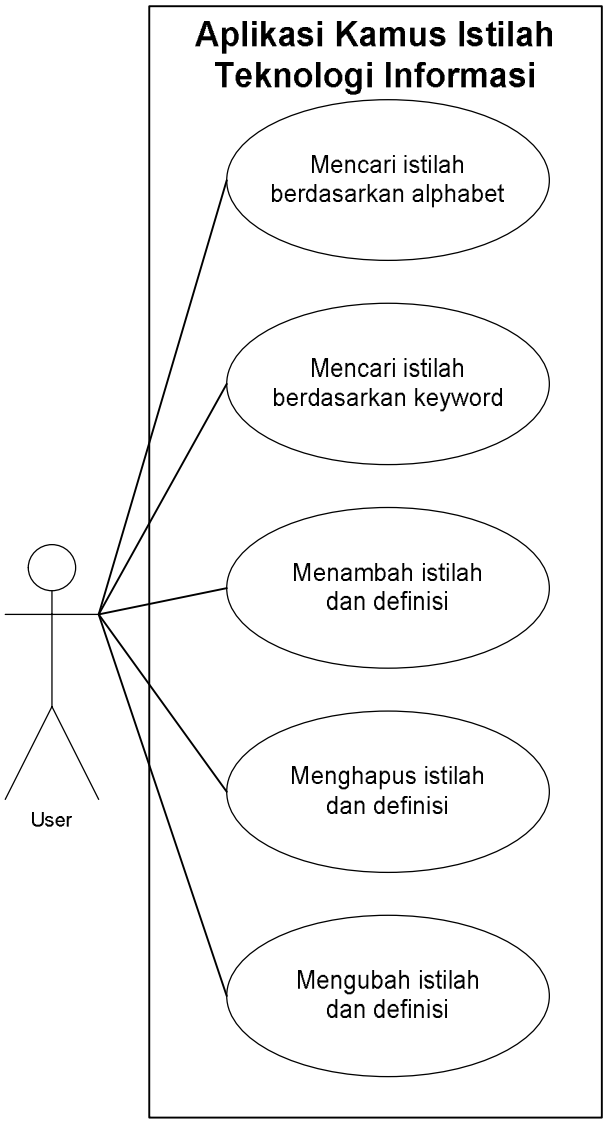
Gambar 3. 1 Deskripsi Sistem Aplikasi Kamus

Deskripsi Sistem Aplikasi Kamus pada Gambar 3.1 menjelaskan tentang proses kerja aplikasi kamus berbasis mobile. Berikut merupakan spesifikasi sistem aplikasi kamus :

1. *User* mencari istilah berdasarkan alphabet dan keyword dengan cara memasukkan kata atau huruf yang ingin dicari
2. *User* menambah istilah dan definisi baru pada kamus
3. *User* mengubah istilah dan definisi yang kurang valid pada kamus
4. *User* menghapus istilah dan definisi yang tidak valid pada kamus
5. Sistem menampilkan istilah dan definisi yang dicari ke *User.*

III.3 Use Case Diagram Aplikasi Kamus

Use Case Diagram aplikasi kamus ini digunakan untuk menggambarkan hubungan sejumlah external aktor dengan use case yang terdapat dalam sistem aplikasi kamus. Use Case Diagram ini hanya menggambarkan keadaan lingkungan sistem yang dapat dilihat dari luar oleh aktor.



Gambar 3. 2 Use Case Diagram Aplikasi Kamus

III.3.1 Skenario Use Case

Dari gambar 3.2 dapat dijelaskan bahwa *use r*mencari istilah kata teknologi informasi dengan tiga cara yakni user dapat mencari istilah berdasarkan alphabet, keyword kemudian *user* dapat mengelola data kamus.

III.3.1.1 Use Case Mencari Istilah Berdasarkan Alphabet

Kondisi awal	:	Istilah ditampilkan sesuai abjad dari A-Z
Skenario	:	*User* mengetik abjad dan kemudian sistem menampilkan berdasarkan abjad yang dipilih
Kondisi akhir	:	Istilah ditampilkan sesuai abjad yang dipilih

III.3.1.2 Use Case Mencari Istilah Berdasarkan Keyword

Kondisi awal	:	Menampilkan menu pencarian istilah kata
Skenario	:	*User* mengetikkan kata yang akan dicari kemudian sistem akan menampilkan istilah berdasarkan kata yang dimasukkan.
Kondisi akhir	:	Istilah ditampilkan sesuai dengan kata yang dimaksud

III.3.1.3 Use Case Menambahkan Istilah dan Definisi

Kondisi awal	:	Istilah dan Definisi baru belum ada
Skenario	:	*User* menambahkan istilah dan definisi baru dan kemudian sistem menyimpan istilah dan definisi baru
Kondisi akhir	:	Istilah dan Definisi baru sudah ditambahkan

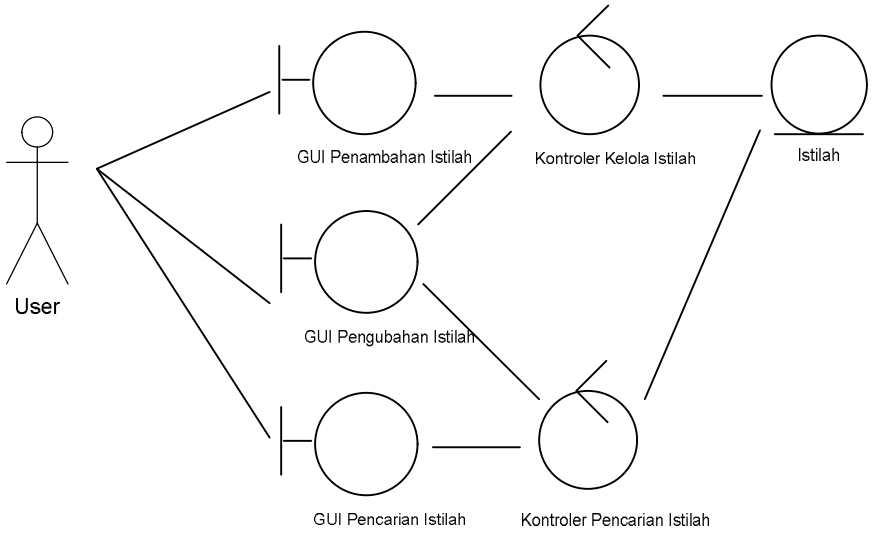
III.3.1.4 Use Case Menghapus Istilah dan Definisi

Kondisi awal	:	Istilah dan Definisi akan dihapus
Skenario	:	*User* menghapus istilah dan definisi yang tidak perlu dan kemudian sistem menghapus istilah dan definisi tersebut
Kondisi akhir	:	Istilah dan Definisi sudah dihapus

III.3.1.5 Use Case Mengubah Istilah dan Definisi

Kondisi awal	:	Istilah dan Definisi akan diubah
Skenario	:	*User* mengubah istilah dan definisi yang tidak valid dan kemudian sistem mengubah istilah dan definisi tersebut
Kondisi akhir	:	Istilah dan Definisi sudah diubah

III.4 Analisis Kelas



Gambar 3. 3 Analisis Kelas Aplikasi Kamus

Berdasarkan gambar 3.3 terlihat bahwa diagram dibagi menjadi tiga kelas, yaitu kelas *boundary*, kelas *control* dankelas *entity*. Rincian kelas-kelas tersebut akan dijelaskan dalam Tabel III.3.

Tabel III. 3Deskripsi Analisis Kelas Aplikasi Kamus

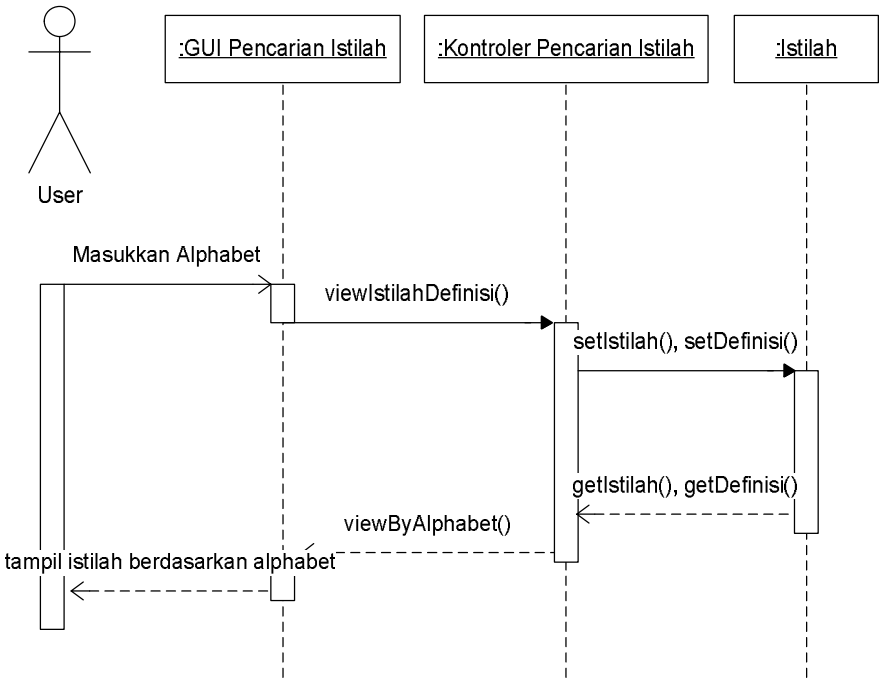
Jenis Kelas	Nama Kelas	Deskripsi	Penanganan Use Case
Kelas Boundary	GUI Penambahan Istilah	Kelas yang berperan sebagai antarmuka untuk menambahkan istilah dan definisi baru pada kamus	Use Case menambah istilah dan definisi
	GUI Pengubahan Istilah	Kelas yang berperan sebagai antarmuka untuk mengubah istilah dan definisi yang ada pada kamus	Use Case mengubah istilah dan definisi
	GUI Pencarian Istilah	Kelas yang berperan sebagai antarmuka untuk mencari istilah pada kamus	Use Case mencari istilah berdasarkan alphabet dan keyword
Kelas Control	Kontroler Kelola Istilah	Kelas yang berperan sebagai penghubung antara GUI	Use Case menambah, mengubah dan

		Penambahan istilah dan GUI Pengubah istilah ke kelas entity istilah dan juga menyimpan fungsi untuk menambah, mengubah dan menghapus istilah	menghapus istilah dan definisi
	Kontroler Pencarian Istilah	Kelas yang berperan sebagai penghubung antara GUI pencarian istilah dengan kelas entity istilah dan menyimpan fungsi untuk mencari istilah berdasarkan alphabet dan keyword	Use Case mencari istilah berdasarkan alphabet dan keyword
Kelas Entity	Istilah	Kelas yang berperan untuk menyimpan istilah-istilah yang akan dipakai oleh kamus	Use Case mencari istilah berdasarkan alphabet dan keyword serta mengelola data pada kamus (insert, update, delete)

III.4.1 Interaction Sequence Diagram

Diagram ini menggambarkan urutan proses yang akan terjadi dalam sistem ini. Diagram ini juga menggambarkan method yang dijalankan oleh masing-masing kelas setiap proses yang terjadi pada sistem.

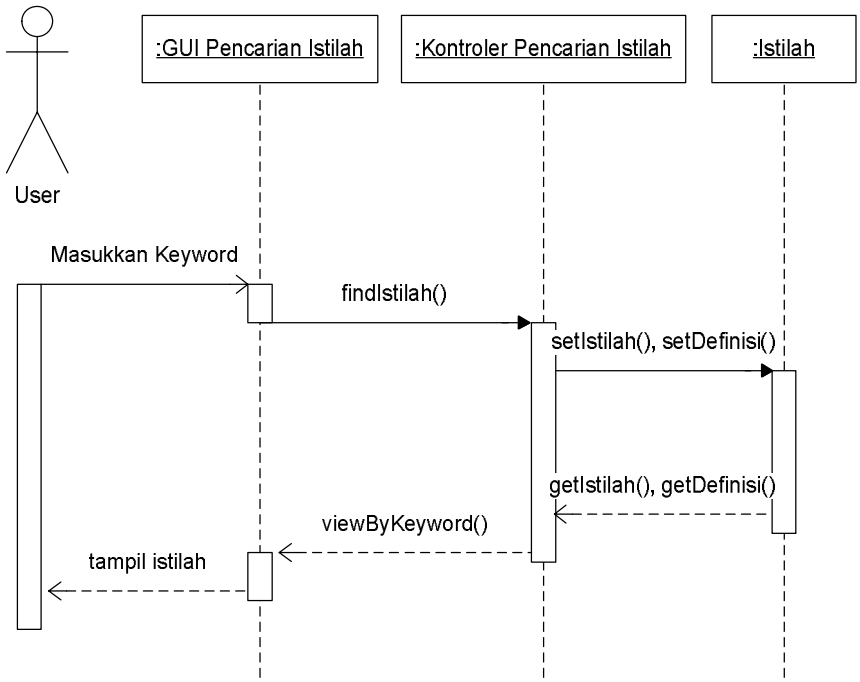
III.4.1.1 Mencari Istilah Berdasarkan Alphabet



Gambar 3. 4 Sequence Diagram Mencari Istilah Berdasarkan Alphabet

Pada gambar 3.4 *user* akan memasukkan alphabet untuk mencari istilah kemudian sistem akan menampilkan berdasarkan alphabet yang dimasukkan oleh *user*.

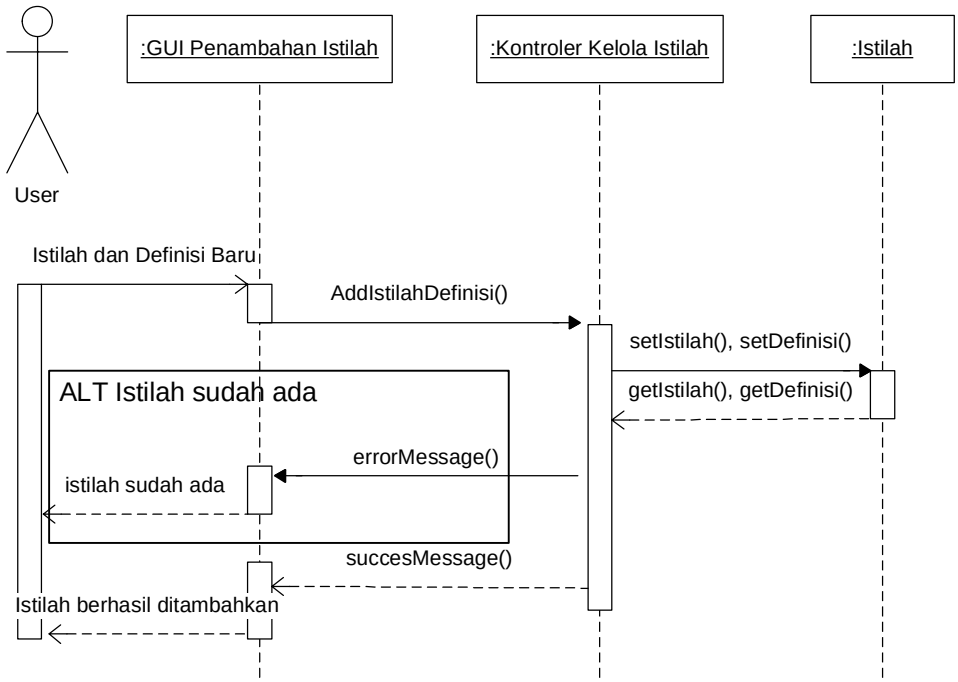
III.4.1.2 Mencari Istilah Berdasarkan Keyword



Gambar 3. 5 Sequence Diagram Mencari Istilah Berdasarkan Keyword

Pada gambar 3.5 user memasukkan huruf untuk mencari istilah kemudian sistem akan mencari berdasarkan keyword yang dimaksud kemudian melakukan penyamaan keyword yang kemudian ditampilkan ke *user*.

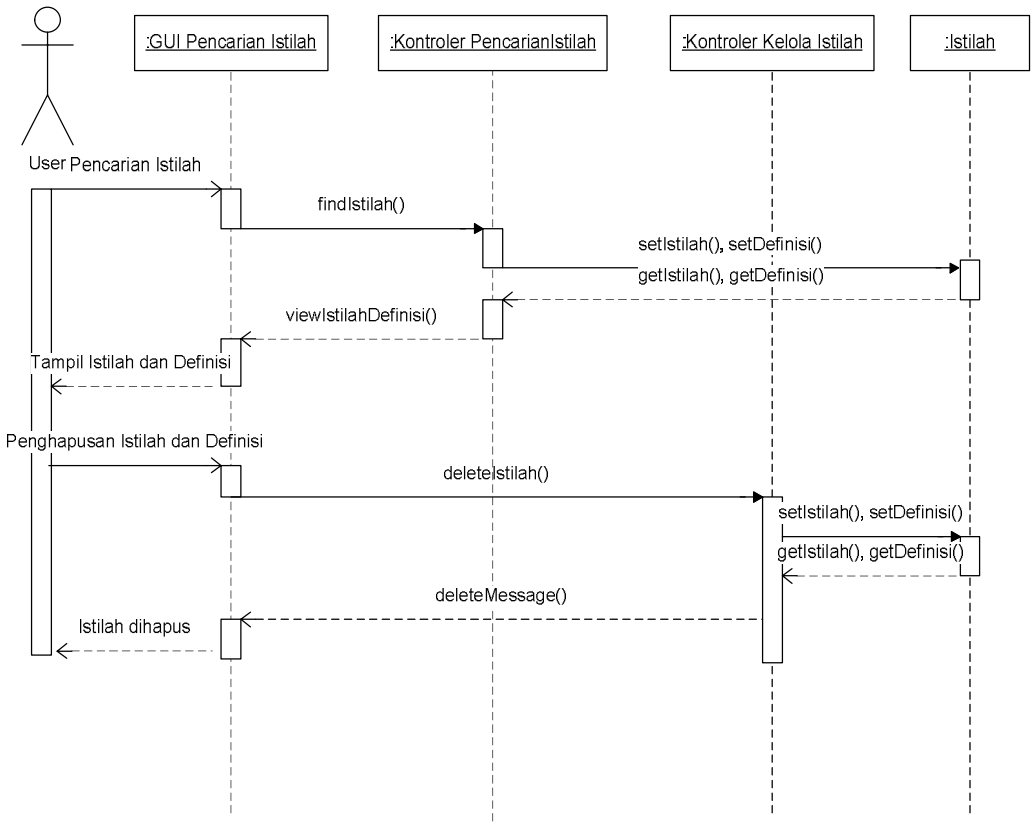
III.4.1.3 Menambahkan Istilah dan Definisi



Gambar 3. 6 Sequence Diagram Menambah Istilah dan Definisi

Pada gambar 3.6 *user* akan memasukkan istilah dan definisi baru ke dalam sistem kemudian sistem akan melakukan pemeriksaan apakah ada istilah yang sama atau tidak, jika ada istilah yang sama maka proses akan berhenti.

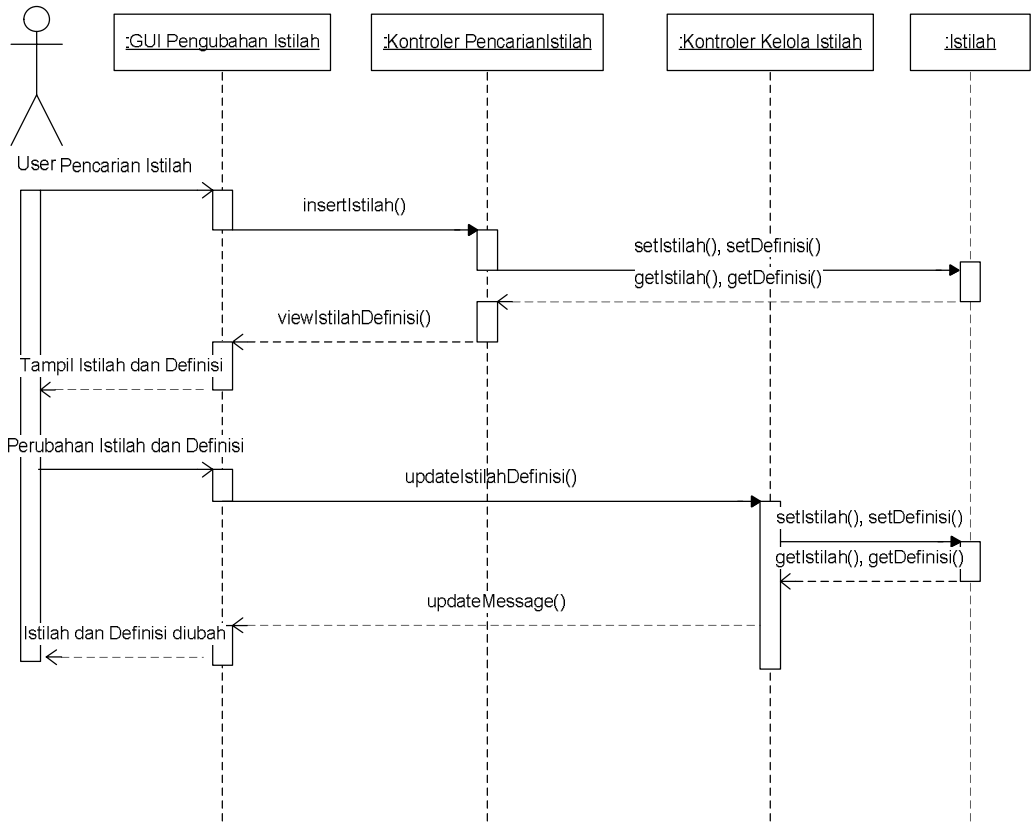
III.4.1.4 Menghapus Istilah dan Definisi



Gambar 3. 7 Sequence Diagram Menghapus Istilah dan Definisi

Pada gambar 3.7 *user* mencari istilah dan definisi yang tidak valid kemudian menjalankan fungsi *delete* untuk menghapus istilah dan definisi. Sistem akan memproses permintaan dan kemudian melakukan penghapusan istilah secara permanen. Setelah berhasil dihapus maka sistem akan mengirimkan pesan ke *user* bahwa istilah telah dihapus.

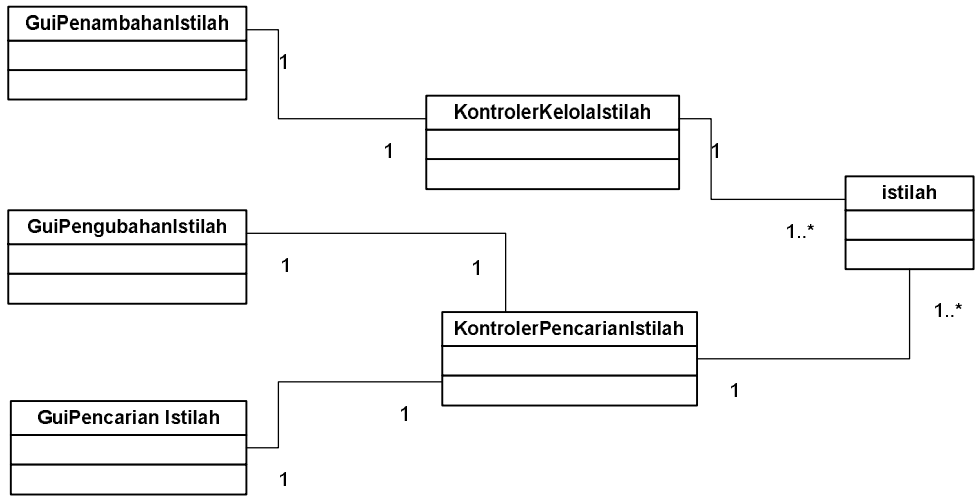
III.4.1.5 Mengubah Istilah dan Definisi



Gambar 3. 8 Sequence Diagram Mengubah Istilah dan Definisi

Pada gambar 3.8 menjelaskan proses mengubah istilah dan definisi. *User* mencari istilah dan definisi yang kurang valid kemudian *user* mengubah istilah dan definisi menjadi valid setelah itu *user* menjalankan fungsi *update* untuk mengubah data. Sistem akan memproses permintaan dan kemudian melakukan pengubahan istilah dan definisi. Setelah proses selesai maka sistem akan mengirimkan pesan ke *user* bahwa data berhasil diubah.

III.5 Class Diagram



Gambar 3. 9 Class Diagram

Pada gambar 3.9 menjelaskan hubungan antar kelas dalam suatu diagram kelas. Diagram kelas diatas menandai kardinalitas dimana kelas KontrolerKelolaIstilah bisa menangani 1 atau banyak data pada kelas istilah yang ditandai dengan 1..* dan kelas istilah hanya bisa ditangani 1 kelas KontrolerKelolaIstilah dan begitu juga dengan kelas-kelas lainnya.

III.6 Rancangan Kelas Rinci

III.6.1 Kelas GuiPenambahanIstilah

GuiPenambahIstilah
- txtIstilah : TextView - txtDefinisi : TextView - edIstilah : EditIstilah - edDefinisi : EditIstilah - btnAdd : Button - btnCancel : Button
+ btnAdd() + btnCancel() + errorMessage() + successMessage()

III.6.2 Kelas GuiPengubahIstilah

GuiPengubahIstilah
- txtIstilah : TextView - txtDefinisi : TextView - edIstilah : EditIstilah - edDefinisi : EditIstilah - btnUpdate : Button - btnDelete : Button - btnCancel : Button
+ btnUpdate() + btnCancel() + btnDelete() + updateMessage() + deleteMessage()

III.6.3 Kelas GuiPencarianIstilah

GuiPencarianIstilah
- edCariIstilah : EditText - btnCari : Button - rdKeyword : RadioButton - rdAlphabet : rdButton - alphaA : Spinner - alphaB : Spinner - alphaC : Spinner
+ clickCariKeyword()

III.6.4 Kelas KontrollerKelolaIstilah

KontrolerKelolaIstilah
- insertIstilahDefinisi : String - updateIstilahDefinisi : String - deleteIstilah :String
+ insertIstilahDefinisi() + updateIstilahDefinisi() + deleteIstilah()

III.6.5 Kelas KontrollerPencarianIstilah

KontrolerPencarianIstilah
- findIstilah : String - viewIstilah : String
+ findIstilah() + viewIstilahDefinisi()

III.6.6 Kelas Istilah

Istilah
- Istilah :String - Definisi : String
+ Istilah(context: Context) + createConnection(db: SQLitedatabase) + getIstilah() + getDefinisi() + setIstilah() + setDefinisi()

III.7 Algoritma

III.7.1 Algoritma KelolaIstilah

Nama Operasi : insertIstilahDefinisi()

Algoritma :

```
 {Menambahkan istilah dan definisi baru}

 Initial state : istilah dan definisi belum ditambahkan
 Final state : istilah dan definisi ditambahkan kedalam database

Inisialisasi istilah
Inisialisasi definisi

If istilah and definisi =’null’, maka akan keluar peringatan
“istilah dan definisi kosong”
Else
Do querry INSERT INTO kamusti (istilah,definisi)VALUES (istilah
sesuai dengan input user,definisi sesuai dengan input user)
```

Nama Operasi : updateIstilahDefinisi()

Algoritma :

```
 {Memperbaharui istilah dan definisi baru}

 Initial state : istilah dan definisi terdapat di database
 Final state : istilah dan definisi diperbaharui

Inisialisasi istilah
Inisialisasi definisi
Inisialisasi istilah_id

Istilah dan definisi diambil sesuai istilah_id dari database untuk
diubah
Do querry UPDATE kamusti SET istilah=sesuai dengan input user,
definisi= sesuai dengan input user WHERE istilah_id=id dari
istilah dan definisi yang dipilih
```

Nama Operasi : deleteIstilah()

Algoritma :

```
 {Menghapus istilah dan definisi yang tidak valid}

 Initial state : istilah dan definisi terdapat di database
 Final state : istilah dan definisi sudah dihapus

Inisialisasi istilah
Inisialisasi definisi
Inisialisasi istilah_id

Istilah dan definisi diambil sesuai istilah_id dari database untuk
diubah
Do querry DELETE FROM kamusti WHERE _id=sesuai dengan istilah dan
definisi yang dipilih
```

III.7.2

III.7.3 Algoritma PencarianIstilsah

Nama Operasi : findIstilah()

Algoritma :

```
 {Mencari istilah dan definisi}

 Initial state : istilah dan definisi terdapat di database
 Final state : menampilkan definisi istilah yang di cari
 ditampilkan

Inisialisasi istilah

Istilah dicari berdasarkan keyword dan alphabet
Istilah dicari berdasarkan keyword dan ditampilkan ke user
Do querry SELECT istilah,definisi FROM kamusti WHERE istilah LIKE
%sesuai dengan inputan user%

Istilah dicari berdasarkan alphabet dan ditampilkan ke user
Do querry SELECT istilah,definisi FROM kamusti WHERE istilah LIKE
sesuai dengan inputan user%
```

Nama Operasi : viewIstilahDefinisi()

Algoritma :

```
 {Menampilkan istilah dan definisi yang di cari}

 Initial state : -
 Final state : menampilkan istilah dan definisi yang di cari

Inisialisasi istilah
Inisialisasi definisi

Istilah dan definisi ditampilkan ke user setelah user menjalankan
```

```
fungsi cari
```

III.7.4 Algoritma Istilah

Nama Operasi : Istilah(context: Context)

Algoritma :

```
 {Menampilkan istilah dan definisi}

 Initial state : -
 Final state : -

 Membuat konstruktor dari kelas istilahdengan parameter context
```

Nama Operasi : createConnection(db: SQLitedatabase)

Algoritma :

```
 {Menampilkan istilah dan definisi}

 Initial state : table belum ditambah
 Final state : table berhasil dibuat

Membuat table dengan sintaks sql create table kamusti (_id integer
primery key autoincrement, istilah text unique, definisi text)
```

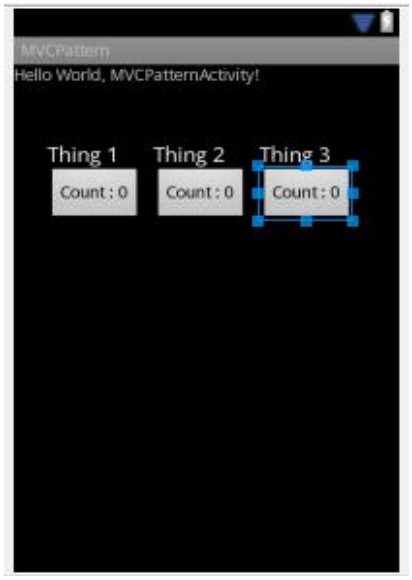
III.8 Struktur MVC

Untuk menggambarkan struktur MVC maka akan dibuat dengan membagi kelas sesuai dengan fungsinya. Untuk itu kelas View dan kelas Controller dijadikan dalam satu kelas dengan kelas View menggunakan file berekstensi .xml yang kemudian dipanggil oleh kelas Aktivity yang merupakan struktur inti yang ada pada android dan juga berfungsi sebagai kelas Controller. Dan untuk kelas Model menggunakan SQLite sebagai databasenya. Pada Gambar 3.10 merupakan Contoh Script View dengan xml

```
main.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<TableLayout android:layout_height="wrap_content" android:id="@+id/tableLayout1" android:layout_width="wrap_content"
    <TableRow android:id="@+id/tableRow1" android:layout_width="wrap_content" android:layout_height="wrap_content">
        <TextView android:id="@+id/textView1" android:layout_width="wrap_content" android:textAppearance="?android:a
        <TextView android:id="@+id/textView2" android:layout_width="wrap_content" android:textAppearance="?android:a
        <TextView android:id="@+id/textView3" android:layout_width="wrap_content" android:textAppearance="?android:a
    </TableRow>
    <TableRow android:id="@+id/tableRow2" android:layout_width="wrap_content" android:layout_height="wrap_content">
        <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/button1" a
        <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/button2" a
        <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/button3" a
    </TableRow>
</TableLayout>
</LinearLayout>
```

Gambar 3. 10 Contoh View dengan xml

Pada Gambar 3.11 merupakan contoh GuiView dengan xml



Gambar 3. 11 Contoh Gui View dengan xml

Pada gambar 3.12 merupakan Contoh Kelas View yang digabung dengan Controller.

main.xml | Model.java | MVCPatternActivity.java

```
package example.mvcpattern;

import java.util.Observable;

public class MVCPatternActivity extends Activity implements Observer,
        OnClickListener {
    private Model mModel;
    private Button mButton1;
    private Button mButton2;
    private Button mButton3;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mModel = new Model();
        mModel.addObserver(this); // add this activity to be the observer of the model
        mButton1 = (Button) findViewById(R.id.button1);//memanggil button 1
        mButton2 = (Button) findViewById(R.id.button2);//memanggil button 2
        mButton3 = (Button) findViewById(R.id.button3);//memanggil button 3
        mButton1.setOnClickListener(this);
        mButton2.setOnClickListener(this);
        mButton3.setOnClickListener(this);
    }

    public void onClick(View v) {
        // TODO Auto-generated method stub
        switch (v.getId()) {
        case R.id.button1:
            mModel.setValueAtIndex(0);
            break;
        case R.id.button2:
            mModel.setValueAtIndex(1);
```

Gambar 3. 12 Contoh Kelas View yang digabung dengan Controller

Pada gambar 3.13 merupakan contoh kelas model tanpa SQLite

main.xml | Model.java

```
package example.mvcpattern;

import java.util.ArrayList;

public class Model extends Observable {
    private List<Integer> mList;

    public Model() {
        mList = new ArrayList<Integer>(3);
        mList.add(0);
        mList.add(0);
        mList.add(0);
    }

    public int getValueAtIndex(final int theIndex)
            throws IndexOutOfBoundsException {
        return mList.get(theIndex);
    }

    public void setValueAtIndex(final int theIndex)
            throws IndexOutOfBoundsException {
        mList.set(theIndex, mList.get(theIndex) + 1);
        setChanged();
        notifyObservers();
    }
```

Gambar 3. 13 Contoh Kelas Model tanpa SQLite

Struktur MVC pada aplikasi implementasi framework MVC berbasis android ini menjelaskan pengelompokkan fungsi-fungsi di framework MVC. Pada Tabel III. 4 akan menjelaskan DetailStruktur MVC yang diterapkan pada android.

Tabel III. 4 Detail Struktur MVC

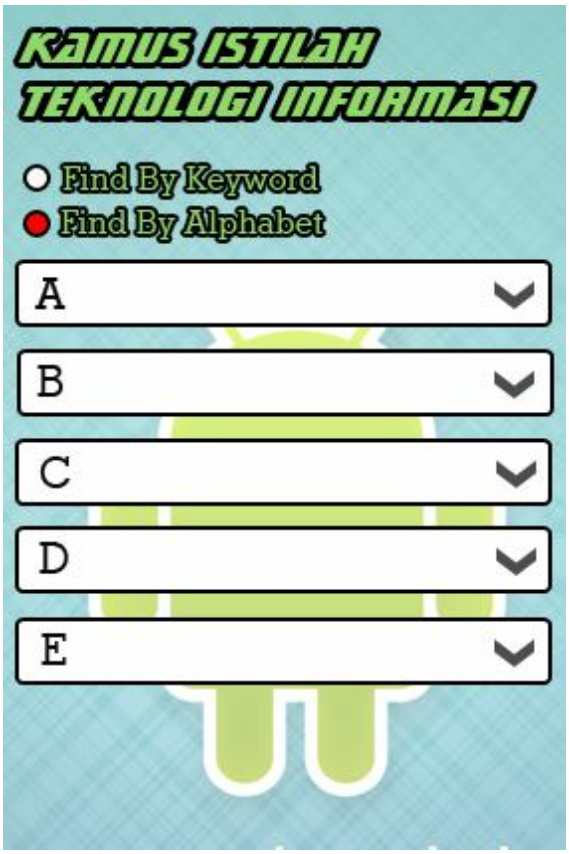
Fungsi	Struktur MVC	Kelas
getIstilah()	Model	Istilah
getDefinisi()	Model	Istilah
setIstilah()	Model	Istilah
set Definisi()	Model	Istilah
findIstilah()	Controller	KontrollerPencarianIstilah
viewIstilahDefinisi()	Controller	KontrollerPencarianIstilah
insertIstilahDefinisi()	Controller	KontrollerKelolaIstilah
updateIstilahDefinisi()	Controller	KontrollerKelolaIstilah
deleteIstilah()	Controller	KontrollerKelolaIstilah
btnAdd()	View	GuiPenambahIstilah
successMessage()	View	GuiPenambahIstilah
errorMessage()	View	GuiPenambahIstilah
btnUpdate()	View	GuiPengubahIstilah
btnDelete()	View	GuiPengubahIstilah
btnCancel()	View	GuiPengubahIstilah
updateMessage()	View	GuiPengubahIstilah
deleteMessage()	View	GuiPengubahIstilah
btnCari()	View	GuiPencarianIstilah

III.9 Perancangan Antarmuka

Aplikasi implementasi framework MVC berbasis android ini memiliki empat gui yaitu Gui pencarian istilah 1, Gui pencarian istilah 2, Gui penambah istilah dan Gui edit istilah.

Gambar 3. 14 Gui Pencarian Istilah 1

Gambar 3.14 menjelaskan komponen-komponen yang terdapat pada antarmuka pencarian istilah dengan *find by keyword.*



Gambar 3. 15 Gui Pencarian Istilah 2

Gambar 3.15 menjelaskan komponen-komponen yang terdapat pada antarmuka pencarian istilah dengan *find by alphabet.*

Gambar 3. 16 Gui Penambah Istilah

Gambar 3.16 menjelaskan komponen-komponen yang terdapat pada antarmuka pencarian istilah dengan dua button add dan cancel.

Gambar 3. 17 Gui Edit Istilah

Gambar 3.17 menjelaskan komponen-komponen yang terdapat pada antarmuka edit istilah dengan tiga button update, delete dan cancel.

Bab IV Implementasi

IV.1 Implementasi Kelas

Berdasarkan perancangan yang telah dilakukan, maka hasil implementasi kelas dan antarmuka yang dibuat secara detail dapat dilihat pada Tabel IV.1.

Tabel IV. 1 Implementasi kelas

No	*Nama Kelas*	*Nama File Fisik*	*Nama File Executable*
1	*PenambahanIstilah*	*PenambahanIstilah.java*	*PenambahanIstilah.class*
2	*PencarianKeywords*	*PencarianKeywords.java*	*PencarianKeywords.class*
3	*PencarianAlphabet*	*PencarianAlphabet.java*	*PencarianAlphabet.class*
4	*PencarianTab*	*PencarianTab.java*	*pencarianTab.class*
5	*Manual*	*Manual.java*	*Manual.class*
6	*PengubahanIstilah*	*PengubahanIstilah.java*	*PengubahanIstilah.class*
7	*KamusTI*	*KamusTI.java*	*KamusTI.class*

Dari perancangan yang telah dilakukan, saat melakukan implementasi menghasilkan 7 kelas yaitu kelas PenambahanIstilah, PencarianKeywords, PencarianAlphabet, PencarianTab, Manual, PengubahanIstilah, dan KamusTI. Dimana kelas tersebut digabung dengan kelas KontrolerKelolaIstilah dan KontrolerPencarianIstilah. Perbedaan antara perancangan dan implementasi dapat dilihat pada Tabel IV.2.

Tabel IV. 2 Perbedaan antara Perancangan dan Implementasi

Nama Kelas Perancangan	*Nama Kelas Implementasi*
GuiPenambahIstilah	*PenambahanIstilah*
GuiPengubahIstilah	*PengubahanIstilah*
GuiPencarianIstilah	*PencarianAlphabet, PencarianKeywords, PencarianTab*
KontrolerKelolaIstilah	*PengubanIstilah, PenambahanIstilah*
KontrolerPencarianIstilah	*PencarianAlphabet, PencarianKeywords, PencarianTab*
Istilah	*KamusTI*

Kelas yang diimplementasikan telah mempresentasikan fungsionalitas yang dibutuhkan oleh aplikasi. Kelas PenambahIstilah menangani fungsionalitas yang berhubungan dengan pengelolaan data seperti penambahan istilah dan definisi.

Kelas PencarianKeywords dan PencarianAlphabet menangani fungsionalitas yang berhubungan dengan pencarian data yaitu pencarian istilah beserta definisinya. Kelas PengubahIstilah menangani fungsionalitas yang berhubungan dengan pengelolaan data seperti meng-*update* istilah dan definisi. Sedangkan kelas KamusTI menangani fungsionalitas yang berhubungan dengan penyimpanan data.

IV.2 Implementasi Antarmuka

Berdasarkan dokumen perancangan yang telah dilakukan, maka hasil implementasi dari antarmuka yang dibuat secara detail dapat dilihat pada tabel IV.3.

Tabel IV. 3 Implementasi Antarmuka

No	*Antarmuka*	*Nama File Fisik*	*Nama File Executable*
1	*PenambahanIstilah*	*tambah.xml*	*tambah.xml*
2	*PencarianIstilah*	*cariKeywords.xml*	*cariKeywords.xml*
3	*PencarianAlphabet*	*cariAlphabet.xml*	*cariAlphabet.xml*
4	*PengubahanIstilah*	*ubah.xml*	*ubah.xml*
5	*Manual*	*Manual.xml*	*Manual.xml*

Pada tahap desain pada awalnya terdapat empat antarmuka yaitu gui pencarian istilah 1, gui pencarian istilah 2, gui penambah istilah dan gui edit istilah, namun pada tahap implementasi terdapat 5 antarmuka yaitu PenambahanIstilah, PencarianKeywords, PencarianAlphabet, PengubahanIstilah dan manual.

IV.3 Perbedaan Perancangan dan Implementasi

Pada sub bab di atas telah terlihat bahwa perancangan berbeda dengan implementasi. Hal tersebut terjadi karena android memiliki sebuah metode untuk membuat antarmuka dengan menggunakan .xml. Berikut adalah contoh antarmuka yang sudah dibuat beserta contoh *coding*:

IV.3.1 Implementasi Pencarian Keywords

Pada Gambar 4.1 merupakan antarmuka pencarian istilah dan definisi by Keywords.

Gambar 4. 1Antarmuka Pencarian Keywords

Berikut merupakan *source code* implementasi Pencarian keywords :

```
public class PencarianKeywords extends Activity {
	private EditText editKeywords;
	private TextView hasilCari;
	private KamusTI kamusTI;
	private ListView listView1;
	private ListAdapter adapter1;
	private SQLiteDatabase db = null;
	private Cursor cursor;
	private Context context;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.carikeywords);

		editKeywords = (EditText)
findViewById(R.id.editKeywords);
		hasilCari = (TextView)
findViewById(R.id.textHasilCari);

		kamusTI = new KamusTI(this);
```

```
            context = this;
            listView1 = (ListView)
findViewById(R.id.listKamusTI1);

            try {
                  kamusTI.createDatabase();
            } catch (Exception error) {
                  // TODO Auto-generated catch block
                  Log.e("Error", "Unable to create database " +
error.getMessage());
            }

            listView1.setSelected(true);
            listView1.setOnItemClickListener(new
OnItemClickListener() {
                  // TODO memberikan perintah jika istilah dan
definisi disentuh
                  public void onItemClick(AdapterView<?> arg0,
View arg1, int arg2,
                              long arg3) {
                        try {
                              db = kamusTI.getReadableDatabase();

                              String sql = "SELECT * FROM kamusti
WHERE istilah LIKE '%"
                                          +
editKeywords.getText().toString().toLowerCase()
                                          + "%' ORDER BY istilah";
                              cursor = db.rawQuery(sql, null);
                              cursor.moveToPosition(arg2);
                              int id =
cursor.getInt(cursor.getColumnIndex("_id"));
                              sendID(id);
                        } catch (SQLException e) {
                              hasilCari.setText(e.toString());
                              Toast.makeText(context,
e.toString(), Toast.LENGTH_LONG)
                                          .show();
                        }
                  }
            });
      }
      // TODO method untuk mencari istilah berdasarkan keywords
      public void findKeywords(View view) {
            try {
                  db = kamusTI.getWritableDatabase();
                  String sql = "SELECT * FROM kamusti WHERE
istilah LIKE '%"
                              + editKeywords.getText().toString()
+ "%' ORDER BY istilah";
                  cursor = db.rawQuery(sql, null);
                  adapter1 = new SimpleCursorAdapter(this,
                              android.R.layout.simple_list_item_2,
cursor, new String[] {
                                          "istilah", "definisi" },
new int[] {
```

```
                                                android.R.id.text1,
android.R.id.text2 });
                  listView1.setAdapter(adapter1);
                  listView1.setTextFilterEnabled(true);
                  String sql1 = "SELECT COUNT(istilah) FROM
kamusti WHERE istilah LIKE '%"
                              + editKeywords.getText().toString()
+ "%' ORDER BY istilah";
                  cursor = db.rawQuery(sql1, null);
                  cursor.moveToFirst();
                  hasilCari.setText("Hasil Pencarian : " +
cursor.getInt(0));
            } catch (SQLException e) {
                  hasilCari.setText(e.toString());
            }
      }
      public void sendID(int id) {
            Intent send = new Intent(this,
PengubahanIstilah.class);
            send.putExtra("ids", id);
            this.startActivity(send);
      }
      // TODO membuat option menu
      public boolean onCreateOptionsMenu(Menu menu) {
            menu.add(0, 1, 0, "Menu Tambah");
            menu.add(0, 2, 0, "Manual");
            menu.add(0, 3, 0, "Exit");
            return true;
      }
      // TODO memberikan perintah jika option menu disentuh
      public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case 1:
                  Intent addMenu = new Intent(this,
PenambahanIstilah.class);

      addMenu.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                  startActivity(addMenu);
                  return true;
            case 2:
                  Intent manMenu = new Intent(this, Manual.class);

      manMenu.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                  startActivity(manMenu);
                  return true;
            case 3:
                  finish();
                  return true;
            }
            return false;
      }
}
```

IV.3.2 Implementasi Pencarian Alphabet

Pada Gambar 4.3 merupakan antarmuka pencarian istilah dan definisi by Alphabet.

Gambar 4. 2 Antarmuka Pencarian Alphabet

Berikut merupakan *source code* implementasi PencarianIstilah Alphabet :

```
public class PencarianAlphabet extends Activity {
	private EditText editAlphabet;
	private TextView hasilCari;
	private KamusTI kamusTI;
	private ListView listView2;
	private ListAdapter adapter2;
	private SQLiteDatabase db = null;
	private Cursor cursor;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.carialphabet);
		editAlphabet = (EditText)
findViewById(R.id.editAlphabet);
		hasilCari = (TextView)
findViewById(R.id.textHasilCari);

		kamusTI = new KamusTI(this);
```

```
            listView2 = (ListView)
findViewById(R.id.listKamusTI2);

            try {
                  kamusTI.createDatabase();
            } catch (Exception error) {
                  // TODO Auto-generated catch block
                  Log.e("Error", "Unable to create database " +
error.getMessage());
            }
            listView2.setSelected(true);
            listView2.setOnItemClickListener(new
OnItemClickListener() {

                  // TODO memberikan perintah jika istilah dan
definisi disentuh
                  public void onItemClick(AdapterView<?> arg0,
View arg1, int arg2,
                              long arg3) {
                        try {
                              db = kamusTI.getReadableDatabase();
                              String sql = "SELECT * FROM kamusti
WHERE istilah LIKE '"
                                          +
editAlphabet.getText().toString().toLowerCase()
                                          + "%' ORDER BY istilah";
                              cursor = db.rawQuery(sql, null);
                              cursor.moveToPosition(arg2);
                              int id =
cursor.getInt(cursor.getColumnIndex("_id"));
                              sendID(id);
                        } catch (SQLException e) {
                              hasilCari.setText(e.toString());
                        }
                  }
            });
      }
      // TODO method untuk mencari istilah berdasarkan alphabet
      public void findAlphabet(View view) {
            try {
                  db = kamusTI.getWritableDatabase();
                  String sql = "SELECT * FROM kamusti WHERE
istilah LIKE '"
                              + editAlphabet.getText().toString()
+ "%' ORDER BY istilah";
                  cursor = db.rawQuery(sql, null);
                  adapter2 = new SimpleCursorAdapter(this,
                              android.R.layout.simple_list_item_2,
cursor, new String[] {
                                          "istilah", "definisi" },
new int[] {
                                          android.R.id.text1,
android.R.id.text2 });
                  listView2.setAdapter(adapter2);
                  listView2.setTextFilterEnabled(true);
                  String sql1 = "SELECT COUNT(istilah) FROM
```

```
kamusti WHERE istilah LIKE '"
                              + editAlphabet.getText().toString()
+ "%' ORDER BY istilah";
                  cursor = db.rawQuery(sql1, null);
                  cursor.moveToFirst();
                  hasilCari.setText("Hasil Pencarian : " +
cursor.getInt(0));
            } catch (SQLException e) {
                  hasilCari.setText(e.toString());
            }
      }
      public void sendID(int id) {
            Intent send = new Intent(this,
PengubahanIstilah.class);
            send.putExtra("ids", id);
            this.startActivity(send);
      }
      // TODO membuat option menu
      public boolean onCreateOptionsMenu(Menu menu) {
            menu.add(0, 1, 0, "Menu Tambah");
            menu.add(0, 2, 0, "Manual");
            menu.add(0, 3, 0, "Exit");
            return true;
      }
      // TODO memberikan perintah jika option menu disentuh
      public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case 1:
                  Intent addMenu = new Intent(this,
PenambahanIstilah.class);

      addMenu.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                  startActivity(addMenu);
                  return true;
            case 2:
                  Intent manMenu = new Intent(this, Manual.class);

      manMenu.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                  startActivity(manMenu);
                  return true;
            case 3:
                  finish();
                  return true;
            }
            return false;
      }
}
```

IV.3.3 Implementasi Pengubahan Istilah

Pada implementasi Pengubahan Istilah terdapat komponen-komponen seperti Update, Delete dan Cancel. Pada Gambar 4.3 merupakan antarmuka Enable Edit dan Gambar 4.4 merupakan antarmuka Pengubahan Istilah.

Gambar 4. 3 Antarmuka Enable Edit

Gambar 4. 4 Antarmuka Pengubahan Istilah

Berikut merupakan *source code* implementasi Pengubahan Istilah :

```
public class PencarianKeywords extends Activity {
	private EditText editKeywords;
	private TextView hasilCari;
	private KamusTI kamusTI;
	private ListView listView1;
	private ListAdapter adapter1;
	private SQLiteDatabase db = null;
	private Cursor cursor;
	private Context context;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.carikeywords);

		editKeywords = (EditText)
findViewById(R.id.editKeywords);
		hasilCari = (TextView)
findViewById(R.id.textHasilCari);

		kamusTI = new KamusTI(this);
		context = this;
		listView1 = (ListView)
findViewById(R.id.listKamusTI1);

		try {
			kamusTI.createDatabase();
		} catch (Exception error) {
			// TODO Auto-generated catch block
			Log.e("Error", "Unable to create database " +
error.getMessage());
		}

		listView1.setSelected(true);
		listView1.setOnItemClickListener(new
OnItemClickListener() {
			// TODO memberikan perintah jika istilah dan
definisi disentuh
			public void onItemClick(AdapterView<?> arg0,
View arg1, int arg2,
					long arg3) {
				try {
					db = kamusTI.getReadableDatabase();
					String sql = "SELECT * FROM kamusti
WHERE istilah LIKE '%"
							+
editKeywords.getText().toString().toLowerCase()
							+ "%' ORDER BY istilah";
					cursor = db.rawQuery(sql, null);
					cursor.moveToPosition(arg2);
					int id =
cursor.getInt(cursor.getColumnIndex("_id"));
					sendID(id);
				} catch (SQLException e) {
					hasilCari.setText(e.toString());
```

```
                                    Toast.makeText(context,
e.toString(), Toast.LENGTH_LONG)
                                                .show();
                              }
                        }
                  });
      }
      // TODO method untuk mencari istilah berdasarkan keywords
      public void findKeywords(View view) {
            try {
                  db = kamusTI.getWritableDatabase();
                  String sql = "SELECT * FROM kamusti WHERE
istilah LIKE '%"
                              + editKeywords.getText().toString()
+ "%' ORDER BY istilah";
                  cursor = db.rawQuery(sql, null);
                  adapter1 = new SimpleCursorAdapter(this,
                              android.R.layout.simple_list_item_2,
cursor, new String[] {
                                          "istilah", "definisi" },
new int[] {
                                          android.R.id.text1,
android.R.id.text2 });
                  listView1.setAdapter(adapter1);
                  listView1.setTextFilterEnabled(true);
                  String sql1 = "SELECT COUNT(istilah) FROM
kamusti WHERE istilah LIKE '%"
                              + editKeywords.getText().toString()
+ "%' ORDER BY istilah";
                  cursor = db.rawQuery(sql1, null);
                  cursor.moveToFirst();
                  hasilCari.setText("Hasil Pencarian : " +
cursor.getInt(0));
            } catch (SQLException e) {
                  hasilCari.setText(e.toString());
            }
      }
      public void sendID(int id) {
            Intent send = new Intent(this,
PengubahanIstilah.class);
            send.putExtra("ids", id);
            this.startActivity(send);
      }
      // TODO membuat option menu
      public boolean onCreateOptionsMenu(Menu menu) {
            menu.add(0, 1, 0, "Menu Tambah");
            menu.add(0, 2, 0, "Manual");
            menu.add(0, 3, 0, "Exit");
            return true;
      }
      // TODO memberikan perintah jika option menu disentuh
      public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case 1:
                  Intent addMenu = new Intent(this,
PenambahanIstilah.class);
```

```
        addMenu.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    startActivity(addMenu);
                    return true;
              case 2:
                    Intent manMenu = new Intent(this, Manual.class);

        manMenu.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    startActivity(manMenu);
                    return true;
              case 3:
                    finish();
                    return true;
              }
              return false;
        }
}
```

IV.3.4 Implementasi Penambahan Istilah

Pada implementasi GuiPenambahanIstilah terdapat komponen-komponen seperti Add dan Cancel. Pada Gambar 4.4 merupakan antarmuka GuiPenambahanIstilah.

Gambar 4. 5 Antarmuka Penambahan Istilah

Berikut merupakan *source code* implementasi Penambahan Istilah :

```
public class PenambahanIstilah extends Activity {
	private EditText edIstilah;
	private EditText edDefinisi;
	private KamusTI kamusTI;
	private SQLiteDatabase db;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.tambah);

		kamusTI = new KamusTI(this);
		db = kamusTI.getWritableDatabase();

		edIstilah = (EditText) findViewById(R.id.editIstilah);
		edDefinisi = (EditText)
findViewById(R.id.editDefinisi);
	}
	public void insertIstilahDefinisi() {
		String inIstilah =
edIstilah.getText().toString().toLowerCase();
		String inDefinisi =
edDefinisi.getText().toString().toLowerCase();
		try {
			if (inIstilah.equals("") ||
inDefinisi.equals("")) {
				Toast.makeText(this, "Kotak Istilah dan
Definisi harus diisi",
						Toast.LENGTH_LONG).show();
			} else {
				// TODO perintah untuk menambahkan istilah
dan definisi ke
				// database
				ContentValues add = new ContentValues();
				add.put("istilah", inIstilah);
				add.put("definisi", inDefinisi);
				db.insertOrThrow("kamusti", null, add);

				// TODO memunculkan progress dialog
				ProgressDialog addDialog = new
ProgressDialog(this);
				addDialog.setMessage("Harap
Menunggu....");
				addDialog.show();

				// TODO menampilkan pesan jika istilah dan
definisi berhasil
				// ditambahkan
				Toast.makeText(this,
						"Istilah dan Definisi berhasil
ditambahkan",
						Toast.LENGTH_LONG).show();

				// TODO ketika proses selesai maka akan
pindah ke
```

```
                        // PencarianKeywords
                        Intent key = new Intent(this,
PencarianTab.class);

      key.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                        startActivity(key);
                        finish();
                  }
            } catch (SQLException e) {
                  // TODO menampilkan kesalahan
                  Toast.makeText(this, "Gagal Menambah Istilah " +
e.toString(),
                              Toast.LENGTH_LONG).show();
            }
      }
      // TODO membatalkan
      public void cancel() {
            finish();
      }
      public boolean onCreateOptionsMenu(Menu menu) {
            menu.add(0, 1, 0, "Add");
            menu.add(0, 2, 0, "Cancel");
            return true;
      }
      public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case 1:
                  insertIstilahDefinisi();
                  return true;
            case 2:
                  cancel();
                  return true;
            }
            return false;
      }
}
```

Dari *coding* di atas terlihat bahwa pada method onCreate() terdapat fungsi setContentView(), fungsi ini adalah memanggil antarmuka yang dibuat dengan menggunakan xml oleh karena itu view dan controller digabung menjadi satu kelas dan diberi nama PencarianKeywords, PencarianAlphabet, PencarianTab, PenambahanIstilah dan PengubahanIstilah. Kelas entity pada perancangan diberi nama Istilah namun pada implementasi diberi nama KamusTI, hal ini terjadi untuk mengurangi ambigu karena entity tidak hanya menampung istilah tetapi juga definisi.

Berikut merupakan source code database pada kamus istilah teknologi informasi agar bisa dijalankan di mobile.

```
public class KamusTI extends SQLiteOpenHelper {

	// TODO inisialisasi variable
	private static final String DB_PATH =
"/data/data/efrizal.app.kamusti/databases/";
	private static final String DB_NAME = "itdict.sqlite";
	private static final int DB_VER = 1;

	public static final String TABLE_NAME = "kamusti";
	public static final String COL_ID = "_id";
	public static final String COL_ISTILAH = "istilah";
	public static final String COL_DEFINISI = "definisi";

	private Context myContext;

	// TODO construtor dari kelas KamusTI
	public KamusTI(Context context) {
		super(context, DB_NAME, null, DB_VER);
		myContext = context;
	}

	// TODO menyiapkan database
	public void createDatabase() throws IOException {
		if (checkDatabase()) {
			// TODO jika database sudah ada maka tidak
melakukan apa-apa
		} else {
			// TODO membaca database
			this.getReadableDatabase();
			try {
				// TODO memanggil method copyDatabase
				copyDatabase();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				throw new Error("Error coping database");
			}
		}
	}

	// TODO memeriksa database apakah sudah ada di sistem
android atau belum
	private boolean checkDatabase() {
		SQLiteDatabase checkDb = null;
		try {
			String myPath = DB_PATH + DB_NAME;
			checkDb = SQLiteDatabase.openDatabase(myPath,
null,
					SQLiteDatabase.OPEN_READONLY);
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		if (checkDb != null) {
```

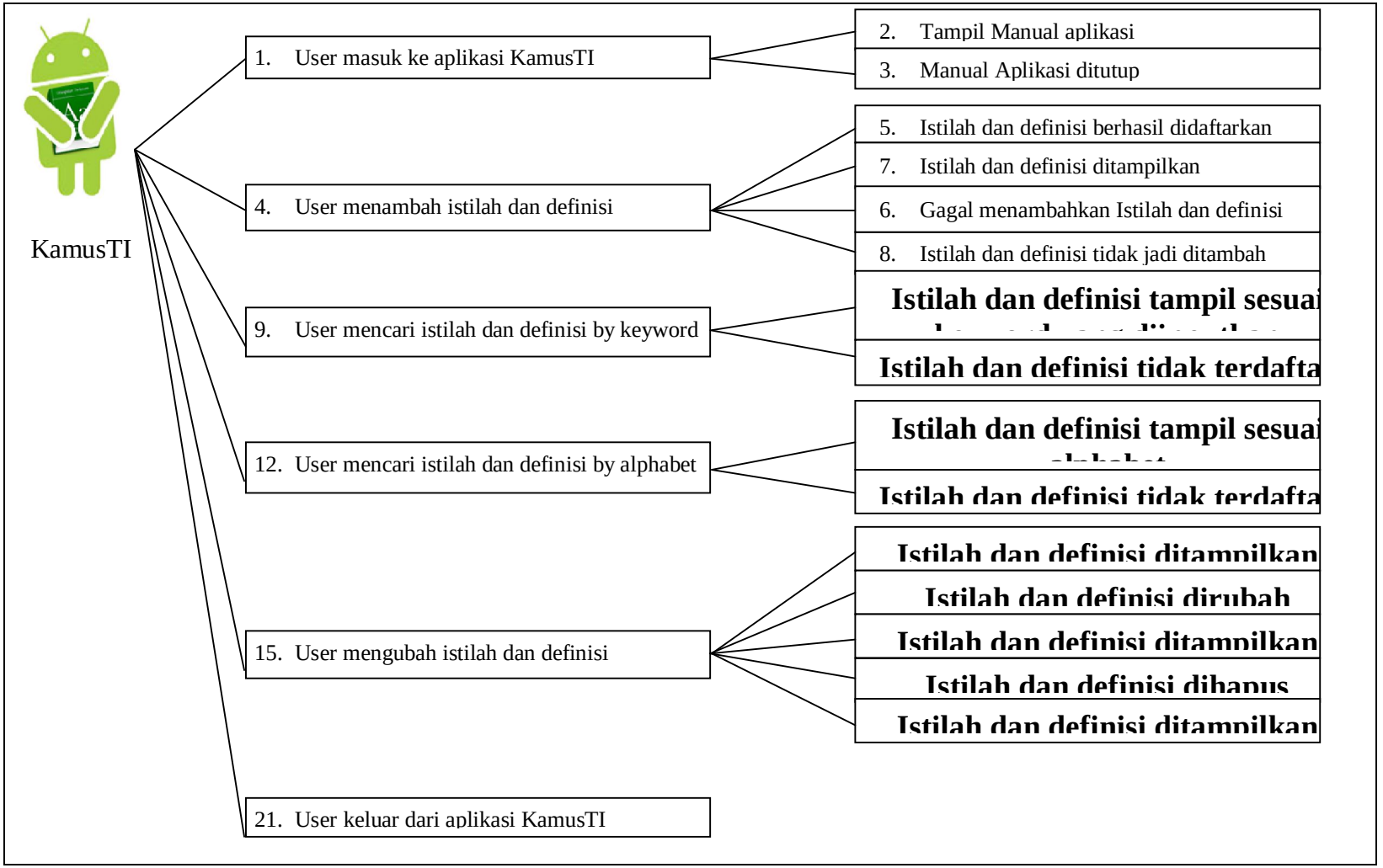
```
                checkDb.close();
            }
            if (checkDb != null)
                return true;
            else
                ;
            return false;

      }

      // TODO menyalin database dari apk ke sistem android
      private void copyDatabase() throws IOException {
            InputStream myInput =
myContext.getAssets().open(DB_NAME);

            String outFileName = DB_PATH + DB_NAME;

            OutputStream myOutput = new
FileOutputStream(outFileName);

            byte[] buffer = new byte[1024];
            int length;
            while ((length = myInput.read(buffer)) > 0) {
                myOutput.write(buffer, 0, length);
            }
            myOutput.flush();
            myOutput.close();
            myInput.close();
      }

      @Override
      public void onCreate(SQLiteDatabase db) {

      }

      @Override
      public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {

      }

}
```

Untuk mempermudah pemahaman *coding* di atas, method createDatabase() akan memeriksa apakah database sudah ada di sistem android atau belum, jika sudah ada maka tidak akan melakukan apa-apa, jika tidak ada maka akan membaca database kemudian akan menyalin database ke sistem android sesuai dengan path yang telah didefinisikan diatas. Method copyDatabase() akan menyalin database yang ada di folder /assets untuk disalin ke sistem android dengan path yang telah didefinisikan diatas

IV.4 Pengujian

IV.4.1 Skenario Pengujian pada Emulator

Skenario pengujian dilakukan terhadap fungsi yang tersedia dalam aplikasi yaitu untuk mengetahui bagaimana aplikasi menangani data masukan dari user dan penanganan kesalahan pada aplikasi. Pengujian aplikasi dilakukan terhadap fungsi yang terdapat pada aplikasi, yaitu fungsi memasukkan istillah teknologi informasi beserta definisinya. Skenario pengujian implementasi framework MVC berbasis android pada emulator dapat dilihat pada Gambar 4.5.



Gambar 4. 6 Skenario Pengujian pada Emulator

IV.4.2 Hasil Rincian Pengujian

Tabel IV. 4 Hasil Rincian Pengujian

No.	Kelas	Fungsi	Use Case	Skenario	Data Uji	Target	Hasil Pengujian	Emulator	HP
1	Manual.java	Select		User masuk aplikasi KamusTI	Tampil manual aplikasi KamusTI	Manual aplikasi KamusTI tampil	Tampil Manual aplikasi	✓	✓
						Keluar dari manual aplikasi KamusTI	Tutup manual aplikasi	✓	✓
2	View dan Controller (GUIPenambahanIstilah, KontrolerKelolaIstilah)	Insert	Menambah istilah dan definisi	User Menambah istilah dan definisi baru	**Data benar :** Menambah istilah TI **boot** kegiatan menghidupkan komputer dengan bootstrap loader	Istilah teknologi informasi dengan nilai masukkan boot berhasil ditambahkan	Istilah dan definisi berhasil didaftarkan	✓	✓
						Menampilkan istilah dan definisi yang baru ditambahkan	Istilah dan definisi ditampilkan	✓	✓
					Data salah : Menambahkan istilah TI **boot** kegiatan menghidupkan komputer dengan bootstrap loader	Istilah dan definisi dengan nilai masukkan boot tidak berhasil ditambahkan karena sudah ada nama istilah dengan nama boot	User membatalkan penambahan istilah dan definisi	✓	✓
3	View dan controller (GUIPencarianIstilah, KontrolerPencarianIstila)	Select	Mencari istilah berdasarkan keyword	User mencari istilah dan definisi by keyword	**Data Benar :** **Nilai masukkan = ram** Hasil pencarian : 3 **ramdac (random access memory digital to analog converter)**	Istilah dan definisi dengan nilai masukkan keyword ram tampil yaitu video ram	Tampil istilah dan definisi yang dicari sesuai keyword yang diinginkan user	✓	✓

No.	Kelas	Fungsi	Use Case	Skenario	Data Uji	Target	Hasil Pengujian	Emulator	HP
					Chip pengontrol VGA yang mengatur color pallete dan convert data dari memori ke sinyal analog di monitor **SDRAM (Synchronised Dynamic RAM)** Adalah pengganti DRAM yang spektakuler. Siklus akses memori ini disinkronkan dengan clock processor. Dengan demikian mengurangi waktu tunggu antara prosessor dan memori **Video ram** Tipe 68 pecial dari DRAM yang memungkinkan akses direct high speed memory melalui sirkuit video. Jenis memori ini lebih mahal bila dibandingkan chips DRAM yang konvensional				
					Data Salah : **Nilai masukkan android** Hasil Pencarian : 0	Istilah teknologi informasi dengan nilai masukan android tidak berhasil tampil karena nilai masukkan tidak terdaftar dalam database KamusTI	Istilah dan definisi tidak terdaftar	✓	✓

No.	Kelas	Fungsi	Use Case	Skenario	Data Uji	Target	Hasil Pengujian	Emulator	HP
			Mencari istilah dan definisi berdasarkan alphabet	User mencari istilah dan definisi by alphabet	**Data Benar :** **Nilai masukkan a** Hasil pencarian : 5 **Abend (Abnormal end)** Penghentian sebuah program atau proses yang tidak normal diakibatkan oleh terjadinya kesalahan *input* data oleh *user*atau *crash* program **Abi (Application Binary Interface)** Pemafaranspesifikasi perangkat keras dan sistem operasi yang sedang digunakan **AbiWord** Aplikasi GNOME *Office* untuk mengolah kata (*word processing*). AbiWord tergolong dalam salah satu perangkat lunak *open source* yang dilisensi dengan GNU GPL (*General Public License*). Karena bisa	Istilah dan definisi dengan nilai masukkan alphabet a tampil	Tampil istilah dan definisi yang dicari sesuai alphabet yang diinginkan user	✓	✓

No.	Kelas	Fungsi	Use Case	Skenario	Data Uji	Target	Hasil Pengujian	Emulator	HP
					digunakan sebebas-bebasnya termasuk juga melakukan modifikasi sesuai kebutuhan **abort** Perintah untuk membatalkan jalannya suatu program secara paksa dan mengembalikan ke sistem operasi. **access** Kegiatan mengambil atau menyimpan data dari atau ke memori atau ke *disk drive* **aliasing** Suatu efek yang muncul karena contoh citra atau sinyal dilakukan pada tingkatan yang terlalu rendah. Efek ini menyebabkan area *high texture* (*rapid change*) dalam citra terlihat sebagai *slow change*. Bila *aliasing* muncul akan sulit untuk mereproduksi contoh citra ke dalam citra asli yang akurat				

No.	Kelas	Fungsi	Use Case	Skenario	Data Uji	Target	Hasil Pengujian	Emulator	HP
					Data salah : **Nilai masukkan ad** Hasil Pencarian : 0	Istilah dan definisi dengan masukkan ad tidak tampil karena tidak sesuai dengan alphabet	Istilah dan definisi tidak terdaftar	✓	✓
4	View dan controller (GUIPengubahanIstila, KontrolerKelolaIstilah)	Update	Mengubah istilah dan definisi	Mengubah istilah dan definisi	**Data benar :** Nilai masukkan yang ingin diubah adalah istilah boot menjadi booting **booting** kegiatan menghidupkan komputer dengan bootstrap loader	Istilah dan definisi dengan nilai masukkan boot berhasil dirubah	Tampil pesan konfirmasi dan Istilah dan definisi berhasil dirubah	✓	✓
						Istilah dan definisi tidak jadi dirubah	Tampil pesan konfirmasi dan istilah dan definisi tidak jadi dirubah	✓	✓
					Data salah : Nilai masukkan yang ingin diubah adalah access ke boot **boot** kegiatan menghidupkan komputer dengan bootstrap loader	Istilah dan definisi dengan nilai masukkan access tidak berhasil dirubah ke nilai masukkan booting karena nama istilah tidak boleh sama	User membatalkan pengubahan istilah dan definisi	✓	✓
		Delete	Menghapus istilah dan definisi	Menghapus istilah dan definisi	**Data uji :** Isitilah yang ingin dihapus adalah access **access** Kegiatan mengambil atau menyimpan data dari atau ke	Istilah dan definisi berhasil dihapus sesuai istilah dan definisi yang diinginkan user	Tampil pesan konfirmasi dan Istilah dan definisi berhasil di hapus	✓	✓

No.	Kelas	Fungsi	Use Case	Skenario	Data Uji	Target	Hasil Pengujian	Emulator	HP
					memori atau ke *disk drive*				
					Data uji : Isitilah yang ingin dihapus adalah browser **Network** Sekelompok komputer yang terhubung yang bisa saling berbagi sumber daya (seperti printer atau modem) dan data	Istilah dan definisi tidak jadi di hapus	User membatalkan menghapus istilah dan definisi	✓	✓
5	View dan controller (GUIPencarianIstilah, KontrolerPencarianIstilah)	Select		Exit		Keluar aplikasi KamusTI	User keluar dari aplikasi KamusTI	✓	✓

Bab V Kesimpulan dan Saran

V.1 Kesimpulan

Setelah mengimplementasikan framework MVC berbasis android mulai dari tahap analisis, perancangan dan implementasi maka kesimpulan yang dapat diambil adalah :

1. Framework MVC dapat diterapkan pada platform android dan aplikasi kamus teknologi informasi.
2. Pada aplikasi ini tidak menangani kesalahan dari istilah teknologi informasi yang menjadi bahan studi kasus.
3. Untuk menjalankan perintah DML bisa menggunakan syntax, bisa juga menggunakan sebuah method dari kelas ContentValues.
4. Dari analisis framework MVC dan pemrograman OO dapat dipadukan sehingga menghasilkan kelas-kelas yang memiliki karakteristik tersendiri. Kelas istilah yang berfungsi sebagai kelas model, kelas Kontroler Penambahan Istilah yang berfungsi sebagai kelas kontroler dan kelas Penambahan Istilah sebagai view. Saat implementasi di Android, kelas view dan controller digabung dan dinamakan Penambahan Istilah karena untuk view di Android diperlukan sebuah file yang bernama tambah.xml yang kemudian dipanggil di kelas Penambahan Istilah dan kelas model dinamakan kelas Istilah.
5. Aplikasi Kamus Teknologi Informasi berhasil dijalankan menggunakan emulator dan handphone android dengan spesifikasi minimum adalah OS 2.2(froyo).Untuk menjalankan aplikasi teknologi informasi di handphone android, diperlukan sebuah method tersendiri untuk memanggil database atau dikenal dengan istilah pre-built database yang mana database sudah disediakan oleh programmer.

V.2 Saran

Untuk pengembangan aplikasi ini pada masa yang akan datang, maka saran yang dapat diberikan adalah framework MVC ini dapat dikembangkan dengan menggunakan Struts dan Spring.

Daftar Pustaka

[1] _, "*MVC*". Tersedia : http://id.wikipedia.org/wiki/MVC, di akses pada tanggal 26 September 2011

[2] _, "*Android (Sistem Operasi)*". (Tersedia :http://id.wikipedia.org/wiki/Android_(sistem_operasi), di akses pada tanggal 27 September 2011

[3] TeknoJurnal. Nugraha, Firman, "*Framework MVC*", (2010, February 24). Tersedia : http://www.teknojurnal.com/2010/02/24/framework-mvc-apa-pengaruhnya-bagi-para-programmer/, di akses pada tanggal 27 September 2011

[4] Maryono. Y, Istiana. B Patmi (2007). *Teknologi Informasi & Komunikasi*. Yudhistira, Jakarta.

[5] _, "*Teknologi Informasi*". Tersedia : http://id.wikipedia.org/wiki/Teknologi_informasi, di akses pada tanggal 29 September 2011

[6] Maseleno. Andino, "*Kamus Istilah Komputer dan Informatika*", (2003). Tersedia : blog.fitb.itb.ac.id/usepm/wp-content/uploads/2010/.../andinokamusti.pdf, diakses pada tanggal 29 September 2011

[7] Action Research. Hendra, "Berbagai Definisi Teknologi Informasi", (2006, Oktober 9). Tersedia : http://www.hdn.or.id/index.php/research/2006/berbagai_definisi_teknologi_informasi_1, di akses pada tanggal 29 September 2011

[8] _, "*Eclipse (Perangkat Lunak)*". Tersedia : http://id.wikipedia.org/wiki/Eclipse_%28perangkat_lunak%29, di akses pada tanggal 29 September 2011

[9] Manage, Margareth (2010). *Penerapan Metode ontologi Taksonomi pada Istilah Teknologi Informasi*. Batam: Politeknik Negeri Batam

[10] _, "*Simple Example of MVC (Model View Controller) Design Pattern for Abstraction*", (2008, April 8). Tersedia : http://www.codeproject.com/KB/tips/ModelViewController.aspx, di akses pada tanggal 4 Oktober 2011

[11] _, "*Android GUI Arsitecture*". Tersedia : http://androidapps.org.ua/i_sect13_d1e11121.html, di akses pada tanggal 4 Oktober 2011

[12] _, "*Model View Controller Pattern*". Tersedia di : http://www.enode.com/x/markup/tutorial/mvc.html, di akses pada tanggal 4 Oktober 2011

[13] _, "*Penjualan Perangkat Berbasis Android Meningkat di Kuartal Kedua 2011*" (2011, Agustus 12). Tersedia : http://www.wowkeren.com/berita/tampil/00009964.html, di akses pada tanggal 5 Oktober 2011

[14] http://masputih.com/2009/05/mvc-sederhana-untuk-pemula, di akses pada 11 Oktober 2011

[15] F, Priyanta. Subari (2011). *Pemograman ANDROID untuk Pemula*. Cerdas Pustaka, Jakarta.

MANUAL APLIKASI

Kamus Istilah Teknologi Informasi

(Studi Kasus Aplikasi Kamus Teknologi Informasi)

Perhatian: Untuk menjalankan aplikasi ini diperlukan handphone dengan sistem operasi android versi 2.2 (froyo) ke atas. Berikut tampilan aplikasi dijalankan dengan menggunakan emulator android.

1. Menjalankan file .apk

Setelah anda mengunduh file ---.rar, ekstrak file tersebut dan cari KamusTI.apk, masukkan file tersebut ke android dan install (mohon maaf tidak ada gambar). Setelah di-install, carilah aplikasi yang sudah di-install dengan nama KamusTI dan jalankan.

2. Tampilan Awal Aplikasi

Menu ini menampilkan manual aplikasi dan tekan button OK untuk menutup manual dan akan dinavigasikan ke menu berikutnya.

3. Mencari Istilah

Untuk mencari istilah terdapat dua jenis pencarian, yaitu mencari berdasarkan keywords dan mencari berdasarkan alphabet.

a. Tampilan Menu Mencari Istilah Berdasarkan Keyword

Menu ini menampilkan istilah yang mirip berdasarkan keyword yang ditulis oleh pengguna dan akan mengembalikan nilai 0 jika keyword yang dicari tidak ditemukan.

b. <u>Tampilan Menu Mencari Istilah berdasarkan Alphabet</u>

Menu ini menampilkan istilah berdasarkan huruf awal yang ditulis oleh pengguna dan akan mengembalikan nilai 0 jika tidak ada.

4. <u>Menu Untuk Pindah ke Menu Lain</u>

Untuk mengakses ini tekan tombol menu dan akan muncul menu navigasi.

INSERT MENU : Untuk menambahkan istilah baru ke dalam kamus

MANUAL: Menampilkan kembali manual aplikasi

EXIT: Keluar dari aplikasi

5. **Tampilan Menu Tambah**

Menu ini untuk menambahkan istilah baru ke dalam kamus. Jika istilah dan definisi sudah diisi tekan tombol add untuk menambahkannya

6. **Mengubah Istilah**

Untuk mengubah istilah dan definisi. Hal yang harus dilakukan pertama kali adalah mencari istilah apa yang ingin diubah dan kemudian pilih istilah tersebut dan akan dinavigasikan ke menu pengubahan istilah.

a. **Tampilan Menu Ubah Istilah**

Gambar diatas menunjukkan istilah yang ingin diubah adalah abi(application binary interface), setelah di pilih menu berganti ke menu pengubahan istilah. Disini pengguna hanya dapat melihat istilah dan definisi. Untuk menghapus atau mengubah, tekan tombol menu maka akan muncul menu Enable Edit seperti gambar di bawah ini :

b. <u>Mengaktifkan Menu Ubah Istilah</u>

Pilih Enable Edit maka kotak istilah dan definisi sudah bisa di sunting. Untuk mengubah tekan tombol update dan untuk menghapus tekan tombol delete.