# APLIKASI KAMUS BAHASA MADURA-INDONESIA-JAWA *ONLINE* DENGAN TEKNOLOGI XHTML MP DAN WCSS

**SKRIPSI**

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik


Disusun Oleh:

**ILHAMI PUJI LESTARI**

**NIM. 0310630067**

**KEMENTERIAN PENDIDIKAN NASIONAL**
**UNIVERSITAS BRAWIJAYA**
**FAKULTAS TEKNIK**
**MALANG**
**2010**

# APLIKASI KAMUS BAHASA MADURA-INDONESIA-JAWA *ONLINE* DENGAN TEKNOLOGI XHTML MP DAN WCSS

**SKRIPSI**

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Teknik


Disusun oleh:
**ILHAMI PUJI LESTARI**
**NIM. 0310630067**

DOSEN PEMBIMBING:

| | |
|---|---|
| **Ir. Heru Nurwarsito, M.Kom.** | **Ir. Sutrisno, MT.** |
| **NIP. 19650402 199002 1 001** | **NIP. 19570325 198701 1 001** |

# LEMBAR PENGESAHAN

# APLIKASI KAMUS BAHASA MADURA-INDONESIA-JAWA *ONLINE* DENGAN TEKNOLOGI XHTML MP DAN WCSS

**SKRIPSI**

**KONSENTRASI TEKNIK INFORMATIKA DAN KOMPUTER**

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Oleh
**ILHAMI PUJI LESTARI**
**NIM. 0310630067-63**

Skripsi ini telah diuji dan dinyatakan lulus pada
Tanggal 04 Agustus 2010

Majelis Penguji:

**Ir. Bambang Siswojo, MT.**
**NIP. 19621211 198802 1 001**

**Ir. Muhammad Aswin, MT.**
**NIP. 19640626 199002 1 001**

**Adharul Muttaqin, ST, MT.**
**NIP. 19760121 200501 1 001**

Mengetahui:
**Ketua Jurusan Teknik Elektro**

**Rudy Yuwono, ST., MSc.**
**NIP. 19710615 199802 1 003**

# KATA PENGANTAR

Alhamdulillah, segala puji bagi Allah SWT semata, sholawat serta salam semoga terlimpahkan untuk Nabi Muhammad SAW, beserta para sahabat yang setia mengikuti jejak beliau. Berkat hidayah dan inayah Allah SWT, skripsi berjudul "Aplikasi Kamus Bahasa Madura-Indonesia-Jawa *Online* dengan Teknologi XHTML MP dan WCSS" dapat diselesaikan.

Ucapan terima kasih yang mendalam penulis sampaikan kepada pihak-pihak yang telah membantu secara langsung maupun tidak langsung, dan telah memberikan dukungan sehingga penyusunan skripsi ini dapat selesai pada waktunya. Atas dukungan dan bimbingannya, pada kesempatan yang berbahagia ini, penulis mengucapkan banyak terima kasih kepada :

1. Ayahanda Silowiyono dan Ibunda St. Megayati Warna tercinta yang telah memberikan doa, motivasi dan dukungan material, serta kakak-kakak dan adikku tersayang Ilham Khalid Setiawan, Hamida Kurnia Sari dan Dian Pramita Utari.
2. Bapak Rudy Yuwono, ST, Msc., selaku Ketua Jurusan Teknik Elektro.
3. Bapak M. Azis Muslim, ST., MT.,Ph.D, selaku Sekretaris Jurusan Teknik Elektro.
4. Bapak Heru Nurwarsito, Ir., MKom., selaku Dosen Pembimbing I, Bapak Sutrisno, Ir., MT., selaku Dosen Pembimbing II, dan Bapak Waru Djuriatno, ST., MT., selaku Ketua Kelompok Dosen Keahlian (KKDK) Konsentrasi Teknik Informatika dan Komputer yang telah meluangkan waktu, tenaga, dan pikiran untuk membimbing penulis dalam penyusunan skripsi ini.
5. Bapak Soemarwanto selaku Dosen Wali yang telah membimbing penulis.
6. Seluruh Dosen dan Staf Administrasi Jurusan Teknik Elektro.
7. Irwanda Yuni Pungkiarto atas bantuan dan dukungan moril.
8. Prima, Sono, Dadack, Astri, Anggi, Aliek, Ejha, Handri, Mbak Nafiri, Anya, Rosi, Iink, atas bantuan, doa, dan semangat.
9. Semua teman Silvergen dan Teknik Elektro.
10. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang telah membantu dalam penyelesaian skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karenanya penulis sangat mengharapkan masukan berupa kritik dan saran yang membangun. Semoga skripsi ini memberikan manfaat bagi semua pihak sesuai dengan fungsinya.

Malang, 4 Agustus 2010

Penulis

# DAFTAR ISI

## DAFTAR GAMBAR

# DAFTAR TABEL

# ABSTRAK

**ILHAMI PUJI LESTARI. 2010. : Aplikasi Kamus Bahasa Madura-Indonesia-Jawa Online dengan Teknologi xHTML MP dan WCSS. Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing : Ir. Heru Nurwarsito, M.Kom dan Sutrisno, Ir, MT.**

Masalah bahasa dan sastra di Indonesia berkenaan dengan tiga masalah pokok, yaitu masalah bahasa nasional, bahasa daerah dan bahasa asing. Bahasa daerah termasuk salah satu budaya nasional yang harus dikembangkan dan dilestarikan. Untuk itu, haruslah terlebih dahulu mandapatkan pemahaman kosakata bahasa daerah yang ingin dikuasainya tersebut. Salah satu solusi tepat untuk mewujudkan pengembangan dan pelestarian bahasa daerah di tengah perkembangan teknologi yang semakin pesat adalah dengan menyediakan aplikasi perangkat lunak yang merepresentasikan sebuah kamus daerah secara *online*.

Perancangan dan implementasi Aplikasi *Web* kamus Madura-Indonesia-Jawa dilakukan dengan menggunakan bahasa pemrograman PHP (versi 5.2.5) dan basis data MySQL (versi 5.0.51a). aplikasi ini terdiri dari dua kategori akses sistem yaitu kategori Admin dan kategori *user* eksternal.

Pengujian Aplikasi kamus Madura-Indonesia-Jawa dilakukan pada setiap aplikasi sistem untuk mengetahui proses yang dilakukan oleh tiap-tiap aplikasi sistem tersebut. Hasil dari pengujian aplikasi sistem dapat diketahui bahwa aplikasi sistem pada kamus Madura-Indonesia-Jawa dapat melakukan proses sesuai dengan kegunaannya masing-masing. Pengujian juga dilakukan terhadap koneksi basis data dan waktu akses *query* yang digunakan oleh Aplikasi Kamus Madura-Indonesia-Jawa. Aplikasi Kamus Madura-Indonesia-Jawa yang dihubungkan dengan koneksi basis data MySQL pada *port* 3306 dapat melakukan proses manipulasi terhadap data-data di dalam basis data. Pengujian waktu akses *query* dilakukan pada masing-masing tabel dengan 500, 1000, 2000, dan 4000 data entri dan 5 kali percobaan. Dari hasil pengujian yang dilakukan terhadap koneksi basis data dan waktu akss *query* menunjukkan bahwa Aplikasi kamus Madura-Indonesia-Jawa dapat berfungsi dengan baik.

**Kata Kunci** : Kamus, xHTML MP, PHP, MySQL, Aplikasi *Web*.

# BAB I
# PENDAHULUAN

## 2.1 Latar Belakang

Masalah bahasa dan sastra di Indonesia berkenaan dengan tiga masalah pokok, yaitu masalah bahasa nasional, bahasa daerah dan bahasa asing. Dalam pasal 32 Undang-undang dasar 1945 ditetapkan bahwa pemerintah memajukan kebudayaan nasional Indonesia dan dalam penjelasan pasal 36 ditetapkan bahwa bahasa-bahasa daerah seperti bahasa jawa dan Madura merupakan sebagian dari kebudayaan. Dengan demikian pemerintah memajukan juga bahasa daerah. Ini berarti bahasa daerah diberi hak hidup berdampingan dengan bahasa Indonesia, dijaga kelestariannya, dibina dan dikembangkan pemakaiannya dalam masyarakat. [EKO-93]

Untuk mengembangkan dan melestarikan bahasa daerah, haruslah terlebih dahulu mandapatkan pemahaman kosakata bahasa daerah yang ingin dikuasainya tersebut. Kemudian dilanjutkan dengan mempelajari tata bahasa yang benar. Sedangkan fasilitas yang tersedia untuk mempelajari kosakata dan tata bahasa daerah tersebut sangat sedikit. Salah satu alternatif untuk dapat dengan mudah belajar dan memahami bahasa daerah adalah dengan menggunakan kamus. Dari kamus tersebut, dapat dicari kata-kata yang ingin diterjemahkan artinya ke dalam Bahasa daerah tertentu atau Bahasa Indonesia. Cara ini cukup mudah namun tidak efektif, karena umumnya sebuah kamus mempunyai halaman yang cukup tebal. Hal ini sangat merepotkan apabila penggunanya harus selalu membawa dan kemudian mencari kata-kata di dalam kamus tersebut. [ADR-06]

Solusi yang tepat dalam permasalahan di atas adalah dengan meyediakan suatu aplikasi perangkat lunak yang dapat menterjemahkan suatu kata dari Bahasa Madura ke Bahasa Indonesia dan sebaliknya. Sehingga kita dapat menterjemahkan suatu kata tanpa harus membawa kamusnya itu sendiri. Aplikasi ini merupakan pengembangan dari aplikasi yang pernah dirancang dan dibuat sebelumnya, yaitu kamus *online* bahasa Jawa. Maka, Kamus *online* Bahasa Madura menjadi alternatif pengembangan aplikasi tersebut yang tidak hanya dapat menterjemahkan bahasa Madura ke Bahasa Indonesia dan sebaliknya, tetapi juga dapat diterjemahkan ke dalam bahasa Jawa, sehingga menjadi lebih aplikatif dan lebih kompleks untuk sebuah aplikasi berbasis *web* yang dapat diakses secara cepat dan mudah dengan menggunakan jaringan internet di mana saja (*online*).

Selain hal tersebut di atas, Saat ini belum ada aplikasi *web* yang menyediakan fasilitas kamus bahasa daerah yang bersifat multibahasa yang menyediakan layanan

penerjemah dari bahasa daerah yang satu ke bahasa daerah yang lain. Kamus *online* yang ada, pada umumnya hanya menyediakan layanan penerjemah dua bahasa saja yaitu bahasa daerah ke bahasa Indonesia atau sebaliknya, dan mereka hanya menyajikan kata berdasarkan abjad dalam Bahasa Indonesia dan sekaligus padanan katanya dalam Bahasa daerah di dalam satu halaman. Sehingga apabila kita ingin mencari padanan kata, kita harus mencari kata tersebut satu per satu secara berurutan ke bawah. Aplikasi yang akan dirancang dan dibuat ini adalah aplikasi yang memberikan kecepatan dan kemudahan pada penggunanya dalam mencari padanan kata ke dalam bahasa yang diinginkan. Sehingga dalam implementasinya, *user* cukup memilih arah bahasa yang diinginkan dan mengetikkan kata yang ingin diterjemahkan.

Dengan semakin berkembangnya pengguna perangkat *mobile*, dalam hal ini perangkat telepon seluler (ponsel), maka aplikasi ini juga dirancang untuk memenuhi kebutuhan tersebut dengan menggunakan teknologi xHTML MP yaitu *Extensible Markup Language Mobile Profile*, merupakan sebuah bahasa markup yang didefinisikan dalam *Wireless Aplication Protokol* (WAP) 2.0, sebuah protokol komunikasi untuk aplikasi-aplikasi nirkabel yang dibuat oleh WAP forum. xHTML sendiri merupakan gabungan antara *Hypertext Markup Language* (HTML) dan *Extensible Markup language* (XML). Penambahan istilah *Mobile Profile* berarti xHTML MP merupakan bahasa pemrograman yang dikhususkan untuk membangun aplikasi-aplikasi yang dapat dibaca melalui perangkat *mobile* seperti telepon seluler. Aplikasi *mobile* yang dibangun dengan menggunakan xHTML MP ini selain dapat dibaca melalui *browser* yang ada di dalam telepon seluler juga dapat dibaca melalui internet *browser*. Jadi, sebenarnya tujuan utama xHTML MP adalah menggabungkan teknologi *browser* yang ada pada *mobile* dan *World Wide Web* (WWW). [JUS-08]

Dari latar belakang di atas, diharapkan aplikasi ini akan bersifat lebih *reliable* dan memberikan lebih banyak kemudahan sehingga menjadi sarana pembelajaran yang cukup efektif untuk masyarakat pada umumnya yang ingin belajar bahasa daerah khususnya bahasa Madura dan Jawa dengan menggunakan teknologi *web* yang bersifat *realtime* dan *up to date*. Dengan demikian, aplikasi kamus *online* ini tidak hanya akan memberi kemudahan dalam pencarian padanan kata, tetapi juga kemudahan penggunaan karena bisa diakses *online* dengan *mobile browser*, di samping menyediakan padanan kata dalam bahasa Madura dan bahasa jawa yang menjadikan aplikasi ini bisa digunakan secara lebih luas.

## 1.2. Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dikhususkan pada :

1. Membuat aplikasi *web* dengan teknologi XHTML MP dan WCSS
2. Merancang dan mengimplementasikan basis data untuk aplikasi kamus bahasa Madura-Indonesia-Jawa *online*
3. Mengimplementasikan dan menguji aplikasi kamus bahasa Madura-Indonesia-Jawa *online*

## 1.3. Batasan Masalah

Dalam perencanaan dan pembuatan skripsi ini perlu dilakukan pembatasan masalah. Pembatasan masalah yang diajukan dalam penyusunan tugas akhir ini antara lain:

a. Pembahasan difokuskan pada perancangan dan pembuatan aplikasi *web* dan berkomunikasi dengan sistem *database* MySQL.
b. Sistem informasi dibuat dengan menggunakan Sistem Operasi Microsoft Windows XP Professional Service Pack 3, XAMPP for Windows Version 1.6.6a yang mencakup program aplikasi, Web Server Apache 2.2.8 (Win32), database MySQL 5.0.51a dan bahasa pemrograman PHP 5.2.5 berorientasi objek
c. *Browser* yang digunakan Mozilla Firefox 3.5.3 dan Opera Mini 4.0.
d. Data yang digunakan adalah Bahasa Indonesia, Madura dan Jawa yang populer di kalangan masyarakat.

## 1.4. Tujuan

Tujuan dari tugas akhir ini adalah untuk dapat merancang dan membuat kamus Bahasa Madura-Indonesia-Jawa *online* dengan menggunakan bahasa pemrograman PHP dan sistem basis data MySQL.

## 1.5. Manfaat

Manfaat yang bisa didapatkan dari tugas akhir ini adalah:

1. Memudahkan dan mempercepat proses administrasi basis data dalam jaringan internet yang dilakukan secara terpusat.
2. Memudahkan dan mempercepat penggunanya dalam mencari padanan kata dari Bahasa Madura ke Bahasa Indonesia dan dari Bahasa Indonesia ke Bahasa Madura, serta dari bahasa Indonesia ke bahasa Jawa dan sebaliknya.

## 1.6. Sistematika Penulisan

Pembahasan tugas akhir ini terdiri dari 7 bab yang disusun dengan sistematika penulisan sebagai berikut:

**Bab I Pendahuluan**

Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat serta sistematika penulisan.

**Bab II Dasar Teori**

Dalam bab ini akan menjelaskan tentang teori-teori yang menunjang penulisan laporan tugas akhir ini, seperti basis data dan bahasa pemrograman yang akan dipakai.

**Bab III Metodologi**

Membahas metodologi yang digunakan dalam penulisan yang terdiri dari studi literatur, analisis dan perancangan, implementasi, pengujian, pengambilan kesimpulan, dan penulisan laporan.

**Bab IV Perancangan**

Membahas analisis kebutuhan dan perancangan sistem yang sesuai dengan teori yang ada serta membahas pembuatan sistem yang dirancang.

**Bab V Implementasi**

Membahas lingkungan implementasi, batasan implementasi, *file-file* implementasi, algoritma operasi yang diimplementasikan.

**Bab VI Pengujian**

Membahas pengujian basis data dan sistem yang diimplementasikan.

**Bab VII Penutup**

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran-saran untuk pengembangan lebih lanjut**.**

# BAB II

# TINJAUAN PUSTAKA

Dalam merancang dan merealisasikan aplikasi kamus bahasa daerah ini dibutuhkan pemahaman mengenai berbagai hal yang mendukung. Pemahaman pengetahuan yang mendukung perancangan dan perealisasian aplikasi ini meliputi teori dasar kamus, teori dasar basis data, HTML dan XHTML MP, WCSS, PHP, dan basis data MySQL.

## 2.1 Kamus

Kamus adalah buku acuan yg memuat kata dan ungkapan yang disusun menurut abjad berikut keterangan tentang maknanya, pemakaiannya, atau terjemahannya serta memuat kumpulan istilah atau nama yang disusun menurut abjad beserta penjelasan tentang makna dan pemakaiannya. [PUR-90]

Kamus berdasakan bentuknya dibedakan menjadi dua macam, yaitu kamus manual dan kamus elektronik. Kamus manual adalah buku referensi atau rujukan yang menerangkan kata-kata yang dicetak di lembaran kertas. Kamus manual mempunyai beberapa ukuran, yaitu kamus mini atau kamus saku, kamus kecil dan kamus besar. Pada umumnya kamus manual mempunyai jumlah halaman yang banyak. Sedangkan kamus elektronik adalah kumpulan kata-kata yang menerangkan kata-kata lain yang diciptakan dalam bentuk perangkat lunak atau *software*. Kamus elektronik dapat diciptakan berbasis *handhold* dan juga berbasis web. [ADR-06]

Adapun kelebihan kamus elektronik berbasis web dibandingkan dengan kamus manual dan kamus elektronik berbasis *handhold* adalah:

1. Pencarian padanan kata menjadi lebih cepat.
2. Penggunaan lebih efektif, karena bisa diakses melalui jaringan intranet dan internet
3. Isi atau data dari kamus elektronik dapat dimanipulasi (ditambah atau dikurangi) dengan mudah. [ADR-06]

### 2.1.1 Bahasa Madura

Bahasa Madura mempunyai jumlah penutur yang cukup besar dan distribusi wilayah yang cukup luas. Daerah pemakaian bahasa Madura tidak hanya terbatas di kawasan Madura saja, tetapi juga di kawasan pantai utara Jawa Timur, seperti Bondowoso, Jember, Situbondo dan Banyuwangi, bahkan di beberapa tempat di pulau Bali juga dijumpai adanya pemakaian bahasa Madura. Sebagaimana lazimnya bahasa yang masih hidup, artinya masih dituturkan oleh tiap-tiap anggota masyarakat di dalam kehidupan atau pergaulan sehari-hari, bahasa Madura mengalami perkembangan yang sangat pesat seiring dengan perkembangan masyarakatnya. Ada kecenderungan faktor-faktor luar bahasa turut mempengaruhi pemakaian dan perkembangan sebuah bahasa. [SIT-98]

Bahasa Madura sebagai bahasa daerah yang dipakai oleh orang-orang Madura yang berbeda tingkat sosialnmya dalam masyarakat, dibedakan pemakaiannya atas tiga tingkatan tutur (*level of speech*) yang dalam bahasa Madura disebutkan sebagai :

1. ***Bhasa Enja'-iya***, yaitu jenis tingkatan tuturan kasar umumnya dipakai oleh :
   - Sesama teman yang sangat akrab dalam pergaulan sehari-hari
   - Orang-orang yang menempatkan diri pada status sosial tinggi terhadap orang-orang yang dianggap berstatus sosial lebih rendah.
2. ***Bhasa Engghi-Enten***, yakni jenis tingkatan tuturan sedang, umumnya dipakai oleh :
   - Sesama teman yang berkedudukan sederajat.
   - Orang yang berkedudukan dituakan terhadap orang yang dianggap muda.
3. ***Bhasa Engghi-Bhunten***, yakni jenis tingkatan tuturan tinggi yang umumnya dipakai oleh :
   - Sesama teman berstatus tinggi atau mereka yang berstatus priyayi.
   - Seorang bawahan atau mereka yang berstatus lebih rendah terhadap orang dengan status sosial lebih tinggi. [SIT-98]

Tingkat bahasa sedang dan tingkat bahasa halus dalam bahasa Madura tidak terlalu banyak, sebab tidak semua kata memiliki tingkat halus dan sedang. Untuk itu, dalam penerapan bahasa harus diketahui pada siapa sebuah kata atau kalimat digunakan. Kalimat yang ditujukan untuk diri sendiri, jika tidak memiliki tingkat

bahasa sedang, bias digantikan dengan tingkat kasar, bukan dengn tingkat bahasa halus. Sedangkan untuk orang yang diajak bicara, jika tidak memiliki tingkat halus, bias diganti dengan tingkat sedang.

### 2.1.2 Bahasa Jawa

Salah satu aspek sasaran pembinaan bahasa Jawa adalah pemakaian tingkat tutur krama yang termasuk ragam bahasa baku. Yang dimaksud tingkat tutur krama adalah tingkat tutur yang menyatakan santun bahasa yang halus, yang berbeda dengan tingkat tutur ngoko yang menyatakan santun bahasa yang biasa. Tingkat tutur adalah variasi bahasa yang perbedaan antara satu dan yang lainnya dibedakan oleh perbedaan sikap santun yang ada pada diri pembicara terhadap lawan bicara dan yang dibicarakan. [EKO-93]

Satuan dasar sebuah tuturan adalah kalimat. Setiap kalimat tersusun dari komponen segmental dan suprasegmental yang di antaranya berupa intonasi. Komponen segmental kalimat itu adalah kata. Dalam sebuah kalimat paling tidak terdapat sebuah kata, tetapi pada umumnya dalam sebuah dalam sebuah kalimat terdapat lebih dari satu kata. Peranan kosakata dalam kaitannya dengan ragam-ragam bahasa Jawa sangatlah besar. Satuan ragam bahasa disebut ngoko atau krama, baik ngoko lugu atau krama lugu, didasarkan pada jenis kata pembentuk kalimat itu. Apabila semua katanya ngoko, ragam itu adalah ragam ngoko. Apabila semua katanya krama, maka ragam itu adalah ragam krama, dan apabila ragam ngoko dan krama itu dimasukkan kata krama inggil, ragam itu masing-masing disebut ngoko halus dan krama halus.

Dari uraian di atas tampak bahwa ragam-ragam santun bahasa Jawa ditandai oleh kata-kata khusus. Kata-kata khusus itu ada empat macam, yakni :

1. Kata ngoko
2. Kata krama
3. Kata krama inggil (inggil)
4. Kata netral

Satu kata digolongkan sebagai kata ngoko, krama, krama inggil atau netral atas dasar segi semantis sosiolinguistik, yakni adanya nilai santun dengan kadar yang berbeda-beda pada masing-masing penanda ragam itu. [EKO-93]

## 2.2 Basis Data

### 2.2.1 Definisi Basis Data

Basis data adalah sebuah cara mendokumentasikan berbagai macam data yang kemudian dimanajemen dengan sebuah sistem untuk kemudian disimpan dalam sebuah media penyimpanan. Dengan demikian data-data tersebut dapat diakses dengan mudah dan cepat. Media penyimpanan tersebut dapat kita ibaratkan sebagai sebuah *storage* penyimpanan, misalnya hardisk.

Dalam basis data, data yang ada tidak hanya diletakkan dan disimpan begitu saja dalam sebuah media penyimpanan, akan tetapi dikelola dengan sebuah sistem pengaturan basisdata yang sering disebut dengan *Database Management System* (DBMS). Dengan begitu suatu data dengan jumlah besar dan kompleks dapat tersusun sangat baik sehingga memungkinkan pengaksesan data dengan mudah dan cepat oleh pengguna. Basis data, dapat juga disebut *database*, adalah sekumpulan informasi yang sangat komplek yang berguna untuk mengatur semua data yang ada didalamnya sehingga dapat diakses oleh oleh pengguna dengan mudah dan cepat [NUG-05].

### 2.2.2 Perancangan Basis Data

Sebuah sistem basis data dibuat untuk mengatasi permasalahan-permasalahan yang nantinya akan memberikan kemudahan dan kecepatan dalam pengaksesan suatu informasi. Perancang sistem basis data harus memperhatikan struktur penyimpanan informasi dan mekanisme untuk memanipulasi informasi tersebut. Selain itu, perancang sistem basis data juga harus bisa menjamin keamanan informasi yang disimpan ketika terjadi *crash* atau terjadi manipulasi oleh orang yang tidak berhak. Jika data yang disimpan akan di-*share* kepada beberapa *users* maka sistem basis data juga harus menghindari keganjilan-keganjilan yang mungkin terjadi.

Perancang sistem basis data sebaiknya menyembunyikan kompleksitas data agar interaksi antara *user* dan sistem dapat dibuat sesederhana mungkin. Basis data yang ditampilkan kepada *user* merupakan gambaran abstrak dari data yang sesungguhnya. Tingkatan/level tentang bagaimana basis data dilihat dan

direpresentasikan dalam sebuah sistem basis data, biasa dikenal dengan istilah abstraksi data. Ada tiga tingkat /level abstraksi data, [FAT-04] di antaranya :

***1. View Level***

Merupakan level tertinggi dari abstraksi data yang hanya menunjukkan sebagian dari basis data. Banyak pemakai dalam sistem basis data tidak akan terlibat (*concern*) dengan semua data/informasi yang tersimpan. Para pemakai umumnya hanya membutuhkan sebagian data/informasi dalam basis data yang kemunculannya di mata pemakai diatur oleh aplikasi pengguna basis data. Aplikasi ini juga dapat berfungsi untuk mengkonversi data asli / fisik menjadi data bermakna bagi pemakai.

***2. Conceptual Level***

Merupakan level berikutnya dalam abtraksi data yang menggambarkan data apa yang sebenarnya (secara fungsional) disimpan dalam basis data dan hubungannya dengan data yang lain. *Level* ini digunakan oleh *"database administrator"*.

***3. Physical Level***

Merupakan level terendah dalam abstraksi data, yang menunjukkan bagaimana sesungguhnya suatu data diorganisasi dan disimpan. Secara fisik, basis data dapat disimpan dalam organisasi relasional, hirarkis, dsb. Tabel sebagai komponen utama dari basis data secara fisik akan direpresentasikan sebagai gabungan dari struktur dan datanya sendiri. Struktur tabel dan dan komponen non-data (seperti *indeks tabel , view, constraint, rule, procedure,* dll), dalam basis data juga harus disimpan secara fisik dan itu terpisah dari data lojik, dan biasa dikenal dengan sebutan meta data atau kamus data.

### 2.2.3 SQL (*Structured Query Language*)

*Stuctured Query Language* (SQl) merupakan standar yang digunakan untuk mengakses *database relational*. Beberapa perangkat lunak yang menggunakan SQL sebagai perintah untuk mengakses data, di antanya DB2, Ingres, Informix, ORACLE, Microsoft Access, PostgreSQL, Rdb, Sybase dan lain-lain. SQL adalah sebuah bahasa permintaan *database* yang terstruktur. Bahasa SQL dibuat sebagai bahasa yang dapat merelasikan antar *database*.

Bahasa SQL ditulis langsung dalam sebuah program *database* sehingga seorang pengguna dapat lemihat langsung permintaan yang diinginkan, sekaligus melihat hasilnya.

SQL (*Stuctured Query Language*) dibagi menjadi dua bentuk *query*, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML). [NUG-08]

#### 2.2.3.1 DDL (*Data Definition Language*)

DDL adalah sebuah metode *query* SQL yang berguna untuk mendefinisikan data pada sebuah *database*, adapun *query* yang dimiliki adalah :

| | | |
|---|---|---|
| create | : | digunakan untuk melakukan pembuatan tabel dan *database* |
| drop | : | digunakan untuk melakukan penghapusan tabel maupun *database*. |
| alter | : | digunakan untuk melakukan pengubahan struktur tabel yang telah dibuat, baik menambah *field* (*add*), mengganti nama *field* (*change*) ataupun menamakannya kembali (*rename*), serta menghapus (*drop*) |
| comments on | : | perintah ini digunakan untuk memberikan komentar terhadap sebuah kolom. komentar ini berguna apabila nama dari kolom tidak mengidentifikasi secara jelas isi dari kolom atau tabel tersebut. |
| grant | : | perintah untuk memberikan hak akses tabel dan *view* kepada user. perintah ini dapat diletakkan pada sebuah program aplikasi atau secara interaktif. |
| revoke | : | perintah revoke akan mencabut hak akses pada tabel dan *view* dari *user*. |
| label on | : | perintah ini digunakan untuk memberikan judul tabel dan kolom. |

#### 2.2.3.2 DML (*Data Manipulation Language*)

DML adalah sebuah metode *quey* yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari *query* ini adalah untuk melakukan pemanipulasian

*database* yang telah ada atau telah dibuat sebelumnya. Adapun *query* yang termasuk I dalamnya adalah ;

insert : digunakan untuk melakukan penginputan/pemasukan data pada tabel *database*

update : digunakan untuk melakukan pengubahan atau peremajaan terhadap data yang ada pada tabel

delete : digunakan untuk melakukan penghapusan data pada tabel. Penghapusan ini dapat dilakukan secara sekaligus (seluruh isi tabel) maupun hanya beberapa *recordset*.

commit : Menuliskan perubahan ke dalam disk

rollback : Membatalkan perubahan yang dilakukan setelah perintah commit yang terakhir.

select : Perintah ini digunakan untuk menampilkan data yang sudah tersimpan dalam tabel.

fetch : Mengembalikan output tempalate

### 2.2.4 MySQL

MySQL adalah sebuah sistem manajemen basis data relasional yang *open source* untuk memproses data di dalam basis data. MySQL menyediakan API (*Application Protocol Interfaces*) untuk bahasa pemrograman C, C++, Eiffel, Java, Perl, PHP, dan Python.

MySQL adalah sebuah sistem manajemen basis data. MySQL banyak digunakan pada aplikasi-aplikasi berbasis *web* dan aplikasi-aplikasi *standalone* yang terintegrasi dengan basis data dan saat ini MySQL telah menjadi sebuah alternatif sistem manajemen basis data dikarenakan kecepatan dan relabilitasnya. MySQL dapat dijalankan pada sistem operasi Unix, Windows, dan Mac OS.

Sebuah basis data relasional menyimpan data pada tabel-tabel yang terpisah dan tidak menyimpan seluruh data yang ada di dalamnya ke dalam sebuah ruang penyimpanan yang besar. Sehingga hal ini dapat menambah kecepatan dan fleksibilitas sistem manajemen basis data. [NUG-08]

#### 2.2.4.1 Tipe data pada MySQL

Beberapa tipa data yang sering digunakan antara lain:

- *Integer*

  Kegunaan untuk bilangan bulat dengan range-2147483648 s/d 2147483647.
- *Float*

  Kegunaan untuk bilangan pecahan.
- *Date*

  Kegunaan untuk tanggal dengan format YYYY-MM-DD
- *Date/Time*

  Kegunaan untuk tanggal dan waktu dengan format YYYY-MM-DD HH:MM:SS.
- *Char*

  Kegunaan untuk variabel string dengan panjang tetap sesuai dengan yang ditentukan. Panjangnya antara 1-255 karakter.
- *Varchar*

  Kegunaan untuk variabel string dengan panjang yang berubah-ubah sesuai dengan yang disimpan pada saat itu. Panjangnya antara 1-255 karakter.

#### 2.2.4.2 Keunggulan MySQL 5.0.51a

Beberapa keunggulan dari MySQL antara lain:

- Kemampuan untuk menangani jumlah *user* yang tidak terbatas secara simultan.
- Memiliki kapasitas untuk menampung *record* sebanyak lebih dari 50.000.000 *record*.
- Eksekusi perintah yang sangat cepat, atau mungkin yang tercepat di pasaran.
- Sistem menajemen user yang efisien dan mudah.

## 2.3 Teori Perancangan Perangkat Lunak

Sebagai penunjang penyusunan skripsi ini dipelukan dasar teori perancangan perangkat lunak yang terdiri dari *Object Oriented Development* dan *Entity-Relationship Diagram*.

### 2.3.1 *Object Oriented Development*

Adalah suatu cara pengembangan perangkat lunak dan system informasi berdasarkan abstraksi objek-objek yang berada di dunia nyata. Abstraksi adalah menemukan serta memodelkan fakta-fakta dari suatu objek yang penting bagi suatu aplikasi. Dengan metode pemrograman yang disebut dengan *Object Oriented Programming*, *Object Oriented Programming* adalah suatu cara baru untuk berfikir dan berlogika dalam menghadapi masalah-masalah yang-akan dicoba atasi dengan bantuan komputer [PRA-04].

*Object Oriented programming* memiliki *standard* bahasa pemodelan yang disebut dengan UML (*Unified Modelling Language*). UML menyediakan bermacam teknik dengan notasi grafik (diagram) memungkinkan *developers* untuk menyajikan *system model software* dalam bentuk yang berbeda dan dari sudut pandang yang berbeda. Meskipun UML terdiri dari 12 teknik, tetapi tidak seluruhnya diaplikasikan didalam *project* sistem, penggunaannya hanya sebagian saja terkait dengan kebutuhan sistem. Berikut akan dijelaskan beberapa teknik, terkait dengan tugas akhir ini [PRA-04].

#### 2.3.1.1 *Use-case Diagram*

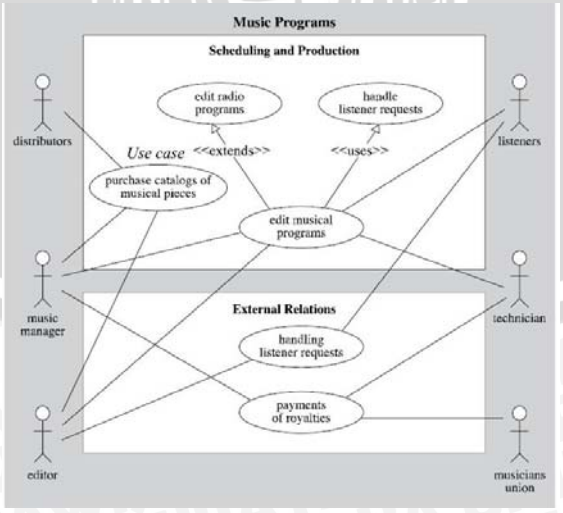
Diagram *Use-case* menunjukan interaksi antara sistem satu dengan sistem lainnya maupun dengan *user* sistem, yang disebut dengan aktor. Diagram ini menggambarkan siapa yang akan menggunakan sistem dan bagaimana *user* berinteraksi dengan sistem untuk mencapai tujuan yang diinginkan. Biasanya, diagram disertai dengan deskripsi narasi yang dijelaskan secara detil dengan cara yang terstruktur untuk langkah-langkah setiap interaksi. Teknik ini memungkinkan menciptakan/membuat deskripsi inisial kebutuhan *user*, sehingga nantinya perilaku sistem bisa ditetapkan dengan menggunakan arti yang lain. Contohnya, *sequence* atau *collaboration diagram*. Tabel 2.2 Menunjukan komponen-komponen di dalam *use-case*

**Tabel 2.1** Komponen *Use-case Diagram*

| Notation | Meaning |
|---|---|
| Use Case | **Use Case :** Gambar ellipse menunjukan single use case. Nama use case dituliskan di dalam ellipse |
| Actor | **Aktor :** menunjukan setiap element yang berhubungan dengan sistem. Bisa digambarkan User atau sistem lain yang berhubungan dengan penjelasan sistem di dalam use case |
| | **Ordinary Relationship :** Menunjukan koneksi, sebagai saluran transfer informasi antara actor dengan use case, atau use case dengan use case |
| uses | **Uses Relationship :** menunjukan ketergantungan use case, yaitu use case satu menggunakan use case yang lain. |
| extends | **Extends Relationship :** Use case complex, yang banyak memiliki aktifitas yang sulit dipahami, dan bisa berubah menjadi simple use case |
| System | **System Boundary :** Kotak menunjukan batas system. Didalam kotak adalah use case, dan bagian luar adalah actor yang berhubungan dengan system. |

Sumber: [SUS-07]

Gambar 2.1 memperlihatkan diagram *use case* untuk contoh *program music*. Diagram tersebut terdiri dari 6 *use case*, yang dibagi dalam 2 subsistem, yang masing-masing subsistemnya berisi : **Assignment and Productions** dan **External Relations.** Pembagian ini memungkinkan 2 *team* mengembangkan sistem. Seperti yang bisa di lihat subsistem **"Assignment and Productions"** terdiri atas 4 *use case*, termasuk **"Edit Musical Program"** *use case* ini melanjutkan *use case* **"Edit Radio Program"** dan menggunakan "*Handle Listener Request*" Subsistem "*External Relations*" terdiri atas 2 *use case*, Sistem secara keseluruhan memiliki 6 aktor.




**Gambar 2.1** *Use case diagram* untuk contoh *music programs*

### 2.3.1.2 *Class Diagram (statis)*

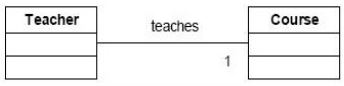
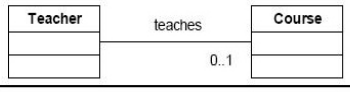
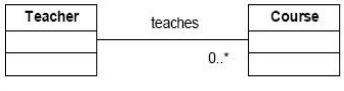
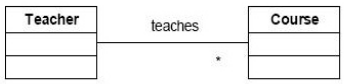
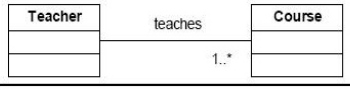
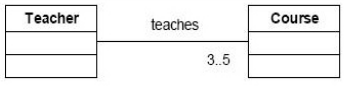
*Class Diagram* digunakan untuk menguraikan sistem struktur *static*. Yang menunjukan kelas *object*, atribut, *methods*, dan bermacam tipe hubungan. Kelas diagram digunakan dalam seluruh metodologi *OO Development*. Kelas dibagi dalam tiga bagian : Bagian atas – Nama Kelas, Bagian Tengah – Atribut Kelas, Bagian Bawah – Fungsi Kelas. Hubungan antar kelas ditunjukan dengan garis penghubung. Tabel 2.2 dan 2.3 menunjukan notasi UML. [R

**Tabel 2.2** Komponen Kelas Diagram

| Notation | Meaning |
|---|---|
| Class Name<br>-attribute:String<br>+doOperationInParameter:1 Double: (Integer) | **Kelas**: Dibagi dalam 3 bagian: (1) Nama kelas, (2) Atribut: Setiap atribut memiliki nama dan tipe data, dan (3) Fungsi: Setiap fungsi memiliki nama, parameter, tipe data. |
| +Role1 -Role2<br>*.1 5.2 | **Ordinary Relationship**: Hubungan ini ditunjukkan dengan garis; multiplisitas ditunjukkan di akhir garis. |
|  | **Inheritance**: Simbol ini menunjukkan hubungan super kelas dan sub kelas. Arah panah menunjukkan super kelas. |
|  | **Composite Agregation**: Simbol ini menunjukkan hubungan "seluruh bagian". Bentuk kepala garis panah menunjukkan "seluruh" kelas, bentuk panah berwarna hitam menunjukkan satu unit dan tidak bisa dipisahkan (ex: badan manusia). |
|  | **Shared Agregation**: Simbol ini menunjukkan hubungan "seluruh bagian". Perbedaannya adalah bagian dari shared agregation bisa menunjuk ke "wholes" yang berbeda |

Sumber : [R

**Tabel 2.3** *Multiplicity Constraints* dalam *UML*

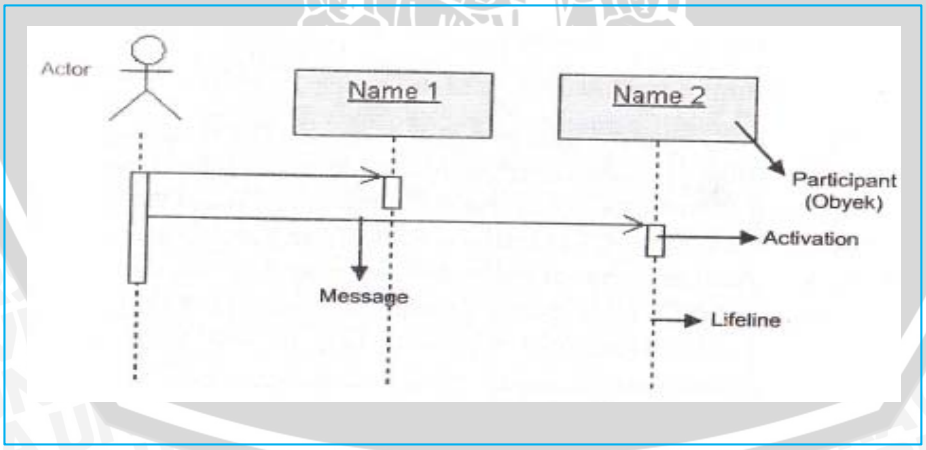
| Type | Notation | Example | Meaning |
|---|---|---|---|
| Exactly 1 | 1 or nothing |  | **Guru hanya mengajar satu kursus - tidak ada lagi.** |
| 0 or 1 | 0..1 |  | **Guru hanya mengajar satu kursus atau tidak sama sekali** |
| 0 or more | 0..* or * |  <br>  | **Guru mengajar banyak kursus atau tidak sama sekali.** |
| 1 or more | 1..* |  | **Guru mengajar banyak kursus tetatapi minimal 1.** |
| A defined range of numbers | 3..5 |  | **Guru hanya mengajar antara 3 sampai 5 kursus** |

Sumber : [R

### 2.3.1.3 *Sequence Diagram (dinamis)*

*Sequence diagram* menggambarkan sebuah skenario interaksi antar objek dalam *use case*. Interaksinya digambarkan oleh *message* yang terkirim dari 1 objek ke objek lainnya. *Sequence diagram* bisa digunakan sebagai cara untuk mengetahui metode apa yang perlu dilampirkan kelas-kelas objek. *Sequence Diagram* bisa digunakan untuk menggambarkan interaksi antara sistem dan elemen external dalam kasus ini disebut dengan "*Black Box Diagram*", dan ini melengkapi diagram *use case*. Diagram ini menggambarkan, setiap *use case*-nya, *subevent* yang dibuat, permintaan *event* dan reaksi sistem yang mungkin terjadi. Satu-satunya komponen dalam diagram adalah elemen eksternal dan sistem.

Penggunaan *sequence diagram* yang lain adalah untuk menggambarkan interaksi antara objek sistem dan *message* yang ditukar. *Message-message* tersebut diminta secara kronologis. Setiap *object* yang ada memiliki "*Life line*" (ditunjukan dengan garis putus-putus dibawah kotak yang menandakan *object*) Menandakan bahwa objek sedang aktif selama *event* berjalan. *Message* sebelumnya ditunjukan dengan tanda panah, dimana terhubung dengan garis objek lain. Apabila *message* sudah terkirim, berarti objek yang sudah di terima memiliki fungsi yang penting untuk melakukan layanan kebutuhan.




**Gambar 2. 2** Simbol-simbol yang ada pada *sequence diagram*
**Sumber:** [MUN-05]

### 2.3.1.4 *StateChart Diagram (dinamis)*

*State chart* menggambarkan perilaku dinamis dari suatu objek termasuk masa hidup dalam sistem. Sebuah objek memungkinkan didalam *state* yang berbeda dari masa hidupnya, tergantung dari macam-macam *events* yang terjadi dan perubahan *state* yang terjadi sekarang. *State chart* menunjukan sebuah objek yang mungkin pada *state*. Dihubungkan dengan peralihan *link* yang diketahui dari *event* yang mungkin terjadi dan oleh karena itu objek *state* berubah. *State chart* juga bisa digunakan untuk *model interface*, *control* dan *reactive system*. Digunakan utama untuk model perilaku dari objek yang memiliki *real time system*, di dalam *real time event* memiliki efek di *state* objek [WAH-05].

**Tabel 2.4** State Chart Komponen

| Notation | Meaning |
|---|---|
| State Name | **State :** Menunjukan state dari objek. Nama menunjukan state yang dituliskan didalam kotak. |
| when: Event [condition] / Action | **Transition :** Menunjukan peralihan antar state. Denagn setiap peralihan dispesifikasikan sebagai:<br>1. Event pemacu sebagai awal dari peralihan<br>2. Kondisi yang membatasi constraint yang harus dipertemukan untuk memunculkan peralihan<br>3. Action akan di bawa keluar ketika peralihan mengambil alih. |
| ● | **Initial State :** Menunjukan Initial state dari objek |
| ◉ | **Final State :** Menunjukan Objek State terakhir |

Sumber: [WAH-05]

*Object Oriented Programming* memiliki 4 macam karakteristik :

- **Pembungkusan (*Encapsulation*)**

Pembungkusan (penyembunyian Informasi) berarti meninggalkan aspek eksternal dari objek yang dapat dimasup (diakses) oleh objek lain dan memfokuskan diri pada implementasi internal suatu objek. Rincian implementasi internal dari suatu objek tersembunyi dari objek-objek lain dan terpisah dari implementasi eksternal, yaitu antarmuka (*interface*) satu objek dengan objek lainnya. Oleh karena itu, implementasi internal suatu objek dapat dirubah tanpa mempengaruhi aplikasi yang menggunakannya. Pembungkusan sebenarnya tidak unik pada pemrograman berorientasi objek tapi kemampuannya untuk menggabungkan struktur data dan perilaku (baca: fungsi atau prosedur) dalam suatu entitas tunggal membuat bahasa berorientasi objek lebih berdaya guna

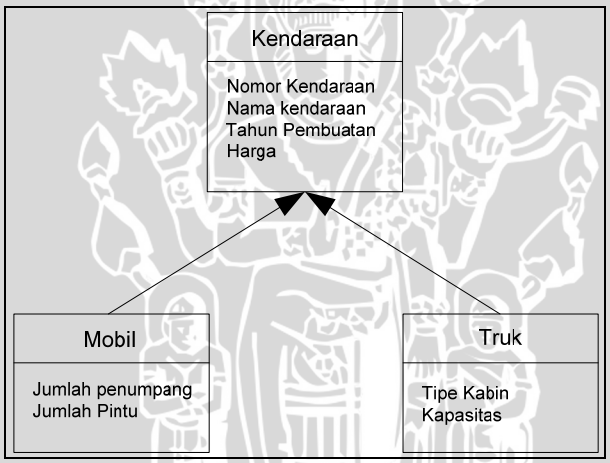
dibandingkan bahasa konvensional yang memisahkan struktur data dan perilaku [PRA-04].

- **Penyembunyian Informasi (*Information Hiding*)**

Penyembunyian implementasi mengacu perlindungan implementasi internal objek. Objek disusun dari antarmuka *public* dan bagian *private* yang merupakan kombinasi data dan *method internal*. Manfaat utama adalah bagian internal dapat berubah tanpa mempengaruhi bagian-bagian program yang lain.

- **Generalisasi dan Pewarisan (*Inheritance*)**

Generalisasi serta Pewarisan adalah suatu cara yang sangat berdayaguna untuk berbagi apa yang dimiliki suatu kelas (atau objek) bagi kelas-kelas (atau objek-objek) yang lain. Misalkan, kita ambil contoh kelas kendaraan bermotor. Mobil, truk, dan lain-lain bisa berbagi atribut yang sama, misalnya : atribut model, tahun pembuatan, jumlah gigi transmisi, dan sebagainya.

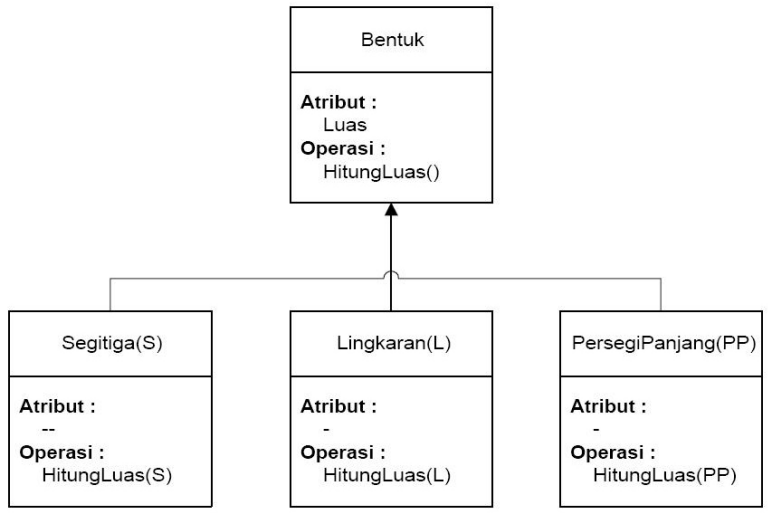



**Gambar 2.3** Pewarisan Atribut dari Kelas Pendahulu ke Kelas Turunan
[Sumber: PRA-04]

Dari keterangan gambar diatas dapat dijelaskan bahwa generalisasi adalah proses *"bottom-up"* ("bawah ke atas") sedangkan spesialisasi adalah proses *"top-bottom"* ("atas ke bawah") dimana pewarisan adalah berbagi atribut yang sama diantara dua kelas dibawahnya (mobil dan truk) dalam kelas yang lebih atas (kendaraan). Perlu diperhatikan bahwa pewarisan memungkinkan atribut-atribut yang cukup dituliskan sekali saja pada superkelas dan tidak perlu ditulis ulang pada subkelas yang mewarisi atribut-atribut yang sama itu.

- **Polimorfisme (*Polymorphism*)**

*Polymorphism* berasal dari bahasa Yunani yang berarti banyak bentuk. Dalam PBO, konsep ini memungkinkan digunakannya suatu *interface* yang sama untuk memerintah objek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda. Dalam konsep yang lebih umum sering kali polymorphism disebut dalam istilah satu *interface* banyak aksi.




Gambar 2.4 Pewarisan dari Atribut yang memiliki Kelas berbeda

Sumber: [PRA-04]

Pada contoh diatas *class* dasar adalah *class* bentuk yang memiliki atribut berupa Luas dan operasi hitung luas, *class* tersebut dapat diturunkan kedalam berbagai macam *class* bentuk seperti Segitiga, Lingkaran, Persegi panjang. *Class* Segitiga, Lingkaran, Persegi panjang memiliki atribut "Luas" dari hasil penurunan *class* bentuk, akan tetapi operasi hitung luas pada masing masing *class* akan berbeda-beda, inilah yang disebut sebagai *polymorphism.* [PRA-04].

### 2.3.2 *Entity-Relationship Diagram* (Diagram E-R)

Diagram *Entity-Relationship* menjelaskan data-data yang ada pada dunia nyata. Diagram *Entity-Relationship* menterjemahkan atau mentransformasikan data dengan memanfaatkan sejumlah model konseptual. Komponen-komponen pembentuk diagram *Entity-Relationship*, antara lain:

1. Entitas dan Himpunan Entitas

   Entitas merupakan individu yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Sekelompok entitas yang sejenis dan

berada dalam lingkup yang sama membentuk sebuah himpunan entitas. Entitas menunjuk pada individu suatu objek, sedangkan himpunan entitas menunjuk pada rumpun dari individu tersebut. Himpunan entitas digambarkan dengan persegi panjang dalam diagram *Entity-Relationship*.

2. Atribut

   Entitas pasti memiliki atribut yang mendeskripsikan karakteristik dari entitas tersebut. Penentuan atau pemilihan atribut-atribut yang relevan bagi sebuah entitas merupakan hal penting lain dalam pembentukan model data. Atribut digambarkan dengan elips dalam diagram *Entity-Relationship*.

3. Relasi dan Himpunan Relasi

   Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Kumpulan semua relasi di antara entitas-entitas yang terdapat pada himpunan entitas-himpunan entitas tersebut membentuk suatu himpunan relasi. Himpunan relasi digambarkan dengan belah ketupat dalam diagram *Entity-Relationship*.

4. Kardinalitas atau Derajat Relasi

   Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi merujuk kepada hubungan maksimum yang terjadi dari himpunan entitas yang satu ke himpunan entitas yang lain. Kardinalitas relasi yang terjadi di antara dua himpunan entitas (misal A dan B) dapat berupa:

   - Satu ke Satu (*One to One*)

     Entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B dan sebaliknya.

   - Satu ke Banyak (*One to Many*)

     Entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B tetapi tidak sebaliknya. Setiap entitas pada himpunan entitas B berhubungan paling banyak dengan satu entitas pada himpunan entitas A.

   - Banyak ke Satu (*Many to One*)

     Entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B tetapi tidak sebaliknya. Setiap

entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

- Banyak ke Banyak (*Many to Many*)

  Entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B dan sebaliknya.

Diagram E-R selalu dibuat bertahap. Tahapan pertama yang dapat dilakukan adalah dengan membuat diagram E-R awal. Tujuan dari pentahapan ini adalah untuk mendapatkan sebuah rancangan basis data minimal yang dapat mengakomodasi kebutuhan penyimpanan data terhadap sistem yang sedang ditinjau. Langkah-langkah yang harus dilakukan antara lain: [RM-03]:

1. Mengidentifikasi dan menetapkan seluruh himpunan entitas yang akan terlibat.
2. Menentukan atribut-atribut *key* dari masing-masing himpunan entitas.
3. Mengidentifikasi dan menetapkan seluruh himpunan relasi di antara himpunan entitas-himpunan entitas yang ada beserta *foreign key*-nya.
4. Menentukan derajat atau kardinalitas relasi untuk setiap himpunan relasi.
5. Melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif (*non-key*).

## 2.4 Teori Dasar Web

Penyusunan skripsi ini memerlukan teori dasar *web* sebagai dasar teori yang akan memberikan penjelasan tentang *Hypertext Preeprocessor* (PHP), xHTML (*Extensible Markup Language*), HTTPS (*HyperText Transport Protocol Secure*), *Apache Web Server*.

### 2.4.1 PHP (*Hypertext proprocessor*)

PHP adalah bahasa *scripting* yang menyatu dengan HTML dan dijalankan pada *server side*. Ini berarti bahwa semua script PHP diletakkan di server dan diterjemahkan oleh web server terlebih dahulu, kemudian hasil terjemahannya dikirim ke browser client. Sebagian besar perintahnya berasal dari bahasa C, Java dan Perl dengan beberapa fungsi tambahan khusus PHP. Bahasa ini

memungkinkan para pembuat aplikasi web menyajikan halaman HTML dinamis dan interaktif dengan cepat dan mudah yang dihasilkan *server*. [NUG-08]

#### 2.4.1.1 Variabel Pada PHP

Dalam PHP setiap nama variabel diawali tanda dollar "$". Misalnya nama variable a, dalam PHP akan ditulis dengan $a. Jenis suatu variable ditentukan pada saat jalannya program (*runtime*) dan tergantung pada konteks yang digunakan. Contoh penggunaan variabel pada PHP:

```
<?php
$a=”5”;
$b=”2”;
$hasil=$a+$b;
echo($hasil);
?>
```

PHP mendukung beberapa jenis variabel sebagai berikut:

1. *Integer*

   Variabel berjenis *integer* bertujuan untuk menyimpan bilangan bulat.

2. *Double*

   Untuk menyimpan bilangan bernilai pecahan dan juga bilangan pemangkatan.

3. *String*

   String merupakan jenis data karakter yang disimpan sebagai nomor pada memori komputer. Nilai yang disimpan adalah nilai ASCII karakter string tersebut.

4. *Array*

   Adalah sebuah set variabel yang mempunyai jenis data sama. Array mengandung komponen yang disebut elemen dan disimpan pada lokasi tertentu pada memory .

5. *Object*

   Jenis variabel objek adalah berdasarkan gambaran objek pada dunia nyata yang mempunyai status dan tingkah laku.

6. PDFDoc dan PDFInfo (hanya bila dukungan PDF diaktifkan).

#### 2.4.1.2 Script PHP

Ada dua cara yang sering digunakan untuk menulis *script* PHP, yaitu:

1. *Embedded Script*

   Cara ini dilakukan dengan meletakkan *script* PHP di antara *tag-tag* HTML. Berikut contoh penggunaannya:

```
<html>
<head>
<title> embedded script </title>
</head>
<body>
<?
echo “Ini adalah contoh embedded script”;
?>
</body>
</html>
```

2. *Non-Embedded Script*

   *Non-embedded script* merupakan pembuatan program murni PHP, dimana *tag-tag* HTML yang diletakkan didalamnya. Berikut contoh penggunaannya:

```
<?
    echo”<html>”;
    echo”<head>”;
    echo”<title> non embedded script </title>”;
    echo”</head>”;
    echo”<body>”;
    echo”Ini adalah contoh non embedded script”;
    echo”</body>”;
    echo”</html>”;
?>
```

#### 2.4.1.3 Tag PHP

Penulisan program PHP dapat dilakukan dengan menggunakan berbagai *tag* tetapi tidak akan mempengaruhi hasil program yang dibuat. *Tag* yang dapat digunakan adalah sebagai berikut:

1. *Style* Standar

   *Style* standar PHP sangat mirip dengan penulisan program XML, yakni diawali dengan <**?php** dan diakhiri dengan **?>.** Contoh penulisannya adalah

   ```
   <html>
   <head>
   <title> style standar </title>
   </head>
   <body>
   <?php
   echo “Ini adalah contoh style standar PHP”;
   ?>
   </body>
   </html>
   ```

2. *Short Style*

   *Style* ini cukup praktis bila dibandingkan dengan *style* sebelumnya. Contoh penulisannya adalah:

   ```
   <html>
   <head>
   <title> short style </title>
   </head>
   <body>
   <?
   echo “Ini adalah contoh short style PHP”;
   ?>
   </body>
   </html>
   ```

3. *Style* JavaScript

   *Style* ini digunakan apabila telah terbiasa menggunakan pemrograman JavaScript. Contoh penulisannya adalah:

   ```
   <html>
   ```

<head>
<title> style JavaScript </title>
</head>
<body>
<SCRIPT LANGUAGE = 'JavaScript'>
echo "Ini adalah contoh style Javascript";
</SCRIPT>
</body>
</*html*>

4. *Style* ASP

   Diawali dengan <**%** dan diakhiri dengan **%**>. Contoh penulisannya adalah:

   <html>
   <head>
   <title> style ASP </title>
   </head>
   <body>
   <%
   echo "Ini adalah contoh style ASP";
   %>
   </body>
   </html>

#### 2.4.1.4 Kelebihan PHP 5

PHP membuat proses pengembangan aplikasi menjadi mudah karena kelebihan-kelebihannya, yaitu:

1. *Script* (kode program) terintegrasi dengan file HTML, sehingga developer bisa berkonsentrasi langsung pada penampilan dokumen webnya.
2. Tidak ada proses *compiling* dan *linking*.
3. Berorientasi obyek (*object oriented*).
4. Sintaks pemrogramannya mudah dipelajari, sangat menyerupai C dan Perl.

Integrasi yang sangat luas ke berbagai *server* basis data. Menulis web yang terhubung ke basis data menjadi sangat sederhana. Basis data yang didukung oleh PHP antara lain Oracle, Sybase, mSQL, MySQL, Solid, ODBC, PostgreSQL, dBase, UNIX dbm dan sebagainya. [NUG-08]

### 2.4.2 Framework

Framework dapat diartikan sebagai koleksi atau kumpulan potongan-potongan program yang disusun atau diorganisasikan sedemikian rupa, sehingga dapat digunakan untuk membantu membuat aplikasi utuh tanpa harus membuat semua kodenya dari awal. Sebuah framework selain menyediakan lingkungan pengembangan sendiri-sendiri juga menyediakan berbagai macam fungsi siap pakai yang bisa kita gunakan dalam pembuatan *website*. Sehingga tidak perlu kaget jika akan banyak kode atau fungsi yang terlihat tidak seperti biasanya, karena fungsi-fungsi tersebut merupakan fungsi bawaan framework dan bukan fungsi asli dari PHP. Fungsi tersebut terkadang merupakan pengembangan atau penyesuaian fungsi asli PHP agar lebih mudah digunakan atau agar lebih sesuai dengan kebutuhan pengguna. [PRI - 10]

Framework yang berkembang saat ini di antaranya Zend Framework, Cake PHP, Trax, Symfony dan lain lain. Jenis framework yang digunakan dalam perancangan aplikasi kamus Madura- Indonesia-Jawa *online* ini adalah CodeIgniter.

#### 2.4.2.1 Codeigniter

CodeIgniter adalah sebuah Aplication Development Framework (sebuah alat kerja) yang di gunakan untuk membuat website dengan menggunakan bahasa pemrograman PHP. Tujuan utamanya adalah memberikan kemampuan untuk mengembangkan suatu proyek lebih cepat dari pada membuat kode program dari awal, dengan memberijan banyak library untuk fungsi fungsi umum, juga dengan struktur interface dan struktur logika yang sederhana untuk menggunakan library ini. CodeIgniter terfokus pada bagaimana meminimalkan jumlah kode program yang di gunakan untuk melakukan suatu tugas.

CodeIgniter menggunakan arsitektur Model, View, Controller (MVC) yang memudahkan dalam pemisahan antara script yang bersifat logika, tampilan dan model yang berhubungan dengan database. Hal ini sangat menguntungkan dalam pengembangan suatu aplikasi yang besar, karena antara tampilan, logika dan data sudah di pisahkan dengan jelas.

Kelebihan frameworks CodeIgniter dibandingkan dengan frameworks PHP lain yang adalah sebagaiberikut :

1. Menggunakan Arsitektur MVC (Model View Controller).
2. Dapat berjalan di server PHP 4 maupun PHP 5.
3. Mudah dimengerti dan dikembangkan.
4. Menyediakan banyak library untuk pengembangan.
5. Tidak membutuhkan installasi.
6. Dapat melakukan abstraksi database.
7. Memiliki dokumentasi yang lengkap.
8. Memiliki komunitas pengguna yang besar.

### 2.4.2.2 konsep MVC (*Model View Controller*)

Seperti sudah disebutkan di muka bahwa CodeIgniter menerapkan lingkungan pengembangan dengan metode MVC (Model View Controller). MVC memisahkan antara logika pembuatan kode dengan pembuatan *template* atau tampilan *website*. Penggunaan MVC membuat pembuatan sebuah proyek *website* menjadi lebih terstruktur dan lebih sederhana. Secara sederhana konsep MVC terdiri dari tiga bagian yaitu bagian *Model*, bagian *View* dan bagian *Controller*. Didalam *website* dinamis setidaknya terdiri dari 3 hal yang paling pokok, yaitu basis data, logika aplikasi dan cara menampilkan halaman wesite. 3 hal tersebut direpresentasikan dengan MVC yaitu model untuk basis data, *view* untuk cara menampilkan halaman *website* dan *controller* untuk logika aplikasi.

**1. Model**

Merepresantiskan struktur data dari *website* yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks atau file xml. Biasanya didalam model akan berisi *class* dan fungsi untuk mengambil, melakukan *update* dan menghapus data *website*. Karena sebuah website biasanya memnggunakan basis

data dalam menyimpan data maka bagian *Model* biasanya akan berhubungan dengan perintah-perintah *query* SQL. Model bisa dibilang khusus digunakan untuk melakukan koneksi ke basis data oleh karena itu logika-logika pemrograman yang berada didalam *model* juga harus yang berhubungan dengan basis data. Misalnya saja pemilihan kondisi tetapi untuk memilih melakukan *query* yang mana.

**2. View**

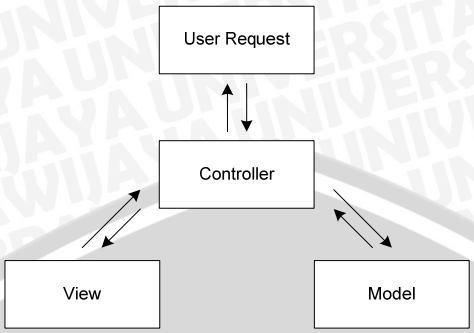
Merupakan informasi yang ditampilkan kepada pengunjung *website*. Sebisa mungkin didalam *View* tidak berisi logika-logika kode tetapi hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. *View* bias dibilang adalah halaman website yang dibuat menggunakan HTML dengan bantuan CSS atau JavaScript. Didalam *view* jangan pernah ada kode untuk melakukan koneksi ke basis data. View hanya dikhususkan untuk menampilkan data-data hasil dari *model* dan *controller*.

**3. Controller**

*Controller* merupakan penghubung antara *Model* dan *View*. Didalam *Controller* inilah terdapat *class* dan fungsi-fungsi yang memproses permintaan dari View kedalam struktur data didalam *Model*. *Controller* juga tidak boleh berisi kode untuk mengakses basis data. Tugas *controller* adalah menyediakan berbagai variabel yang akan ditampilkan di view, memanggil model untuk melakukan akses ke basis data, menyediakan penanganan *error*, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap *input*.

Jadi secara singkat urutan dari sebuah *request* adalah sebagai berikut : *user* berhubungan dengan *view*, dimana didalam *view* inilah semua informasi ditampilkan. Saat user melakukan permintaan atau request, misal klik tombol maka request tersebut akan diproses oleh *Controller*. Apa yang harus dilakukan, data apa yang diinginkan, apakah ingin melihat data, atau memasukan data atau mungkin melakukan validasi data terlebih dahulu, semua diproses oleh *Controller*. Kemudian Controller akan meminta *Model* untuk menyelesaikan request, entah itu melakukan *query* atau apapun. Dari *Model*, data akan dikirim kembali untuk di proses lebih lanjut di dalam *Controller* dan baru dari *Controller*

data akan ditampilkan di *View*. Alur program aplikasi berbasis framework codeigniter dapat dilihat pada gambar 2.5.




Gambar 2.5 Model – View - Controller
Sumber : [PRI-10]

### 2.4.3 XHTML MP

*Extensible Hypertext Markup Language Mobile Profile* (XHTML MP) merupakan sebuah bahasa *markup* yang didefinisikan dalam *Wireless Aplication Protocol* 2.0, yaitu sebuah *protocol* komunikasi untuk aplikasi-aplikasi nirkabel yang dibuat oleh WAP forum. XHTML sendiri sebenarnya merupakan gabungan antara *Hypertext Markup Language* (HTML) yang telah umum digunakan sebagai bahasa pemrograman untuk membuat situs-situs *internet* dewasa ini dan *Extensible Markup Language* (XML). Penambahan istilah *Mobile Profile* berarti XHTML MP merupakan bahasa pemrograman yang dikhususkan untuk membangun aplikasi-aplikasi yang dapat dibaca melalui perangkat-perangkat *mobile*, seperti telepon seluler (ponsel), PDA ataupun *smartphone*. Aplikasi *mobile* yang dibangun dengan menggunakan XHTML MP ini selain dapat dibaca melalui *browser* yang ada di dalam ponsel juga dapat dibaca melalui *internet browser* [JUS-08]

#### 2.4.3.1 Struktur Dasar Dokumen XHTML MP

Struktur dokumen XHTML MP terdiri atas beberapa bagian penting seperti terlihat dibawah ini:

<?xml version=”1.0”?>
<!DOCTYPE html PUBLIC “-//WAPFORUM//DTD XHTML *Mobile* 1.0//EN” ”http://www.wapforum.org/DTD/xhtml-*mobile*10.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>

```
<head>
 <title>JUDUL</title>
 <meta name=”author” content=”Nama”/>
</head>
<body>
 <p>isi . . . </p>
</body>
</html>
```

Dari *script* di atas secara umum dokumen XHTML MP terdiri atas tiga bagian utama yaitu:

- DOCTYPE,
- Head, dan
- Body

Baris pertama dari sebuah dokumen XHTML MP adalah deklarasi XML. Deklarasi ini dibutuhkan karena XHTML MP merupakan turunan dari XML. Seperti terlihat pada *script* diatas deklarasi xml diikuti dengan versi dari dokumen yaitu versi 1.0. dan juga jenis *encoding* karakter yang digunakan bisa dispesifikasikan dalam baris pertama. Deklarasi DOCTYPE pada baris kedua mutlak harus ada dalam dokumen XHTML MP. Diletakkan di antara deklarasi xml dan elemen <html>. Deklarasi ini menspesifikasikan nama dari DTD (*Document Type Definision*) dan url dimana DTD tersebut berada. DTD menyimpan semua informasi tentang sintaks dari bahasa *markup* dan mendefinisikan elemen-elemen, atribut, dan aturan-aturan yang harus digunakan. Baris selanjutnya yaitu elemen <html>, <head>, <title>, dan <body> yang merupakan batang tubuh dari dokumen XHTML MP. [JUS-08]

a. Elemen <html> merupakan elemen utama. Secara umum semua dokumen HTML menggunakan elemen <html>, secara khusus XHTML MP menambahkan sebuah atribut xmlns dalam elemen <html>. Atribut ini berfungsi untuk mendefinisikan *namespace* bagi XHTML MP.
b. Elemen <head> berfungsi untuk menyimpan semua informasi tentang dokumen XHTML MP.

c. Elemen <body> berisi semua elemen-elemen yang akan ditampilkan dalam halaman situs XHTML MP.
d. Elemen  digunakan untuk menampung sebuah paragraf dalam halaman XHTML MP. Jadi setiap WAP *browser* membaca elemen , maka akan memulai *text* tersebut pada baris yang baru. Tetapi, XHTML MP tidak memiliki elemen untuk membuat paragraf rata kanan, kiri, atau tengah.
e. Metadata dalam XHTML MP diletakkan dalam sebuah tag <meta />, tag ini harus dituliskan di dalam tag <head>...</head>. Dalam XHTML MP metadata berfungsi untuk menjelaskan pemilik dokumen, menentukan kapan sebuah dokumen akan di*refresh*, dan pengaturan akses terhadap *cache memory*.

#### 2.4.3.2 Keuntungan menggunakan XHTML MP sebagai bahasa pemrograman untuk semua aplikasi *mobile* internet

a. Sebuah situs XHTML dapat dibaca melalui *mobile browser* maupun *internet browser*, hal ini berarti tidak diperlukan teknologi berbeda untuk membangun aplikasi yang dapat diakses melalui WWW atau WAP 2.0.
b. Sebuah situs dapat diakses melalui dua media sekaligus maka hanya dibutuhkan sebuah *tool* untuk membangun situs yang dapat diakses melalui WWW dan WAP 2.0.
c. Pada dasarnya XHTML adalah HTML dengan aturan-aturan ketat XML sehingga tidak sulit untuk membangun situs *internet* dengan XHTML.
d. Kehadiran WAP CSS (WCSS) pada XHTML MP sangat memudahkan pengaturan tata letak dari sebuah situs. Karena WCSS memungkinkan adanya pemisahan antara isi dari sebuah situs dengan pengaturan tata letaknya. [JUS-08]

#### 2.4.3.3 Perbedaan XHTML MP dan HTML

a. Elemen-elemen dalam XHTML MP harus tersarang dengan benar .
    hello world! , bukan
    hello world! 
b. Elemen-elemen dalam XHTML MP harus tertutup.
    hello world! 

Elemen-elemen lain dalam XHTML MP yang membutuhkan penutupan secara benar adalah elemen-elemen kosong seperti berikut ini:

Elemen untuk pindah baris: <br />

c. Elemen-elemen dalam XHTML MP harus tertulis dengan huruf kecil.

 hello world! , bukan

<P> hello world! </P>

d. Nilai dari setiap atribut pada XHTML MP harus ditulis dalam tanda petik.

 ini paragraph 1 , atau

 ini paragraph 1 , bukan

 ini paragraph 1 

e. XHTML MP tidak mengijinkan adanya minimasi atribut.

Contoh:

<input type=”radio” value=”1” checked=”checked”> XHTML MP 1 <br />

<input type=”radio” value=”2”> XHTML MP 2 <br />, bukan

<input type=”radio” value=”1” checked> XHTML MP 1 <br />

<input type=”radio” value=”2”> XHTML MP 2 <br />

XHTML MP adalah bahasa pemrograman untuk membuat desain halaman web dan merupakan penyempurnaan bahasa HTML. Tujuan XHTML MP adalah agar tampilan bisa lebih konsisten disemua browser. Dokumen XHTML MP dapat divalidasi dengan mudah Pada HTML tidak semua browser mensupport sintaks HTML yang sama. HTML merupakan tag untuk memformat tampilan atau menitikberatkan pada unsur presentasi.

### 2.4.4 WCSS

WCSS (WAP *Cascading Style Sheet*) adalah versi *mobile* dari CSS. Ini adalah sub bagian dari CSS2 (bahasa CSS dari *World Wide Web*) ditambah beberapa ekstensi spesifik WAP. Fitur CSS2 dan kelengkapannya yang tidak banyak berguna untuk aplikasi *internet mobile* tidak termasuk dalam WAP CSS. WAP CSS adalah kerabat dari XHTML *Mobile Profile* (XHTML MP). Keduanya didefinisikan dalam spesifikasi WAP 2.0 yang dibuat oleh forum pembuat WAP (saat ini adalah *Open Mobile* Alliance atau OMA). Ada beberapa WAP 2.0 yang memungkinkan digunakan untuk telepon selular saat ini.

Ide dari CSS adalah sederhana, pola dan tampilan informasi dari suatu halaman didefinisikan dengan peraturan pola CSS dan ditempatkan terpisah dari isi halaman (contohnya: peraturan pola tersimpan dalam *file* eksternal). Untuk merubah tampilan dari suatu halaman pada suatu *browser*, hanya butuh merubah lembar pola CSS. Fitur ini sangat berguna dalam dunia tanpa kabel yang mana bermacam-macam peralatan *mobile* seperti telepon selular yang mempunyai karakteristik yang bervariasi seperti ukuran layar. Dengan WCSS dapat mengendalikan tampilan dari halaman yang ada pada peralatan *mobile* yang berbeda dengan menggunakan pemisah dari lembar pola CSS dan tidak harus merubah isi *file*.

Tipe MIME (*Multipurpose Internet Mail Extentions*) dan ekstensi *file* dari WCSS adalah teks/CSS dan ekstensi *file*nya adalah ".css" . Tipe ini sama dengan CSS *web*.

## 2.5 Apache Server

Untuk membuat sebuah pemrograman *web* dinamis, diperlukan sebuah *web server*. Ada banyak *web server* yang berkembang dan sering digunakan dalam membangun aplikasi berbasis *web*, salah satunya adalah Apache. Apache memiliki beberapa kelebihan antara lain adalah:

- *Free of Charge*, berarti tidak harus membayar lisensi kepada pembuat untuk menggunakannya.
- Dapat diakses (API ke berbagai *scripting language*) dan digabung dengan berbagai aplikasi lain (databaseserver, ssl, ext) dan sebagainya.
- Waktu pemrosesan lebih cepat dan tangguh denga konfigurasi yang benar.
- Dapat dilakukan setting dan instalasi sesuai dengan kebutuhan dengan adanya *modules* dan DSO-nya.
- Memiliki kemampuan *advanched setting* dan *configuration support.*

Dengan berbagai keunggulan tersebut, Apache sangat bagus jika dikombinasikan dengan aplikasi lainnya. Penggabungan yang paling sering adalah dengan menggabungkan Apache, PHP dan MySQL yang berjalan di *server* Linux atau yang terkenal dengan dengan LAMP (Linux, Apache, Mysql, PHP).

Sementara bagi pengguna windows Apache, PHP dan MySQL juga bisa diinstal pada OS tersebut.

## 2.6 HTTPS

HTTPS adalah versi aman dari HTTP, protokol komunikasi dari *World Wide Web*. Ditemukan oleh *Netscape Communications Corporation* untuk menyediakan autentikasi dan komunikasi tersandi dan penggunaan dalam komersi elektris.

Selain menggunakan komunikasi plain text, HTTPS menyandikan data sesi menggunakan protokol SSL (*Secure Socket layer*) atau protokol TLS (*Transport Layer Security*). Kedua protokol tersebut memberikan perlindungan yang memadai dari serangan eavesdroppers, dan man in the middle attacks. Pada umumnya port HTTPS adalah 443.

Tingkat keamanan tergantung pada ketepatan dalam mengimplementasikan pada browser web dan perangkat lunak server dan didukung oleh algorithma penyandian yang aktual.

Oleh karena itu, pada halaman web digunakan HTTPS, dan URL yang digunakan dimulai dengan 'https://' bukan dengan 'http://'

Kesalahpahaman yang sering terjadi pada pengguna kartu kredit di web ialah dengan menganggap HTTPS "sepenuhnya" melindungi transaksi mereka. Sedangkan pada kenyataannya, HTTPS hanya melakukan enkripsi informasi dari kartu mereka antara browser mereka dengan web server yang menerima informasi. Pada web server, informasi kartu mereke secara tipikal tersimpan di database server (terkadang tidak langsung dikirimkan ke pemroses kartu kredit), dan server database inilah yang paling sering menjadi sasaran penyerangan oleh pihak-pihak yang tidak berkepentingan. [ANO-05]

# BAB III

# METODE PENELITIAN

Penyusunan dan penelitian yang digunakan dalam penelitian tugas akhir ini didasarkan pada metode studi literatur tentang web basis data pada jaringan internet dan intranet. Langkah-langkah yang akan dilakukan untuk pembuatan perangkat lunak aplikasi ini antara lain:

## 3.6. Studi Literatur

Studi literatur yang dilakukan bertujuan untuk mengkaji landasan teori yang digunakan dalam mendukung setiap perencanaan dan implementasi aplikasi yang dibuat yaitu kajian pustaka mengenai:

a. Kamus
b. Basis data
c. Teori dasar *web*
*d. Apache Server*
e. HTTPS

Landasan teori yang digunakan dapat diperoleh dari sumber bacaan berupa *text book*, buku panduan pemrograman, tugas akhir, *paper*, tutorial pemrograman, dan sumber bacaan *softcopy* lain yang didapatkan dari *Internet*, maupun sumber-sumber bacaan yang lain.

## 3.7. Analisis dan Perancangan Sistem

Pada tahap ini akan dibuat seluruh *use case diagram* dan *use case specification* dari sistem. Kemudian beberapa *workflow* dari sistem aplikasi diubah menjadi *activity diagram* untuk mempermudah pemahaman proses kerja sistem aplikasi kamus online.

Selain *use case diagram* dan *activity diagram*, perlu dibuat ER diagram untuk memperjelas desain basis data yang benar.

*Class diagram*, *sequence diagram* dan *state diagram* kemudian dibuat dari setiap *use case*. Pembuatan *class diagram*, *sequence diagram* dan *state diagram* dibuat secara garis besar untuk membantu dalam fase implementasi sistem.

Berikut Perancangan dan Pembuatan Perangkat Lunak:

A. Perancangan Sistem

- Menentukan Aktor
- Merancang daftar kebutuhan Fungsional Sistem Informasi

- Pemodelan dalam *use case diagram*
- Spesifikasi use case
- Class diagram
- Sequence diagram

B. Perancangan dan Pembuatan Website
- Penentuan kontent website
- Perancangan Design
- Penyusunan coding

C. Perancangan dan Pembuatan Basis Data
- Perencanaan tabel awal
- Normalisasi
- Diagram ER

### 3.8. Implementasi

Implementasi adalah tahapan setelah proses analisis dan perancangan, yaitu mengimplementasikan hasil rancangan ke dalam kode-kode program dan algoritma sehinga dapat dijalankan sebagai program utuh untuk dapat dilakukan pengujian di tahap selajutnya. Pada aplikasi ini, bahasa pemrograman yang digunakan adalah PHP yang ternormalisasi ke dalam basis data MySQL.

Penyusunan *query* sql dan implementasi system yang dibuat meliputi:

1. Proses pada aplikasi kamus bahasa untuk administrator:
   - Proses melakukan setup tabel yang digunakan di dalam basis data aplikasi kamus *online*.
   - Proses manipulasi data (memasukkan, menghapus, mengubah) data di dalam basis data kamus online yang ternormalisasi.
2. Proses pada aplikasi kamus untuk user yaitu melakukan pencarian data berupa padanan kata yang dapat dilakukan empat arah di antaranya Jawa-Indonesia, Indonesia-Jawa, Indonesia-Madura dan Madura-Indonesia.

### 3.9. Pengujian Rancangan

Pengujian aplikasi dilakukan untuk mengetahui apakah aplikasi kamus *online* dapat digunakan untuk mencari padanan kata dalam bahasa daerah sesuai dengan kata kunci yang diberikan. Metode pengujian yang digunakan adalah metode pengujian *White Box* dan *Black box*. Pengujian dimulai dari pengujian

unit, kemudian dilanjutkan dengan pengujian integrasi, dan berakhir pada validasi atau pengujian sistem. Pengujian aplikasi dilakukan untuk mengetahui kesesusaian analisis kebutuhan dengan implementasi dari aplikasi yang dibuat, yaitu:

a. Pengujian klas-klas pembangun aplikasi dengan menggunakan metode *White Box* dengan teknik *Basis-Path.*
b. Pengujian terintegrasi hasil implementasi aplikasi dengan mengunakan metode *Scenario-Based.*

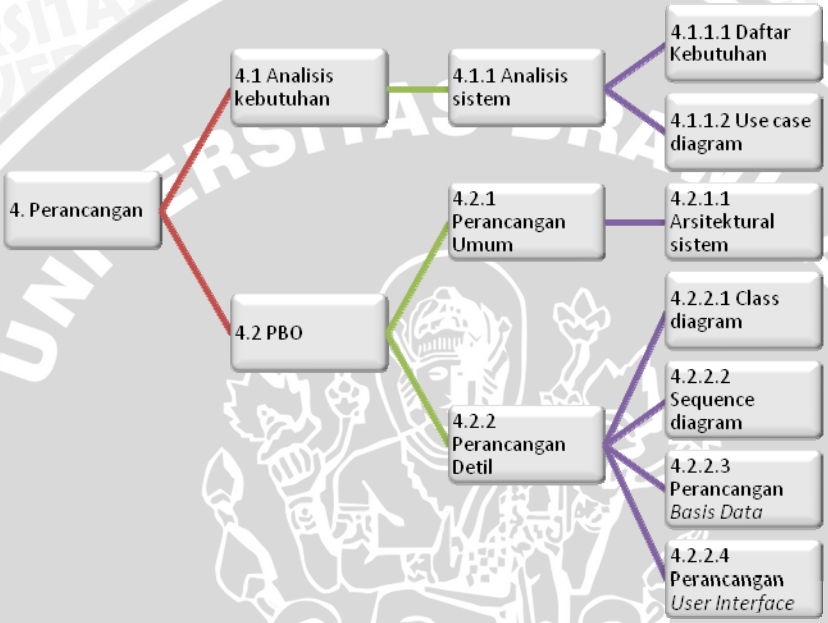
Pengujian validasi dilakukan untuk mengetahui validitas aplikasi berdasarkan kebutuhan fungsional dan non fungsional yang ditentukan dengan menggunakan metode *Black box*.

### 3.10. Pengambilan Kesimpulan dan Saran

Setelah mendapatkan hasil dan analisis dari pengujian perangkat lunak, maka langkah berikutnya adalah penarikan kesimpulan dari hasil pengujian dan memberi saran untuk pengembangan aplikasi lebih lanjut.

# BAB IV
# PERANCANGAN

Bab ini membahas analisis kebutuhan dan perancangan pada aplikasi kamus *online*. Tahap analisis kebutuhan menggunakan pemodelan *use case diagram*. Tahap perancangan aplikasi terdiri dari 5 tahap, yang berisi perancangan sistem, *class diagram*, *sequence diagram*, perancangan basis data dan perancangan antarmuka.




**Gambar 4.1** Diagram Pohon Perancangan
**Sumber :** [Perancangan]

## 4.1 Analisis Kebutuhan

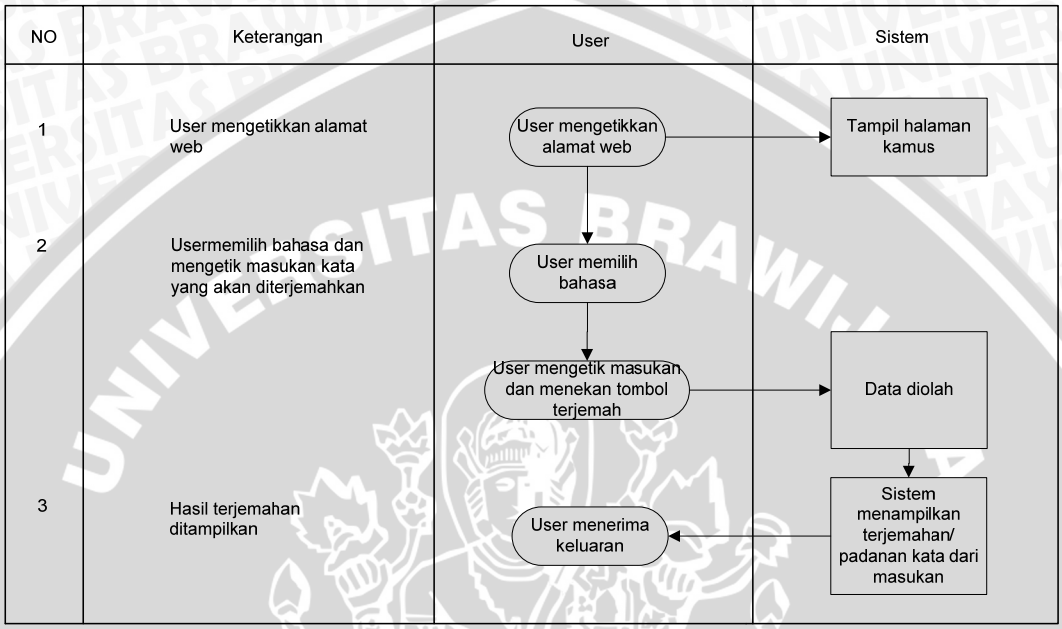
Analisis kebutuhan perangkat lunak adalah aktifitas rekayasa perangkat lunak yang menjembatani antara kebutuhan ditingkat sistem dan perancangan perangkat lunak. Digunakan untuk mendapatkan, menganalisis, dan memvalidasi kebutuhan-kebutuhan sistem.

### 4.1.1 Analisis Sistem

Pengembangan sebuah perangkat lunak bertujuan untuk menghasilkan perangkat lunak yang dapat memenuhi kebutuhan *user*. Setiap pengembangan sebuah sistem perangkat lunak memerlukan adanya dokumentasi terhadap kebutuhan-kebutuhan *user* agar tujuan tersebut tercapai. Tahap analisis berorientasi obyek dilakukan dengan membuat alur bisnis sistem yang kemudian diurutkan dalam sebuah

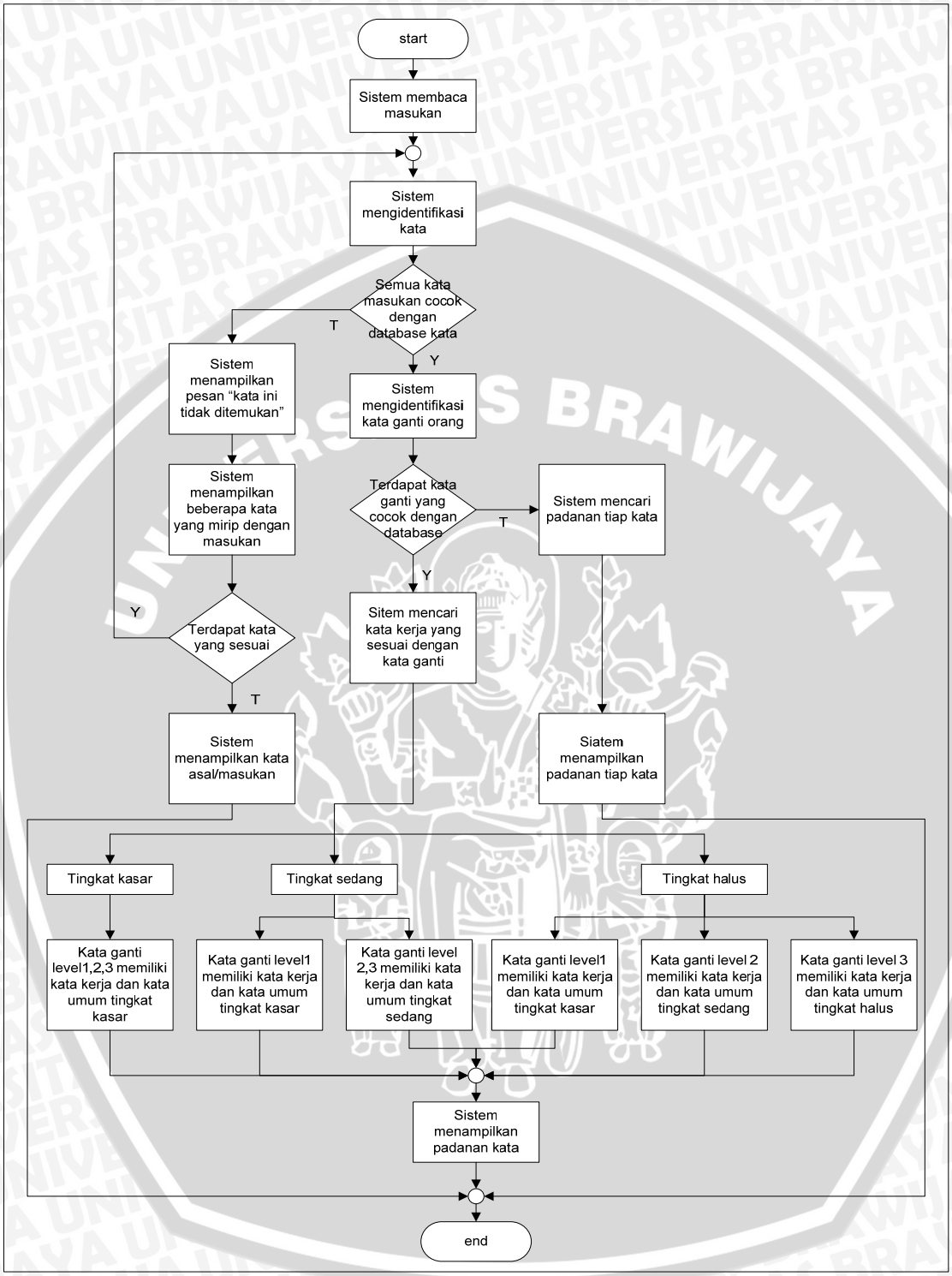
daftar kebutuhan. Tahap selanjutnya adalah memodelkan kebutuhan *user* ke dalam *use case diagram. Use case diagram* bertujuan untuk menggambarkan kebutuhan-kebutuhan fungsional yang harus disediakan oleh sistem agar dapat memecahkan permasalahan yang dihadapi *user*.

**Alur proses penggunaan sistem**




**Gambar 4.2** *Bussiness Process* untuk menjalankan proses sistem.
**Sumber:** [analisis]

## Alur Proses Pencarian Data




**Gambar 4.3** *Bussiness Process* untuk proses mengolahan data sistem.
**Sumber:** [analisis]

#### 4.1.1.1 Daftar Kebutuhan

Daftar kebutuhan merupakan daftar yang menguraikan kebutuhan-kebutuhan pengguna yang harus disediakan oleh perangkat lunak baik kebutuhan fungsional maupun non fungsional. Sebelum dapat membuat daftar kebutuhan, perlu dirumuskan terlebih dahulu aktor-aktor yang menggunakan sistem ini.

Tabel 4.1 dibawah ini akan memperlihatkan aktor-aktor beserta penjelasannya yang merupakan hasil dari proses identifikasi aktor.

Tabel 4.1 Deskripsi Aktor

| No | User | Penjelasan |
|---|---|---|
| 1. | User | Aktor yang menggunakan aplikasi kamus online tanpa proses *login* |
| 2. | Administrator | Aktor yang telah melakukan *login* pada sistem, bertugas untuk memanipulasi data dan memiliki hak-hak tertentu sesuai kapasitas dan keperluannya. Termasuk perbaikan sistem apabila terjadi gangguan pada sistem |

Sumber : [Analisis]

Daftar kebutuhan ini terdiri dari sebuah kolom yang menguraikan kebutuhan yang harus disediakan oleh sistem, dan pada kolom yang lain akan menunjukkan nama *use case* yang akan menyediakan fungsionalitas masing-masing kebutuhan tersebut. Daftar kebutuhan fungsional dan non fungsional sistem yang dikembangkan pada tugas akhir ini ditunjukkan pada Tabel berikut

Tabel 4.2. Daftar Kebutuhan fungsional

| No. | *Use Case* | Kebutuhan |
|---|---|---|
| 1. | *Login* administrator | Sistem harus memberikan fasilitas untuk *login* untuk administrator untuk memudahkan manajemen sistem. |
| 2. | *Logout* administrator | Sistem harus menyediakan fasilitas untuk *logout* agar administrator dapat keluar dari sistem. |
| 3. | Melihat data kamus | Sistem harus menyediakan fasilitas untuk |
| 4. | Menambah data kamus | Sistem harus menyediakan fasilitas untuk menambah referensi kata pada kamus online |
| 5. | Menghapus data kamus | Sistem harus menyediakan fasilitas untuk menghapus data |
| 6. | Mengubah data kamus | Sistem harus menyediakan fasilitas untuk mengubah data |
| 7. | Mencari padanan kata | Sistem harus menyediakan fasilitas untuk mencari padanan kata dan menyediakan pilihan pengarah bahasa |

**Sumber**: [analisis]

**Tabel 4.3** Daftar kebutuhan non fungsional

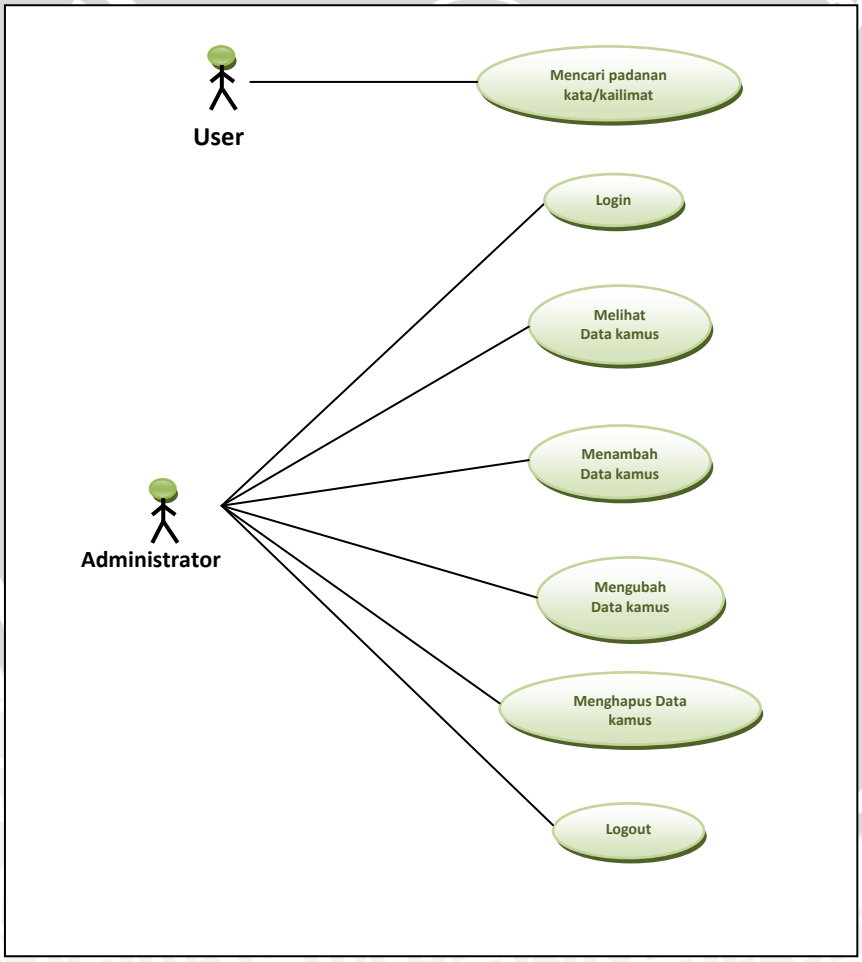
| No | Kebutuhan |
|---|---|
| 1. | Sistem dikembangkan dengan menggunakan bahasa PHP |
| 2. | Sistem harus dapat diakses dengan menggunakan *web browser* dan *mobile browser* |

**Sumber**: [analisis]

### 4.1.1.2 Use Case Diagram

*Use Case Diagram* adalah salah satu diagram dalam UML untuk memodelkan kebutuhan dari sistem. Masing-masing *use case diagram* menunjukkan sekumpulan *use case*, aktor dan hubungannya.

Gambar berikut adalah *use case diagram* untuk aplikasi kamu *online* yang melibatkan 2 aktor dan 7 *usecase*. Aktor yang terlibat dalam sistem antara lain administrator dan user.




**Gambar 4.4** *Use Case Diagram*.
**Sumber:** [analisis]

Berikut adalah spesifikasi dari *use case* aplikasi kamus *online*

1. Spesifikasi *use case* untuk *login administrator*

| **Nama use case** | *Login* |
|---|---|
| **Aktor** | *Administrator* |
| **Deskripsi** | Sistem harus mampu memberikan fasilitas untuk login administrator, sehingga hanya administrator saja yang berhak melakukan manajemen data dan manajemen web. |
| **Pra-kondisi** | Tampil halaman utama kamus |
| **Pasca-kondisi** | Kondisi benar |
| **Aliran Utama** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| 1. Menekan link login admin | 2. Tampil halaman login |
| 3. memasukkan *username* dan *password* kemudian menekan tombol *login* | 4. validasi *username* dan password |
| | 5. Menampilkan halaman administrator |
| **Aliran alternatif 1: *username* dan *password* salah** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| | 1. Menampilkan pesan bahwa *username* dan *password* salah |
| | 2. Kembali pada langkah no 1 aliran utama |
| **Aliran alternatif 2: *user*name atau *password* kosong** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| | 1. Menampilkan pesan bahwa field yang kosong tersebut harus diisi<br>2. Kembali pada langkah no 1 aliran utama |

2. Spesifikasi *use case* untuk *logout administrator*

| **Nama use case** | *Logout* |
|---|---|
| **Aktor** | *Administrator* |
| **Deskripsi** | Sistem harus menyediakan fasilitas untuk *logout* agar aktor yang telah login dapat keluar dari sistem. |
| **Pra-kondisi** | *administrator* telah *login* |
| **Pasca-kondisi** | Tampil halaman *login* |
| **Aliran Utama** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| 1. menekan tombol *logout* | 2. Menampilkan halaman login |
| | |

3. Spesifikasi *use case* untuk melihat data kamus

| **Nama *use case*** | Melihat data kamus |
|---|---|
| **Aktor** | *Administrator* |
| **Deskripsi** | Sistem harus menyediakan fasilitas untuk menampilkan semua data dalam sistem. |
| **Pra-kondisi** | Tampil halaman administrator |
| **Pasca-kondisi** | Sistem menampilkan data kamus |
| **Aliran Utama** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| 1. Memilih menu "data" | 2. Menampilkan isi menu "data" yang telah ada dalam sistem. Sistem juga menampilkan tombol tambah data, ubah, hapus. |

4. Spesifikasi *use case* untuk menambah data kamus

| **Nama *use case*** | Menambah data kamus |
|---|---|
| **Aktor** | *Administrator* |
| **Deskripsi** | Sistem harus menyediakan fasilitas untuk menambah data kamus |
| **Pra-kondisi** | *Administrator* telah menjalankan *use case* tampil data |
| **Pasca-kondisi** | Data kamus ditambahkan ke dalam sistem |
| **Aliran Utama** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| 1. *Administrator* menekan tombol tambah data | 2. Menampilkan form masukan yang berupa field kata. |
| 3. *Administrator* mengisi form masukan | 4. Validasi masukan data dari *admininstrator.* (Form tidak boleh kosong dan *kata* tersebut belum ada dalam sistem) |
| | 5. Menampilkan pesan bahwa data baru telah ditambahkan |
| **Aliran alternatif 1: masukan data kosong** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| | 1. Menampilkan pesan bahwa field tersebut harus diisi. |
| | 2. Kembali ke langkah no 2 dari aliran utama. |
| **Aliran alternatif 2: masukan data telah terdaftar dalam sistem** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| | 1 Menampilkan pesan bahwa masukan data |

| | |
|---|---|
| | *kata* tersebut telah terdaftar. |
| | 2 Kembali ke langkah no 2 dari aliran utama. |
| **Aturan khusus** | |
| 1. Masukan *kata* tidak boleh kosong. | |

5. Spesifikasi *use case* untuk menghapus data kamus

| | |
|---|---|
| **Nama *use case*** | Menghapus data *kamus* |
| **Aktor** | *Administrator* |
| **Deskripsi** | Sistem harus menyediakan fasilitas untuk menghapus data kamus. |
| **Pra-kondisi** | *Administrator* telah menajalankan *use case* "tampil data" |
| **Pasca-kondisi** | Data kamus dihapus dari sistem |
| **Aliran Utama** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| 1. Memilih satu baris data pada baris data daftar kata dari hasil menjalankan *use case* "melihat data kamus" kemudian menekan tombol "hapus" | 2. Data dihapus dari sistem kemudian sistem me-*refresh* daftar kata. |
| | 3. Menampilkan pesan "satu kata telah berhasil dihapus" |

6. Spesifikasi *use case* untuk mengubah data kamus

| | |
|---|---|
| **Nama *use case*** | Mengedit data *kamus* |
| **Aktor** | *Administrator* |
| **Deskripsi** | Sistem harus menyediakan fasilitas untuk mengedit data kamus. |
| **Pra-kondisi** | *Administrator* telah menajalankan *use case* "tampil data" |
| **Pasca-kondisi** | Data yang telah diedit disimpan ke dalam sistem |
| **Aliran Utama** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| 1. Memilih satu baris data pada baris data daftar kata dan dari hasil menjalankan *use case* "melihat data kamus" kemudian menekan tombol "ubah" | 2. Menampilkan field kata dari baris data yang dipilih oleh *administrator.* |
| *3.* Mengedit field kata. | 4. Validasi masukan data, (field *kata* tersebut belum ada dalam sistem) |
| | 5. Menampilkan pesan "satu data telah berhasil diubah". |

| **Aliran alternatif 1: masukan data kosong** | |
|---|---|
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| | 1. Menampilkan pesan bahwa field tersebut harus diisi. |
| | 2. Kembali ke langkah no 2 dari aliran utama |
| **Aliran alternatif 2: masukan data telah terdaftar dalam sistem** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| | 1. Menampilkan pesan bahwa masukan kata tersebut telah terdaftar. |
| | 2. Kembali ke langkah no 2 dari aliran utama. |
| **Aturan khusus**<br>1. Masukan kata tidak boleh kosong | |

7. Spesifikasi *use case* untuk mencari padanan kata/kalimat

| **Nama *use case*** | Mencari padanan kata/kalimat |
|---|---|
| **Aktor** | *user* |
| **Deskripsi** | Sistem harus menyediakan fasilitas untuk mencari padanan kata/kalimat. |
| **Pra-kondisi** | *User* telah membuka halaman awal sistem yang berisi form masukan |
| **Pasca-kondisi** | Padanan kata/kalimat yang dicari ditampilkan oleh sistem |
| **Aliran Utama** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| 1. Mengisi form masukan, menentukan arah bahasa dan menekan tombol cari | 2. Menampilkan field pencarian dimana pencarian data dilakukan berdasarkan kata masukan yang diketik user |
| **Aliran alternatif 1: masukan data *tidak terdapat dalam database sistem*** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| | 1. Melakukan pengecekan dan menampilkan kata yang mirip dengan masukan |
| | 2. Menampilkan padanan kata dengan menampilkan ulang kata yang tidak terdapat dalam database sistem |
| **Aliran alternatif 2: masukan data *kosong*** | |
| **Aksi dari aktor** | **Tanggapan dari sistem** |
| | 1. Menampilkan pesan bahwa masukan kata tidak boleh kosong. |
| | 2. Kembali ke halaman awal. |

## 4.3 Perancangan Berorientasi Obyek

Perancangan berorientasi obyek (*Object Oriented Design/OOD)* berhubungan dengan pengembangan model berorientasi objek dari sistem perangkat lunak untuk mengimplementasikan kebutuhan-kebutuhan yang telah teridentifikasi pada analisis kebutuhan (*Object Oriented Analysis*/*OOA*).

Perancangan umum menggambarkan relasi klas (*class)* sebagai pemodelan sistem secara keseluruhan. Perancangan detail menggunakan *class diagram*, *sequence diagram* sebagai pemodelan perangkat lunak. Perancangan detail melakukan perancangan terhadap pola hubungan antar komponen-komponen detail (dalam konteks berorientasi objek adalah klas dan objek) sehingga mampu membentuk sebuah fungsi kerja yang mampu memberikan pelayanan terhadap kebutuhan aktor

### 4.2.1 Perancangan Umum

Perancangan umum menggambarkan relasi antar paket (*package)* dan klas (*class)* sebagai pemodelan sistem secara keseluruhan. Perancangan umum memberikan pandangan keseluruhan sistem tanpa melihat detil dari masing-masing subsistem yang ada.

#### 4.2.1.1 Arsitektural Sistem

Suatu perangkat lunak dapat dibuat menjadi banyak *layer* atau lapisan. Pembagian sebuah perangkat lunak menjadi banyak *layer* menyebabkan kompleksitas sistem dapat dikurangi. Suatu sistem yang dibuat dengan banyak *layer*, maka setiap *layer* akan memiliki sebuah tanggung jawab yang utuh.

Aplikasi Kamus Madura – Indonesia –Jawa online ini dibagi menjadi 3 buah *layer*, yaitu :

1. *Presentation layer*
2. *Bussiness logic layer*
3. *Data source layer*

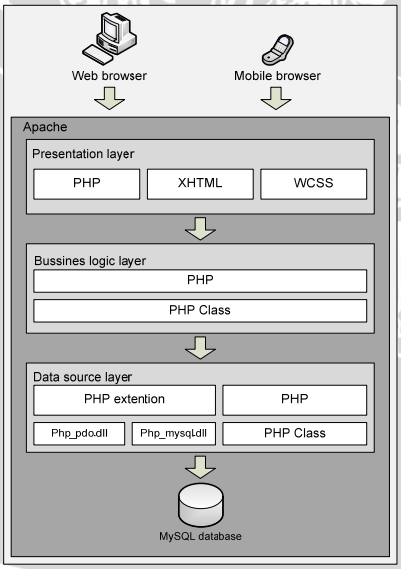
*Presentation layer* adalah suatu lapisan yang bertanggung jawab untuk memproses masukan dari aktor atau menampilkan sesuatu pada aktor. *Presentation layer* pada Aplikasi Kamus Madura–Indonesia–Jawa *online* menggunakan xHTML,WCSS dan PHP.

*Business logic layer* adalah suatu lapisan letak logika-logika dari sistem (bisnis proses). *Business logic layer* pada Aplikasi Kamus Madura – Indonesia –Jawa online menggunakan PHP.

*Data source layer* adalah suatu lapisan yang bertanggung jawab untuk berkomunikasi dengan database. *Data source layer* pada Aplikasi Kamus Madura – Indonesia –Jawa online menggunakan PHP.

Aktor yang mengakses sistem ini menggunakan *web browser* atau *mobile browser*. *Browser* akan mengirimkan *request* pada *server* dengan menampilkan sesuatu dalam format xHTML. Tanggapan *server* ini dihasilkan oleh *presentation layer*. *Presentation layer* dalam memproses *request* dari aktor akan menggunakan *business logic layer* untuk memproses bisnis proses yang bersesuaian dengan *request* aktor. *Business logic layer* akan menggunakan *data source layer* bila *busniness logic layer* membutuhkan suatu data yang berhubungan dengan database.

Gambar berikut menjelaskan arsitektur Sistem Aplikasi Kamus Madura – Indonesia – Jawa online yang terdiri dari tiga *layer*. Ketiga *layer* tersebut adalah *presentation layer*, *business logic layer* dan *data access layer*. Ketiga *layer* tersebut berada dalam *web server* Apache.



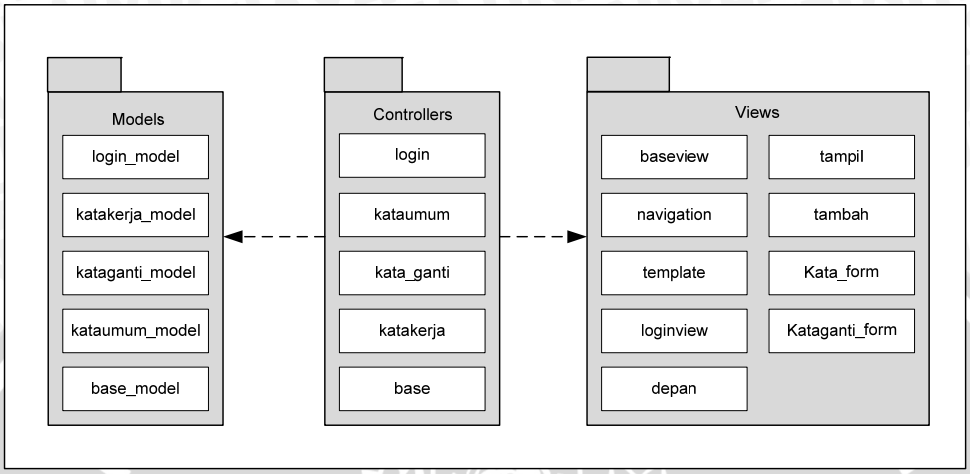

**Gambar 4.5** Arsitkertur Sistem.
**Sumber:** [perancangan]

### 4.2.2 Perancangan Detil

Pada perancangan detil dilakukan spesifikasi dari tipe-tipe atribut, bagaimana fungsi-fungsi beroperasi, dan bagaimana suatu obyek berhubungan dengan obyek lainnya. Perancangan detil dimodelkan dengan *class diagram* dan *sequence diagram*.

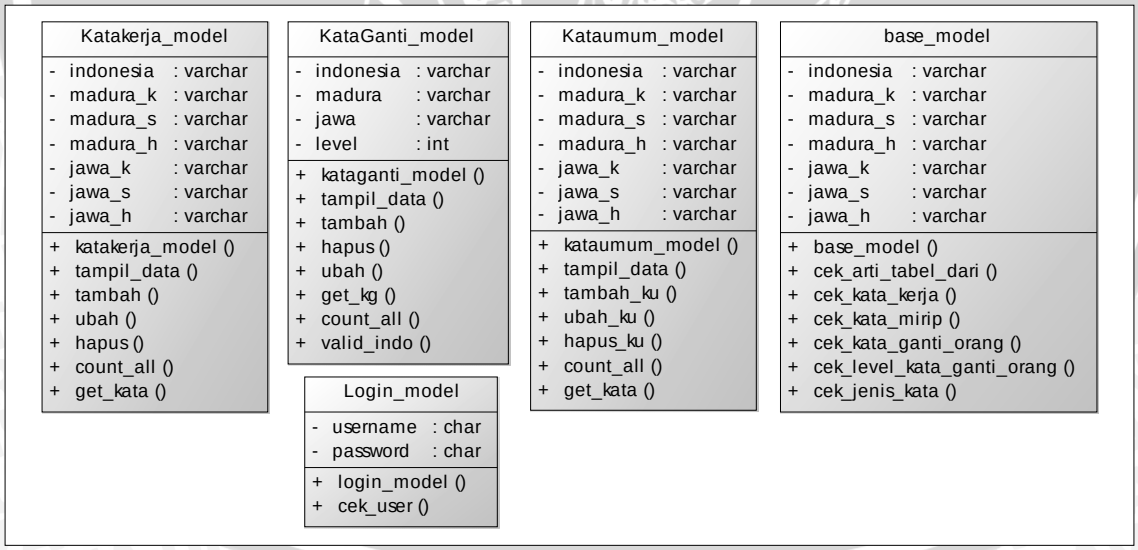
### 4.2.2.1 *Class Diagram*

Kelas adalah kumpulan dari obyek yang berbagi atribut, operasi relasi dan semantik yang sama. *Class diagram* adalah representasi berbentuk diagram dari kelas-kelas yang membentuk perangkat lunak [BOR-05].

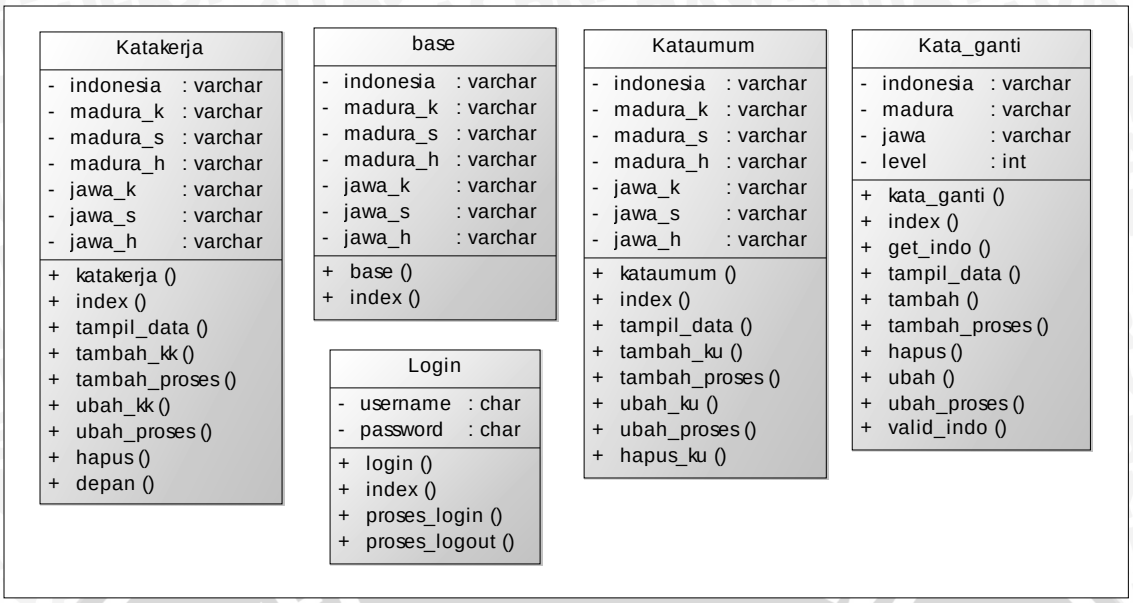



Gambar 4.6. *Class Diagram*
Sumber : perancangan

*Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain. [DHA-03]




**Gambar 4. 7** *Class-class diagram* dari *models*
**Sumber:** [Perancangan]



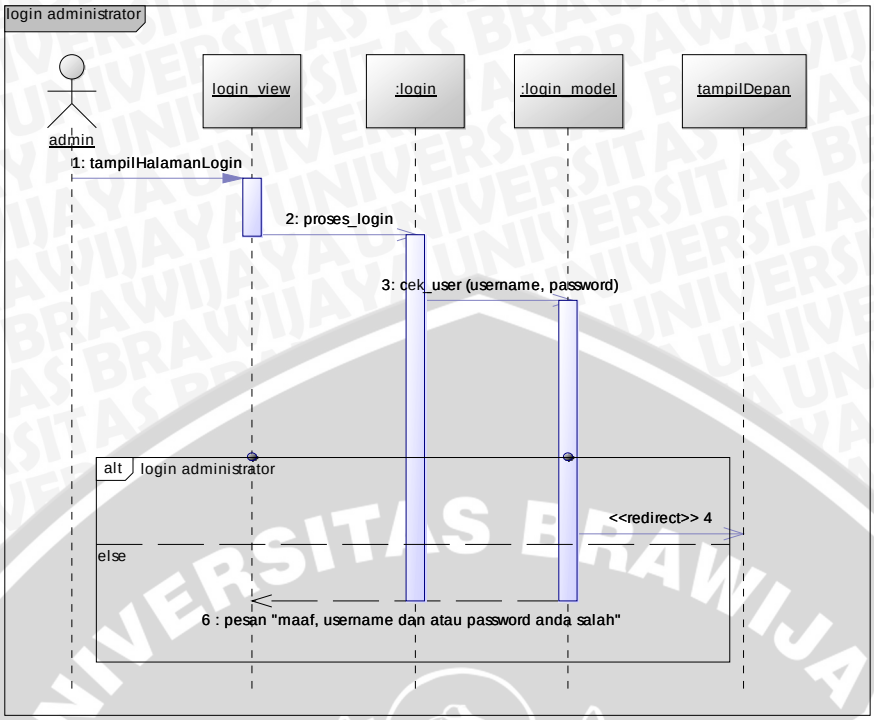

**Gambar 4. 8** *Class-class diagram* dari *controller*
**Sumber:** [Perancangan]

### 4.2.2.2 Sequence Diagram

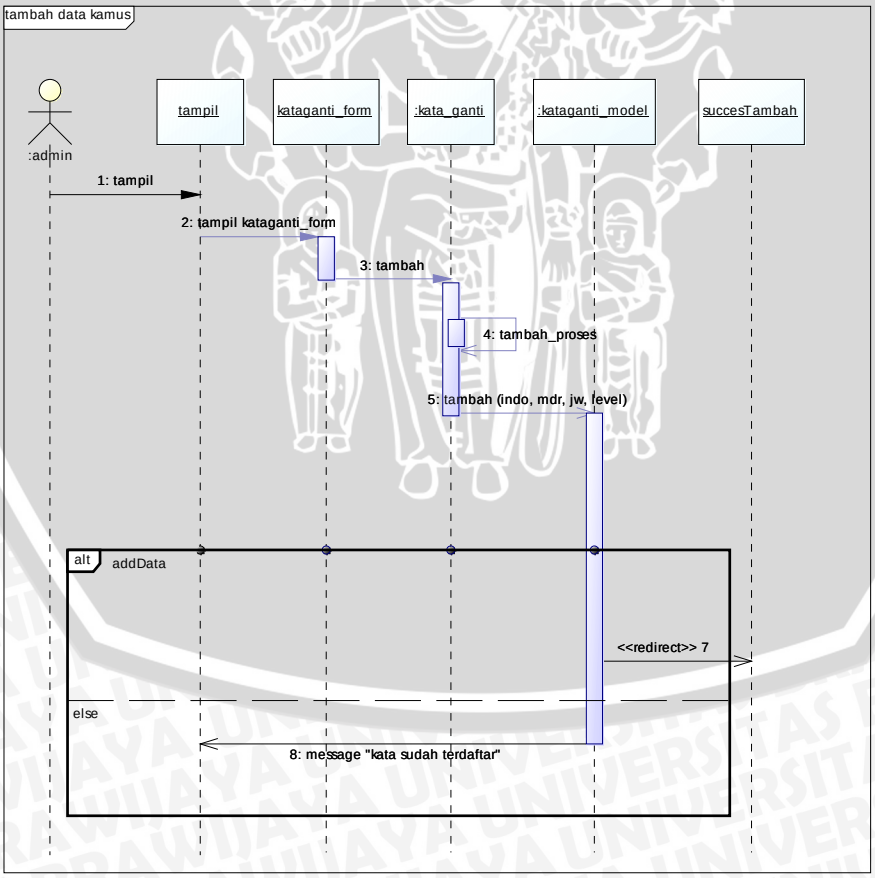
Hubungan antar klas yang digambarkan pada diagram klas termasuk dalam pemodelan statis, tanpa disertai dengan gambaran aliran proses atau data antarklas yang satu dengan klas yang lain. *Sequence diagram* berguna untuk menampikan aliran jalannya proses yang ditunjukkan dengan interaksi antar objek atau klas, dan disusun berdasarkan urutan waktu. *Sequence diagram* dirancang dengan mengambil acuan pada *use case* serta operasi – operasi dalam klas yang menjadi implementasi dari fungsionalitas yang digambarkan pada *use case* tersebut.

Sub bab 4.2.2.2 ini tidak memodelkan keseluruhan *sequence diagram* untuk tiap *use case*, akan tetapi diambil beberapa contoh *sequence diagram* untuk *use case* tertentu. Sub bab 4.2.2.2 ini memodelkan *sequence diagram* untuk use case berikut :

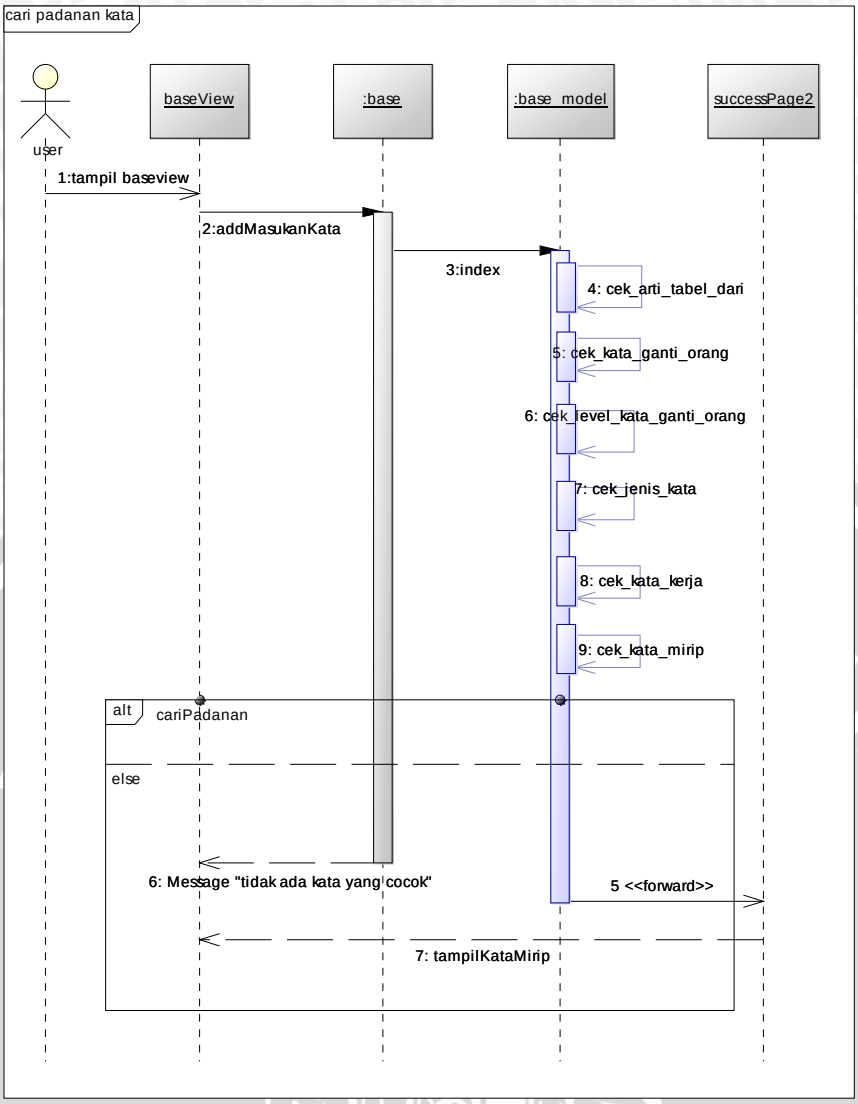
1. *Use case* Login administrator
2. *Use case* menambah data kamus
3. *Use case* mencari padanan kata




**Gambar 4.9** Sequence Diagram untuk use case login administrator.
**Sumber:** [perancangan]




**Gambar 4.10** Sequence Diagram untuk use case tambah data kamus.
**Sumber:** [perancangan]




**Gambar 4.11** Sequence Diagram untuk use case cari padanan kata.
**Sumber:** [perancangan]

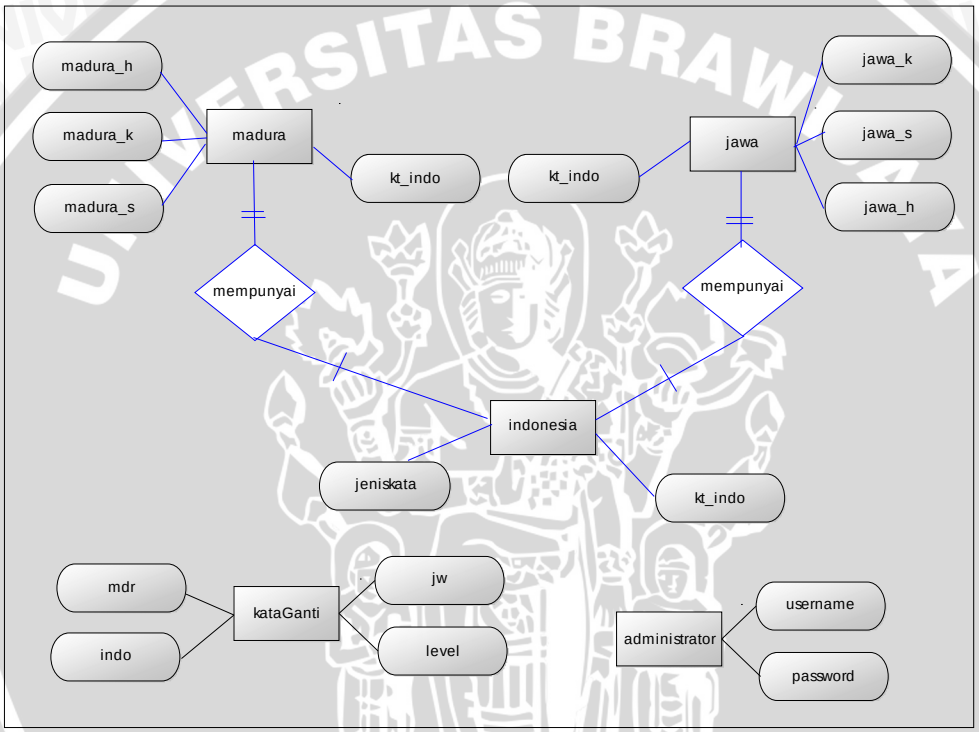
### 4.2.2.3 Basis Data

Sistem Informasi yang akan dikembangkan ini memerlukan suatu basis data yang berfungsi untuk menampung data-data yang terkait. Perancangan basis data dilakukan agar basis data dapat efisien dalam penggunaan ruang penyimpanan, cepat dalam pengaksesan, dan mudah dalam pemanipulasian data. Perancangan basis data dapat dilakukan dengan menggambarkan diagram hubungan antar entitas atau biasa dikenal dengan *Entity –Relationship Diagram* (ER-Diagram), dan normalisasi. Normalisasi merupakan teknik dalam mengelompokkan atribut dari suatu relasi sehingga terbentuk struktur yang baik. Dalam ER-Diagram akan tampak relasi antar entitas yang saling terkait.

#### 4.2.2.3.1 Diagram Entitas Relasional

Diagram Entitas Relasional ini digunakan untuk menggambarkan hubungan entitas satu dengan lainnya dengan memperlihatkan hubungan antar *key* untuk berelasi antar tabel. Diagram Entitas Relasional dari aplikasi kamus *online* digambarkan pada gambar 4.12.

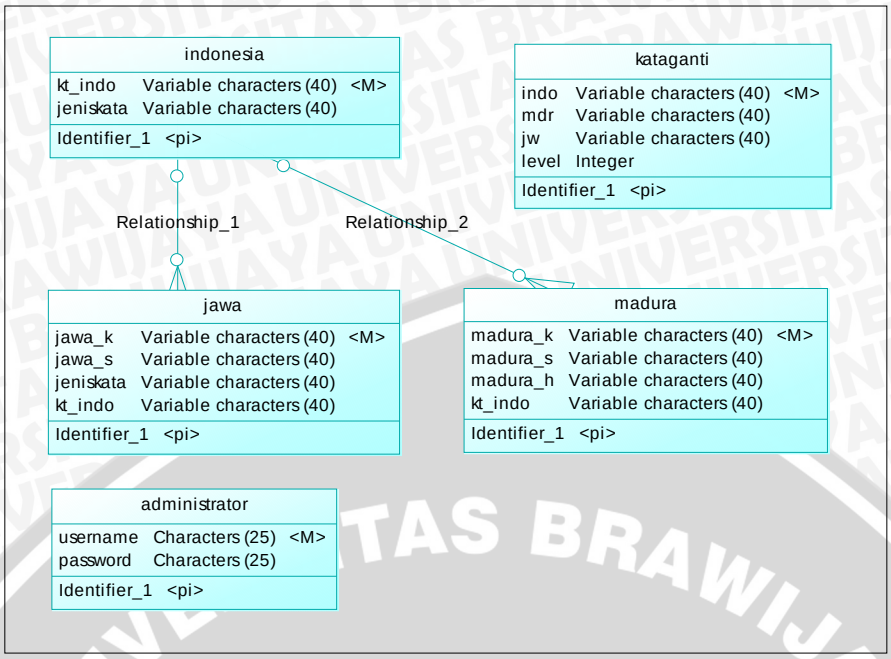
Diagram Relasi Antar Tabel disajikan untuk menampilkan relasi tabel-tabel yang digunakan dalam sistem ini. Diagram Relasi Antar Tabel digambarkan melalui *Conceptual Data Model* pada gambar 4.13 dan *Physical Data Model* pada gambar 4.14




**Gambar 4.12** *ER* Diagram [Perancangan]

#### 4.2.2.3.2 *Conceptual Data Model*

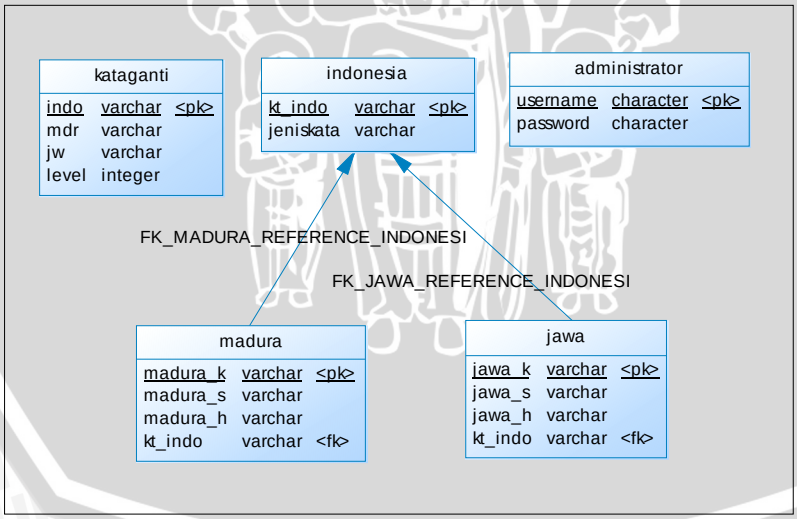
*Conceptual Data Model* menggambarkan struktur basis data secara detail dalam bentuk logika dan terdiri dari objek yang belum diimplementasikan secara langsung.




**Gambar 4.13** *Conceptual Data Model* dari Aplikasi kamus online [Perancangan]

### 4.2.2.3.3 *Physical Data Model*

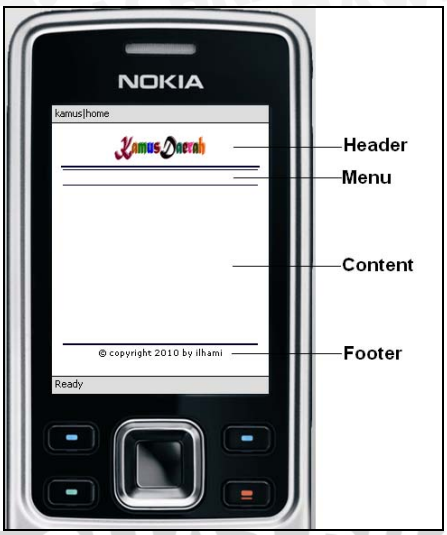
*Physical Data Model* menggambarkan basis data secara detail dalam bentuk fisik dan memperlihatkan struktur penyimpanan data yang digunakan sesungguhnya.



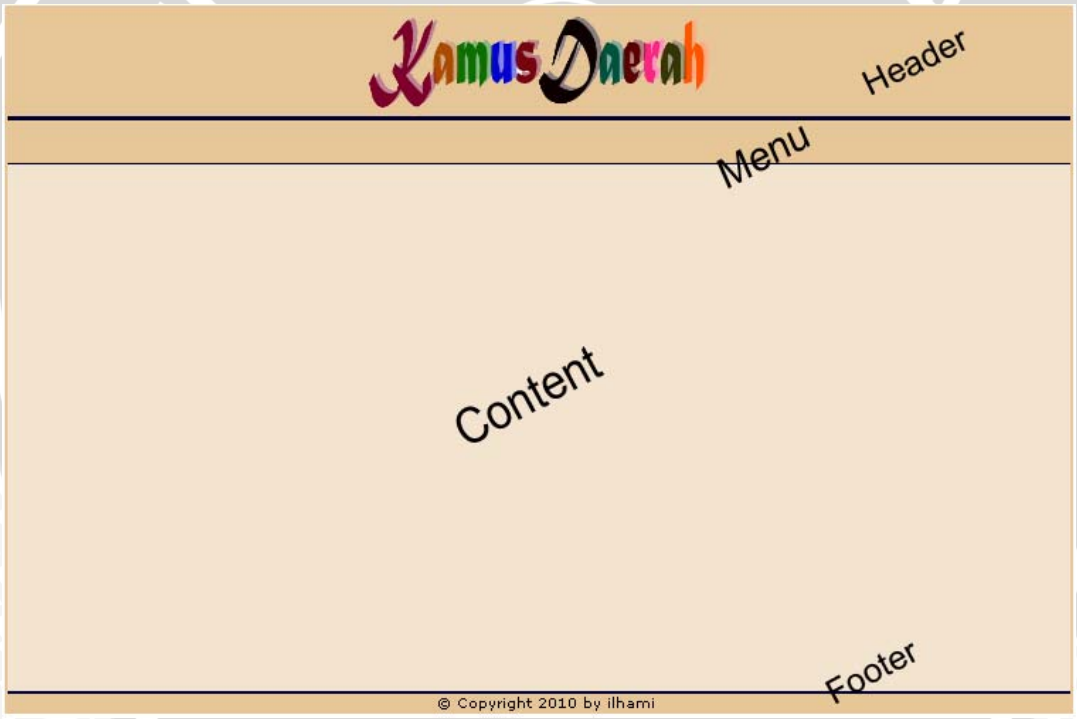

**Gambar 4.14** *Physical Data Model* dari Aplikasi Kamus *Online* [Perancangan]

### 4.2.2.4 Perancangan Antarmuka

Pada sistem kamus online ini, halaman antarmuka terdiri dari beberapa komponen, yaitu: header, menu, content, dan footer. Rancangan dari halaman antarmuka untuk konsumen melalui mobile browser dapat dilihat pada Gambar 415, dan rancangan dari halaman antarmuka untuk administrator ditunjukkan pada Gambar 4.16.



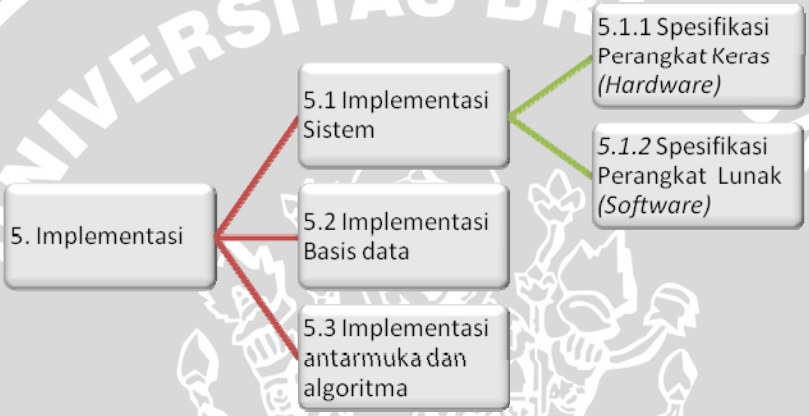

**Gambar 4. 15** Rancangan antarmuka konsumen
**Sumber:** [Perancangan]




**Gambar 4. 16** Rancangan antarmuka administrator
**Sumber:** [Perancangan]

# BAB V
# IMPLEMENTASI

Bab ini membahas tentang implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak sebelumnya. Pembahasan terdiri dari penjelasan implementasi perancangan basis data, dan implementasi perancangan perangkat lunak Aplikasi Kamus Bahasa Madura-Indonesia-Jawa *Online*. Diagram alir implementasi perangkat lunak ditunjukkan dalam gambar 5.1 berikut.




**Gambar 5.1** Diagram Alir Implementasi
**Sumber**: [Implementasi]

## 5.1. Implementasi Sistem

Agar bisa berjalan sesuai dengan kebutuhan sistem Kamus Bahasa Madura-Indonesia-Jawa *Online*, sistem diimplementasikan pada perangkat keras dan perangkat lunak dengan spesifikasi tertentu.

### 5.1.1. Spesifikasi Perangkat Keras

Sistem Kamus Bahasa Madura-Indonesia-Jawa *Online* ini merupakan aplikasi *client-server*. Pada skripsi ini baik *client* maupun *server* terdapat pada satu komputer yang sama. Pengembangan sistem Kamus Bahasa Madura-Indonesia-Jawa *Online* menggunakan komputer dengan spesifikasi sebagai berikut:

**Tabel 5. 3** Spesifikasi perangkat keras

| | |
|---|---|
| Prosesor | Intel Core Duo T4200 |
| Memori (RAM) | 1 GB DDR3 |
| Harddisk | 250 GB |
| VGA card | Intel GMA 4500M |

**Sumber:** [Implementasi]

### 5.1.2. Spesifikasi Perangkat Lunak

Pengembangan aplikasi Kamus Bahasa Madura-Indonesia-Jawa *Online* menggunakan perangkat lunak dengan spesifikasi sebagai berikut:

**Tabel 5. 4** Spesifikasi perangkat lunak

| | |
|---|---|
| Sistem operasi | Windows XP Professional Service Pack 3 |
| Bahasa pemrograman | XHTML MP<br>WCSS<br>PHP 5.2.5 |
| Web server | Apache 2.2.8 |
| Database | MySQL 5.0.51a |
| IDE (*Integrated Development Environment*) | Macromedia Dreamweaver CS3, Nokia series 40 5th edition SDK emulator |

**Sumber:** [Implementasi]

### 5.1.3. Perkiraan Jumlah Kata yang Dapat Ditampung *Database*.

Tabel 5.6 berikut menunjukkan perkiraan data yang dapat dimuat oleh database kamusdb.

**Tabel 5.5** Perkiraan jumlah data

| Keterangan | Size | Dalam byte |
|---|---|---|
| Kapasitas HD yang terbaca | 232 GB | 249108103168 |
| Kebutuhan Drive C | | |
| ➔ untuk OS dan program aplikasi | 20 GB | 21474836480 |
| Kebutuhan Drive D | 212 GB | 227633266688 |
| ➔ Instalasi xampp version 1.6.6a | 206 MB | 216006656 |
| ➔ Space untuk data | | 227417260032 |
| Kapasitas file untuk tipe data varchar = L + 1 byte | | 41 |
| **Data yang bisa ditampung mysql** | | 5546762440 |

Sumber : [implementasi]

Table perkiraan jumlah data di atas berlaku jika memenuhi ketentuan sebagai berikut:

1. Harddisk komputer hanya terbagi menjadi dua partisi dengan masing-masing partisi drive C = ±20 GB dan drive D = ±212 GB
2. Drive D hanya digunakan untuk penyimpan data dan hanya terinstall xampp versi 1.6.6a.
3. Head pada masing-masing partisi diabaikan.
4. Semua data yang dimasukkaa dalam *database* dianggap memiliki 40 karakter.
5. Tidak mempertimbangkan kecepatan akses data dan hanya untuk tujuan mengetahui jumlah kata yang dapat dimuat database.

## 5.2 Implementasi Perancangan Basis Data

Aplikasi ini dirancang untuk dapat terhubung ke *server* basis data MySQL. Implementasi perancangan basis data `kamusdb` dilakukan sesuai dengan *Entity Relationship Diagram*. Implementasi perancangan basis data menggunakan *query* SQL. *Query* SQL digunakan untuk mengimplementasikan rancangan basis data ke dalam sistem basis data MySQL.

*Query* SQL yang digunakan dalam membentuk basis data `kamusdb` ditunjukkan pada Gambar 5.2.

```
CREATE DATABASE kamusdb;
```

Gambar 5.2 *Query* untuk membuat basis data `kamusdb`
**Sumber**: Implementasi

*Query* SQL yang digunakan dalam membentuk tabel `madura` ditunjukkan pada Gambar 5.3.

```
CREATE TABLE IF NOT EXISTS madura (
  kt_indo varchar(40) NOT NULL,
  madura_k varchar(40) NOT NULL,
  madura_s varchar(40) NOT NULL,
  madura_h varchar(40) NOT NULL,
  PRIMARY KEY  (Madura_k),
  );
```

Gambar 5.3 *Query* untuk membuat tabel `madura`
**Sumber**: Implementasi

*Query* SQL yang digunakan dalam membentuk tabel `jawa` ditunjukkan pada Gambar 5.4.

```
CREATE TABLE IF NOT EXISTS jawa (
  kd_indo varchar(40) NOT NULL,
  jawa_k varchar(40) NOT NULL,
  jawa_s varchar(40) NOT NULL,
  jawa_h varchar(40) NOT NULL,
  PRIMARY KEY  (jawa_k),
);
```

Gambar 5.4 *Query* untuk membuat tabel `jawa`
**Sumber**: Implementasi

*Query* SQL yang digunakan dalam membentuk tabel `kataganti` ditunjukkan pada Gambar 5.5.

```
CREATE TABLE IF NOT EXISTS kataganti (
  indo varchar(40) NOT NULL,
  jw varchar(40) NOT NULL,
  mdr varchar(40) NOT NULL,
  level varchar(1) NOT NULL,
  PRIMARY KEY  (indo)
);
```

Gambar 5.5 *Query* untuk membuat tabel `kataganti`
**Sumber**: Implementasi

*Query* SQL yang digunakan dalam membentuk tabel `indonesia` ditunjukkan pada Gambar 5.6.

```
CREATE TABLE IF NOT EXISTS indonesia (
  kt_indo varchar(40) NOT NULL,
  jeniskata varchar(40) NOT NULL,
  PRIMARY KEY  (kt_indo)
);
```

Gambar 5.6 *Query* untuk membuat tabel `indonesia`
**Sumber**: Implementasi

*Query* SQL yang digunakan dalam membentuk tabel `administrator` ditunjukkan pada Gambar 5.7.

```
CREATE TABLE administrator (
  username char(25) not null,
  password char(25) not null,
  PRIMARY KEY(username),
  );
```

Gambar 5.7 *Query* untuk membuat tabel `administrator`
**Sumber**: Implementasi

## 5.3 Implementasi Antarmuka dan Algoritma

Subbab implementasi antarmuka dan algoritma ini akan menjelaskan semua implementasi antarmuka yang diperoleh dari daftar kebutuhan fungsional kamus *online*.

### 5.3.1 Implementasi Antarmuka Kebutuhan Fungsional *Login*

Implementasi dilakukan dengan menyediakan dua buah *text field* untuk masing-masing *username* dan *password*, dan satu tombol *login*.

Pengguna akan *login* sesuai peran yang dimiliki, jika pengguna berhasil melakukan *login* maka *user* akan memiliki hak tertentu sesuai peran yang dimiliki.

Implementasi kebutuhan Fungsional *login* dapat dilihat pada gambar 5.8 dibawah ini.


**Gambar 5.8** Halaman untuk Melakukan Login
**Sumber**: [Implementasi]

Algoritma pada tombol "login" diatas ditunjukan secara sederhana melalui *pseudocode* berikut ini :

```
function process_login()

Declaration:
Username IS string
Password IS string
data IS Array

CALL form_validation->set_rules('username', 'Username', 'required');
CALL form_validation->set_rules('password', 'Password', 'required');

        if validation = TRUE
                username <- CALL input->post('username');
                password <- CALL input->post('password');
        if check_user = TRUE
                data <- array('username' => $username, 'login' => TRUE);
                CALL session->set_userdata($data);
                redirect('kata');
        else
                CALL session->set_flashdata('message', 'Maaf, username dan atau
                password Anda salah');
                redirect('login/index');
        else
                CALL load->view('login_view');

End Login
```

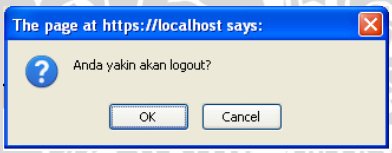
**Gambar 5. 9** *Pseudocode method* login()
**Sumber:** [Implementasi]

### 5.3.2 Implementasi Antarmuka Kebutuhan Fungsional *logout.*

Implementasi Kebutuhan Fungsional *logout* dilakukan dengan menyediakan satu buah tombol yang hanya muncul apabila telah melakukan login. Setelah melakukan *logout*, pengguna akan kembali ke halaman login.

Implementasi kebutuhan Fungsional *logout* dapat dilihat pada gambar 5.10 dan 5.11 dibawah ini.


**Gambar 5.10** implementasi antarmuka *Logout*
**Sumber**: [Implementasi]




**Gambar 5.11** implementasi antarmuka pesan *Logout*
**Sumber**: [Implementasi]

Algoritma fungsi *logout* ditunjukkan secara sederhana melalui pseudocode pada gambar 5.12 berikut:

```
function process_logout()

        CALL session->sess_destroy();
        redirect('login', 'refresh');

End procces_logout
```

**Gambar 5.12** *Pseudocode method* logout()
**Sumber:** [Implementasi]

### 5.3.3 Implementasi Antarmuka Tampil Data.

Halaman tampil data ini menampilkan data dari 3 jenis kata yang diklasifikasikan menjadi kata ganti, kata kerja dan kata umum. Gambar 5.13 dan

gambar 5.14 adalah implementasi antarmuka tampil_data saat tidak satu pun data yang ada dalam basis data `kamusdb`. Pada halaman ini, administrator bisa melakukan aktifitas tambah-data dengan memilih tombol tambah data.


**Gambar 5.13** implementasi antarmuka tampil_data untuk kata ganti
**Sumber**: [Implementasi]


**Gambar 5.14** implementasi antarmuka tampil_data
**Sumber**:[Implementasi]

Algoritma tampil_data ditunjukkan secara sederhana melalui pseudocode berikut:

```
function tampil_data()

Declaration:
data IS Array

        data['title'] <- "kamus | tampil Data";
        data['main_view'] <- 'tampil';

        uri_segment <- 3;
        offset <- CALL uri->segment($uri_segment);

        // Load data
        query <- CALL kata_model->tampil_data($this->limit , $offset)->result();
        num_rows <- CALL kata_model->count_all();

        if ($num_rows > 0)
```

```
        //set pagination
        config['base_url'] <- site_url('kata/tampil_data');
        config['total_rows'] <- $num_rows;
        config['per_page'] <- $this->limit;
        config['uri_segment'] <- $uri_segment;
        CALL <- pagination->initialize($config);
        data['pagination'] <- $this->pagination->create_links();

        // Table
        /*Set table template for alternating row 'zebra'*/
        tmpl <- array('table_open'    => '<table border="0" cellpadding="0"
        cellspacing="0" align="center">',
        'row_alt_start' => '<tr class="zebra">',
        'row_alt_end'   => '</tr>'
        );
        CALL table->set_template($tmpl);

        /*Set table heading */
        CALL table->set_empty(" ");
        CALL table->set_heading('no', 'indonesia', 'madura kasar', 'madura sedang',
        'madura halus', 'jawa kasar', 'jawa sedang', 'jawa halus', 'Actions');
        $i = 0 + $offset;
        foreach ($query as $row)
        CALL table->add_row(++$i, $row->indonesia, $row->madura_k, $row->madura_s,
        $row->madura_h, $row->jawa_k, $row->jawa_s, $row->jawa_h,
        anchor('kata/ubah/'.$row->indonesia,' ',array('class' => 'ubah')).' '.
        anchor('kata/hapus/'.$row->indonesia,' ',array('class'=>
        'hapus','onclick'=>"return confirm('Anda yakin akan menghapus data ini?')"))
                                                                        );

        data['table'] <- $this->table->generate();

        else
        data['message'] <- 'Tidak ditemukan satupun data!';
        data['link'] <- array('link_add' => anchor('kata/tambah/','tambah data',
        array('class' => 'tambah'))
                                                        );
        // Load view
        CALL load->view('template', $data);


End Tampil_data
```

**Gambar 5.16** pseudocode tampil_data
**Sumber**:[Implementasi]

### 5.3.4 Implementasi Antarmuka Tambah Data kata ganti

Halaman tambah data menyediakan fasilitas untuk administrator menambah data kamus. Data yang diisikan adalah Indonesia, Madura, jawa dan level. Gambar 6.3 menunjukkan tampilan halaman menambah data.


**Gambar 5.17** implementasi antarmuka tambah data
**Sumber**:[Implementasi]


**Gambar 5.18** implementasi antarmuka tambah data
**Sumber**:[Implementasi]

Algoritma untuk proses tambah data dijelaskan secara sederhana pada pseudocode berikut:

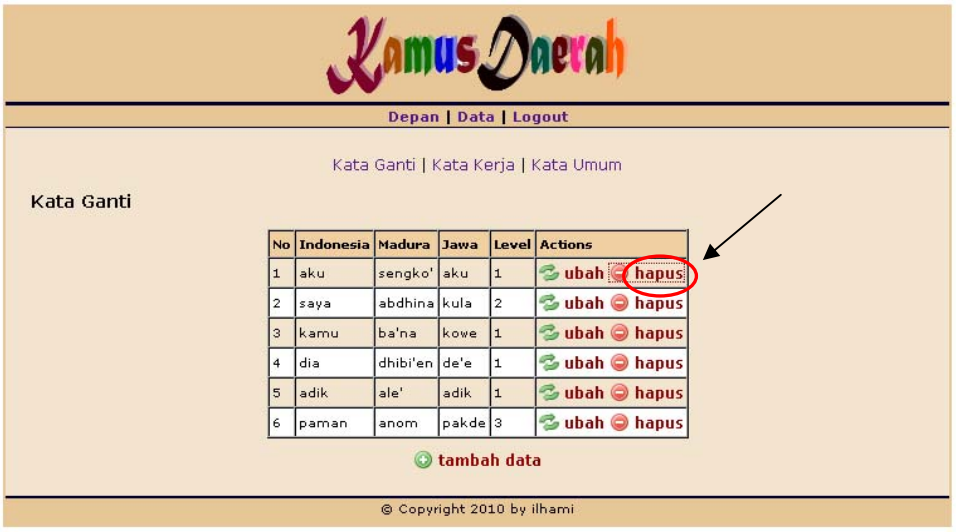
```
function tambah_proses()
Declaration:   data, tmpl  IS Array
               indo, mdr, jw IS varchar
               level IS int

       data['title']  <- 'Kamus | Kata Ganti';
       data['h2_title'] <-'Absen > Tambah Data';
       data['main_view'] <-'kata_ganti/kataganti_form';
       data['form_action']<- site_url('kata_ganti/tambah_proses');
       data['link']   <- array('link_back' =>
       anchor('kata_ganti/tampil_data/','kembali', array('class' => 'back')));
       CALL form_validation->set_rules('indo', 'indo', 'required');
       CALL form_validation->set_rules('mdr', 'mdr', 'required');
       CALL form_validation->set_rules('jw', 'jw', 'required');
       CALL form_validation->set_rules('level', 'level', 'required');
       if (CALL form_validation->run() == TRUE)
       tmpl <- array('indo'   => CALL input->post('indo'),
                     'mdr'   => CALL input->post('mdr'),
                     'jw'    => CALL input->post('jw'),
                     'level' => CALL input->post('level')
                                          );
       CALL kataganti_model->tambah($kg);
       CALL session->set_flashdata('message', 'Satu data absen berhasil
       disimpan!');
       redirect('kata_ganti/tambah');
       else
       CALL load->view('template', $data);
End Tambah_proses
```

**Gambar 5.19** pseudocode tambah_proses
**Sumber**:[Implementasi]

### 5.3.5 Implementasi Antarmuka Hapus Data.

Pada halaman tampil_data terdapat link hapus untuk menghapus data kamus dari database, gambar 5.20 menunjukkan link hapus. Saat admin menjalankan fungsi ini, akan muncul konfirmasi hapus yang ditunjukkan pada gambar 5.21 yang selanjutnya pesan bahwa data telah dihapus ditunjukkan pada gambar 5.22.




**Gambar 5.20** implementasi antarmuka *Logout*
**Sumber**: [Implementasi]


**Gambar 5.21** implementasi antarmuka *Logout*
**Sumber**: [Implementasi]


**Gambar 5.22** implementasi antarmuka *Logout*
**Sumber**: [Implementasi]

Algoritma hapus data dijelaskan secara sederhana dalam pseudocode berikut:

```
function hapus()

Declaration:
Indo IS varchar

              CALL kataganti_model->hapus($indo);
              CAL session->set_flashdata('message', '1 data berhasil dihapus');
              redirect('kata_ganti/tampil_data');

End Hapus
```

**Gambar 5.23** *Pseodocoe* hapus
**Sumber**: [Implementasi]

### 5.3.6 Implementasi Antarmuka Ubah Data.

Pada halaman tampil_data terdapat link ubah untuk mengubah data kamus dari database. Saat admin menjalankan fungsi ini, akan muncul konfirmasi ubah yang ditunjukkan pada gambar 5.24 yang selanjutnya pesan bahwa data telah dihapus ditunjukkan pada gambar 5.25.


**Gambar 5.24** implementasi antarmuka ubah
**Sumber**: [Implementasi]


**Gambar 5.25** implementasi antarmuka ubah
**Sumber**: [Implementasi]

Algoritma ubah data dijelaskan secara sederhana dalam pseudocode berikut:

```
function ubah_proses()

declaration:
tmpl, data IS ARRAY
indo, mdr, jw, level IS STRING


        // Inisialisasi data umum
        data['title']  <- 'Kamus | Kata Ganti';
        data['h2_title'] <- 'Kata Ganti > Update Data';
        data['main_view'] <- 'kataganti_form';
        data['form_action']<- site_url('kata_ganti/ubah_proses');
```

```
    // Set validation rules
        CALL form_validation->set_rules('indo', 'indo', 'required');
        CALL form_validation->set_rules('mdr', 'mdr', 'required');
        CALL form_validation->set_rules('jw', 'jw', 'required');
        CALL form_validation->set_rules('level', 'level', 'required');

        // jika proses validasi sukses, maka lanjut update absen
         if (CALL form_validation->run() == TRUE)

         // Simpan data
        tmpl <- array('indo'=> CALL input->post('indo'),
                      'mdr'=> CALL input->post('mdr'),
                      'jw'=> CALL input->post('jw'),
                      'level' => CALL input->post('level')
                      );
         CALL Kataganti_model->ubah CALL session-userdata('indo'), $kg);

        // set pesan
        CALL session->set_flashdata('message', 'Satu data berhasil diubah!');

        redirect('kata_ganti');
        else
        CALL load->view('template', $data);

END ubah_proses
```

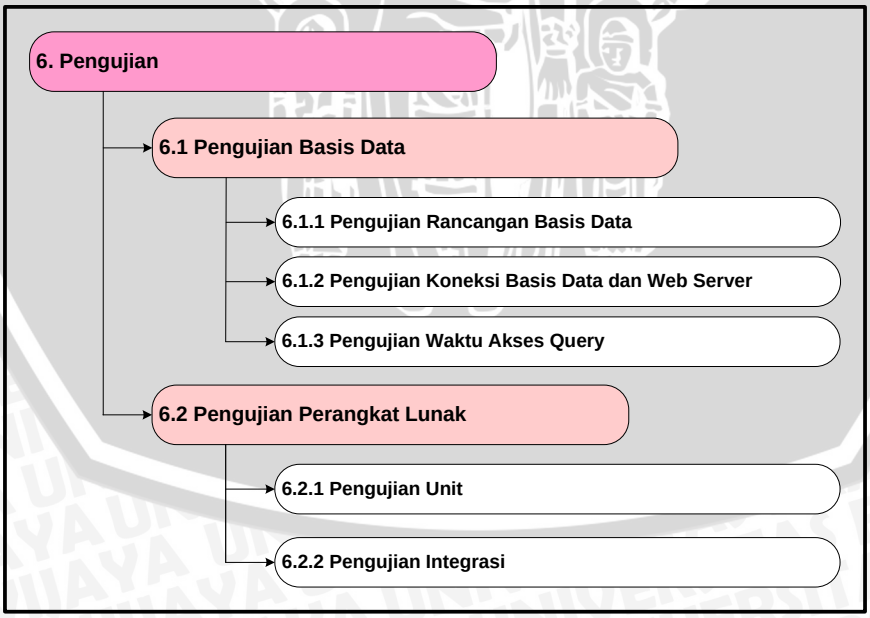
**Gambar 5.26** pseudocode ubah
**Sumber**: [Implementasi]

# BAB VI
# PENGUJIAN

Pada Bab ini dilakukan proses pengujian pada Aplikasi Kamus Bahasa Madura-Indonesia-Jawa. Proses pengujian yang akan dilakukan terbagi menjadi dua bagian, yaitu pengujian perangkat lunak dan pengujian basis data.

Pengujian basis data meliputi pengujian rancangan basis data, pengujian koneksi basis data, dan pengujian waktu akses *query*. Pengujian rancangan basis data bertujuan untuk menguji apakah implementasi perancangan basis data telah sesuai dengan *Entity Relational Diagram*. Pengujian koneksi basis data dilakukan untuk memastikan bahwa komputer *client* dapat melakukan koneksi dengan *database* MySQL yang berada pada komputer *server*.

Pengujian Perangkat lunak terbagi menjadi dua tahap, yaitu pengujian unit dan pengujian integrasi. Pada pengujian unit akan digunakan teknik pengujian *white box (white box Testing)*. Pada pengujian integrasi akan digunakan teknik pengujian *black box (black box Testing)*. Gambar 6.1 menunjukkan langkah-langkah proses pengujian.




**Gambar 6. 1** Diagram pohon pengujian sistem
**Sumber:** [Pengujian]

## 6.1 Pengujian Basis Data

Pengujian basis data kamus ini meliputi tiga macam pengujian, yaitu pengujian rancangan basis data, pengujian koneksi basis data, dan pengujian waktu akses *query*.

### 6.1.1 Pengujian Rancangan Basis Data

#### 6.1.1.6. Tujuan

Pengujian Rancangan basis data bertujuan untuk mengetahui kebenaran rancangan basis data yang telah dibuat. Pengujian ini dilakukan untuk memastikan bahwa tabel pada basis data hasil perancangan sesuai dengan tabel hasil implementasi.

#### 6.1.1.7. Alat yang Digunakan

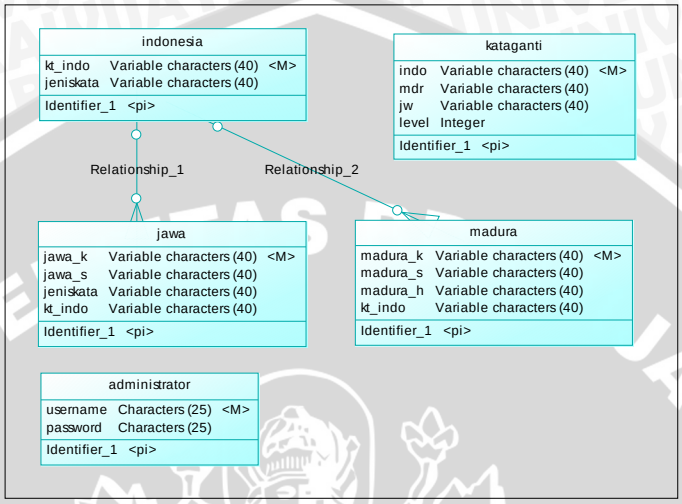
Pengujian perancangan basis data dilakukan dengan menggunakan perangkat lunak Sybase PowerDesigner 12.5.

#### 6.1.1.8. Prosedur Uji dan Pelaksanaan

Prosedur Uji pada pengujian rancangan basis data adalah sebagai berikut:

1. Sebuah *window* Command Prompt dijalankan dari:
   ```
   Start | Run... | Open: cmd.exe
   ```
2. *Server* basis data MySQL dijalankan sebagai *service* dengan memberikan perintah:
   ```
   C:\Documents and Settings\Administrator>net start mysql
   ```
3. Memasuki SQL *Shell* dengan perintah berikut:
   ```
   C:\xampp\mysql\bin>mysql -u root -p kamusdb
   Enter password:
   ```
4. Tabel-tabel yang terdapat pada basis data kamusdb ditampilkan dengan menggunakan perintah SQL berikut:
   ```
   mysql>show tables;
   ```
5. Membuka *software* Sybase PowerDesigner 12.5 dengan cara berikut:
   ```
   Start|All Programs|Sybase|PowerDesigner 12.5|
   PowerDesigner
   ```

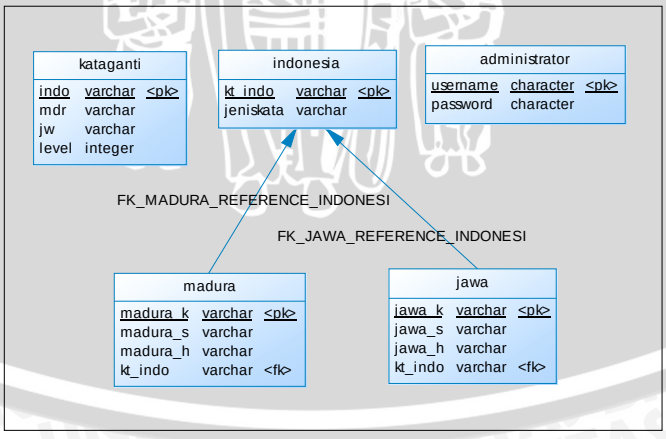
6. Menggambarkan *Entity Relationalship Diagram* pada area kerja *Conceptual Data Model* (CDM).
7. Memeriksa diagram ER tersebut dengan cara menekan tombol *Check Model* pada *toolbar*. Hasil pemeriksaan ini disebut dengan CDM *Object* yang ditunjukkan dalam Gambar 6.2.




**Gambar 6.2** *Conceptual Data Model Object*

**Sumber:** [Pengujian]

8. Untuk mengubah diagram ER dari CDM *Object* menjadi *Physical Data Model* (PDM) *Object*, dilakukan proses *generate* dengan menekan *link Generate Physical Data Model* pada *toolbar Tools*. PDM *Object* ditunjukkan dalam Gambar 6.3.




**Gambar 6.3** *Physical Data Model*

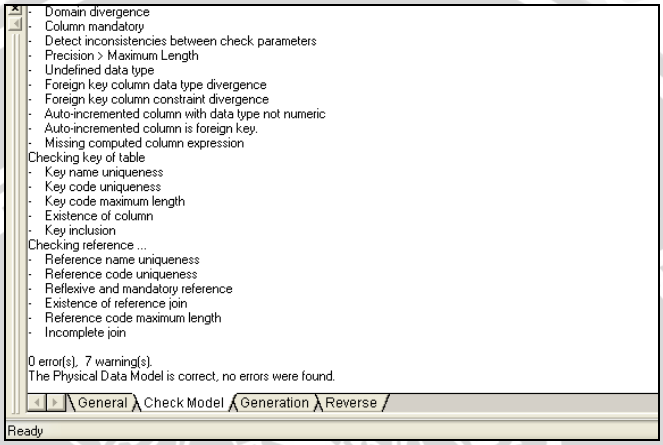
**Sumber:** [Pengujian]

9. Setelah berubah menjadi PDM *Object*, dilakukan pemeriksaan kembali dengan cara menekan tombol *Check Model* pada *toolbar*.

10. Setelah berubah menjadi PDM *Object*, dilakukan *generate* ke *database* MySQL dengan cara menekan *link Generate Database* pada *toolbar Tools*.

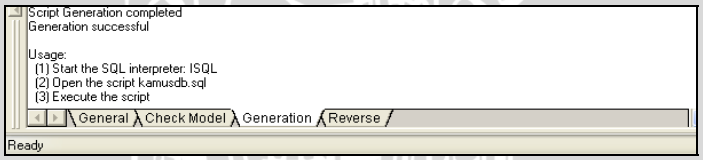
#### 6.1.1.9. Hasil Pengujian

Hasil pemeriksaan PDM dengan cara menakan tombol *check model* pada *toolbar powerdesigner* ditunjukan pada gambar 6.4.




Gambar 6.4 *Laporan Check Model*
Sumber: [Pengujian]

Hasil generate database proses pengujian basis data ditunjukkan pada gambar 6.5.




Gambar 6.5 *Laporan Check Model*
Sumber: [Pengujian]

#### 6.1.1.10. Analisis Hasil Pengujian

Tabel pada basis data `kamusdb` berhasil di *generate*, dan sesuai dengan tabel hasil implementasi. Hasil *generate* bisa dilihat dengan perintah `C:\xampp\mysql\bin>mysql -u root -p kamusdb` sesuai dengan prosedur uji, point 1 sampai dengan point 4.

### 6.1.2 Pengujian Koneksi Basis Data dan *Web Server*

#### 6.1.2.4. Tujuan

Tujuan pengujian koneksi basis data dan web server adalah untuk mengetahui apakah komputer *client* dapat melakukan koneksi dengan *database* MySQL pada komputer *server*.

#### 6.1.2.5. Alat yang digunakan

Pengujian Koneksi basis data dan web *server* dilakukan dengan menggunakan dua buah PC yang masing-masing sebagai PC *server* dan PC *client*. Pengujian dilakukan dengan menjalankan window command promt dan server database MySQL.

#### 6.1.2.6. Prosedur Uji dan Pelaksanaan

Pelaksanaan pengujian dilakukan dari sisi server dan client dengan prosedur uji sebagai berikut:

PC *Server* Aplikasi:

1. Sebuah *window* Command Prompt dijalankan dari:
   Start | Run... | Open: cmd.exe
2. *Server database* MySQL dijalankan sebagai *service* dengan memberikan perintah:
   C:\>net start mysql
3. Aplikasi yang sedang berjalan dan koneksi yang sedang aktif ditampilkan dengan memberikan perintah:
   C:\>netstat –an

PC *Client*:

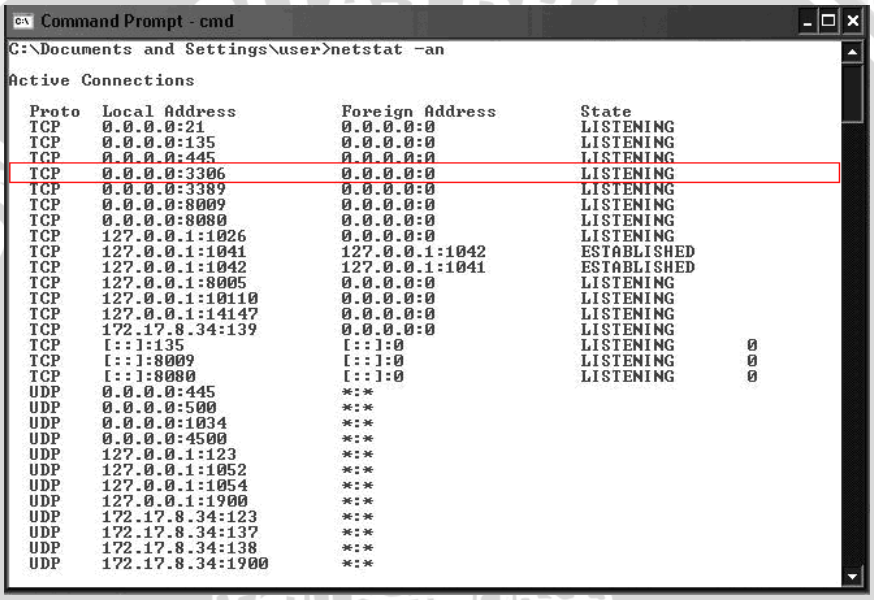
1. Membuka aplikasi kamus *Madura-Indonesia-Jawa*
2. Melakukan proses *login* sebagai Administrator.
3. Aplikasi yang sedang berjalan dan koneksi yang sedang aktif ditampilkan dengan memberikan perintah:
   C:\>netstat –an

PC *Server* Aplikasi:

1. Aplikasi yang sedang berjalan dan koneksi yang sedang aktif ditampilkan kembali dengan memberikan perintah:
   C:\>netstat –an

#### 6.1.2.6. Hasil Pengujian

Hasil dari penggunaan perintah netstat –an pada komputer *server* aplikasi sebelum ada koneksi dengan komputer *client* ditunjukkan dalam Gambar 6.6. Perintah tersebut digunakan untuk menampilkan koneksi yang sedang aktif. Dari gambar tersebut terlihat bahwa *database server* MySQL (mysqld-nt.exe) memiliki kondisi (*state*) LISTENING pada alamat lokal 0.0.0.0:3306. Hal tersebut berarti bahwa *database server* MySQL telah siap untuk menerima sebuah koneksi *database* pada *port* TCP 3306.




**Gambar 6.6** Koneksi yang sedang aktif pada komputer *server* sebelum aplikasi dijalankan pada komputer *client*
**Sumber:** [Pengujian]

#### 6.1.2.7. Analisis Hasil Pengujian

Hasil pengujian koneksi menunjukkan bahwa komputer *client* dapat melakukan koneksi dengan komputer *server*.

### 6.1.3 Pengujian Waktu Akses *Query*

#### 6.1.3.6. Tujuan

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan dalam melakukan *query* pada basis data `kamusdb` dan untuk

mendapatkan perbandingan waktu *query* yang dilakukan terhadap jumlah data yang berbeda pada basis data `kamusdb`.

### 6.1.3.7. Alat yang digunakan

Pengujian dilakukan menggunakan perangkat lunak PHP dengan mengakses file `query_tes.php`.

### 6.1.3.8. Prosedur dan Pelaksanaan

Prosedur yang dijalankan dengan mengakses `query_tes.php`. File `query_tes.php` dibuat khusus untuk melakukan pengujian waktu akses query dengan menggunakan fungsi `microtime()` pada PHP. Pengujian dijalankan dengan jumlah data *entry* sebanyak 500, 1000, 2000, dan 4000 data *entry*.

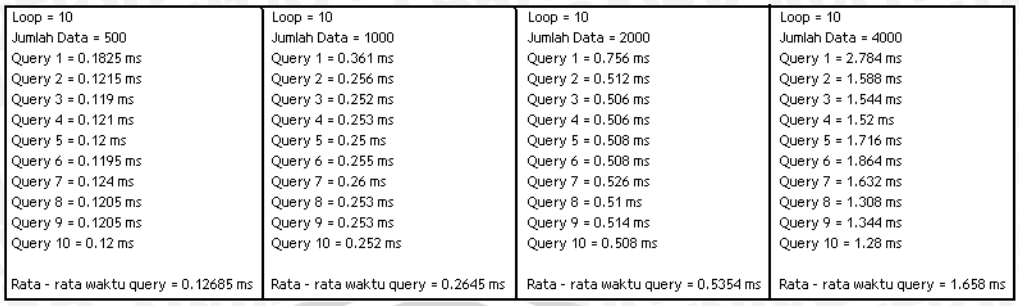
### 6.1.3.9. Hasil pengujian

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `indonesia` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.7 berikut.

| | | | |
|---|---|---|---|
| Loop = 10 | Loop = 10 | Loop = 10 | Loop = 10 |
| Jumlah Data = 500 | Jumlah Data = 1000 | Jumlah Data = 2000 | Jumlah Data = 4000 |
| Query 1 = 0.1765 ms | Query 1 = 1.444 ms | Query 1 = 0.722 ms | Query 1 = 1.792 ms |
| Query 2 = 0.1265 ms | Query 2 = 0.617 ms | Query 2 = 0.514 ms | Query 2 = 1.304 ms |
| Query 3 = 0.124 ms | Query 3 = 0.347 ms | Query 3 = 0.5 ms | Query 3 = 1.344 ms |
| Query 4 = 0.14 ms | Query 4 = 0.327 ms | Query 4 = 0.506 ms | Query 4 = 1.332 ms |
| Query 5 = 0.1265 ms | Query 5 = 0.323 ms | Query 5 = 0.506 ms | Query 5 = 1.268 ms |
| Query 6 = 0.125 ms | Query 6 = 0.306 ms | Query 6 = 0.502 ms | Query 6 = 1.276 ms |
| Query 7 = 0.1285 ms | Query 7 = 0.327 ms | Query 7 = 0.506 ms | Query 7 = 1.3 ms |
| Query 8 = 0.1655 ms | Query 8 = 0.304 ms | Query 8 = 0.502 ms | Query 8 = 1.26 ms |
| Query 9 = 0.126 ms | Query 9 = 0.319 ms | Query 9 = 0.502 ms | Query 9 = 1.3 ms |
| Query 10 = 0.1245 ms | Query 10 = 0.287 ms | Query 10 = 0.502 ms | Query 10 = 1.264 ms |
| Rata - rata waktu query = 0.1363 ms | Rata - rata waktu query = 0.4601 ms | Rata - rata waktu query = 0.5262 ms | Rata - rata waktu query = 1.344 ms |

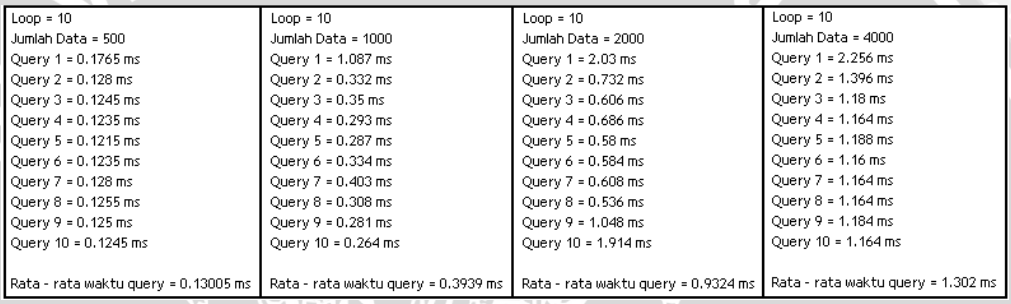
**Gambar 6.7** Hasil Pengujian Waktu Akses *Query* untuk Tabel `indonesia`
**Sumber**: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `kataganti` dengan jumlah data sebanyak data sebanyak 500, 1000, 2000, dan 4000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.8 berikut.



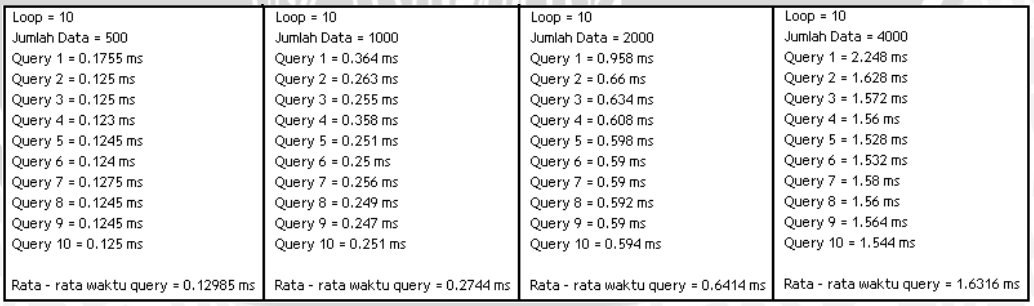

**Gambar 6.8** Hasil Pengujian Waktu Akses *Query* untuk Tabel `kataganti`
**Sumber**: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `madura` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.9 berikut.




**Gambar 6.9** Hasil Pengujian Waktu Akses *Query* untuk Tabel `madura`
**Sumber**: [Pengujian]

Hasil pengujian waktu akses *query* dalam *milisecond*(ms) terhadap tabel `jawa` dengan jumlah data sebanyak 500, 1000, 2000, dan 4000 data *entry* dengan konfigurasi selama 10 kali perulangan ditunjukkan pada Gambar 6.10 berikut.




**Gambar 6.10** Hasil Pengujian Waktu Akses *Query* untuk Tabel `jawa`
**Sumber**: [Pengujian]

Hasil rata-rata pengujian waktu akses *query* terhadap basis data `kamusdb` dengan jumlah data sebanyak 500 dan 1000 data *entry*

memberikan hasil yang cukup stabil. Data rata-rata pengujian terhadap basis data `kamusdb` ditunjukkan dalam Tabel 6.1.

**Tabel 6.1** Pengujian Waktu Akses *Query*

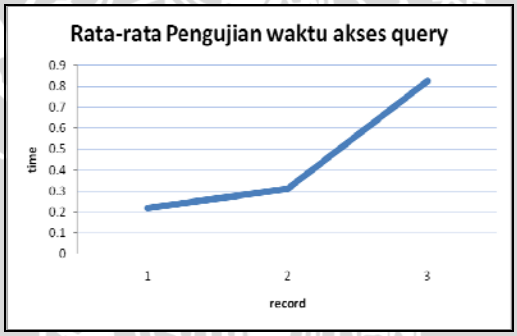
| Nama Tabel | Waktu Akses dengan 500 Data *Entry* (ms) | Waktu Akses dengan 1000 Data *Entry* (ms) | Waktu Akses dengan 2000 Data *Entry* (ms) | Waktu Akses dengan 4000 Data *Entry* (ms) |
|---|---|---|---|---|
| indonesia | 0.1363 | 0.4601 | 0.5262 | 1.344 |
| kataganti | 0.12685 | 0.2645 | 0.5354 | 1.658 |
| madura | 0.13005 | 0.3939 | 0.9324 | 1.302 |
| jawa | 0.12985 | 0.2744 | 0.6414 | 1.6316 |

**Sumber**: [Pengujian]

**Tabel 6.2** Rata-rata Pengujian Waktu Akses Query

| | $\Delta_1$ | $\Delta_2$ | $\Delta_3$ |
|---|---|---|---|
| | 0.3238 | 0.0661 | 0.8178 |
| | 0.13765 | 0.2709 | 1.1226 |
| | 0.26385 | 0.5385 | 0.3696 |
| | 0.14455 | 0.367 | 0.9902 |
| Δ | 0.2174625 | 0.310625 | 0.82505 |

**Sumber**: [Pengujian]




**Gambar 6.14** Grafik Rata-rata Pengujian Waktu

**Sumber**: [Pengujian]

### 6.1.3.10. Analisis Hasil Pengujian

Dari grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* pada Gambar 6.14 menunjukkan bahwa semakin banyak data *entry* maka semakin lama waktu yang dibutuhkan untuk mengakses.

## 6.2 Pengujian Perangkat Lunak

Pengujian perangkat lunak sistem ini dibagi menjadi dua tahap, yaitu pengujian unit dan pengujian integrasi.

### 6.2.1 Pengujian Unit

Dalam sistem yang dibangun dengan pemrograman berorientasi objek, pengujian unit diterapkan untuk suatu metode (operasi) dari suatu *class*.

#### 6.2.1.5. Tujuan Pengujian Unit

Tujuan Pengujian unit adalah untuk mengetahui kompleksitas siklomatis dari metode operasi suatu *class,* agar mudah dilakukan pelacakan apabila terjadi kesalahan pada *statement* program sehingga program bisa berjalan dengan baik sesuai hasil yang diharapkan.

#### 6.2.1.6. Prosedur dan Pelaksanaan

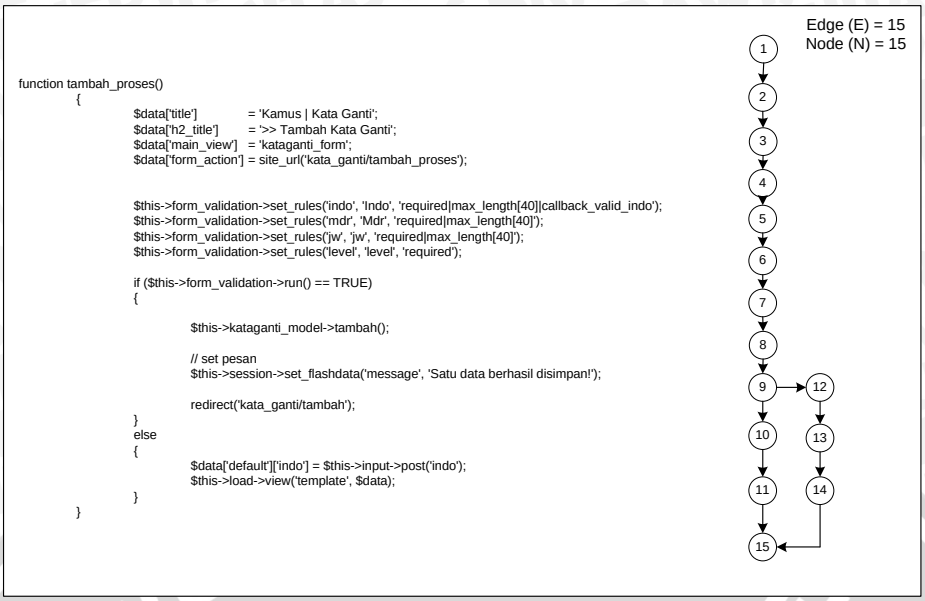
Pada pengujian unit ini, digunakan teknik pengujian *white box (white box testing)* dengan teknik *basis path testing*. Pada teknik *basis path testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph*, menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity),* menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan.

#### 6.2.1.7. Hasil Pengujian

Di dalam penulisan laporan skripsi ini, hanya dicantumkan hasil pengujian unit untuk algoritma dari beberapa metode (operasi) saja (tidak untuk keseluruhan metode).

##### 6.2.1.7.1 Pengujian Unit untuk Operasi tambah_proses

Operasi tambah_proses merupakan salah satu metode yang terdapat dalam class Kata_ganti. Pada Gambar 6.15 memperlihatkan proses pemodelan dalam *flow graph* pada operasi tambah_proses pada class Kata_ganti.




```
function tambah_proses()
	{
		$data['title']		= 'Kamus | Kata Ganti';
		$data['h2_title']	= '>> Tambah Kata Ganti';
		$data['main_view']	= 'kataganti_form';
		$data['form_action'] = site_url('kata_ganti/tambah_proses');


		$this->form_validation->set_rules('indo', 'Indo', 'required|max_length[40]|callback_valid_indo');
		$this->form_validation->set_rules('mdr', 'Mdr', 'required|max_length[40]');
		$this->form_validation->set_rules('jw', 'jw', 'required|max_length[40]');
		$this->form_validation->set_rules('level', 'level', 'required');

		if ($this->form_validation->run() == TRUE)
		{

			$this->kataganti_model->tambah();

			// set pesan
			$this->session->set_flashdata('message', 'Satu data berhasil disimpan!');

			redirect('kata_ganti/tambah');
		}
		else
		{
			$data['default']['indo'] = $this->input->post('indo');
			$this->load->view('template', $data);
		}
	}
```

**Gambar 6.15** *Flowgraph* pada Operasi tambah_proses
**Sumber**: [Pengujian]

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.15 yang telah dilakukan terhadap operasi menambahAccount, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

$$V(G) = E - N + 2$$
$$= 15 - 15 + 2$$
$$= 2$$

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 2 ditentukan dua basis set dari jalur independen yaitu:

Jalur 1 : 1-2-3-4-5-6-7-8-9-10-11-15

Jalur 2 : 1-2-3-4-5-6-7-8-9-12-13-14-15

Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.3.

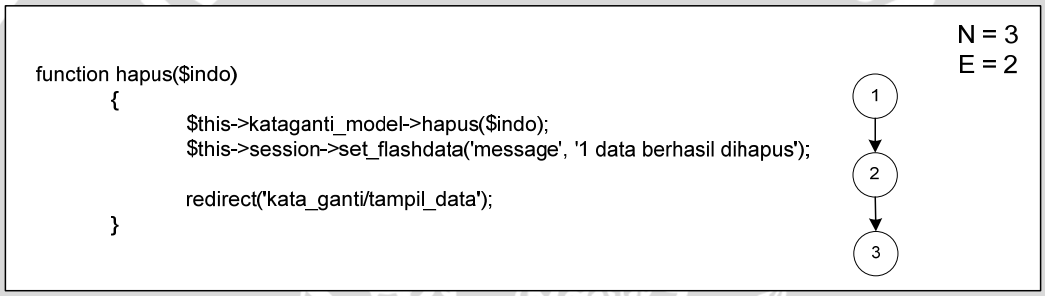
**Tabel 6.3** *Testcase* untuk Pengujian Unit Operasi tambah_proses

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Data yang dimasukkan sudah ada dalam database | Sistem Menampilkan Kembalian Berupa String | Sistem Menampilkan Kembalian Berupa String |
| 2 | Memasukkan data kata ganti (indo, mdr,jw,level) | Sistem Menampilkan Kembalian Berupa String | Sistem Menampilkan Kembalian Berupa String |

**Sumber**: [Pengujian]

### 6.2.1.7.2 Pengujian Unit untuk Operasi hapus

Operasi hapus merupakan salah satu metode yang terdapat dalam *class* kata_ganti. Pada Gambar 6.16 memperlihatkan proses pemodelan dalam *flow graph* pada operasi hapus pada *class* kata_ganti.




**Gambar 6.16** *Flowgraph* pada Operasi hapus
**Sumber**: [Pengujian]

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.16 yang telah dilakukan terhadap operasi hapus, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

$$V(G) = E - N + 2$$

$$= 2 - 3 + 2$$

$$= 1$$

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 1 ditentukan dua basis set dari jalur independen yaitu:

Jalur : 1-2-3

Penentuan kasus uji *(test case)* untuk jalur tersebut dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.4.

**Tabel 6.4** *Testcase* untuk Pengujian Unit Operasi hapus

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Memilih submenu hapus pada tampilan data | Sistem Menampilkan Kembalian Berupa String | Sistem Menampilkan Kembalian Berupa String |

**Sumber**: [Pengujian]

#### 6.2.1.8. Analisis Hasil Pengujian

Dari hasil pengujian unit *method* tambah_proses() dan method hapus() yang dilakukan dapat diketahui bahwa unit ini dapat bekerja dengan baik karena keluaran hasil yang didapatkan sesuai dengan hasil yang diharapkan.

### 6.2.2 Pengujian Integrasi

Pengujian Integrasi diterapkan pada kebutuhan sistem yang mengintegrasikan fungsionalitas dari beberapa class untuk memenuhi suatu kebutuhan sistem.

#### 6.2.2.5. Tujuan Pengujian Integrasi

Tujuan dari pengujian integrasi ini adalah untuk menguji kebenaran interaksi antara bagian-bagian sistem perangkat lunak serta menemukan kesesuaian antara kinerja sistem dengan *Use-case*., dalam hal ini menguji interaksi dengan *database.*

#### 6.2.2.6. Prosedur dan Pelaksanaan

Pada pengujian Integrasi yang dijadikan objek uji adalah *Use-case* yang merupakan hasil dari analisis kebutuhan sistem.

Pengujian Integrasi didahului oleh pengujian pada unit-unit yang akan diintegrasikan. Pengujian integrasi ini menggunakan metode pengujian *Black box Testing*, karena pengujian ini lebih ditekankan untuk menemukan kesesuaian antara kinerja sistem dengan *Use-case*.

### 6.2.2.7. Hasil Pengujian

Penulisan Laporan Skripsi ini hanya dicantumkan hasil pengujian untuk *Use-case* menambah dan mengubah data saja.

#### 6.2.2.7.1 Pengujian Integrasi untuk Usecase Menambah Data

Dalam memenuhi kebutuhan yang terdapat pada *Use-case* Menambah Data diintegrasikan dua *class* yaitu *class* Kata_ganti dan *class* Kataganti_model. Metode-metode yang dilibat untuk menjawab kebutuhan dari *Use-case* Menambah Data adalah tambah dan tambah_proses.

Tujuan pengujian integrasi ini adalah menguji apakah data yang dimasukan dapat tersimpan dengan baik pada basis data.

Kasus uji untuk pengujian *Use-case* Menambah Data ditampilkan pada tabel dibawah ini.

**Tabel 6.5** Kasus Uji untuk pengujian integrasi pada *Use-case* Menambah Data

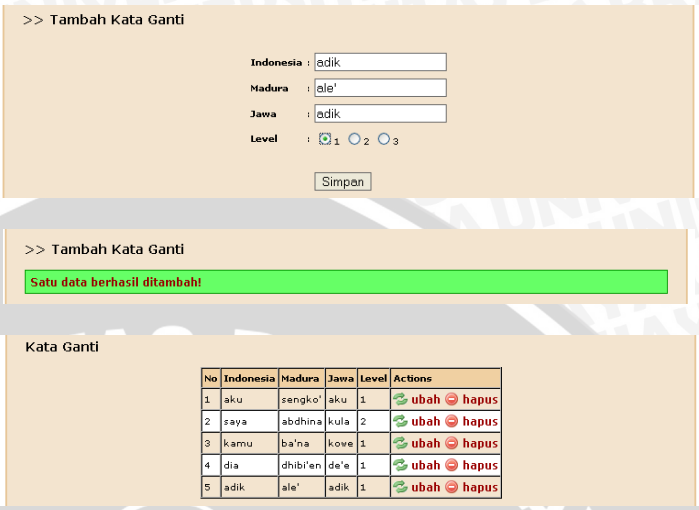
| | | | |
|---|---|---|---|
| **Kasus Uji** | **:** | **Menjalankan prosedur untuk menambah data kata ganti pada aplikasi kamus Madura-Indonesia-Jawa.** | |
| **Prosedur** | **:** | **1** | **Administrator login** |
| | | **2** | **Administrator memilih submenu data** |
| | | **3** | **Adminitrator memilih submenu tambah data** |
| | | **4** | **Administrator mengisi form data** |
| | | **5** | **Administrator memilih submenu simpan** |
| **Hasil yang diharapkan** | **:** | **Sistem dapat menyimpan Data yang telah dibuat ke dalam basis data dengan baik** | |

**Sumber** : [Pengujian]

Hasil yang diperoleh dari hasil pengujian melalui kasus uji pada tabel diatas ditampilkan pada tabel dibawah ini.

**Tabel 6.6** Hasil pengujian integrasi pada *Use-case* Menambah Data

| | | |
|---|---|---|
| **Hasil yang diharapkan** | **:** | **Sistem dapat menyimpan Data kata ganti yang telah dibuat kedalam basisdata dengan baik** |




**Sumber** : [Pengujian]

### 6.2.2.7.2 Pengujian Integrasi untuk Usecase Mengubah Data

Dalam memenuhi kebutuhan yang terdapat pada *Use-case* Menambah Data diintegrasikan dua *class* yaitu *class* Kata_ganti dan *class* Kataganti_model. Metode-metode yang dilibatkan untuk menjawab kebutuhan dari *Use-case* Mengubah Data adalah ubah dan ubah_proses.

Tujuan pengujian integrasi ini adalah menguji apakah data yang sudah tersimpan dapat diubah dengan baik pada basis data.

Kasus uji untuk pengujian *Use-case* Menambah Data ditampilkan pada tabel dibawah ini.

**Tabel 6.7** Kasus Uji untuk pengujian integrasi pada *Use-case* Mengubah Data

| | | | |
|---|---|---|---|
| **Kasus Uji** | **:** | **Menjalankan prosedur untuk mengubah data kata ganti pada aplikasi kamus Madura-Indonesia-Jawa.** | |
| **Prosedur** | **:** | **1** | **Administrator login** |
| | | **2** | **Administrator memilih submenu data** |
| | | **3** | **Dministrator memilih submenu ubah pada baris data** |
| | | **4** | **Administrator mengubah data** |
| | | **5** | **Administrator memilih submenu simpan** |

| | | |
|---|---|---|
| **Hasil yang diharapkan** | **:** | **Sistem dapat menyimpan Data yang telah diubah ke dalam basis data dengan baik** |

**Sumber** : [Pengujian]

Hasil yang diperoleh dari hasil pengujian melalui kasus uji pada tabel diatas ditampilkan pada tabel dibawah ini.

**Tabel 6.8** Hasil pengujian integrasi pada *Use-case* Mengubah Data

| | | |
|---|---|---|
| **Hasil yang diharapkan** | **:** | **Sistem dapat menyimpan Data kata ganti yang telah diubah kedalam basisdata dengan baik**   |

**Sumber** : [Pengujian]

### 6.2.2.8. Analisis Hasil Pengujian

Dari hasil pengujian integrasi pada *Use-case* Menambah Data dan mengubah data diketahui sistem dapat menyimpan data kata ganti kedalam basis data.

# BAB VII

# KESIMPULAN DAN SARAN

## 7.1 Kesimpulan

Berdasarkan hasil Perancangan, Implementasi, dan Pengujian yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut :

1. Tabel basis data hasil implementasi pada aplikasi Kamus Madura-Indonesia-Jawa sesuai dengan tabel pada basis data hasil perancangan.
2. Sistem dapat melakukan proses pengolahan dan penyimpanan data sesuai dengan hasil perancangan.
3. Waktu akses *query* dengan data entry 500, 1000, 2000, dan 4000 menunjukkan peningkatan linier dengan rata-rata waktu akses $\Delta_1$ = 0.2174625 ms, $\Delta_2$ = 0.310625 ms, dan $\Delta_3$ = 0.82505 ms.
4. Sistem yang dikembangkan adalah benar sesuai dengan perancangan.

## 7.2 Saran

Saran yang dapat diberikan untuk pengembangan lebih lanjut dari Aplikasi kamus Madura-Indonesia-Jawa ini adalah :

1. Aplikasi ini bisa dikembangkan menjadi sebuah aplikasi kamus yang memiliki lebih dari dua bahasa daerah.

# DAFTAR PUSTAKA


[ADR-05] Adriati, Rieke. 2005. *Pengembangan Kamus Bahasa Jawa Online Menggunakan PHP dan MySQL*. Tugas Akhir. Fakultas Teknik Jurusan Teknik Elektro. Universitas Brawijaya. Malang.

[DWI-06] Dwi, Didik P. 2006. *Solusi Menjadi Web Master Melalui manajemen Web dengan PHP*. PT.Gramedia. Jakarta

[EKO-93] Ekodarwono, karno. 1993. *Kaidah Penggunaan Ragam Krama Bahasa Jawa*. Pusat Pembinaan dan Pengembangan Bahasa, Jakarta.

[FAT-04] Fathansyah. 2004. *Sistem Basis Data*. Penerbit Informatika, Bandung.

[JUS-08] Jusak. 2008. *Kreasi Situs Mobile Internet dengan xHTML MP*. Prestasi Pustaka Publisher, Jakarta.

[KAD-08] Kadir, Abdul. 2008. *Belajar Database Menggunakan MySQL*. Penerbit Andy, Yogyakarta.

[MEH-08] Mehta, Nirav. 2008. *Mobile Web Development*. Packt Publishing. Mumbai. http://www.packtpub.com/mobile-web-development/book

[MUK-07] Muakmam, dkk. 2007. *Kamus bahasa Madura-madura_Indonesia*. Tim Pakem Maddhu, Pamekasan.

[MUN-05] Munawar. *Pemodelan Visual dengan UML*. Yogyakarta: Graha Ilmu. 2005.

[NUG-05] Nugoho, Bunafit. 2005. *Database Relasional dengan Mysql*. Andi Offset, Yogyakarta.

[NUG-08] Nugroho, Bunafit, 2008, Latihan Membuat Aplikasi Web PHP dan MySQL dengan Dreamweaver, Gava Media, Yogyakarta

[SHO-06] Sholiq. 2006. *Pemodelan Sistem Informasi Berorientasi Objek dengan UML*. Graha Ilmu, Yogyakarta.

[SIT-98] Sitanggang, SRH. 1998. *Geografi Dialek Bahasa Madura*. Pusat Pembinaan dan Pengembangan Bahasa, Jakarta.

[TIM-01] Tim Penyusun Balaibahasa Yogyakarta. 2001. *Kamus Bahasa Jawa, Bausastra Jawa.* Kanisius. Yogyakarta.

[WAH-05] Wahyono, Teguh, 2005, *Pemrograman Web Dinamis dengan PHP*, Elex Media Komputindo, Jakarta