

## TENTANG PENULIS



Kurnia Sandi, seorang mahasiswa program studi Diploma-IV Teknik Informatika yang *falling in love* pada bidang *web development* dan *database engineer*. Memulai petualangan dalam dunia pemrograman pada tahun 2016 (semester 1). Lebih senang dengan dunia pemrograman daripada bidang IT lainnya. Senang berbagi pengetahuan baru terkait pemrograman khususnya *web development* dan teknologi tentang perintah-perintah pada *database* (DDL dan DML).

Memiliki *skill* pemrograman khususnya *Hypertext Pre-Processor* (PHP) dan Javascript untuk *front-end* dan *back-end* serta *skill* Bahasa pemrograman *Structure Query Language* (SQL) guna manajemen arsitektur *database* dalam hal *Data Definition Language* (DDL), *Data Controll Language* (DML) dan *Data Controll Language* (DCL).

Saat ini *doi* masih menempuh Pendidikan semester 7, selama proses Pendidikan semasa kuliah penulis sudah mengerjakan 3 *project* dalam kampus yaitu Manajemen Keuangan Berbasis Web, Toko Buku *Online* Chocobooks dan Bank Sampah Istimewa versi 1.0 serta beberapa *project* lainnya di luar kampus. Untuk mengetahui informasi lebih lanjut tentang penulis, silahkan email ke [kurniya097@gmail.com](mailto:kurniya097@gmail.com).

## PENGANTAR

Dengan nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, Penulis panjatkan puja dan puji syukur atas kehadiran-Nya, yang telah melimpahkan rahmat, taufiq, serta hidayah-Nya kepada penulis, sehingga penulis dapat menyelesaikan buku **“Membangun Aplikasi Bank Sampah Istimewa Menggunakan Framework PHP Codeigniter dan DBMS MySQL”** ini. Penulis menyadari sepenuhnya masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasa. Oleh karena itu, dengan tangan terbuka penulis menerima segala saran dan kritik dari pembaca agar menjadi lebih baik untuk ke depannya. Akhir kata penulis berharap buku ini dapat memberikan manfaat maupun inspirasi terhadap pembaca.

## **DAFTAR ISI**

# **BAB I**

## **PENDAHULUAN**

Sampah merupakan masalah klasik untuk negara berkembang seperti Indonesia, kepadatan penduduk yang tinggi dan aktivitas manusia yang makin berkembang mengakibatkan jumlah sampah yang diproduksi juga meningkat dan bervariasi. Setiap tahun, dipastikan bahwa volume sampah akan selalu bertambah seiring dengan tingkat konsumtif masyarakat yang semakin meningkat. Menurut catatan dari Kementerian Lingkungan Hidup (KLH) bahwa rata-rata masyarakat di Indonesia memproduksi sekitar 2,5 liter sampah per hari atau sekitar 625 juta liter dari jumlah total penduduk. Kondisi ini akan terus bertambah sesuai dengan kondisi lingkungannya [1]. Menurut Statistik Lingkungan Hidup Indonesia (2016) jumlah timbulan sampah di Indonesia mencapai 65.200.000 ton per tahun dengan penduduk sebanyak 261.115.456 orang. Proyeksi penduduk Indonesia menunjukkan angka penduduk yang terus bertambah dan tentunya akan meningkatkan jumlah timbulan sampah. Harus dilakukan suatu upaya agar Target SDGs 12.5 yang menyatakan negara secara substansial mengurangi timbulan sampah melalui pencegahan, pengurangan, daur ulang, dan penggunaan kembali dapat dicapai. Langkah pemerintah tertuang dalam Pepres 97 Tahun 2017 yang menargetkan pengurangan sampah rumah tangga dan sampah sejenis sampah rumah tangga sebesar 30 persen dan penanganannya sebesar 70 persen [2].

Menurut Singhirunnusorn dkk. (2012), perubahan cara berpikir masyarakat mengenai pengelolaan sampah rumah tangga untuk mengurangi sampah di sumber melalui partisipasi warga harus diintegrasikan ke dalam proyek bank sampah yang berbasis masyarakat. Sesuai dengan filosofi mendasar mengenai pengelolaan sampah sesuai dengan ketentuan dalam Undang Undang Nomor 18 Tahun 2008 tentang Pengelolaan Sampah, kini perlu perubahan cara pandang masyarakat mengenai sampah dan cara memperlakukan atau mengelola sampah. Cara pandang masyarakat pada sampah seharusnya tidak lagi memandang sampah sebagai hasil buangan yang

tidak berguna. Sampah seharusnya dipandang sebagai sesuatu yang mempunyai nilai guna dan manfaat. Dalam rangka melaksanakan Peraturan Pemerintah No. 81 Tahun 2012 tentang Pengelolaan Sampah Rumah Tangga dan Sampah Sejenis Sampah Rumah Tangga, maka praktek mengolah dan memanfaatkan sampah harus menjadi langkah nyata dalam mengelola sampah [3]. Oleh karena itu, penulis membangun sebuah system tabungan sampah yang dinamakan “Bank Sampah Istimewa (BASIS) versi 1.0” yang berorientasi pada tabungan berupa *gram* emas bukan rupiah yang nantinya diharapkan system ini dapat digunakan dengan bijak oleh para instansi atau pihak terkait yang ingin membangun bank sampah berupa tabungan emas.

Adapun ruang lingkup dalam membangun system ini sehingga anda juga dapat menjadikannya sebagai acuan dalam membangun system ini yaitu sebagai berikut:

1. Perancangan aplikasi BASIS ini hanya sebatas media pencatatan tabungan yang dikonversi dalam bentuk *gram* emas.
2. Buku tabungan emas masih dicatat secara manual namun tetap disesuaikan dengan system.
3. Data yang digunakan dalam membangun aplikasi ini adalah data kategori sampah, sub kategori sampah, data nasabah, harga beli emas (saat ini) dan harga jual emas (saat ini), data pengguna, dan data *outlet*. Proses yang dilakukan meliputi proses pengolahan data kategori dan subkategori sampah, proses pengolahan data nasabah, proses pengolahan data pengguna dan proses transaksi (tabung dan ambil).
4. Dalam menganalisis system menggunakan BPMN dengan *tools* Bizagy Modeller.

5. Dalam merancang aplikasi BASIS v1.0 ini menggunakan bahasa pemrograman *Hypertext Pre-Processor* (PHP) dan menggunakan DBMS *MySQL*, dengan *editor Atom*.

Adapun sistematika pembahasan dari buku ini yang disajikan berupa BAB pembahasan yang akan memberikan penjelasan dan pemahaman mendalam tentang Bahasa yang digunakan, *tools* pendukung, model perancangan yang dipakai serta manual book dari system yang akan dibangun. Berikut point-pointnya:

1. Pedahuluan, berisi tentang pengantar dari secara umum dari isi buku.
2. Landasan Teori dan Pengenalan Komponen Penyusun Sistem, berisi tentang teori-teori pendukung system seperti Bahasa pemrograman yang digunakan, paket-paket web server, DBMS yang dipakai dan *tools* pengelola *database* serta *editor* yang digunakan.
3. Perancangan Sistem, Basis Data dan Antarmuka, berisi tentang penggunaan *tools* perancangan system, pengelolaan basis data dan antarmuka dalam merancang system yang akan dibangun. Pada buku ini, perancangan system menggunakan *tools* UML, pengelolaan Basis Data menggunakan HeidiSQL dan PHPMyAdmin serta untuk perancangan Antarmuka (*User Interface*) menggunakan.
4. Implementasi dan Uji Coba Aplikasi.
5. Alur/Urutan Penggunaan Aplikasi.

# **BAB II**

## **LANDASAN TEORI DAN KOMPONEN PENYUSUN SISTEM**



## 2.1 Pengenalan PHP

### 2.1.1 Pengertian PHP

Menurut Sibero (2012:49), “PHP (*Personal Home Page*) adalah pemrograman (interpreter) yang melakukan proses penerjemahan baris sumber menjadi kode mesin yang dimengerti oleh komputer secara langsung saat baris kode dijalankan”. [4]

Menurut Kustiyaningsih (2011:114), “PHP (atau resminya PHP: *Hypertext Pre-Processor*) adalah skrip yang bersifat *server-side* yang ditambahkan ke dalam HTML. PHP sendiri merupakan singkatan dari *Personal Home Page Tools*. Skrip ini akan membuat suatu aplikasi dapat diintegrasikan ke dalam HTML sehingga suatu halaman web tidak lagi bersifat statis, namun menjadi bersifat dinamis. Sifat *server-side* berarti pengerjaan kode program dilakukan di *server*, baru kemudian hasilnya akan dikirim ke *browser*”. [5]

Berdasarkan pengertian di atas, dapat ditarik kesimpulan bahwa PHP (PHP: *Hypertext Pre-Processor*) adalah suatu Bahasa pemrograman bersifat *server side* yang digunakan untuk menerjemahkan sejumlah baris kode (bisa ditambahkan ke dalam HTML) program menjadi kode mesin dimengerti oleh mesin komputer.

### 2.1.2 Sejarah PHP

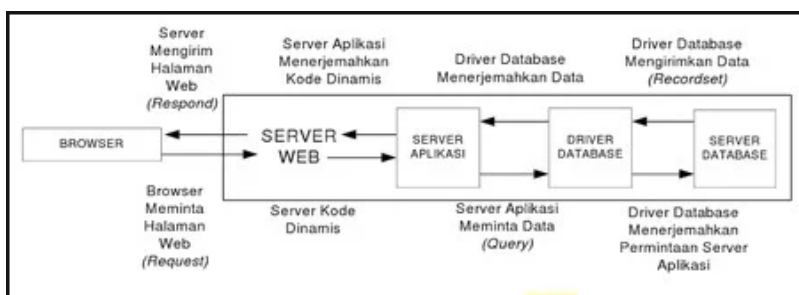
Pada mulanya PHP merupakan singkatan dari *Persnoal Home Page* (Situs Personal). PHP sendiri pertama kali ditemukan oleh Rasmus Lerdorf pada tahun 1995. Saat itu PHP masih bernama *Form Interpreted* (FI), yang bentuknya masih berupa sekumpulan *script* dan digunakan untuk mengolah data *form* dari web.

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang interpreter PHP menjadi lebih bersih, lebih baik dan lebih cepat. Kemudian (Juni 1998), perusahaan tersebut merilis interpreter baru untuk Bahasa pemrograman PHP dan meresmikan *relade version* sebagai PHP 3.0 dan singkatan PHP berubah menjadi *Hypertext Preprocessing*.

PHP difokuskan pada *scripting server-side*. *Scripting server-side* adalah teknik yang digunakan dalam pengembangan web yang melibatkan penggunaan skrip pada server web yang menghasilkan respons yang disesuaikan untuk permintaan setiap pengguna (klien) ke situs web. Artinya, anda dapat melakukan apa yang bisa dilakukan CGI dengan menggunakan PHP seperti mengambil data inputan *form*, meng-*generate* konten halaman dinamis, mengirim dan menerima *cookies* serta masih banyak lagi. Kemampuan dan dukungannya untuk *database* juga sangat bisa diandalkan.

### 2.1.3 Prinsip Kerja PHP

Bahasa pemrograman PHP merupakan Bahasa pemrograman yang dikategorikan dalam kelompok *Server Side Programming*, yang artinya Bahasa pemrograman ini memerlukan penerjemah dalam hal ini *web server* untuk menjalankannya. Adapun penjelasan tentang cara kerja Bahasa pemrograman PHP digambarkan pada Gambar 2.1.



Gambar 2.1 Prinsip Kerja PHP

Berdasarkan gambaran prinsip kerja PHP pada Gambar 2.1 menurut Supono dan Viridiandry P. (2016:4), dapat dijelaskan sebagai berikut:

1. Client/user mengirimkan file PHP (menggunakan browser) melalui Web Server (Seperti Internet Explorer, Mozilla Firefox, Google Chrome, dll).
2. Web Server mendapatkan request atau permintaan dari user lalu meneruskan ke server melalui jaringan internet.
3. Web Server lalu meneruskan permintaan file PHP tersebut ke PHP processor dapat berupa modul (bagian dari web-server) atau terpisah (sebagai CGI/Fast - CGI).
4. Permintaan diproses oleh PHP dan diteruskan ke database (jika terdapat permintaan ke database), kemudian hasilnya dikirim kembali ke web-server.
5. Web Server memaket kembali hasil tersebut dengan menambahkan HTTP header dan dikirim kembali ke browser melalui jaringan internet.
6. Browser memproses HTTP paket dan menampilkannya kembali pada user sebagai file HTML.

#### **2.1.4 Kelebihan PHP**

Bahasa pemrograman PHP merupakan Bahasa pemrograman yang paling banyak digunakan, tentu karena berbagai alasan, salah satunya adalah mempunyai beberapa kelebihan dibandingkan dengan Bahasa pemrograman lainnya yang sejenis. Berikut ini kelebihan Bahasa pemrograman PHP: [6]

1. PHP adalah bahasa multiplatform yang artinya dapat berjalan di berbagai mesin dan sistem operasi (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah sistem lainnya.
2. PHP bersifat open source yang berarti dapat digunakan oleh siapa saja secara gratis.
3. Web server yang mendukung PHP dapat ditemukan dimana-mana dari mulai apache, IIS, Lighttpd, nginx, hingga Xitami dengan konfigurasi yang relatif mudah dan tidak berbelit-belit, bahkan banyak yang membuat dalam bentuk paket atau package (PHP, MySQL dan Web Server).
4. Dalam sisi pengembangan lebih mudah, karena banyaknya milis-milis. Komunitas dan developer yang siap membantu dalam pengembangan.
5. Dalam sisi pemahaman, PHP adalah bahasa scripting yang paling mudah karena memiliki referensi yang banyak.
6. Banyak bertebaran Aplikasi dan Program PHP yang Gratis dan Siap pakai seperti WordPress, PestaShop, dan lain-lain.
7. Dapat mendukung banyak database, seperti MySQL, Oracle, MS-SQL, dan seterusnya.

### 2.1.5 Kekurangan PHP

Seperti layaknya manusia, Bahasa pemrograman PHP juga memiliki kekurangan dari sekian banyak kelebihan yang telah disebutkan sebelumnya. Berikut ini kekurangan Bahasa pemrograman web PHP yang mungkin dapat dijadikan sebagai bahan pertimbangan dalam memilih Bahasa pemrograman temuan Rasmus Lerdorf ini. [6]

1. PHP tidak mengenal *Package*
2. Jika tidak di-*encoding*, maka kode PHP dapat dibaca semua orang dan untuk meng-*encoding*-nya dibutuhkan tool dari Zend yang mahal sekali biayanya.
3. PHP memiliki kelemahan keamanan. Jadi programmer harus jeli dan berhati-hati dalam melakukan pemrograman dan konfigurasi PHP.

## 2.2 Pengenalan Paket Web Server

### 2.2.1 Paket-Paket Web Server

Paket *Web-Server* adalah *server* yang berupa *software* yang digunakan untuk menerima permintaan dalam bentuk situs *web* melalui HTTP atau HTTPS dari klien. [6]

Berikut ini beberapa *web-server* yang banyak dijumpai dan banyak digunakan oleh para *developer* dalam pengembangan *website*.

1. XAMPP terdiri Apache web server, MySQL, PHP, Perl, FTP Server dan PHPMyAdmin. Apache sendiri dapat diinstall di berbagai sistem operasi Linux, Solaris, Windows dan Mac OS X.
2. WampServer terdiri dari Apache, PHP5 dan MySQL. Wamp hanya mendukung system operasi Windows. WampServer dilengkapi

dengan *manage service*, dengan tray icon yang memudahkan dalam mengelola server.

3. EasyPHP, paket ini terdiri dari Apache server, MySQL dan PHPMyAdmin.
4. PHPTriad terdiri dari Apache, MySQL, PHP dan PHPMyAdmin. Web server ini hanya dapat digunakan di sistem operasi Windows.
5. FoxServ terdiri dari Apache, MySQL, PHP, PEAR, Zend dengan versi yang terbaru. Web server ini mendukung system operasi Windows dan Linux.
6. PHPDev terdiri dari PHP, Apache, MySQL, PERL, phpMyAdmin. Phpdev secara kontinyu memperbaharui versi terbaru dari semua paket yang ada.
7. AppServ terdiri dari Apache, MySQL, PHP, dan PHPMyAdmin dengan semua komponen yang lengkap. AppServ mendukung system operasi Windows dan Linux.
8. Server2go terdiri dari Apache, PHP dan MySQL. Server2go dapat secara mandiri sebagai server diinstall dalam CD-ROM, USB Flash-Disk. Ukuran file sekitar 6-45 MB tergantung paket yang ingin digunakan.
9. Apache2Triad terdiri dari Apache2, MySQL, PostgreSQL, OpenSSL, Xmail, SlimFTPd, PHP, Perl dan Pytho + Apache2TriadCP, PHPmyadmin, PHPpgAdmin, AWStats, UebiMiau, PHPXmail, PHPSFTPd. Semua isi dalam paket Ap.ache2Triad tersebut dalam versi stabil
10. VertigoServ terdiri dari PHP development dan server environment untuk Windows secara komplit, juga Apache 2.x.x, PHP 5.x.x, MySQL 5.x.x, dan PHPMyAdmin.

11. Uniform Server terdiri dari paket terakhir dari Apache2, Perl5, PHP5, MySQL5, dan phpMyAdmin. Web server ini dapat dijalankan di system operasi Windows tanpa, tanpa diinstalasi, cukup di-*unpack* dan jalan.
12. MAMP terdiri dari Apache, PHP dan MySQL. MAMP hanya digunakan untuk system operasi OSX.
13. TYPO3. Terdiri dari Apache Webserver, PHP dan MySQL. System Operasi yang didukung Windows.

### 2.2.2 Web Server

Berdasarkan penjelasan pada point 2.2.1, bahwa untuk menjalankan file program yang ditulis menggunakan Bahasa pemrograman PHP dibutuhkan sebuah *web-server*. Adapun paket *web-server* yang digunakan pada buku ini adalah XAMPP versi terbaru. Anda dapat mengunduh *software* tersebut di situs resminya secara gratis (<https://www.apachefriends.org/index.html>).

### 2.2.3 XAMPP



Xampp merupakan paket perangkat lunak (*software*) yang tersedia secara gratis sehingga bebas untuk digunakan tanpa perlu menggunakan *licence* dari pengembang *software*. XAMPP berfungsi sebagai server yang berdiri sendiri (localhost) yang terdiri dari Apache HTTP Server, MySQL database dan penerjemah Bahasa yang ditulis dengan Bahasa pemrograman PHP dan Perl. XAMPP saat ini dikembangkan oleh perusahaan apache friends yang biasanya digunakan untuk simulasi pengembangan *website*.

Tool pengembangan web ini mendukung teknologi web populer seperti PHP, MySQL dan Perl. Dengan menggunakan perangkat lunak XAMPP, programmer web dapat menguji aplikasi web yang dikembangkan kemudian mempresentasikannya ke pihak lain secara langsung dari komputer, tanpa harus terkoneksi ke internet. XAMPP juga dibekali dengan fitur manajemen *database* yaitu PHPMyAdmin seperti pada server *hosting* sungguhan, sehingga pengembang *web* dapat membagikan aplikasi *web* berbasis *database* dengan mudah.

Istilah XAMPP diambil dari kata “X” yang berarti empat system operasi (Windows, Linux, MacOS dan Solaris), sedangkan “A” diambil dari kata Apache, kemudian “M” merupakan singkatan dari kata MySQL, kemudian huruf “P” kepedekan dari PHP dan huruf “P” yang terakhir adalah akronim dari kata Perl.

Apache adalah aplikasi web server yang bertugas menampilkan halaman *web* yang benar kepada *user* berdasarkan kode HTML, PHP atau yang lainnya.

MySQL adalah aplikasi database server. SQL merupakan kependekan dari *Structured Query Language*. SQL sendiri merupakan Bahasa terstruktur yang digunakan untuk mengelola database. MySQL dapat digunakan untuk membuat serta mengelola *database* beserta isinya (DDL dan DML). *Programmer* atau *User* dapat memanfaatkan MySQL untuk proses *Create*, *Read*, *Update* dan *Delete* (atau sering disingkat sebagai CRUD) pada data yang berada dalam *database*.

PHP merupakan kependekan dari *Personal Home Page* (Berubah menjadi *Hypertext Preprocessing* pada tahun 1998 [baca: 2.1.2 Sejarah PHP]). Bahasa pemrograman PHP digunakan untuk membuat *aplikasi* berbasis *web* yang bersifat *server-side scripting*. PHP memungkinkan untuk

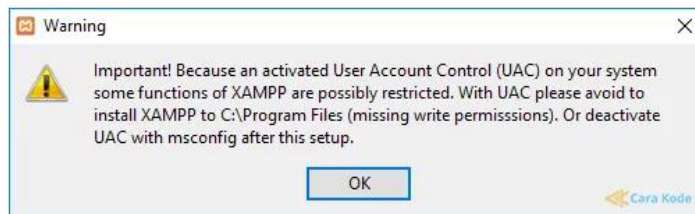


membuat halaman *web* yang bersifat dinamis. System manajemen basis data yang sering digunakan Bersama PHP adalah MySQL namun PHP juga mendukung system manajemen *database* atau disebut juga *database management system* (DBMS) seperti Oracle, Microsoft Access, Interbase, d-base, PostgreSQL, dan sebagainya. [7]

#### 2.2.4 Instalasi Web-Server XAMPP

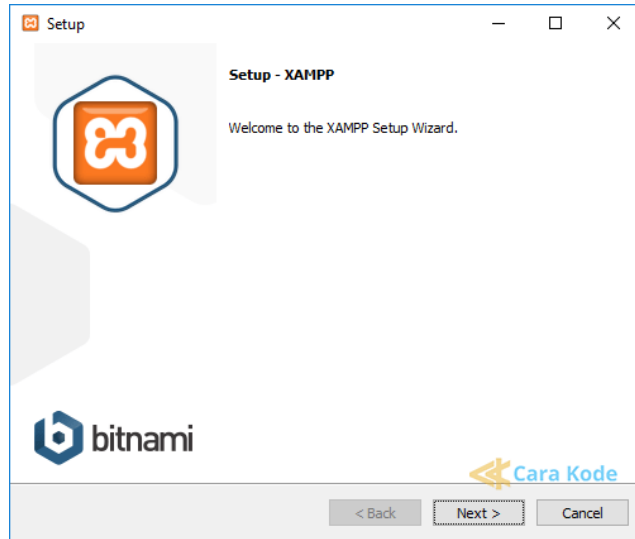
Setelah berkenalan dengan tool pengembang web yang populer ini, saatnya kita melakukan instalasi perangkat lunak ini ke dalam system komputer kita. Pada pembahasan buku ini, penulis menggunakan system operasi Windows versi 10 64-bit. untuk system operasi lain silahkan mencari sumber lainnya baik dari buku maupun internet. Baiklah, tanpa basa-basi lagi, berikut adalah langkah-langkah instalasi atau pemasangan XAMPP:

1. Download program file master *web-server* XAMPP di situs resminya (<https://www.apachefriends.org/download.html>).
2. Jalankan setup.exe XAMPP.
3. Jika muncul *warning UAC* seperti gambar di bawah ini, tidak perlu dihiraukan, klik OK.



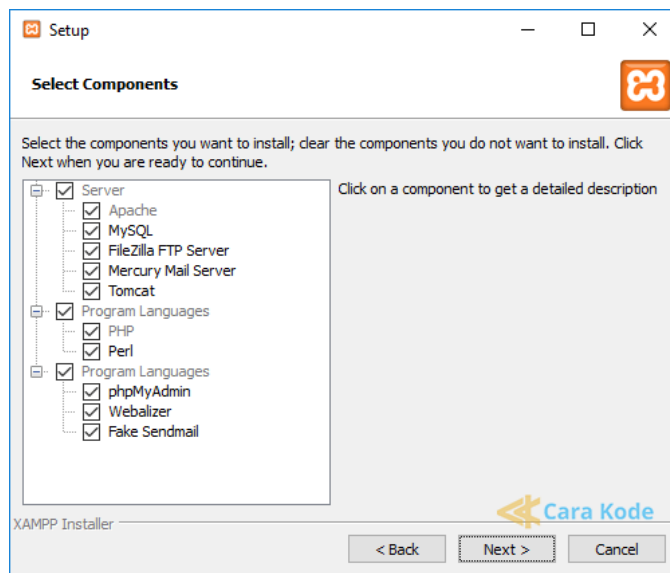
Gambar 2.2 Install XAMPP - Warning

4. Setelah itu akan muncul gambar *Welcome Screen* XAMPP *installer*.



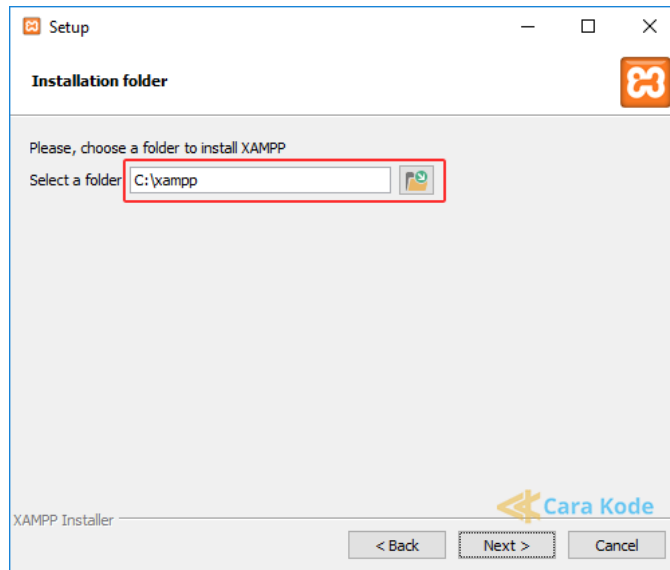
Gambar 2. 3 Install XAMPP – *Welcome Screen*

5. Pilih komponen yang akan diinstal. Lalu klik Next.



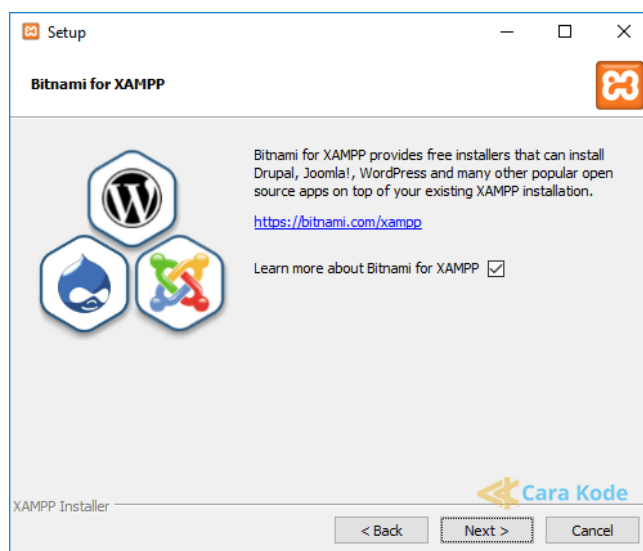
Gambar 2. 4 Install XAMPP – *Select Component*

6. Pilih *destination folder* (default `c:\xampp\`). Fungsi pemilihan ini adalah untuk menentukan di mana nanti folder *root* disimpan. Lalu klik Next.



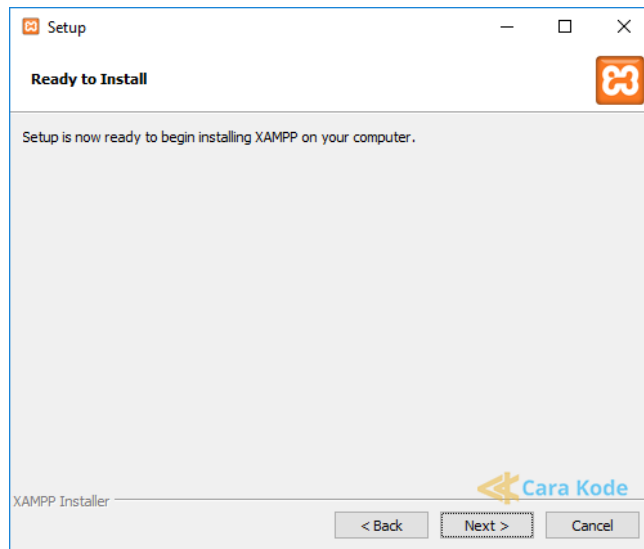
Gambar 2. 5 Install XAMPP – *Select Destination Folder*

7. Proses ini kita bebas untuk *check* atau *uncheck* Learn more about Bitnami for XAMPP. Jika kita memilih opsi *check* maka setelah proses installasi selesai kita akan diarahkan ke halaman Bitnami dari XAMPP itu sendiri. Jika di *uncheck* maka saat instalasi selesai system tidak akan mengalihkan ke halaman Bitnami.



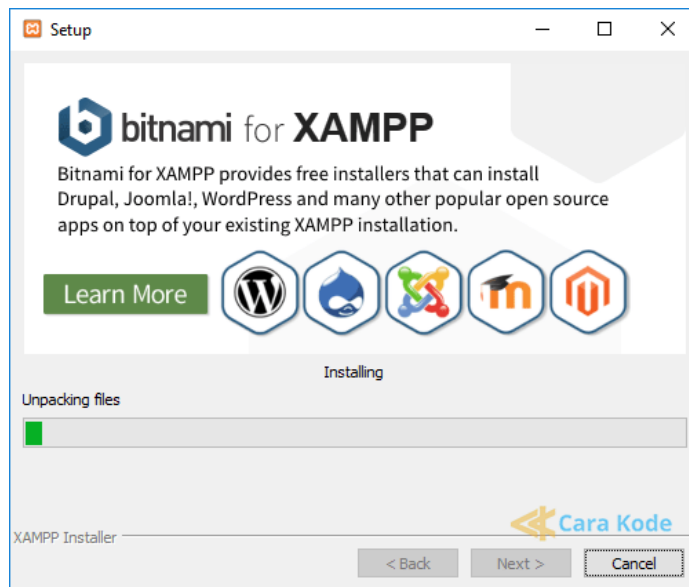
Gambar 2. 6 Install XAMPP – *Select Option*

8. Setelah itu, akan tampil XAMPP ready to Install, klik Next.



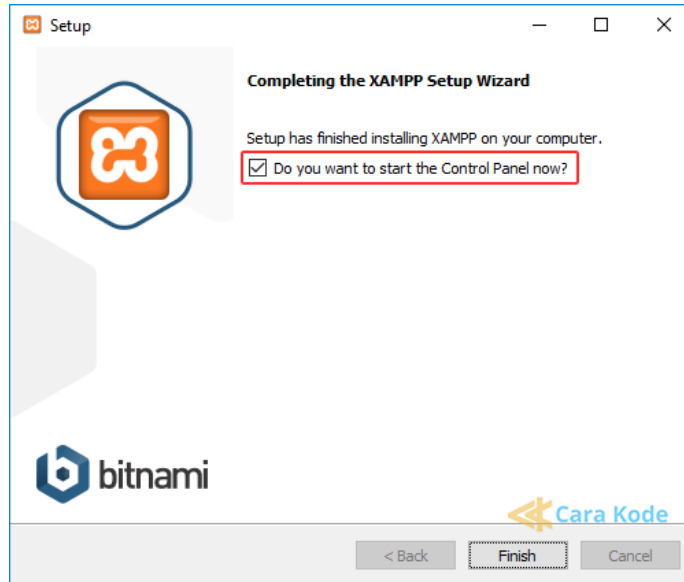
Gambar 2. 7 Install XAMPP – *Ready to Install*

9. Proses instalasi XAMPP. Silahkan ditunggu hingga proses selesai.



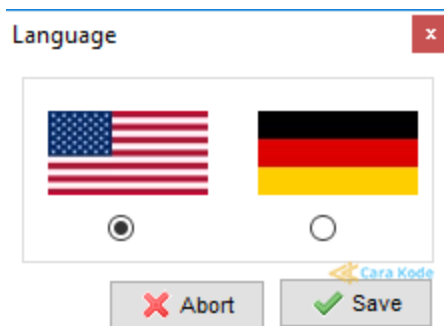
Gambar 2. 8 Install XAMPP – *Installing XAMPP*

10. Proses instalasi selesai. *Checklist* “Do you want to start the Control Panel now?” jika ingin menampilkan control panel xampp. Lalu pilih Finish.



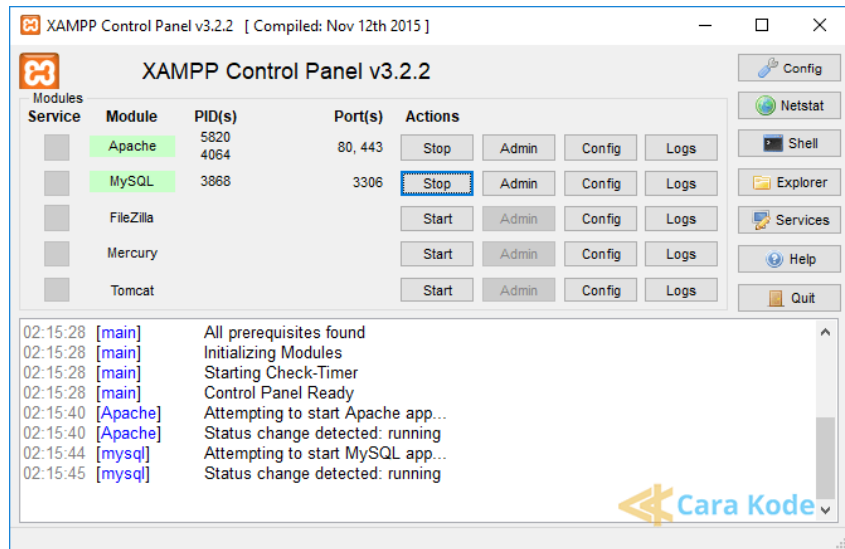
Gambar 2. 9 Install XAMPP – Instalasi selesai

11. Jika muncul kotak dialog untuk memilih Bahasa, maka silahkan pilih Bahasa yang diinginkan (Pada pembahasan ini, penulis memilih Bahasa inggris), kemudian klik Save.



Gambar 2. 10 Install XAMPP – Pilih bahasa

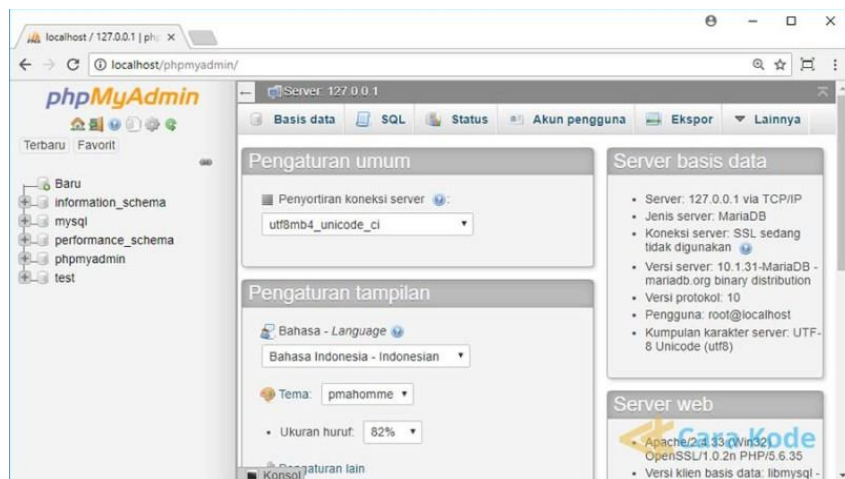
12. Setelah semua langkah dilakukan maka Xampp Controll Panel akan langsung terbuka karena sebelumnya memilih *checklist* Checklist “Do you want to start the Control Panel now?” yang artinya setelah proses instalasi sukses maka secara otomatis Control Panel dari XAMPP akan langsung terbuka.



Gambar 2. 11 Install XAMPP – Control Panel

Silakan klik “start” pada *module* Apache dan MySQL.

13. Untuk melakukan *testing* silahkan buka *browser* favorit. Kemudian pada *address bar* masukkan url “<http://localhost/phpmyadmin/>”.



Gambar 2. 12 Install XAMPP – PHPMyAdmin

Jika pada *browser* anda telah tampil halaman PHPMyAdmin seperti pada Gambar 2. 12, maka *web-server* XAMPP telah berhasil terinstall.

## 2.3 Text Editor: Atom

### 2.3.1 Pengenalan Atom

Bagi developer, memilih text editor sebagai aplikasi untuk menulis koding sangatlah penting, karena hal tersebut dapat membantu dalam penulisan program dengan lebih cepat. Namun kebanyakan *developer*, jikalau sudah terbiasa menggunakan satu text editor, malas untuk mencoba yang lain, padahal semakin banyak kita mencoba, semakin tahu mana text editor yang paling baik. [8]

### 2.3.2 Kenapa Atom?

Editor seperti *Sublime* dan *TextMate* menawarkan kenyamanan tetapi memiliki ekstensibilitas terbatas. Di tempat lain seperti spektrum, Emacs dan Vim menawarkan fleksibilitas yang lebih *ekstream*, tetapi mereka tidak terlalu mudah didekati dan hanya dapat dikustomisasi dengan bahasa scripting tujuan yang khusus.

Kami pikir kami bisa melakukan yang lebih baik. Tujuan kami adalah kombinasi tanpa kompromi dari peretasan dan kegunaan: editor yang akan menyambut siswa sekolah dasar pada hari pertama mereka belajar kode, tetapi juga alat yang tidak akan mereka kalahkan saat mereka berkembang menjadi peretas berpengalaman.

Karena kami telah menggunakan Atom untuk membangun Atom, apa yang dimulai sebagai percobaan secara bertahap telah matang menjadi alat yang tidak bisa kita hidupi tanpa. Di permukaan, Atom adalah editor teks desktop modern yang Anda harapkan. Pasang tudung, dan Anda akan menemukan sistem yang meminta untuk diretas. [9]

### **2.3.3 Inti Atom**

Web bukan tanpa kesalahannya, tetapi pengembangan selama dua dekade telah membentuknya menjadi platform yang luar biasa lunak dan kuat. Jadi ketika kami mulai menulis editor teks yang kami ingin memperluas, teknologi web adalah pilihan yang jelas. Tapi pertama-tama, kita harus membebaskannya dari rantai. [9]

### **2.3.4 Web Asli**

Browser web bagus untuk menjelajahi halaman web, tetapi menulis kode adalah kegiatan khusus yang membutuhkan alat khusus. Lebih penting lagi, browser sangat membatasi akses ke sistem lokal untuk alasan keamanan, dan bagi kami, editor teks yang tidak bisa menulis file atau menjalankan subproses lokal adalah non-starter.

Karena alasan ini, kami tidak membangun Atom sebagai aplikasi web tradisional. Alih-alih, Atom adalah varian khusus Chromium yang dirancang untuk menjadi editor teks dan bukan browser web. Setiap jendela Atom pada dasarnya adalah halaman web yang dibuat secara lokal. [9]

Semua API yang tersedia untuk aplikasi Node.js juga tersedia untuk kode yang berjalan dalam konteks JavaScript setiap jendela. Hibrida ini memberikan pengalaman pengembangan sisi klien yang unik.

Karena semuanya lokal, Anda tidak perlu khawatir tentang pipa aset, penggabungan skrip, dan definisi modul asinkron. Jika Anda ingin memuat beberapa kode, cukup minta di bagian atas file Anda. Sistem modul Node memudahkan untuk memecah sistem menjadi banyak paket kecil yang fokus. [9]



### **2.3.5 Javascript, Penemuan C++**

Berinteraksi dengan kode asli juga sangat sederhana. Sebagai contoh, kami menulis pembungkus di sekitar mesin ekspresi reguler Oniguruma untuk dukungan tata bahasa TextMate kami. Di browser, itu akan membutuhkan petualangan dengan NaCl atau Esprima. Integrasi node membuatnya mudah.

Selain Node API, kami juga mengekspos API untuk dialog asli, menambahkan item menu aplikasi dan konteks, memanipulasi dimensi jendela, dll. [9]

### **2.3.6 Web Tech: Bagian Menyenangkan**

Manfaat besar lainnya, yang dilengkapi dengan penulisan kode untuk Atom, adalah jaminan bahwa itu berjalan pada versi terbaru Chromium. Itu berarti kita dapat mengabaikan masalah seperti kompatibilitas browser dan polyfill. Kita dapat menggunakan semua fitur web yang mengkilap di masa depan, hari ini.

Misalnya, tata letak ruang kerja dan panel kami didasarkan pada flexbox. Ini adalah standar yang muncul dan telah melalui banyak perubahan sejak kami mulai menggunakannya, tetapi tidak ada yang penting selama itu berhasil.

Dengan seluruh industri mendorong teknologi web ke depan, kami yakin bahwa kami sedang membangun Atom di tanah subur. Teknologi UI asli datang dan pergi, tetapi web adalah standar yang menjadi lebih mampu dan ada di mana-mana setiap tahun. Kami senang menggali lebih dalam kotak alatnya. [9]

### 2.3.7 Editor Teks Terbuka

Kami melihat Atom sebagai pelengkap sempurna bagi misi utama GitHub untuk membangun perangkat lunak yang lebih baik dengan bekerja bersama. Atom adalah investasi jangka panjang, dan GitHub akan terus mendukung pengembangannya dengan tim yang berdedikasi kedepannya. Tetapi kami juga tahu bahwa kami tidak dapat mencapai visi kami untuk Atom saja. Seperti yang telah ditunjukkan oleh Emacs dan Vim selama tiga dekade terakhir, jika Anda ingin membangun komunitas yang berkembang dan bertahan lama di sekitar editor teks, itu harus *open source*.

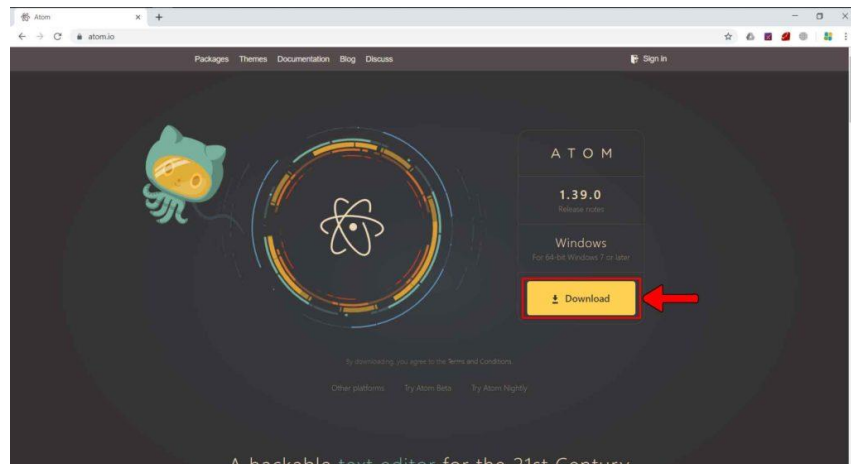
Seluruh editor Atom adalah sumber bebas dan terbuka dan tersedia di bawah organisasi <https://github.com/atom>. [9]

### 2.3.8 Instalasi Atom

Sebelumnya, Atom Editor merupakan editor khusus Sistem Operasi MacOS. Namun sekarang, bagi pengguna system operasi Windows sudah dapat menggunakan Editor ini dikarenakan pengembang telah menyediakan versi Windows namun hanya untuk versi 7 64 bit ke atas. Jadi, bagi pengguna Windows dibawah windows 7 64 bit silahkan menggunakan Text Editor alternatif seperti *Sublime Text*, *Visual Studio Code*, Notepad++, dan sebagainya yang menurut anda nyaman digunakan.



Baiklah, tanpa basa-basi lagi, berikut adalah langkah-langkah instalasi atau pemasangan Text Editor Atom pada Sistem Operasi Windows versi 10:

1. Download file master .exe di situs resminya (<https://atom.io/>).



Gambar 2.13 Install Atom – Download Atom Editor

2. Jalankan file hasil download (biasanya bernama **AtomSetup-x64.exe**).

Name	Date modified	Type	Size
 AtomSetup-x64	1/17/2020 1:45 AM	Application	183,403 KB
 Scratch Desktop Setup 3.6.0	1/1/2020 9:06 PM	Application	175,701 KB

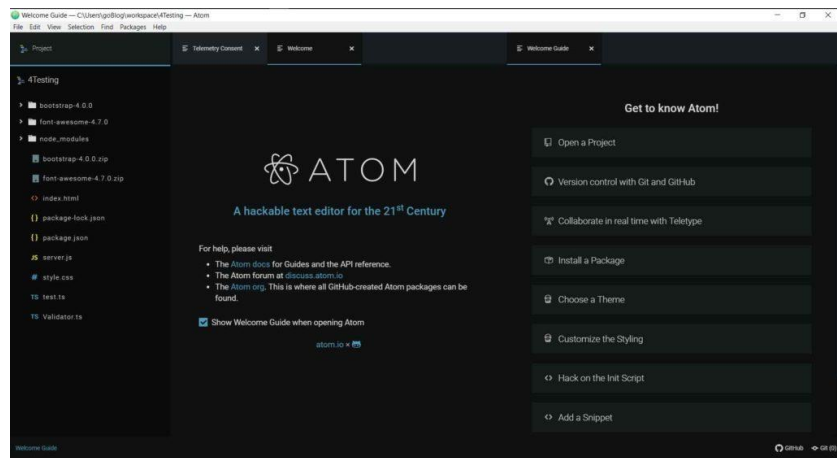
Gambar 2.14 Install Atom – File Hasil Download Atom Editor

3. Proses instalasi Atom Editor



Gambar 2.15 Install Atom – Instalasi Atom Editor

4. Setelah proses instalasi selesai dan berhasil tanpa *error* maka Aplikasi Text Editor Atom akan langsung terbuka secara otomatis.



Gambar 2.16 Install Atom – Instalasi selesai

### 2.3.9 Package-Package Yang Diperlukan

Ketika bekerja dengan Text Editor Atom, *package* atau paket adalah sesuatu yang diperlukan oleh *developer* dalam menulis kodenya. Adapun beberapa *package* yang cukup diperlukan dan penulis menggunakannya juga, berikut *list package* yang digunakan:

1. Autocomplete-paths

Memunculkan nama file yang tersimpan dalam project ketika hendak dipanggil saat dibutuhkan.

2. Color-picker

Berguna saat menulis kode CSS untuk mempercantik tampilan web. Dengan menekan tombol CTRL+Shift+C maka akan muncul menu warna dan warna hexa. Tinggal dipilih maka akan langsung terinput di kode editor.

### 3. Pigments

Masih tentang warna. *Package* ini berfungsi untuk menampilkan warna apa yang sedang aktif atau sedang digunakan pada code CSS.

### 4. Emmet

Salah satu *package* yang powefull dalam atom untuk auto complete kode yang digunakan. Untuk lebih jelasnya silahkan kunjungi situs resminya (<http://emmet.io>).

### 5. Javascript-snippets

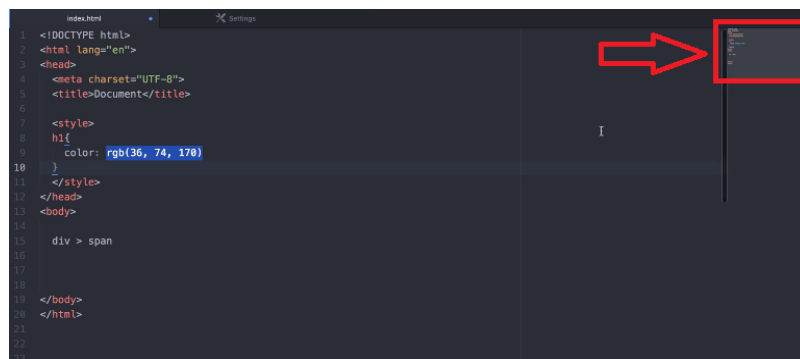
Package ini berguna ketika sedang menulis kode javascript atau Node.js di project.

### 6. Language-blade

Bagi pengguna Laravel dalam membuat aplikasi berbasis web maka *package* ini sangat disarankan. Karena akan mnampilkan sintaks-sintaks blade dapat ter-*highlight*.

### 7. Minimap

*Package* yang berguna untuk menampilkan jendela kecil yang tampilan kode yang telah kita buat.



Gambar 2.17 Package – Minimap

## 8. Terminal-plus

*Package* ini digunakan untuk menampilkan *terminal* atau *command-prompt*. Bagi pengguna Windows *package* ini disarankan untuk dipasang pada editor atom.

## 2.4 Pengenalan Basis Data dan DBMS MySQL

### 2.4.1 Pengertian Basis Data (*Database*) Relasional

Secara bahasa *database* terdiri atas dua kata yaitu “*data*” yang berarti data atau representasi dunia nyata yang belum memiliki makna serta memiliki wujud berupa huruf, angka, gambar, *audio* dan *video*. Sedangkan “*base*” atau basis adalah gudang atau tempat berkumpulnya data.

Secara umum, *database* berarti koleksi data yang saling terkait atau terintegrasi. Semua manusia dan organisasi di dunia membutuhkan informasi. Seorang tukang koran menyimpan alamat pelanggan, rekening koran, tanggal pembayaran, agen koran, dan lainnnnya. Secara praktis, basis data (*database*) dapat dianggap sebagai suatu penyusunan data yang terstruktur yang disimpan dalam media pengingat (*hard disk*) yang tujuannya adalah agar data tersebut dapat diakses dengan mudah dan cepat.

Suatu data yang disimpan dalam media *database* memiliki ketentuan, beberapa ketentuan tersebut seperti harus memiliki *primary key*. *Primary key* adalah kunci utama suatu data yang bersifat *unique* sehingga dapat dibedakan antara baris data yang satu dengan baris data yang lain. Contoh penerapan *primary key* dalam kehidupan ialah menggunakan Nomor Induk Kependudukan (NIK) yang hanya dimiliki oleh satu orang satu NIK.

Jadi, basis data (*database*) adalah himpunan kelompok data (arsip) yang saling berhubungan dan tanpa pengulangan (redudansi) serta

disimpan dalam suatu media elektronik kemudia diorganiasi sedemikian rupa agar kelak dapat dimanfaatkan dengan cepat dan mudah.

Sesungguhnya ada beberapa macam *database*, antara lain yaitu *database* hierarkis, *database* jaringan dan *database* relasional. *Database* relasional merupakan *database* yang populer saat ini dan telah diterapkan pada berbagai *platform*, dari PC hingga minikomputer.

Sebuah *database* relasional tersusun atas sejumlah table. Sebagai contoh, *database* akademis mencakup table-table seperti table dosen, table mahasiswa, table KRS, table nilai, table ruangan, dan lain-lain. Basis data tentang Bank Sampah bisa mencakup info pribadi seperi nama, jenis kelamin, tanggal lahir dan sebagainya serta rekening yang telah didaftarkan. Gambar 2.18 *Gambaran* tabel, baris dan kolom menampilkan sample data mengenai *database* bank sampah.



rek_no	rek_tgl	nik	rek_ket
1234567890123111	2019-12-28 09:11:36	1234567890123111	Aktif
1234567890123321	2019-12-29 03:14:13	1234567890123321	Aktif
1234567890123456	2019-12-28 08:58:30	1234567890123456	Aktif
1332619620000566	2019-12-28 08:56:21	1332619620000566	Aktif
3273076907950000	2020-01-07 04:43:27	3273076907950000	Aktif

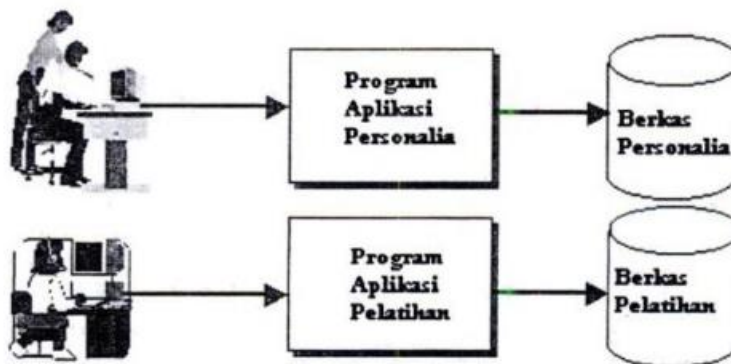
Gambar 2.18 Gambaran tabel, baris dan kolom

## 2.4.2 Sejarah Kemunculan Basis Data (*Database*)

Menurut sejarah, system pemroses basis data terbentuk setelah masa system pemrosesan manual dan system pemrosesan berkas.

System pemrosesan manual (berbasi berkas) merupakan bentuk pemrosesan yang menggunakan dasar berupa setumpuk rekaman yang disimpan pada rak-rak berkas. Jika sesuatu berkas diperlukan, berkas tersebut harus dicari pada rak-rak tersebut. Bentuk seperti ini masih banyak dijumpai dalam kehidupan sehari-hari. Contoh lain adalah buku

telepon saku dimana seseorang relative mudah mencari nama-nama rekannya karena datanya telah disusun secara alfabetis. Namun demikian kemudahan seperti ini tetap saja kurang luwes. Data tidak bisa diurutkan menurut kata atau nomor telepon. Jika hal seperti ini dikehendaki, tidak ada cara lain selain menuliskannya kembali. Tentu saja hal itu tidak praktis.



Gambar 2.19 Gambaran system pemrosesa berkas: masing-masing aplikasi memiliki berkas tersendiri

Pada saat awal penerapan system komputer, sekelompok rekaman disimpan pada sejumlah berkas secara terpisah. System yang menggunakan pendekatan seperti ini biasa disebut sebagai **system pemrosesan berkas**. System ini tentu saja memiliki kelebihan daripada system pemrosesan manual, yaitu dalam hal kecepatan dan keakuratannya. Kelemahannya, perancang system masih didasarkan pada kebutuhan individual pengguna, bukan kebutuhan sejumlah pengguna. Setiap kali ada kebutuhan baru dari seorang pengguna, kebutuhan segera diterjemahkan ke program komputer. Alhasil, setiap program aplikasi menuliskan data sendiri. Sementara itu ada kemungkinan data yang sama juga terdapat pada berkas-berkas lain yang digunakan oleh program aplikasi lainnya.



Konkretnya, system pemrosesan berkas memiliki kekurangan dalam beberapa hal, adapun kekurangannya sebagai berikut:

- Data mubazir,
- Berbagi data terbatas,
- Integritas terhadap data kurang dan tidak konsisten, serta
- Tidak luwes.

**Data mubazir** atau data yang terduplikasi atau duplikasi data pada praktiknya disebabkan oleh setiap program aplikasi menggunakan data tersendiri, sebagaimana telah diutarakan sebelumnya. Sebagai contoh, dalam sebuah perusahaan terdapat bagian Personalia dan bagian Pelatihan. Program aplikasi personalia dipakai untuk mengarsipkan data para pegawai untuk kepentingan personalia semata sedangkan program aplikasi pelatihan dipakai untuk mencatat kepentingan yang menyangkut pelatihan. Data yang terdapat pada kedua program aplikasi tersebut (nomor pegawai, nama pegawai, dan bagian tempat kerja pegawai) akan dicatat. Hal seperti inilah yang memungkinkan adanya duplikasi data.

Data yang mubazir sebenarnya terjadi karena keterbatasan berbagi data. Seandainya suatu data dapat dipakai oleh beberapa program aplikasi, ataupun sejumlah orang maka kemubaziran terhadap data dapat dihindari.

Penyebab **tidak konsisten** suatu data diakibatkan oleh adanya perubahan terhadap data yang sama, tetapi tidak semuanya diubah. Sebagai contoh, bila seseorang telah pindah bagian dan data tentang hal ini telah dicatat oleh bagian Personalia, tetapi tidak dicatat oleh bagian Pelatihan, maka pengguna pada bagian pelatihan tetap dianggap bahwa orang tersebut belum pindah bagian.

Ketidakkonsistenan data berkaitan erat dengan integritas data. Bila data tidak konsisten, maka integritasnya akan berkurang. **Integritas** berarti bahwa data selalu dalam keadaan valid. Sesuatu perubahan, misalnya data alamat pelanggan, seharusnya tidak menimbulkan kerancuan. Hal ini hanya dapat terjadi apabila data tentang hal tersebut hanya ada pada satu tempat. Pada lingkungan *multiuser*, integritas merupakan hal yang sangat kritis, disebabkan tindakan yang dilakukan oleh masing-masing pengguna akan mempengaruhi pengguna yang lain. Bayangkan apa jadinya dua buah biro perjalanan bisa menjual sebuah tiket untuk tempat duduk yang sama pada pesawat terbang kepada dua pelanggan?

Kekurangluwesannya dan/atau tidak luwesnya suatu sistem pemrosesan berkas terletak pada hal pengembangan atau perubahan. Sebagai gambaran, sekiranya terdapat perubahan struktur pada berkas (misalnya jumlah angka suatu data yang menyatakan uang diperbesar) maka setiap program harus diubah. Hal ini disebabkan setiap program berisi definisi data untuk setiap berkas yang diaksesnya. Sekalipun data yang diubah tidak dipakai oleh suatu program, tetap saja program tersebut harus diubah. Keadaan seperti ini banyak dijumpai pada program-program aplikasi lama yang dibangun dengan menggunakan Bahasa pemrograman COBOL.

Pada basis data terdapat istilah independensi data. Independensi data adalah sifat yang memungkinkan perubahan struktur berkas tidak mempengaruhi program, begitupun sebaliknya. Independensi data juga bisa berarti bahwa data bersifat tidak tergantung pada data lain. Bila suatu data dihapus, data lain harus tidak terpengaruh oleh tindakan tersebut.

Sistem pemrosesan basis data dimaksudkan untuk mengatasi kelemahan-kelemahan yang ada pada sistem pemrosesan berkas. Sistem

seperti ini dikenal dengan sebutan DBMS (*Database Management System*). [10]

### 2.4.3 Evolusi Teknologi Basis Data (*Database*)

Perkembangan teknologi basis data tidak lepas dari perkembangan perangkat keras dan perangkat lunak. Perkembangan teknologi jaringan komputer dan komunikasi data merupakan salah satu penyumbang kemajuan penerapan basis data, yang kemudian melahirkan system basis data terdistribusi. Dampak perkembangan ini tentu saja dapat dirasakan dalam kehidupan kita, seperti kemudahan dalam penarikan sejumlah uang dengan fasilitas *Automatic Machine Teller* (ATM) yang banyak diterapkan pada perbankan di Indonesia.

Perkembangan pada dunia perangkat lunak (*software*), seperti kecerdasan buatan (*Artificial Intelligence*), system pakar (*expert system*), dan pemrograman berorientasi objek (*object oriented programming*). Namun, tidak hanya itu perkembangan dunia perangkat lunak tersebut juga mempengaruhi perkembangan basis data, sehingga muncul istilah basis data berorientasi objek (*object oriented database*), dan basis data cerdas (*database intelligence*). **Error! Reference source not found.** memperlihatkan evolusi perkembangan teknologi basis data. [10]

Tabel 2. 1 Evolusi teknologi basis data.

Masa	Perkembangan basis data
1960-an	<ul style="list-style-type: none"><li>• Sistem pemrosesan berkas</li><li>• DBMS</li></ul>

	<ul style="list-style-type: none"> <li>• Layanan informasi secara <i>online</i> berbasis manajemen teks</li> </ul>
1970-an	<ul style="list-style-type: none"> <li>• Penerapan system pakar pada system pengambilan keputusan</li> <li>• Basis data berorientasi objek</li> </ul>
1980-an	<ul style="list-style-type: none"> <li>• Sistem <i>hypertext</i>, yang memungkinkan untuk melihat basis data secara acak menurut kata kunci (sebagaimana yang diterapkan pada internet)</li> </ul>
1990-an	<ul style="list-style-type: none"> <li>• Sistem basis data cerdas</li> <li>• System basis data multimedia cerdas.</li> </ul>

#### 2.4.4 Tujuan Basis Data (*Database*)

Tujuan awal dan tujuan utama dari penggunaan basis data (*database*) adalah untuk memberikan keceatan dan kemudahan dalam menemukan kembali arsi/data yang tersimpan dalam media elektroniknya. Selain itu, tujuan secara lengkap dari basis data (*database*) adalah sebagai berikut:

1. Kecepatan dan Kemudahan (*Speed*)

Basi data mendukung kecepatan dan kemudahan dalam mengakses data untuk membantu pekerjaan.

2. Efisiensi Ruang Penyimpanan (*Space*)

Menekan redudansi data dengan adanya relasi antar kelompok data yang saling berhubungan.

3. Keakuratan (*Accuracy*)

Pembuatan relasi antar data yang disertai penerapan aturan tipe data, domain data dan keunikan data yang secara ketat bisa diterapkan dalam basis data (*database*).

4. Ketersediaan (*Availability*)

Data dalam basis data (*database*) dapat dipilah menjadi data utama/master/referensi, data transaksi dan data histori, hingga data kadaluwarsa sesuai dengan kegunaannya.

5. Kelengkapan (*Completeness*)

Untuk mengakomodasi kebutuhan kelengkapan data yang semakin berkembang, maka kita tidak hanya dapat menambah *record-record* data, tetapi juga dapat melakukan perubahan struktur dalam basis data.

6. Keamanan (*Security*)

Penerapan aspek keamanan dapat ditentukan siapa saja yang boleh menggunakan *database* dan menentukan jenis operasi apa saja yang boleh digunakan.

7. Kebersamaan Pemakaian (*Sharability*)

Basis data (*database*) yang dikelola mendukung *multiuser* dan dapat dipakai bersama-sama oleh beberapa system aplikasi pada saat bersamaan.

### 2.4.5 Penerapan Basis Data (*Database*)

Basis data merupakan salah satu komponen utama dalam setiap system informasi sehingga banyak diterapkan pada bidang-bidang tertentu. Secara nyata, bidang-bidang yang memanfaatkan basis data antara lain sebagai berikut:

1. Kepegawaian

Dalam hal instansi baik formal maupun non-formal.

2. Pergudangan (*Inventory*)

Untuk perusahaan manufaktur (pabrik), grosir(*reseller*), apotik, dan lain-lain.

3. Reservasi

Untuk hotel, restoran, pesawat, kereta api, dan lain-lain.

4. Layanan Pelanggan (*Customer Care*)

Untuk perusahaan yang berhubungan dengan banyak pelanggan seperti bank, konsultan dan penyedia layanan dan lain-lain.

5. Perbankan dan Koperasi

Dalam pengelolaan data nasabah, data tabungan, data pinjaman, pembuatan laporan-laporan akuntansi, pelayanan informasi pada nasabah atau calon nasabah.

6. Asuransi

Dalam melakukan pengelolaan data nasabah atau data pembayaran premi, pemrosesan pengajuan klaim asuransi, dan lain-lain.

7. Rumah Sakit

Dalam melakukan pengelolaan *history* penyakit, pengelolaan pengobatan pasien, menangani pembayaran perawatan, melayani administrasi pasien dan lain-lain.

8. Pendidikan/Sekolah

Dalam melakukan pengelolaan data siswa/mahasiswa, penjadwalan kegiatan belajar mengajar, melayani pembayaran SPP, pengisian KRS online dan lain-lain.

#### **2.4.6 Database Management System (DBMS) dan Structure Query Language (SQL)**

*Database Management System* (DBMS) adalah *software* atau tools dari basis data (*database*) yang dibangun untuk melakukan pengelolaan *database* dengan operasi-operasi yang telah disediakan menggunakan

Bahasa tertentu yang telah disepakati (dalam hal ini adalah SQL). SQL merupakan singkatan dari *Structure Query Language*. Dalam Bahasa Inggris disebut SEQUEL atau ES-KYU-EL [11]. Bahasa ini merupakan standar yang digunakan untuk mengakses *database relasional*.

Standar SQL, mula-mula didefinisikan oleh ISO (*International Standards Organization*) dan ANSI (*the American National Standards Institute*), yang dikenal dengan sebutan SQL86. Namun, seiring berjalannya waktu, sejumlah standar baru telah ditetapkan.

Saat ini banyak perangkat lunak *database* yang menggunakan SQL sebagai perintah untuk mengakses *database*. Beberapa diantaranya sebagai berikut:

- DB2
- Ingres
- Informix
- ORACLE
- Microsoft Access
- MySQL
- PostgreSQL
- Rdb
- Sybase

Pada praktiknya implementasi SQL sangat bervariasi. Tidak semua fitur SQL didukung oleh vendor perangkat-lunak. Beberapa perintah SQL memiliki perbedaan. Sebagian lagi disebabkan sejumlah fitur memang diperuntukkan di masa mendatang, sehingga belum ada yang mengimplementasikannya. Walaupun begitu, pemahaman terhadap SQL akan mempermudah perpindahan dari suatu *database* ke *database* yang

lain karena secara fungsional banyak yang bersifat umum (dapat diterapkan dengan menggunakan perangkat lunak apa pun yang memang mendukungnya).

MySQL sebagai *database* server juga mendukung perintah SQL. Secara khusus, MySQL juga menambahkan sejumlah fungsi yang membuat perintah SQL pada MySQL sangat variatif. Tentu saja tambahan-tambahan tersebut akan membuat keleluasaan dalam mengakses *database* dan melakukan berbagai tindakan lainnya (misalnya untuk mengambil jam sekarang pada server).

Perintah yang dapat dipahami oleh *database* server MySQL disebut dengan istilah pernyataan [11]. Pernyataan adalah sebuah perintah yang dapat dikerjakan oleh MySQL dengan ciri-ciri diakhiri dengan tanda titik koma (;). Begitu anda mengetikkan titik koma kemudian menekan tombol Enter, program klien MySQL akan segera mengirimkannya ke *database* server MySQL akan segera menanggapi.



Gambar 2.20 Permintaan dari klien dan tanggapan dari server MySQL



## 2.4.7 Pembangunan Basis Data (*Database*) Menggunakan DBMS MySQL

### 2.4.7.1 Membuat *Database*

Sebelum mempraktikkan pembuatan table dan berbagai operasi yang terkait dengan table, anda perlu membuat sebuah *database* terlebih dahulu. Sebuah *database* dapat diciptakan dengan menggunakan pernyataan CREATE DATABASE. Perintah berikut akan menghasilkan *database* dengan nama **db\_inventory**:

```
CREATE DATABASE db_inventory;
```

**CATATAN :**

Jangan lupa untuk memberikan tanda titik koma (;) di akhir perintah atau pernyataan. Perintah SQL biasa disebut dengan istilah **pernyataan** (*statement*).

Jika anda menjumpai *output* berupa informasi seperti berikut:

```
Query OK, 1 row affected (0.016 sec)
```

Maka *database* **db\_inventory** berhasil dibuat.

### 2.4.7.2 Memilih *Database*

Sebelum anda bisa mengakses table ataupun hal yang berkaitan dengan *database*, anda perlu mengoneksikan diri ke *database* bersangkutan. Dalam hal ini MySQL *Client* untuk mengakses ke *database* yang hendak dipilih menggunakan perintah:

```
USE db_inventory;
```

### 2.4.7.3 Membentuk Tabel

Setelah *database* tercipta dan pengguna memilih *database* terkait (db\_inventory), maka anda bisa segera mempraktikkan pembuatan table. Dalam hal ini, anda bisa menggunakan pernyataan CREATE TABLE. Kaidah pernyataan ini sebagai berikut:

```
CREATE TABLE nama_tabel (  
  Nama_kolom_1 tipe_data ([ukuran]),  
  Nama_kolom_2 tipe_data ([ukuran]),  
  ...,  
  Nama_kolom_n tipe_data ([ukuran]) );
```

Contoh berikut digunakan untuk menciptakan table dengan nama **tbl\_barang**:

```
CREATE TABLE tbl_barang (  
  id          VARCHAR (5) NOT NULL PRIMARY KEY,  
  nama        VARCHAR(25),  
  jumlah      INTEGER,  
  ket         TEXT(100) );
```

Pada contoh table tbl\_barang terdiri atas 4 buah kolom (*field*) dengan rincian sebagai berikut:

- Kolom (*field*) **id** dengan tipe data VARCHAR (untuk menampung string) dengan Panjang (*length*) maksimal 5 karakter. Digunakan untuk menyatakan id barang. Dalam hal ini, kolom ini digunakan sebagai kunci primer (dinyatakan dengan PRIMARY KEY) dan kolom ini isinya tidak boleh kosong (NOT NULL).
- Kolom (*field*) **nama** dengan tipe data VARCHAR dengan panjang (*length*) maksimal 25 karakter.
- Kolom (*field*) **jumlah** dengan tipe data INTEGER (untuk menampung bilangan bulat) dengan panjang (*length*) default yaitu

maksimal 11 karakter (karena kita tidak menginisiasikan berapa maksimal *length* yang dibutuhkan).

- Kolom (*field*) **ket** dengan tipe data TEXT (digunakan untuk menampung string yang memiliki banyak karakter) dengan panjang (*length*) maksimal 100 karakter.

#### 2.4.7.4 Mengetahui Struktur Tabel

Dalam MySQL ketika anda ingin melihat stuktur tabel, anda dapat menggunakan perintah:

```
DESC nama_tabel;
```

Contoh:

```
DESC tbl_barang;
```

Memberikan hasil berikut:

Field	Type	Null	Key	Default	Extra
id	varchar(5)	NO	PRI	(NULL)	
nama	varchar(25)	YES		(NULL)	
jumlah	int(11)	YES		(NULL)	
ket	tinytext	YES		(NULL)	

Gambar 2.21 Struktur tabel diperoleh via DESC tbl\_barang

#### 2.4.7.5 Memasukkan Data

Tabel `tbl_barang` yang sebelumnya anda buat tentu saja masih berupa tabel yang kosong karena belum ada baris (*row*) data yang sama sekali tersimpan. Oleh karena itu, untuk memasukkan data ke dalam tabel tersebut, kita bisa gunakan pernyataan INSERT. Sebagai contoh, perintah berikut digunakan digunakan untuk memasukkan data barang bernama Buku.

```
INSERT INTO tbl_barang  
VALUES ('B0001','Buku',100,'Tersedia');
```

Urutan data pada VALUES sesuai dengan urutan nama kolom dalam pendefinisian struktur tabel. Untuk data jumlah kenapa tidak menggunakan tanda petik (‘’)? Hal itu disebabkan pada MySQL atau DBMS lain yang menggunakan SQL sebagai Bahasa standar dalam mengelola *database* bahwa tipe data integer atau bilangan bulat berupa angka tidak perlu menggunakan tanda petik, tanda petik tersebut diperuntukkan untuk nilai yang berupa string.

Pernyataan INSERT memungkinkan penambahan baris dengan kolom-kolom tertentu saja yang diisi. Kolom-kolom yang tidak disebutkan akan diisi dengan NULL. Perlu diperhatikan, kolom-kolom yang tidak disebutkan dalam INSERT haruslah yang tidak mengandung NOT NULL dalam pendefinisian struktur tabel.

#### **2.4.7.6 Melihat Isi Tabel**

Sejauh ini anda telah memasukkan data dua buah barang. Kemudian, bagaimana cara anda melihat isi tabel barang yang baru saja anda isi sebelumnya? MySQL memiliki perintah yang dapat kita gunakan ketika hendak melihat isi data pada suatu tabel yaitu dengan perintah SELECT. Sebagai contoh, cobalah pernyataan berikut:

```
SELECT * FROM tbl_barang;
```

Pada pernyataan ini, tanda “\*” merupakan tanda untuk memilih semua kolom sedangkan “tbl\_barang” yang terletak setelah kata FROM menyatakan nama tabel yang diproses.

Contoh berikut memberikan gambaran hasil (*output*) yang diperoleh dari pernyataan di atas.

👉 id	nama	jumlah	ket
B0001	Buku	100	Tersedia
B0002	Pulpen	100	Tersedia

Gambar 2.22 Isi tbl\_barang ditampilkan dengan perintah SELECT

**CATATAN :**

Ada banyak variasi dalam menggunakan pernyataan SELECT. Untuk penggunaan SELECT yang lebih kompleks mungkin akan dibahas pada buku tersendiri tentang *Database*.

#### 2.4.7.7 Mengganti Struktur Tabel

Adakalanya suatu ketika diperlukan untuk mengubah struktur suatu tabel. Oleh karena itu, MySQL menyediakan perintah yang dapat anda gunakan, untuk menjalankan perintah tersebut gunakan pernyataan ALTER TABLE. Sebagai contoh, anda bisa mempraktikkan pengubahan nama kolom **ket** menjadi **keterangan**. Perintah yang diperlukan:

```
ALTER TABLE tbl_barang  
CHANGE ket keterangan TEXT(100);
```

Pada contoh di atas, nama kolom **ket** dalam tabel tbl\_barang diubah menjadi **keterangan** namun dengan tetap memeprtahankan tipe data dan *length*-nya.

Dalam kasus lainnya, terkadang kita bisa saja salah mendefinisikan suatu tipe data pada suatu kolom. Oleh karena itu, anda dapat menggunakan perintah yang sama seperti sebelumnya. Sebagai contoh:

```
ALTER TABLE tbl_barang  
CHANGE keterangan keterangan ENUM('Tersedia','Tidak Tersedia')  
NOT NULL;
```

Membuat tipe data kolom keterangan menjadi ENUM ('Tersedia','Tidak Tersedia') dan kolom harus diisi (NOT NULL). Dengan menggunakan tipe data seperti ini, maka kolom keterangan hanya bisa diisi dengan salah satu nilai yang tercantum dalam ENUM, yaitu "Tersedia" atau "Tidak Tersedia", adapun penulisannya harus sesuai (*case sensitive*).

**CATATAN :**

Ketika kolom **keterangan** diisi selain dengan "Tersedia" atau "Tidak Tersedia" maka MySQL akan mengubah isian di luar nilai dari pada ENUM dengan spasi, sehingga anda akan mendapati isi pada baris tertentu pada kolom keterangan kosong (Bukan Nilai yang diinputkan sebelumnya).

Selain pernyataan sebelumnya yaitu:

```
ALTER TABLE tbl_barang  
CHANGE keterangan keterangan ENUM('Tersedia','Tidak Tersedia')  
NOT NULL;
```

Hasilnya adalah mengubah definisi suatu kolom, dapat disederhanakan menjadi:

```
ALTER TABLE tbl_barang  
CHANGE keterangan ENUM('Tersedia','Tidak Tersedia') NOT NULL;
```

MySQL juga mendukung pernyataan ALTER TABLE untuk menambahkan kolom, contoh:

```
ALTER TABLE tbl_barang  
ADD COLUMN id_jenis_barang INTEGER NOT NULL;
```

Dengan cara seperti di atas, kolom bernama **id\_jenis\_barang** dengan tipe data INTEGER dengan sendirinya ditambahkan pada tabel `tbl_barang`. Gambar 2.23 Struktur tabel `tbl_barang` setelah kolom `id_jenis_barang` ditambahkan

memperlihatkan keadaan struktur tabel setelah perintah `DESC tbl_barang` diberikan.

Field	Type	Null	Key	Default	Extra
<code>id</code>	<code>varchar(5)</code>	NO	PRI	(NULL)	
<code>nama</code>	<code>varchar(25)</code>	YES		(NULL)	
<code>jumlah</code>	<code>int(11)</code>	YES		(NULL)	
<code>ket</code>	<code>tinytext</code>	YES		(NULL)	
<code>id_jenis_barang</code>	<code>int(11)</code>	NO		(NULL)	

Gambar 2.23 Struktur tabel `tbl_barang` setelah kolom `id_jenis_barang` ditambahkan

Tampaknya bahwa kolom **id\_jenis\_barang** telah ditambahkan dan diletakkan diakhir pada struktur tabel.

**CATATAN :**

Ketika kolom **id\_jenis\_barang** dengan perintah berikut:

```
ALTER TABLE tbl_barang
ADD COLUMN id_jenis_barang INTEGER NOT NULL;
```

Tentunya hal ini akan membuat kolom **id\_jenis\_barang** secara default ditambahkan pada akhir sruktur tabel. Namun, apa yang terjadi jika `id_jenis_barang` ingin ditambahkan setelah kolom `id`? MySQL menyediakan pernyataan `BEFORE/AFTER`. `BEFORE` digunakan untuk menambahkan kolom sebelum kolom acuan dan `AFTER` digunakan untuk menambahkan kolom setelah kolom acuan. Penggunaannya dapat dilihat dari perintah berikut:

```
ALTER TABLE tbl_barang
ADD COLUMN id_jenis_barang INTEGER NOT NULL
AFTER id;
```

Pernyataan ALTER TABLE juga bisa digunakan untuk menghapus suatu kolom yang dikehendaki.

Contoh:

```
ALTER TABLE tbl_barang  
DROP id_jenis_barang;
```

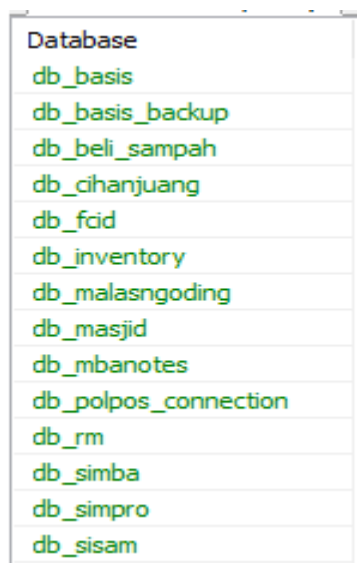
Perintah di atas membuat kolom bernama “**id\_jenis\_barang**” yang sebelumnya terdapat pada tabel tbl\_barang dihapus.

#### 2.4.7.8 Mengetahui Daftar *Database* dan Tabel

Ketika anda menggunakan program klien MySQL dan anda hendak ingin menampilkan daftar *database*, anda dapat menggunakan perintah berikut:

```
SHOW DATABASES;
```

Gambar 2.24 memperlihatkan contoh hasil setelah perintah di atas diberikan:



Database
db_basis
db_basis_backup
db_beli_sampah
db_cihanjuang
db_fcid
db_inventory
db_malasngoding
db_masjid
db_mbanotes
db_polpos_connection
db_rm
db_simba
db_simpro
db_sisam



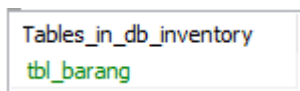
Gambar 2.24 Daftar *database* pada MySQL via SHOW DATABASES.

Selain itu, untuk mengetahui daftar tabel yang terdapat pada suatu *database*, anda bisa memberikan menggunakan perintah berikut:

```
SHOW TABLES;
```

Gambar 2.25 Daftar tabel yang diperoleh via SHOW TABLES

Memberikan contoh hasil dari perintah di atas guna menampilkan tabel yang terdapat pada *database* db\_inventory.



```
Tables_in_db_inventory
tbl_barang
```

Gambar 2.25 Daftar tabel yang diperoleh via SHOW TABLES

Hasil pada Gambar 2.25 menyatakan bahwa hanya ada satu tabel pada *database* db\_inventory.

#### 2.4.7.9 Menghapus Tabel

Sebagai manusia yang pastinya akan melakukan kesalahan, maka kerap kali ada satu atau beberapa tabel yang seharusnya tidak perlukan tapi sebelumnya sudah pernah dibuat/diciptakan. Oleh karena itu, MySQL menyediakan pernyataan DROP TABLE yang dapat anda gunakan untuk menghapus tabel yang sudah tidak dibutuhkan lagi. Untuk menggunakannya, anda bisa menggunakan perintah:

```
DROP TABLE tbl_barang;
```

Pernyataan di atas menghapus tabel tbl\_barang yang telah dibuat sebelumnya. Jangan ragu-ragu melakukan perintah tersebut supaya anda mengerti dan tahu betul efeknya sekarang, sehingga tidak melakukan hal

serupa pada kesempatan lain ketika sebenarnya anda tidak berniat menghapus tabel tersebut. Sekali anda mengeksekusi perintah tersebut, tabel akan hilang dan tidak ada cara untuk mengembalikannya! Namun untuk keperluan latihan, sekali lagi jangan ragu-ragu untuk melakukannya.

#### **2.4.7.10 Menghapus Database**

Apabila anda bermaksud untuk menghapus suatu *database* pada MySQL karena suatu keadaan atau keperluan tertentu, anda dapat melakukannya dengan memakai pernyataan DROP DATABASE. Contoh penerapannya sebagai berikut:

```
DROP TABLE db_inventory;
```

Pernyataan di atas menghapus *database* db\_inventory yang telah dibuat sebelumnya. Namun, sangat disarankan untuk tidak melakukan hal ini karena dalam suatu industri dengan skala besar, menghapus *database* tentunya akan menghapus semua data yang terkandung di dalamnya. Namun, jika anda ingin menghapus *database* tersebut karena keperluan rekonstruksi dan lainnya, maka cukup lakukan backup terlebih dahulu untuk mencegah membangun *database* dari awal.

### **2.5 Framework: Codeigniter**

#### **2.5.1 Say Hello Dengan Framework Codeigniter**

*Welcome to Jumanji. Eh, salah hahaha, maksud saya Selamat datang* di pembahasan tentang Framework Codeigniter. Sebelum era *Object Oriented Programming* (OOP) memegang kendali dunia pemrograman yang populer saat ini. *Programmer* dulunya masih menggunakan paradigma pemrograman Struktural. Apa itu paradigma pemrograman struktural? Lalu, apa itu paradigma pemrograman OOP? Mengapa OOP

bisa menggantikan posisi paradigma pemrograman struktural? Baiklah, akan segera kita bahas.

Paradigma pemrograman struktural atau terstruktur yaitu suatu bahasa pemrograman yang sangat mendukung penuh dalam pembuatan program sebagai kumpulan dari prosedur untuk memecahkan masalah. Prosedur/langkah-langkah ini lah nantinya akan saling memanggil dan dipanggil dari manapun di dalam sebuah program tersebut dengan perintah berbeda dari setiap syntak pemanggilan program [12]. Namun, seiring berjalannya waktu paradigma pemrograman ini sedikit mulai ditinggalkan dan digantikan paradigma baru yaitu OOP. OOP menutup kekurangan dari Pemrograman Terstruktur.

OOP adalah sebuah istilah yang diberikan kepada bahasa pemrograman yang menggunakan tehnik berorientasi atau berbasis pada sebuah obyek dalam pembangunan program aplikasi, maksudnya bahwa orientasi pembuatan program tidak lagi menggunakan orientasi linear melainkan berorientasi pada objek-objek yang terpisah-pisah. Suatu perintah dalam bahasa ini diwakili oleh sebuah Obyek yang didalamnya berisi beberapa perintah-perintah standar sederhana. Obyek ini dikumpulkan dalam Modul form atau Report atau modul lain dan disusun didalam sebuah project [13].

Pada pembahasan ini sesuai dengan judul akan di jelaskan tentang pengertian dan cara menggunakan codeigniter. tentu anda pasti sudah sering mendengar tentang framework codeigniter. Baik itu dari teman, kampus, sekolah atau dari forum-forum pemrograman di internet. Tapi tidak sedikit juga yang masih bertanya-tanya tentang apa sih framework codeigniter itu?

Oleh sebab itu, pada pembahasan khusus BAB *framework codeigniter* ini ini kita akan sedikit berkenalan dengan codeigniter. Dan belajar bagaimana cara melakukan instalasi codeigniter. Kita akan mulai dengan sedikit pengertian dari apa itu *framework*? dan apa itu *framework codeigniter*?

*Baikklah, tanpa basa basi lagi, langsung saja, mari kita mulai...*

*Framework* adalah kumpulan intruksi-intruksi yang di kumpulkan dalam *class* dan *function-function* dengan fungsi masing-masing untuk membantu memudahkan developer/programmer dalam pemanggilannya tanpa harus menuliskan syntax program yang sama secara berulang-ulang. hal ini memiliki kegunaan untuk menghemat waktu dan mencegah penulisan syntax secara berulang-ulang agar tercipta-nya *source code* (kode program) yang bersih dan terstruktur.

Codeigniter adalah sebuah *framework* PHP yang bersifat *open source* dan menggunakan metode Model, View dan Controller (MVC). codeigniter bersifat free alias tidak berbayar jika anda menggunakannya. *Framework* codeigniter di buat dengan tujuan sama seperti *framework* lainnya yaitu untuk memudahkan developer atau programmer dalam membangun sebuah aplikasi berbasis web tanpa harus membuat-nya dari awal.

*Framework* codeigniter juga termasuk salah satu *framework* PHP yang paling banyak digunakan oleh web developer untuk membangun sebuah aplikasi berbasis *website*.

Pengembangan codeigniter oleh developer-nya juga sangat baik. Dan memiliki dokumentasi yang baik dan jelas sebagai panduan kita menggunakan *framework* codeigniter.

## 2.5.2 Pengertian Model, View dan Controller (MVC)

Pada pengertian codeigniter sebelumnya dijelaskan bahwa codeigniter menggunakan metode MVC. Apa itu MVC? kita juga harus mengetahui apa itu MVC sebelum masuk dan lebih dalam mengoprek codeigniter.

MVC adalah teknik atau konsep yang memisahkan komponen utama menjadi tiga komponen yaitu model, view dan controller.

### 1. Model

Model merupakan bagian penanganan yang berhubungan dengan pengolahan atau manipulasi *database*. seperti misalnya mengambil data dari *database*, meng-*input* dan pengolahan *database* lainnya. semua intruksi atau fungsi yang berhubungan dengan pengolahan *database* di letakkan di dalam model.

### 2. View

View merupakan bagian yang menangani halaman *user interface* atau halaman yang muncul pada *user* (pada *browser*). Tampilan dari *user interface* di kumpulkan pada view untuk memisahkannya dengan controller dan model sehingga memudahkan *web designer* dalam melakukan pengembangan tampilan halaman *website*.

### 3. Controller

Controller merupakan kumpulan intruksi aksi yang menghubungkan model dan view, jadi *user* tidak akan berhubungan dengan model secara langsung, intinya data yang tersimpan di *database* (model) di ambil oleh controller dan kemudian controller pula yang menampilkan-nya ke view. Jadi controller lah yang mengolah intruksi.

Dari penjelasan tentang model view dan controller sebelumnya dapat disimpulkan bahwa controller sebagai penghubung view dan model. Jadi,

misalnya pada aplikasi yang menampilkan data dengan menggunakan metode konsep MVC, controller memanggil intruksi pada model yang mengambil data pada *database*, kemudian controller yang meneruskannya pada view untuk di tampilkan. Jadi sudah jelas dan sangat mudah dalam pengembangan aplikasi dengan cara MVC ini karena *web designer* atau *front-end developer* tidak perlu lagi berhubungan dengan controller, dia hanya perlu berhubungan dengan view untuk *men-design* tampilann aplikasi, karena *back-end developer* yang menangani bagian controller dan modelnya. jadi pembagian tugas pun menjadi mudah dan pengembangan aplikasi dapat dilakukan dengan cepat dan terstruktur.

### 2.5.3 Kelebihan *Framework Codeigniter*

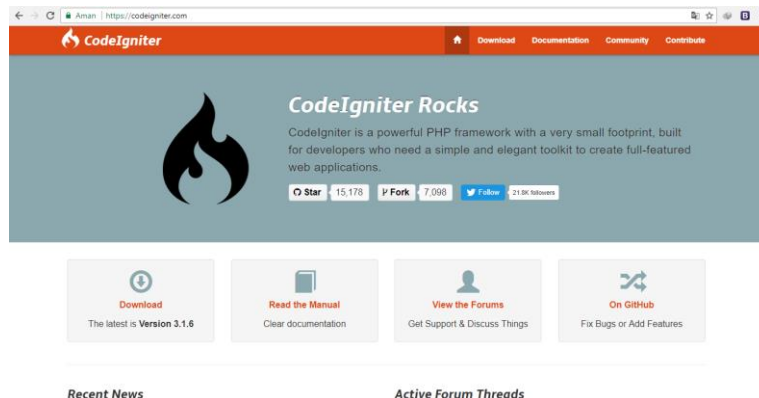
Adapun beberapa kelebihan dari codeigniter adalah :

- *Syntax* yang terstruktur dan bersih
- Kemudahan dalam menggunakannya
- Codeigniter menyediakan fasilitas helper dan library yang dapat membantu developer dalam membuat pagination, session, manipulasi url dan lainnya secara cepat yang akan dipelajari pada tutorial codeigniter selanjutnya.
- keamanan yang sudah lumayan karena user atau pengakses aplikasi tidak berhubungan langsung dengan *database*.

Setelah selesai berkenalan dengan framework codeigniter, selanjutnya kita akan masuk ke tahap belajar cara menginstal framework codeigniter dan mulai menggunakannya.

### 2.5.4 Instalasi *Framework Codeigniter*

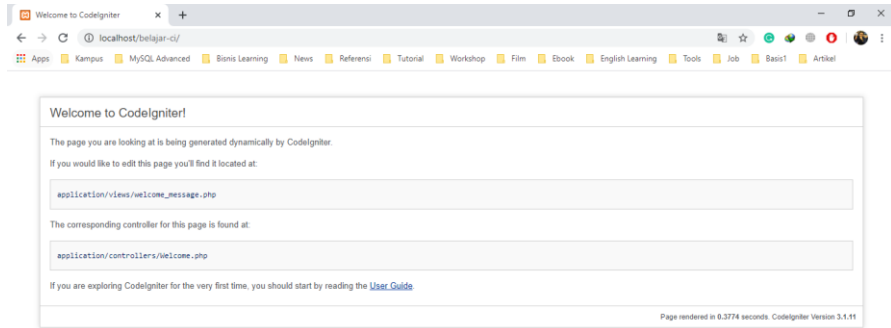
Instalasi Framework codeigniter caranya sangatlah mudah. anda dapat langsung mendownload CodeIgniter pada situs resminya di [www.codeigniter.com](http://www.codeigniter.com).



Gambar 2.26 Halaman utama *website* resmi codeigniter

Setelah selesai mendownload *framework* codeigniter, kemudian ekstrak di folder htdocs (localhost) jika menggunakan XAMPP. Lalu, ubah nama foldernya dengan nama project yang ingin anda buat. Pada contoh ini anda dapat mengubah nama foldernya dari “codeigniter.3.1.5” menjadi “belajar-ci”.

Terakhir, Anda dapat langsung menjalankannya pada *browser* kesayangan anda dengan mengetikan alamat “http://localhost/belajar-ci” pada *address bar browser* anda.



---

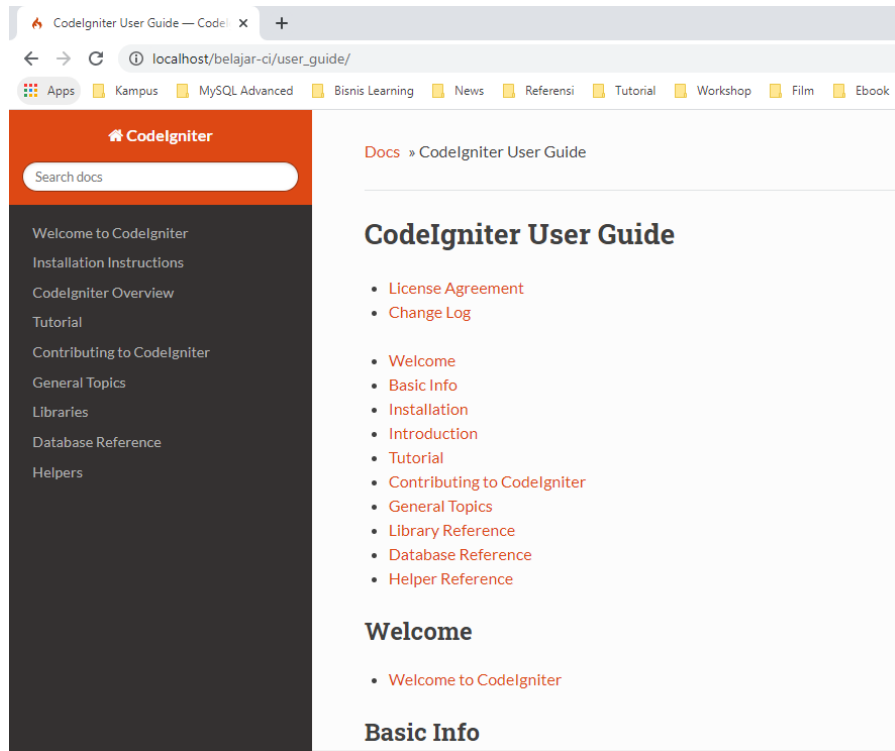
Gambar 2.27 Halaman *welcome codeigniter* dari project belajar-ci

Jika berhasil, maka akan muncul tampilan seperti *Gambar 2.27* Halaman *welcome codeigniter* dari project belajar-ci

yang menandakan bahwa codeigniter telah berhasil di *install* dan siap untuk di gunakan. Tampilan tersebut merupakan halaman *default* dari codeigniter saat pertama kali di-*install*.

Secara *default* Codeigniter menyediakan panduan cara penggunaan codeigniter pada folder codeigniter yang anda *download* sebelumnya, yaitu terdapat pada folder *user\_guide*. Adapun untuk mempelajari panduan dari *user guide* tersebut anda dapat langsung menjalankannya pada browser dengan mengetikkan alamat “*http://localhost/belajar-ci/user\_guide/*” pada *address bar*.





Gambar 2.28 Halaman *user-guide codeigniter* dari project belajar-ci

Segala tentang penggunaan codeigniter bisa di baca pada panduan yang telah di sediakan seperti pada Gambar 2.28.

### 2.5.5 Apa itu Uri Segment?

Pada tutorial codeigniter di buku akan menjelaskan tentang salah satu yang paling terpenting dalam konsep penggunaan codeigniter, yaitu Uri Segment pada codeigniter. Jadi pada tutorial ini kita akan belajar tentang pengenalan dari uri segment codeigniter. apa pengertian uri segment pada codeigniter?

URI adalah singkatan dari *Uniform Resource Identifier*. bisa kita bilang URI yang membantu kita dalam mengambil data melalui url codeigniter. Cara penyebutan uri segment pada codeigniter sendiri

misalnya segment 1, segment 2, segment 3 dan seterusnya. jika teman-teman perhatikan pada url codeigniter pada project codeigniter anda, pasti menjumpai index.php lalu di lanjutkan dengan nama class codeigniter anda kan?. index.php di sini tidak berpengaruh dengan uri segment bahkan index.php dapat kita hilangkan untuk membuat url codeigniter kita lebih rapi.

Untuk tutorial menghilangkan index.php pada codeigniter akan dibahas pada tutorial selanjutnya. Kembali lagi ke penjelasan URI segment, yang di katakan sebagai segment 1 pada codeigniter adalah nama class atau controller yang sedang anda jalankan. segment ke 2 adalah nama method atau function pada dari class/controller anda. dan segment 3 berupa data berbentuk id yang di kirim.

Jadi segment codeigniter di hitung dari setelah index.php pada codeigniter. katakan saja misalnya anda memiliki url seperti contoh berikut ini:

`http://localhost/belajar-ci/index.php/belajar/codeigniter/1`

Baiklah, mari kita perhatikan pada contoh url di atas.

segment 1 = belajar

segment 2 = codeigniter

segment 3 = 1

Untuk memberikan penjelasan yang lebih rinci berikut ini adalah rumus cara mudah mengerti tentang url codeigniter.

`http://localhost/belajar-ci/index.php/controller/method/id`

Data yang di kirimkan melalui url di codeigniter biasanya terletak pada segment 3. Perhatikan contoh berikut ini.

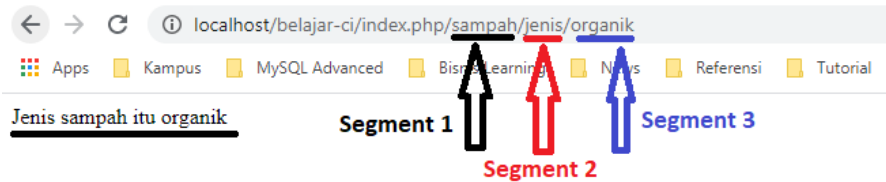
Pertama, buatlah sebuah controller baru dengan nama “Sampah.php” (tanpa tanda petik).

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Sampah extends CI_Controller
{
    public function jenis(){
        echo "Jenis sampah itu " . $this->uri->segment('3');
    }
}
?>
```

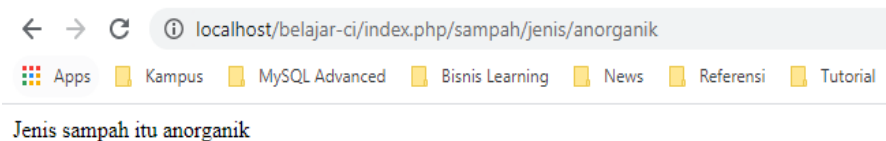
Coba anda perhatikan pada contoh di atas. kita membuat sebuah controller dengan nama sampah.php dan method/function jenis. Pada function jenis ini saya menampilkan isi dari URI segment ketiga ( \$this->uri->segment('3') ).

Oleh karena itu, untuk mengakses uri segment ini, anda dapat menggunakan syntax “\$this->uri->segment(‘urutan segment beberapa’)”. Maka hasilnya akan terlihat pada Gambar 2.29 Belajar uri segment: sampah/jenis/organik



Gambar 2.29 Belajar uri segment: sampah/jenis/organik

Dari Gambar 2.29, bisa anda perhatikan pada segment 3, segment 3 yang digaris bawahhi dengan tanda berwarna biru adalah “organik”. Maka hasil yang muncul adalah sampah itu termasuk jenis samoah organik. Untuk mengujinya lagi, sekarang coba anda ubah pada segment 3 dari “organik” menjadi “anorganik”.



Gambar 2.30 Belajar uri segment: sampah/jenis/anoranik

Perhatikan lagi contoh berikut ini dan semoga dapat memberi pemahaman lebih mendalam tentang uri segment pada codeigniter.

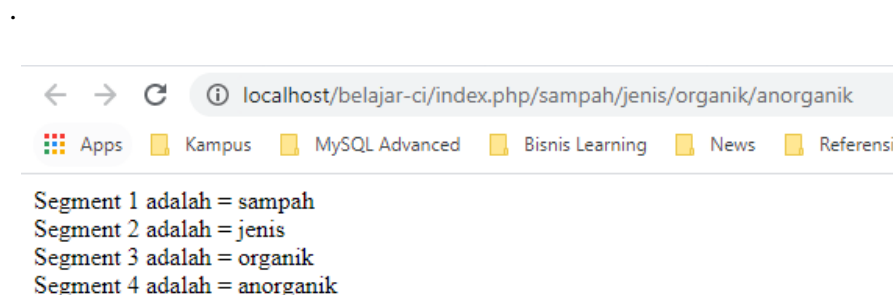
```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Sampah extends CI_Controller
{
    public function jenis(){
        echo "Segment 1 adalah = " . $this->uri->segment('1') .
        "<br/>";
        echo "Segment 2 adalah = " . $this->uri->segment('2') .
        "<br/>";
        echo "Segment 3 adalah = " . $this->uri->segment('3') .
        "<br/>";
        echo "Segment 4 adalah = " . $this->uri->segment('4') .
        ..
    }
}
```

Kemudian, silahkan akses alamat berikut:

<http://localhost/belajar-ci/index.php/sampah/jenis/organik/anorganik>

Maka akan muncul hasil seperti pada Gambar 2.31 Belajar uri segment: sampah/jenis/organik/anorganik



Gambar 2.31 Belajar uri segment: sampah/jenis/organik/anorganik

## 2.5.6 Berkenalan Dengan *Controller* di Codeigniter

Pada tutorial kali ini tentang Berkenalan dengan Controller Codeigniter ini akan menjelaskan tentang cara membuat controller pada

codeigniter, pengaturan dasar dan cara menggunakan controller pada codeigniter. Seperti yang sudah di jelaskan pada tutorial sebelumnya tentang pengertian dan cara menggunakan codeigniter bahwa codeigniter menggunakan metode MVC untuk menciptakan kode atau *syntax* yang bersih.

Sebelum memasuki pengenalan ini, anda di asumsikan untuk memahami konsep OOP terlebih dahulu sebelum mulai mempelajari codeigniter. karena codeigniter di bangun dengan menggunakan konsep OOP.

Controller dalam bahasa indonesia yang berarti pengontrol atau pengatur, yang di maksud pengontrol atau pengatur di sini adalah controller yang berperan sebagai pengatur dari aksi pada aplikasi yang di bangun, seperti misalnya jika di codeigniter controller yang berperan paling penting dari mulai mengirimkan parameter, menangani inputan form (*form handling*), mengatur view dan model, mengaktifkan atau memanggil library dan helper codeigniter dan masih banyak lagi peran controller dalam membangun sebuah aplikasi dengan menggunakan *framework php* codeigniter.

Untuk memahami cara penggunaan controller codeigniter sekarang coba jalankan project Codeigniter yang sudah di install pada localhost, di buku ini nama folder project *root*-nya adalah “belajar-ci”, sehingga untuk menjalankannya pada browser dapat di akses langsung dengan alamat <http://localhost/belajar-ci/>.

Setelah berhasil, maka akan muncul halaman *welcome* codeigniter, hal ini terjadi karena pengaturan default controller yang di jalankan pertama kali pada konfigurasi routes adalah controller welcome. dapat di lihat pada `application/controllers/welcome.php`.

Controller ini memanggil view `welcome_message.php` yang terletak pada folder `view`. view tidak kita jelaskan secara mendetail pada sub judul ini. melainkan akan kita bahas pada tutorial selanjutnya di ebook ini tentang cara membuat view pada codeigniter.

Secara *default* saat pertama kali meng-*install* codeigniter dan menjalankannya, controller codeigniter yang pertama kali di jalankan adalah controller `welcome`, saat anda menjalankannya pada *browser* maka yang pertama kali di jalankan adalah controller `welcome`. Hal ini dapat di lihat pada pengaturan routes codeigniter yang menetapkan controller `welcome` sebagai controller default yang di jalankan. Buka project controller yang sudah anda letakkan pada localhost sesuai dengan tutorial sebelumnya. Kemudian, buka *file* pengaturan routes.php yang terletak di `application/config/routes.php`

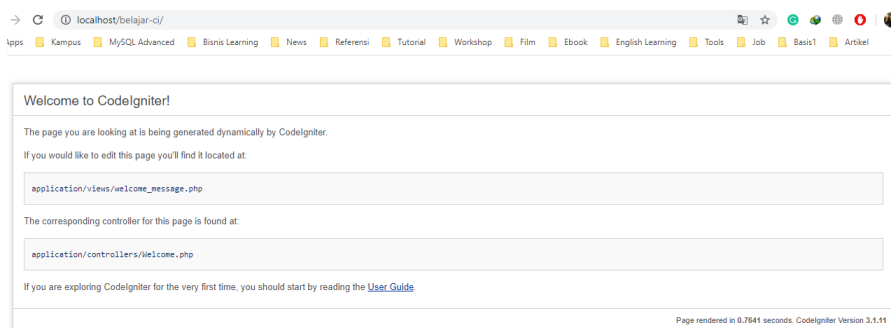
#### **Application/config/routes.php**

```
$route['default_controller'] = 'welcome';  
$route['404_override'] = '';  
$route['translate_uri_dashes'] = FALSE;
```

Dapat dilihat pada pengaturan routes codeigniter di atas, pada pengaturan **default\_controller** di setting controller “ `welcome` “, selain itu, terdapat juga pengaturan untuk menangani halaman 404 atau halaman yang di tampilkan jika tidak di temukannya data suatu url. Anda dapat mengatur halaman 404 anda dengan cara memasukkan controller yang ingin anda jadikan untuk menetapkan halaman 404 pada aplikasi anda.

Pada pengaturan `$route['translate_uri_dashes']=false` adalah pengaturan untuk menetapkan nilai true atau false untuk izin penggunaan tanda “-” (dash) pada controller di url pada saat dijalankan. Namun pada buku ini kita tidak akan terlalu banyak membahasnya.

Controller *default* ini dapat diakses secara langsung dengan alamat <http://localhost/belajar-ci>.



Gambar 2.32 Berkenalan dengan controller: default controller

### 2.5.6.1 Cara Membuat Controller Pada Codeigniter

Jika sebelumnya kita menggunakan controller *default* bawaan saat pertama kali codeigniter di-*install*, maka sekarang kita akan belajar membuat controller baru.

Untuk membuat controller baru pada codeigniter, Silahkan buat sebuah file baru pada `application/controllers/`. untuk contoh di sini kita akan membuat controller “belajar”. Jadi kita harus membuat file `belajar.php` di `application/controllers/`.

Selanjutnya kita akan mendefinisikan class belajar dalam file “**belajar.php**”.



## Application/controllers/belajar.php

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Belajar extends CI_Controller
{
    function __construct(){
        parent::__construct();
    }

    public function index(){
        echo "ini method index pada controller belajar";
    }

    public function halo(){
        echo "ini method halo pada controller belajar";
    }
}
?>
```

### Penjelasan:

Mari perhatikan pada controller yang telah dibuat. Pertama kali yang harus dilakukan adalah meng-*extends* controller baru ini dengan CI\_Controller.

```
Class Belajar extends CI_Controller {
```

Dalam hal ini, nama class harus diawali dengan huruf besar seperti contoh di atas. Nama class harus sesuai dengan nama *file* controller yang dibuat tadi. Karena tadi kita membuat controller **belajar.php** maka penulisan class-nya seperti di atas.

```
defined('BASEPATH') OR exit('No direct script access allowed');
```

*Syntax* di atas berfungsi untuk mencegah akses langsung pada *file* controller. Kemudian, anda dapat membuat function `construct()` untuk menjalankan fungsi yang anda inginkan pada saat controller diakses. biasanya pada fungsi `construct()` di letakkan fungsi untuk memanggil helper atau library (akan dijelaskan pada sub selanjutnya).

Kemudian, untuk contoh method pada controller dalam buku ini, saya membuat dua buah method yaitu `index` dan `halo`.

```
public function index(){
    echo "ini method index pada controller belajar";
}

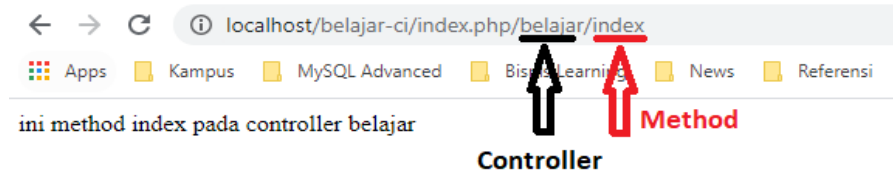
public function halo(){
    echo "ini method halo pada controller belajar";
}
```

Untuk menjalankan method `index` anda bisa mengaksesnya menggunakan alamat:

```
http://localhost/belajar-ci/index.php/belajar
```

Untuk method `index` boleh tidak dituliskan karena pada saat controller di akses, maka yang pertama di jalankan adalah method `index` atau bisa juga dengan menambahkan `index` langsung seperti alamat berikut:

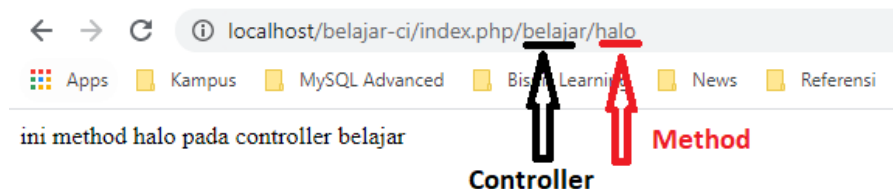
```
http://localhost/belajar-ci/index.php/belajar/index
```



Gambar 2.33 Berkenalan dengan controller: belajar/index

Kemudian, untuk mengakses function atau method halo dapat di akses dengan alamat:

<http://localhost/belajar-ci/index.php/belajar/halo>



Gambar 2.34 Berkenalan dengan controller: belajar/halo

Jika anda ingin menjadikan controller belajar ini sebagai controller *default* anda dapat melakukan *setting*-nya di `application/config/routes.php`.

Ubah pengaturan `default_controller` menjadi belajar (nama controller yang ingin di set default).

```
$route['default_controller'] = 'belajar';
```

## 2.5.7 Berkenalan Dengan View

Setelah mempelajari tentang cara membuat controller pada codeigniter dan pengertian serta cara menggunakan codeigniter pada tutorial sebelumnya, Maka untuk kelanjutan dari tutorial codeigniter dasar kita akan membahas tentang cara membuat view pada codeigniter. Seperti yang sudah di jelaskan bahwa view pada codeigniter berguna untuk meng-*handle* urusan tampilan dari aplikasi yang kita buat menggunakan codeigniter.

#### 2.5.7.1 Pengenalan View

*View* bertugas menampilkan *user interface* kepada *user*, sesuai dengan fungsi MVC yang memisahkan model, controller dan view sehingga memudahkan developer atau programmer dalam membuat pembaharuan serta dapat memudahkan developer bekerja dalam tim pada saat membangun aplikasi berbasis web menggunakan codeigniter. Sehingga *web designer* yang menangani tampilan *user interface* tidak perlu berurusan dengan *back-end* karena jatah kerja *web designer* ada pada view yang terletak pada folder view pada codeigniter. `application/view/`.

#### 2.5.7.2 Cara Membuat View

Cara membuat *view* pada codeigniter caranya sangat mudah, anda hanya perlu membuat file baru pada folder view. Yaitu di `application/view/`.

Kemudian, buat view dengan nama *file*-nya terserah anda. Di contoh ini, kita akan membuat view dengan nama `view_belajar.php` “`application/view/view_belajar.php`” langkah selanjutnya adalah membuat isi dari view. **`application/view/view_belajar.php`**.

```
<html>
<head>
  <title>Cara Membuat View Pada CodeIgniter</title>
</head>
<body>
  <h1>Cara Membuat View Pada CodeIgniter</h1>
  <h2>Ini adalah view view_belajar.php</h2>
  <h3>Ini adalah view yang di tampilkan pada controller belajar,
method halo</h3>
</body>
</html>
```

Dan untuk cara memanggil atau menampilkan view nya silahkan buka controller anda, di sini saya membuat controller belajar.php dan membuat method halo.

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Belajar extends CI_Controller
{
  function __construct(){
    parent::__construct();
  }

  public function index(){
    echo "ini method index pada controller belajar";
  }

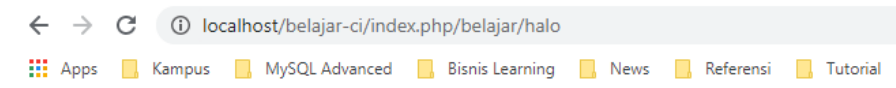
  public function halo(){
    $this->load->view('view_belajar');
  }
}
?>
```

Perhatikan pada method halo di atas. Pada method halo ini, controller memerintahkan system untuk memanggil komponen view view\_belajar.php untuk ditampilkan di *user interface*.

```
public function halo(){
    $this->load->view('view_belajar');
}
```

Pada kasus kali ini, anda tidak perlu lagi menuliskan ekstensi ‘.php’ pada saat memanggil view. syntax `$this->load->view()` dengan otomatis mengakses folder `application/view` codeigniter.

Dan coba kita jalankan method `halo` nya pada browser untuk menampilkan view `view_belajar.php` dengan mengakses alamat <http://localhost/belajar-ci/index.php/belajar/halo>.



## Cara Membuat View Pada CodeIgniter

**Ini adalah view `view_belajar.php`**

**Ini adalah view yang di tampilkan pada controller belajar, method `halo`**

Gambar 2.35 Berkenalan dengan view: `view_belajar.php`

Dapat di perhatikan pada contoh di atas. view `view_belajar.php` pun muncul pada saat kita akses method `halo`. Kemudia muncul pertanyaan, Bagaimana cara membuat view di dalam subfolder?. Misalnya untuk sebagian view ada yang ingin anda kumpulkan pada sebuah folder untuk membuat view lebih rapi dan terstruktur. misalnya anda meletakkan view yang ingin anda panggil di dalam suatu folder. Jika anda meletakkan view `view_belajar.php` di dalam folder “front” di dalam view codeigniter.

**`application/view/front/view_belajar.php`**

Dan cara memanggil view `view_belajar.php` yang terletak di dalam folder `front`, caranya anda hanya perlu memasukkan juga nama folder tempat view yang anda ingin ditampilkan pada berada.

```
$this->load->view('front/view_belajar');
```

### 2.5.7.3 Cara Parsing Data Ke View

Untuk memarsing data dari controller ke view anda dapat memarsingnya dengan menggunakan bantuan array. Jadi data yang akan diparsing kita masukkan ke array.

Perhatikan contoh cara memarsing data ke view codeigniter berikut ini.

Buka controller **belajar.php**

```

<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Belajar extends CI_Controller
{
    function __construct() {
        parent::__construct();
    }

    public function index() {
        echo "ini method index pada controller belajar";
    }

    public function halo() {
        $data['nama_web'] = "Webkita.com";
        $this->load->view('view_belajar',$data);
    }
}
?>

```

### **Application/view/view\_belajar.php**

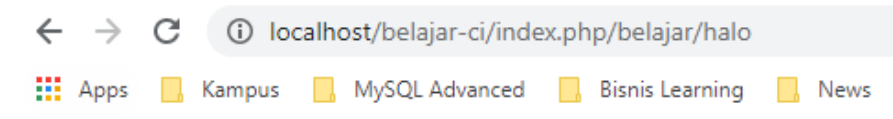
```

<html>
<head>
    <title>Cara Membuat View Pada CodeIgniter</title>
</head>
<body>
    <h1><?php echo $nama_web; ?></h1>
</body>
</html>

```

Untuk menguji hasil parsing data ke view, silahkan akses halaman [“http://localhost/belajar-ci/index.php/belajar/halo”](http://localhost/belajar-ci/index.php/belajar/halo).





## Webkita.com

Gambar 2.36 Berkenalan dengan view: parsing data ke view

Perhatikan pada contoh di atas data di parsing pada dengan memasukkan variabel ke dalam parameter kedua pada syntax:

**`$this->load->view()`.**

```
public function halo() {  
    $data['nama_web'] = "Webkita.com";  
    $this->load->view('view_belajar',$data);  
}
```

Dan dari view tinggal mengakses variabel `$nama_web`.

```
<h1><?php echo $nama_web; ?></h1>
```

Kemudia bisa juga menggunakan cara berikut.

**`application/controller/belajar.php`**

```
defined('BASEPATH') OR exit('No direct script access allowed');  
class Belajar extends CI_Controller  
{  
    function __construct() {  
        parent::__construct();  
    }  
}
```

```

public function index() {
    echo "ini method index pada controller belajar";
}

public function halo() {
    $data = array( 'judul' => "Cara Membuat View Pada
                  CodeIgniter", 'tutorial' => "CodeIgniter" );
    $this->load->view('view_belajar', $data);
}
}
?>

```

### Application/view/view\_belajar.php

```

<html>
<head>
    <title>Cara Membuat View Pada CodeIgniter</title>
</head>
<body>
    <h2><?php echo $judul; ?></h2>
    <h3><?php echo $tutorial; ?></h3>
</body>
</html>

```

Untuk menguji hasil parsing data ke view, silahkan akses halaman “<http://localhost/belajar-ci/index.php/belajar/halo>”.



Gambar 2.37 Berkenalan dengan view: parsing data ke view cara lain

## 2.5.8 Membuat *Template Web* Sederhana Pada Codeigniter

Pada sub bagian kali ini, buku ini menjelaskan tentang bagaimana membuat template *website* sederhana dengan menggunakan codeigniter. pada tutorial ini akan di jelaskan tentang cara menggunakan teknik *multiple view* pada codeigniter untuk membuat template website yang dinamis, sama seperi menggunakan include( ) pada bagian header dan footer pada penggunaan PHP *native*.

Tetapi di codeigniter kita membuat nya dengan men-*load* view codeigniter yang kita pisah-pisahkan sesuai keinginan, misalnya anda bisa memisahkan bagian header, footer, sidebar dan konten untuk mencegah pengulangan penulisan syntax dan memudahkan dalam hal memodifikasi *template website*.

### 2.5.8.1 *Templating* Dengan Codeigniter

Buat sebuah controller yang menampilkan sebuah view. Pada buku ini diarahkan dengan membuat controller web.php dan sebuah view dengan nama v\_index.php.

**application/controller/web.php**

```
<html>
<head>
  <meta charset="UTF-8">
  <title>Templating dengan codeigniter</title>
  <link rel="stylesheet" type="text/css" href="<?php echo
base_url() ?>assets/css/style.css">
</head>
<body>
  <div id="wrapper">
    <header>
      <hgroup>
        <h1>WebKita.com</h1>
        <h3>Membuat Template sederhana dengan codeigniter</h3>
      </hgroup>
    </header>
  </div>
</body>
</html>
```

```

<nav>
  <ul>
    <li><a href="<?php echo base_url(). 'index.php/web'
?>">Home</a></li>
    <li><a href="<?php echo base_url(). 'index.php/web/about'
?>">About</a></li>
  </ul>
</nav>
<div class="clear">

</div>
</header>
<section>
  <h1><?php echo $judul; ?></h1>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
    dolor in reprehenderit in voluptate velit esse cillum dolore eu
    fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
    proident, sunt in culpa qui officia deserunt mollit anim id est
    laborum.
  </p>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
    dolor in reprehenderit in voluptate velit esse cillum dolore eu
    fugiat nulla pariatur.
  </p>
</section>
<footer>
  <a href="#">WebKita</a>
</footer>
</div>
</body>
</html>

```

Sebelum lanjut, jangan lupa untuk setting `base_url()` codeigniternya terlebih dulu agar kita bisa menghubungkan codeigniter dengan file css. file css-nya dibuat pada folder project di dalam folder belajar-ci/assets/css/style.css (Jika folder assets dan css belum ada silahkan buat terlebih dahulu)

#### **assets/css/style.css**

```
body{
  background: #eee;
  color: #333;
  font-family: sans-serif;
  font-size: 15px;
}

#wrapper{
  background: #eee;
  width: 1100px;
  margin: 20px auto;
}

#wrapper header{
  background: #232323;
  padding: 20px;
}

#wrapper header hgroup{
  float: left;
  color: #fff;
}

#wrapper header nav{
  float: right;
  margin-top: 50px;
}

#wrapper header nav ul{
  padding: 0;
  margin: 0;
}
```

```
#wrapper header nav ul li{
    float: left;
    list-style: none;
}

#wrapper header nav ul li a{
    padding: 15px;
    color: #fff;
    text-decoration: none;
}

.clear{
    clear: both;
}

footer{
    background: #232323;
    padding: 20px;
}

footer a{
    color: #fff;
    text-decoration: none;
}

section{
    padding: 20px;
}
```

### 2.5.8.2 Setting Base\_URL Pada Codeigniter

Base\_URL merupakan halaman di mana folder root dari project kita alamatkan guna membantu saat proses templating dan *redirect* pada codeigniter.

Untuk setting base\_url() buka pada folder applications lalu pilih file config.php di application/config/config.php. Kemudia cari syntax berikut:

```
$config['base_url'] = '';
```

Lalu sesuai kan dengan letak project codeigniter anda sehingga menjadi:

```
$config['base_url'] = 'http://localhost/belajar-ci ';
```

Setelah proses setting selesai dan base\_url sudah diarahkan ke lokasi project. Selanjutnya, dapat kita perhatikan pada contoh di atas bahwa untuk menghubungkan file view dengan file css memerlukan bantuan base\_url().

```
<link rel="stylesheet" type="text/css" href="<?php echo  
base_url() ?>assets/css/style.css">
```

Hasil perintah di atas akan sama seperti:

```
<link rel="stylesheet" type="text/css"  
href="http://localhost/belajar-ci/assets/css/style.css">
```

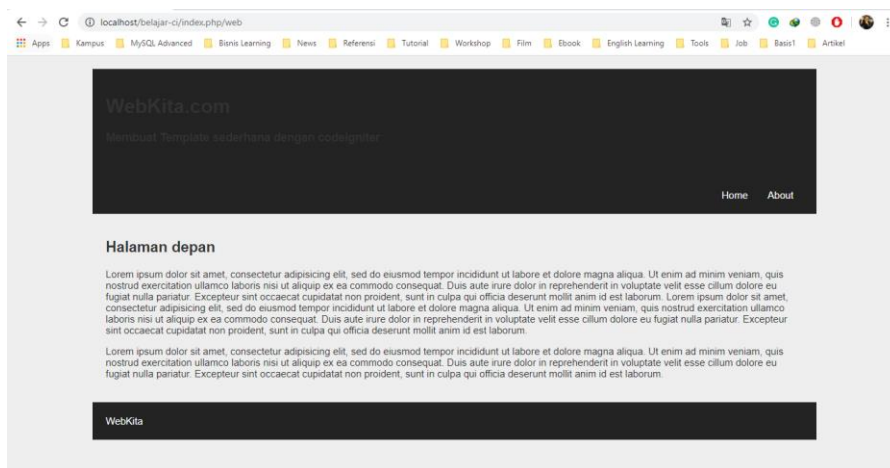
Kenapa demikian? Karena base\_url() pada project sudah ditentukan menjadi "localhost/belajar-ci". Dikarenakan kita menggunakan base\_url() codeigniter, maka kita perlu mengaktifkan juga helper url seperti yang sudah kita buat pada controller web.php. silahkan perhatikan. Kita mengaktifkan helper url pada function construct().

```
function __construct(){  
    parent::__construct();  
    $this->load->helper('url');  
}
```



Jalankan controller web.php untuk melihat hasil dari *template* sederhana yang telah dibuat dengan mengakses halaman <http://localhost/belajar-ci/index.php/web>.

Berikut hasilnya:



Gambar 2.38 *Templating* sederhana: tampilan web sederhana

Akhirnya, tampilan web sederhana pun berhasil kita buat pada codeigniter. Akan tetapi, halaman ini masih statis. kita masih perlu membuat bagian header dan footer secara berulang-ulang pada halaman lainnya.

### 2.5.8.3 Membuat Halaman *Template* Dinamis Dengan Codeigniter

Halaman *template* dinamis merupakan halaman yang dimana membantu developer atau programmer dalam membuat tampilan yang tidak berulang-ulang contoh untuk bagian header dan footer pada project kali ini. Untuk membuat halaman *template* yang dinamis caranya kita harus memecah *template* ini menjadi beberapa bagian. yaitu header dan footer. sehingga template sederhana ini menjadi:

## application/view/v\_header.php

```
<html>
<head>
  <meta charset="UTF-8">
  <title>WebKita | Membuat Template sederhana
codeigniter</title>
  <link rel="stylesheet" type="text/css" href="<?php echo
base_url() ?>assets/css/style.css">
</head>
<body>
  <div id="wrapper">
    <header>
      <hgroup>
        <h1>WebKita.com</h1>
        <h3> Membuat Template Sederhana Codeigniter </h3>
      </hgroup>
      <nav>
        <ul>
          <li><a href="<?php echo base_url().'.index.php/web'
?>">Home</a></li>
          <li><a href="<?php echo base_url().'.index.php/web/about'
?>">About</a></li>
        </ul>
      </nav>
      <div class="clear">

    </div>
  </header>
```

## application/view/v\_footer.php

```
<footer>
  <a href="http://github.com/k-sandi"> Developer Webkita.id</a>
</footer>
</div>
</body>
</html>
```

## application/view/v\_index.php

```
<section>
  <h1><?php echo $judul; ?></h1>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
    dolor in reprehenderit in voluptate velit esse cillum dolore eu
    fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
    proident, sunt in culpa qui officia deserunt mollit anim id est
    laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore magna
    aliqua. Ut enim ad minim veniam, quis nostrud exercitation
    ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis
    aute irure dolor in reprehenderit in voluptate velit esse cillum
    dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat
    non proident, sunt in culpa qui officia deserunt mollit anim id est
    laborum.
  </p>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
    dolor in reprehenderit in voluptate velit esse cillum dolore eu
    fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
    proident, sunt in culpa qui officia deserunt mollit anim id est
    laborum.
  </p>
</section>
```

Sekarang template ini sudah menjadi tiga view dan cara memanggilnya dengan cara memanggil view secara berurut. dari v\_header, v\_index dan kemudian v\_footer pada controller. Berikut cara pemanggilannya:

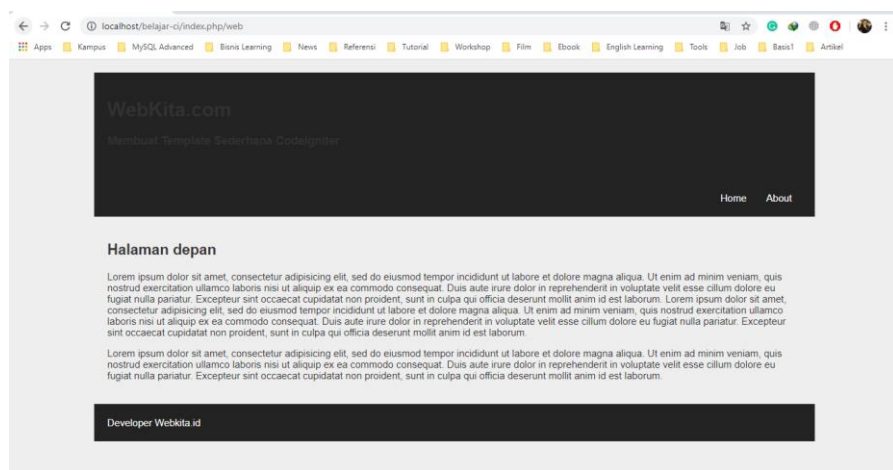
## application/controller/web.php

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Web extends CI_Controller
{
    function __construct() {
        parent::__construct();
        $this->load->helper('url');
    }

    public function index() {
        $data['judul'] = "Halaman depan";
        $this->load->view('v_header',$data);
        $this->load->view('v_index',$data);
        $this->load->view('v_footer',$data);
    }
}
?>
```

Dan hasilnya juga akan sama, tetapi kelebihanannya kita bisa dengan mudah membuat halaman lain dan dengan memanggil v\_header dan v\_footer, hanya perlu mengganti v\_index untuk halaman lainnya.



Gambar 2.39 *Templating sederhana*: Tampilan halaman web dinamis.

Perhatikan lagi pada bagian *hyperlink* tepatnya pada menu *template* di atas (v\_header.php) yang sudah kita setting untuk menuju method index dan method about.

```
<li><a href="<?php echo base_url(). 'index.php/web'
?>">Home</a></li>
<li><a href="<?php echo base_url(). 'index.php/web/about'
?>">About</a></li>
```

Sekarang buat sebuah view lagi dengan nama v\_about.php

```
<section>
  <h1><?php echo $judul ?></h1>
  <p>
    halaman about yang bisa anda ubah sesuai keinginan. - Lorem
    ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
    tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
    minim veniam, quis nostrud exercitation ullamco laboris nisi ut
    aliquip ex ea commodo consequat. Duis aute irure dolor in
    reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
    pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
    culpa qui officia deserunt mollit anim id est laborum.
  </p>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
    eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
    enim ad minim veniam, quis nostrud exercitation ullamco laboris
    nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
    reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
    pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
    culpa qui officia deserunt mollit anim id est laborum. Lorem
    ipsum dolor sit amet, consectetur adipisicing elit, seddo eiusmod
    tempor incididunt ut labore et dolore magna aliqua.
  </p>
  <p>
    webkita.com
  </p>
</section>
```

Kemudian tambahkan lagi *method* about pada controller web untuk membuat halaman about.

#### **application/controller/web.php**

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

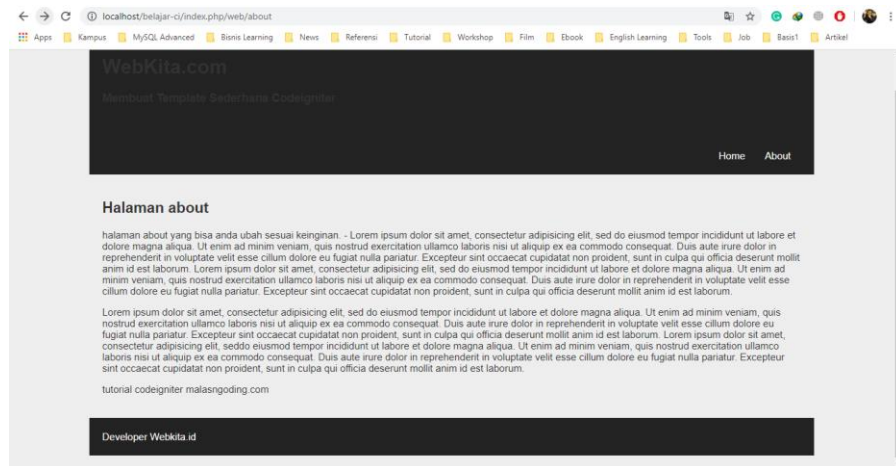
class Web extends CI_Controller
{
    function __construct() {
        parent::__construct();
        $this->load->helper('url');
    }

    public function index() {
        $data['judul'] = "Halaman depan";
        $this->load->view('v_header',$data);
        $this->load->view('v_index',$data);
        $this->load->view('v_footer',$data);
    }

    public function about() {
        $data['judul'] = "Halaman about";
        $this->load->view('v_header',$data);
        $this->load->view('v_about',$data);
        $this->load->view('v_footer',$data);
    }
}
?>
```

Selanjutnya jalankan lagi pada *browser*, Kemudia klik pada menu about atau bisa mengakses langsung halaman <http://localhost/belajar-ci/index.php/web/about>.

Berikut hasilnya:



Gambar 2.40 *Templatig sederhana: Tampilan halaman web about*

#### 2.5.8.4 Kesimpulan

Dengan membuat template *web* dengan *multiple view* atau memecahkan *template* menjadi beberapa bagian dan kemudian memanggilnya secara berurutan seperti contoh di atas yaitu dengan memanggil view header, index dan kemudian footer, maka akan memudahkan kita karena tidak perlu menuliskan syntax berulang-ulang. Misalnya, kita tidak perlu lagi membuat header khusus untuk halaman index dan tidak perlu juga membuat header khusus lagi untuk halaman about.

Pada dasarnya, header dan footer dapat di gunakan secara bersama-sama dan hanya bagian kontennya saja yang berubah-ubah sesuai keinginan dengan cara memanggilnya pada *method* controller web.

## 2.5.9 Fitur *Form Validation* Codeigniter

Pada sub bagian kali ini, buku ini menjelaskan tentang bagaimana membuat atau memanfaatkan *form validation* pada codeigniter. Terkadang kita memerlukan *form validation* dalam penginputan data, seperti misalnya kita menentukan *form* apa saja yang wajib di sisi, *form* dengan format *input* tertentu, membuat konfirmasi *password* dan sebagainya. Oleh karena itu, untuk membuat *form validation* pada codeigniter kita perlu memanggil atau membuka library **form\_validation** pada codeigniter. berikut ini adalah cara membuat form validation pada codeigniter.

### 2.5.9.1 Membuat Form Validation

Untuk contoh membuat form validation pada codeigniter, kita akan membuat sebuah controller dengan nama **form.php**.

**application/controller/form.php**

```
<?php
class Form extends CI_Controller
{
    function __construct() {
        parent::__construct();
        $this->load->library('form_validation');
    }

    function index() {
        $this->load->view('v_form');
    }

    function aksi() {
        $this->form_validation->set_rules('nama','Nama','required');
        $this->form_validation->set_rules('email','Email','required');
        $this->form_validation->set_rules('konfir_email','Konfirmasi
        Email','required');
```



```
if($this->form_validation->run() != false) {  
    echo "Form validation oke"; }else{ $this->load-  
>view('v_form');  
    }  
}  
}  
?>
```

Perhatikan pada controller form.php di atas. pertama kita akan membuat dulu sebuah form pada view yang kita panggil pada method index agar di panggil pertama kali. Lalu, di sini kita membuat view dengan nama v\_form.php.

Kemudian, untuk menggunakan library *form validation* codeigniter kita harus memanggil library form\_validation codeigniter terlebih dahulu. Pada buku ini cara memanggilnya berada pada function construct pada controller form. dapat di perhatikan pada construct() controller form di atas.

## application/view/v\_form.php

```
<html>
<head>
  <title>Membuat form validation dengan Codeigniter</title>
</head>
<body>
  <h1>Membuat Form Validation dengan CodeIgniter</h1>
  <?php echo validation_errors(); ?>
  <?php echo form_open('form/aksi'); ?>
  <label>Nama</label>
  <br/>
  <input type="text" name="nama">
  <br/>
  <label>Email</label>
  <br/>
  <input type="text" name="email">
  <br/>
  <label>Konfirmasi Email</label>
  <br/>
  <input type="text" name="konfir_email">
  <br/>
  <input type="submit" value="Simpan">
</form>
</body>
</html>
```

Pada form di atas kita membuka form dengan function form codeigniter. Perhatikan juga pada function form\_open() pada view di atas. kita menetapkan aksi dari form ke method aksi pada controller form.

```
<?php echo form_open('form/aksi'); ?>
```

Dan kita juga telah membuat tiga buah form, yaitu nama, email dan konfir\_email. di sini saya ingin membuat form input tersebut wajib di isi, maka form validasi nya akan kita buat pada method aksi pada controller form.

```
function aksi() {
    $this->form_validation->set_rules('nama','Nama','required');
    $this->form_validation->set_rules('email','Email','required');
    $this->form_validation->set_rules('konfir_email','Konfirmasi
Email','required');

    if($this->form_validation->run() != false) {
        echo "Form validation oke"; }else{ $this->load-
>view('v_form');
    }
}
```

Kemudian, untuk membuat *form validation* kita harus menentukan dulu *form* yang akan diberikan validasi. Seperti contoh di atas pada buku ini kita membuat form validation pada form input nama, email dan konfir\_email. function `set_rules` di atas berarti kita menetapkan peraturan untuk form. Adapun cara penulisannya pada parameter pertama berikan nama form yang ingin diberi validasi, pada parameter kedua berikan kata yang dimunculkan pada saat validasi, dan parameter ketiga isikan peraturan form. Dalam hal ini, `required` berarti wajib yang artinya form tersebut wajib di isi. Sekarang kita jalankan form validation yang sudah kita buat dengan alamat “<http://localhost/belajar-ci/index.php/form>”. Lalu klik tombol submit dalam keadaan form dibiarkan kosong, maka apa yang terjadi? akan muncul peringatan untuk mengisi form.

← → ↻ ⓘ [:1]/belajar-ci/index.php/form/aksi

Apps Kampus MySQL Advanced Bisnis Learning News Referensi Tutorial

## Membuat Form Validation dengan CodeIgniter

The Nama field is required.

The Email field is required.

The Konfirmasi Email field is required.

Nama

Email

Konfirmasi Email

Simpan

Gambar 2.41 Membuat *form validation: Required form*

Seperti yang anda lihat, letak pesan peringatan form validationnya di tampilkan dengan fungsi berikut pada view form (v\_form).

```
<?php echo validation_errors(); ?>
```

Adapun jika ingin menampilkan pesan dengan manual anda dapat menggunakan fungsi berikut:

```
<?php echo form_error('nama'); ?>  
<?php echo form_error('email'); ?>  
<?php echo form_error('konfir_email'); ?>
```

Berikut contoh full *code*-nya:

```
<html>
<head>
  <title>Membuat form validation dengan Codeigniter</title>
</head>
<body>
  <h1>Membuat Form Validation dengan CodeIgniter</h1>
  <?php echo validation_errors(); ?>
  <?php echo form_open('form/aksi'); ?>
  <label>Nama</label>
  <br/>
  <input type="text" name="nama">
  <?php echo form_error('nama'); ?>
  <br/>
  <label>Email</label>
  <br/>
  <input type="text" name="email">
  <?php echo form_error('email'); ?>
  <br/>
  <label>Konfirmasi Email</label>
  <br/>
  <input type="text" name="konfir_email">
  <?php echo form_error('konfir_email'); ?>
  <br/>
  <input type="submit" value="Simpan">
</form>
</body>
</html>
```

Selengkapnya tentang membuat *form validation* pada codeigniter anda dapat membacanya pada *user\_guide* yang sudah di sediakan oleh codeigniter pada project CI anda.

[http://localhost/belajar-ci/user\\_guide/libraries/form\\_validation.html#the-form](http://localhost/belajar-ci/user_guide/libraries/form_validation.html#the-form)

## 2.5.10 Upload File Pada Codeigniter

Membuat *Upload File* Dengan CodeIgniter, itulah judul yang akan kita bahas pada sesi ini. Sebenarnya tutorial *upload file* dengan codeigniter telah disediakan pada panduan penggunaan codeigniter (*user\_guide*). Anda dapat mengakses halaman tersebut pada folder *user\_guide* di folder project codeigniter yang anda buat sebelumnya.

Berikut *link* URL halaman User Guide tentang panduan *upload file* codeigniter:

[http://localhost/arca/user\\_guide/libraries/file\\_uploading.html](http://localhost/arca/user_guide/libraries/file_uploading.html)

Codeigniter sendiri sudah menyediakan library upload yang dapat digunakan langsung. Biasanya untuk membuat upload file pada codeigniter kita bisa menggunakan *library* ‘upload’ dan *helper* ‘form’, serta *helper* ‘url’ untuk bantuan dalam membuat *upload file* dengan codeigniter. Langsung saja masuk ke persiapan membuat upload file dengan codeigniter.

### 2.5.10.1 Membuat Upload File Dengan Codeigniter

Untuk mulai membuat *upload file* dengan codeigniter, pertama-tama sediakan sebuah form pada view, dan buat juga sebuah controller. Pada buku ini, khususnya pada sub-bab framework codeigniter, tiap controller dibuat terpisah sesuai pembahasan untuk membantu dan memudahkan anda dalam belajar.

## application/controller/upload.php

```
<?php
class Upload extends CI_Controller
{
    function __construct(){ parent::__construct();
        $this->load->helper(array('form', 'url'));
    }

    public function index(){
        $this->load->view('v_upload', array('error' => ' '));
    }

    public function aksi_upload(){
        $config['upload_path'] = './gambar/';
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size'] = 100; $config['max_width'] = 1024;
        $config['max_height'] = 768; $this->load->library('upload',
        $config);

        if ( ! $this->upload->do_upload('berkas')){
            $error = array('error' => $this->upload->display_errors());
            $this->load->view('v_upload', $error);
        }else{
            $data = array('upload_data' => $this->upload->data());
            $this->load->view('v_upload_sukses', $data);
        }
    }
}
?>
```

Dapat anda perhatikan pada controller yang telah dibuat di atas, kita memanggil *helper* url dan *helper* form untuk membantu kita dalam membuat *upload file* di codeigniter. *helper form* dan *url*-nya dipanggil pada function `construct()`.

**Penting!** Helper url paling sering di gunakan saat membuat aplikasi dengan menggunakan codeigniter. Karena helper url berguna dalam penggunaan fungsi **redirect()** yang sudah di sediakan oleh codeigniter sebelumnya. Redirect berguna untuk membuat pengalihan halaman (*hyperlink*).

```
function __construct(){ parent::__construct();  
    $this->load->helper(array('form', 'url'));  
}
```

Kemudian, pada function index kita tampilkan sebuah view v\_upload, pada view v\_upload ini ita akan buat *form upload*-nya.

Selanjutnya, kita buat sebuah view dengan nama v\_upload sesuai dengan yang kita panggil pada method/function index di atas.

#### **application/view/v\_upload.php**

```
<html>  
<head>  
    <title>Webkita.com</title>  
</head>  
<body>  
    <center><h1>Membuat Upload File Dengan  
CodeIgniter</h1></center>  
    <?php echo $error;?>  
    <?php echo form_open_multipart('upload/aksi_upload');?>  
    <input type="file" name="berkas" />  
    <br /><br />  
    <input type="submit" value="upload" />  
    </form>  
</body>  
</html>
```

Pada *form upload* view di atas kita menggunakan function form\_open\_multipart(), function form\_open\_multipart() ini isinya sama dengan syntax berikut:



```
<form action="" enctype="multipart/form-data">
```

Kemudian, pada *form* tersebut kita beri nama “berkas”. Lalu, aksi dari form tersebut kita arahkan ke method `aksi_upload` pada controller `upload`.

```
public function index(){
    $this->load->view('v_upload', array('error' => ' '));
}

public function aksi_upload(){
    $config['upload_path'] = './gambar/';
    $config['allowed_types'] = 'gif|jpg|png';
    $config['max_size'] = 100; $config['max_width'] = 1024;
    $config['max_height'] = 768; $this->load->library('upload',
    $config);

    if ( ! $this->upload->do_upload('berkas')){
        $error = array('error' => $this->upload->display_errors());
        $this->load->view('v_upload', $error);
    }else{
        $data = array('upload_data' => $this->upload->data());
        $this->load->view('v_upload_sukses', $data);
    }
}
```

Hal terpenting berada pada method `aksi_upload`. Method ini-lah yang akan dijadikan sebagai pengatur dari proses *upload file*. Pada variabel `config` di atas berfungsi sebagai pengaturan upload file pada codeigniter.

Adapun contohnya sebagai berikut:

```
$config['upload_path'] = './gambar/';
$config['allowed_types'] = 'gif|jpg|png';
$config['max_size'] = 100;
$config['max_width'] = 1024;
$config['max_height'] = 768;
$this->load->library('upload', $config);
```

`$config['upload_path']` berfungsi sebagai pengatur kemana file akan di upload. Dibuku ini kita menetakannya dalam folder ‘gambar’ pada directory root codeigniter (folder project kita). Oleh karena itu, kita buat dulu foldernya. Buat folder dengan nama “gambar” yang nantinya semua *file* yg di-*upload* akan masuk ke dalam folder “gambar” ini.

application	1/27/2020 12:51 AM	File folder	
assets	1/27/2020 9:55 AM	File folder	
gambar	2/2/2020 10:03 PM	File folder	
system	1/27/2020 12:51 AM	File folder	
user_guide	1/27/2020 12:51 AM	File folder	
.editorconfig	9/19/2019 5:08 AM	EDITORCONFIG File	1 KB
.gitignore	9/19/2019 5:08 AM	Text Document	1 KB
composer.json	9/19/2019 5:08 AM	JSON File	1 KB
contributing.md	9/19/2019 5:08 AM	MD File	7 KB
index.php	9/19/2019 5:08 AM	PHP File	11 KB
license	9/19/2019 5:08 AM	Text Document	2 KB
readme.rst	9/19/2019 5:08 AM	RST File	3 KB

Gambar 2.42 Membuat *upload file*: Menambahkan folder gambar

Kemudian, untuk membatasi ukuran file dan ekstensi file yang diperbolehkan untuk di-*upload* bisa di-*setting* pada baris kode berikut:

```
$config['upload_path'] = './gambar/';
$config['allowed_types'] = 'gif|jpg|png';
$config['max_size'] = 100;
$config['max_width'] = 1024;
$config['max_height'] = 768;
```

Setelah selesai membuat folder sebagai lokasi gambar yang di-*upload* berikutnya perhatikan syntax selanjutnya pada method aksi\_upload.

```
$this->load->library('upload', $config);

if ( ! $this->upload->do_upload('berkas')){
    $error = array('error' => $this->upload->display_errors());
    $this->load->view('v_upload', $error);
}else{
    $data = array('upload_data' => $this->upload->data());
    $this->load->view('v_upload_sukses', $data);
}
```

`$this->load->library('upload', $config)` berfungsi memanggil library upload codeigniter dengan menggunakan pengaturan yang sudah dibuat sebelumnya pada variabel `$config`.

`$this->upload->do_upload()` berfungsi untuk melakukan aksi upload. Dalam parameternya, berikan nama file upload tadi dengan nama 'berkas'. Kemudian system melakukan pengecekan, jika *file* tidak berhasil di-*upload* atau jika *file* yang di-*upload* tidak sesuai dengan pengaturan yang sudah kita buat maka pesan *error* akan muncul. Adapun pesan *error* tersebut kita masukkan dalam variabel `$error` lalu diparsing ke view `v_upload` untuk ditampilkan.

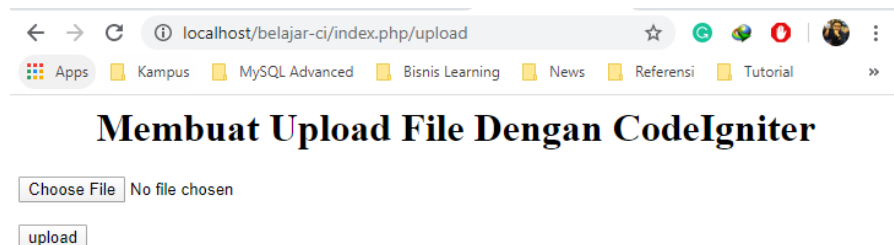
```
$error = array('error' => $this->upload->display_errors());
$this->load->view('v_upload', $error);
```

Dan jika file berhasil di-*upload*, maka data *file* yang di-*upload* dimasukkan ke dalam variabel `$data` untuk diparsing ke view `v_upload_sukses.php`.

## application/view/v\_upload\_sukses.php

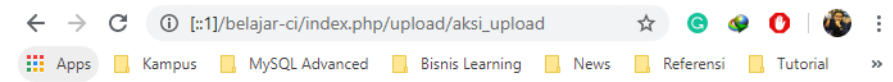
```
<html>
<head>
  <title>Webkita.com</title>
</head>
<body>
  <center><h1>Membuat Upload File Dengan
CodeIgniter</h1></center>
  <ul>
    <?php foreach ($upload_data as $item => $value):?>
      <li><?php echo $item;?>: <?php echo $value;?></li>
    <?php endforeach; ?>
  </ul>
</body>
</html>
```

Sekarang waktunya untuk mencoba apakah berhasil atau tidak.



Gambar 2.43 Membuat *upload file*: view upload

Pilih gambar yang ingin di upload dan submit (klik upload)



## Membuat Upload File Dengan CodeIgniter

- file\_name: Quotes\_Monta.jpg
- file\_type: image/jpeg
- file\_path: C:/xampp/htdocs/belajar-ci/gambar/
- full\_path: C:/xampp/htdocs/belajar-ci/gambar/Quotes\_Monta.jpg
- raw\_name: Quotes\_Monta
- orig\_name: Quotes\_Monta.jpg
- client\_name: Quotes\_Monta.jpg
- file\_ext: .jpg
- file\_size: 99.32
- is\_image: 1
- image\_width: 404
- image\_height: 404
- image\_type: jpeg
- image\_size\_str: width="404" height="404"

Gambar 2.44 Membuat *upload file*: aksi upload

Jika muncul gambar seperti pada Gambar 2.44 maka file berhasil di upload. Adapun yang muncul pada gambar di atas adalah, informasi gambar yang di upload. Sesuai dengan perintah yang kita buat pada view `v_upload_sukses.php` sebelumnya. Silahkan disesuaikan sendiri pesan suksesnya seperti apa.

Local Disk (C:) > xampp > htdocs > belajar-ci > gambar



Gambar 2.45 Membuat *upload file*: Hasil upload

Seperti yang tampil pada Gambar 2.45. Gambar pun berhasil di-*upload* dan sudah masuk ke dalam folder “gambar”. Sesuai dengan intruksi yang didefinisikan sebelumnya pada method `aksi_upload`.

- 2.5.11** Membuat *Upload File* Dengan Codeigniter
- 2.5.12** Membuat *Download File* Dengan Codeigniter
- 2.5.13** Membuat *Library* Codeigniter Sendiri
- 2.5.14** Membuat URL Cantik Dengan Menghilangkan “index.php” Pada Codeigniter
- 2.5.15** Berkenalan Dengan Model, Membuat Hubungan dengan *Database* dan DBMS MySQL
- 2.5.16** Membuat CRUD Dengan Codeigniter
- 2.5.17** Membuat Pagination Dengan Codeigniter
- 2.5.18** Membuat Login Sederhana Dengan Codeigniter

# **BAB III**

## **PERANCANGAN**

### 3.1 Perancangan Sistem

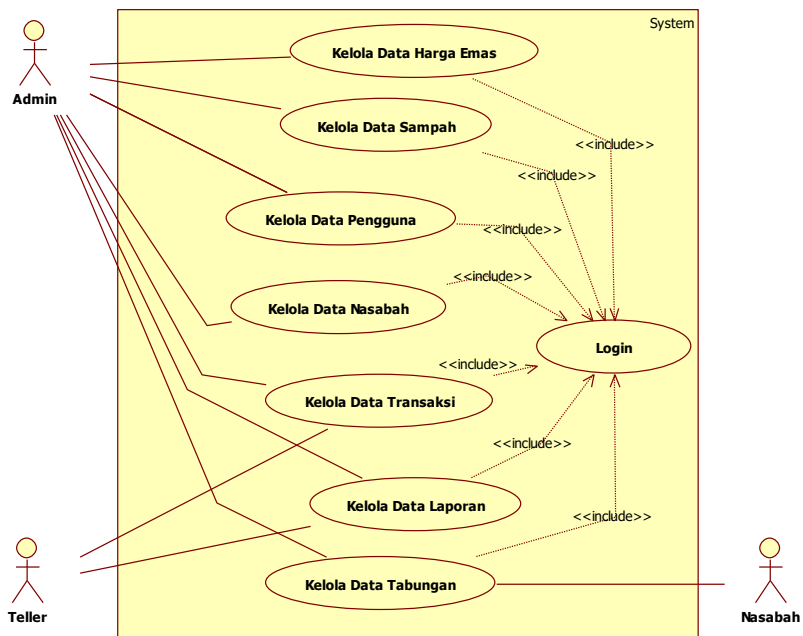
Perancangan merupakan tahap dimana rancangan atau *blueprint* suatu system yang akan dibangun hendak dibuat. Pada BAB ini, kami menitik beratkan pada tahapan rancangan menggunakan *tools* perancangan UML.

*Unified Modelling Language* (UML) adalah *tools* yang sering digunakan dalam merancang suatu system yang menggunakan paradigma *Object Oriented Programming* (OOP). Adapun tahap-tahap dari UML yang digunakan dalam membangun aplikasi Bank Sampah Istimewa ini terdiri atas *use case diagram*, *class diagram*, *sequence diagram*, *collaborative diagram*, *activity diagram*, *statechart diagram*, *component diagram* dan *deployment diagram*.

#### 3.1.1 Use Case Diagram

*Use Case Diagram* yang akan dibangun pada Bank Sampah Istimewa ini mendeskripsikan hubungan yang terjadi antara *Admin*, *Teller* dan Nasabah dengan aktivitas yang terdapat pada sistem. Adapun rancangan *Use Case Diagram* Bank Sampah Istimewa dapat dilihat pada gambar Gambar 3.1.





Gambar 3.1 Perancangan *Use Case Diagram*

### 3.1.1.1 Skenario *Use Case*

Skenario *Use Case* Bank Sampah Istimewa menjelaskan urutan dalam proses bisnis, baik yang dilakukan *Admin*, *Teller* dan *Nasabah* terhadap sistem maupun yang dilakukan oleh sistem terhadap *Admin*, *Teller* dan *Nasabah*. Berikut ini penjelasan dari masing-masing skenario tersebut:

#### 1. Skenario *Use Case Login*

Identifikasi	
Nama	<i>Login</i>
Tujuan	Masuk ke dalam sistem sebagai <i>Admin</i> , <i>Teller</i> dan <i>Nasabah</i>
Aktor	<i>Admin</i> , <i>Teller</i> dan <i>Nasabah</i> .
Deskripsi	Proses <i>login</i> ini untuk masuk ke halaman <i>Admin</i> , Jika <i>Admin</i> yang melakukan

	proses <i>login</i> , Jika <i>Teller</i> yang melakukan proses <i>login</i> maka akan masuk ke halaman <i>Teller</i> dan Jika Nasabah yang melakukan proses <i>login</i> maka akan masuk ke halaman Nasabah.
Skenario utama	
Kondisi awal	<i>Form login</i> di tampilkan
Aksi Aktor	
1. Memasukan <i>Username</i> dan <i>Password</i>	2. Mencocokkan data <i>login</i> dengan data <i>Admin</i> , <i>Teller</i> dan Penjual pada basis data.
	3. Bila valid akan menampilkan halaman <i>Admin</i> , Jika <i>Teller</i> yang melakukan proses <i>login</i> maka akan masuk ke halaman <i>Teller</i> dan Jika Nasabah yang melakukan proses <i>login</i> maka akan masuk ke halaman Nasabah.
Skenario Alternatif (jika gagal)	
Aksi aktor	Reaksi Sistem
	1. Menampilkan pesan
2. Memasukan <i>Username</i> dan <i>Password</i> .	
Kondisi Akhir	<i>Admin</i> melakukan kegiatan pada sistem sesuai kewenangan sebagai <i>Administrator</i>

## 2. Skenario *Use Case* Kelola Data Harga Emas

Identifikasi	
Nama	Kelola Data Harga Emas

Tujuan	Untuk mengelola data Emas yang harus ditambahkan setiap hari.
Deskripsi	Proses data <i>outlet</i> ini digunakan untuk melihat, menambah, mengubah data harga emas harian yang berada di data harga emas.
Aktor	<i>Admin</i>
<i>Use Case</i> yang berkaitan	<i>Login</i>
Kondisi awal	Menampilkan tabel data <i>outlet</i>
Aksi aktor	Reaksi sistem
1. <i>Admin</i> memilih menu master data → data harga emas.	2. Menampilkan tabel data harga emas harian.
3. Menekan tombol tambah data lalu mengisi data harga emas dan menekan tombol simpan.	4. Akan menyimpan data ke dalam <i>database</i> dan jumlah data harga emas harian akan bertambah.

5. Menekan tombol <i>update</i> lalu mengubah data yang diinginkan dan menekan tombol simpan	6. Akan mengubah data didalam <i>database</i> dan data harga emas akan berubah.
7. Menekan tombol <i>delete</i> dan muncul pesan lalu tekan <i>yes</i>	8. Data harga emas telah berhasil diolah.

### 3. Skenario *Use Case* Kelola Data Sampah

Identifikasi	
Nama	Data Sampah
Tujuan	Untuk mengelola data Sampah
Deskripsi	Proses kelola data sampah ini digunakan untuk menambah, merubah dan menghapus data sampah yang ada dalam <i>database</i> . Adapun kelola tersebut dibagi menjadi dua yaitu kategori dan subkategori.
Aktor	<i>Admin</i>

<i>Use Case</i> yang berkaitan	<i>Login</i>
Kondisi awal	Menampilkan menu <i>Admin</i>
Aksi aktor	Reaksi sistem
1. <i>Admin</i> memilih menu master data → data kategori dan/atau data subkategori.	2. Menampilkan tabel data sampah (kategori dan/atau subkategori).
3. Menekan tombol Tambah Data. <i>Admin</i> lalu mengisi data sampah berdasarkan kategori dan subkategori lalu menekan tombol <i>save</i>	4. Akan menyimpan data sampah ke dalam <i>database</i> .

5. Menekan tombol <i>edit</i> lalu mengubah data yang diinginkan dan menekan tombol <i>save</i>	6. Akan mengubah data didalam <i>database</i> dan data sampah pada master data akan berubah.
7. Menekan tombol delete dan muncul pesan lalu pilih yes.	8. Akan menghapus data berdasarkan <i>record</i> didalam <i>database</i> dan data sampah yang dipilih akan dihapus.
Kondisi akhir	Data Sampah telah berhasil diolah

4. Skenario *Use Case* Kelola Data Pengguna

Identifikasi	
Nama	Data Pengguna
Tujuan	Untuk mengelola data pengguna
Deskripsi	Proses kelola data pengguna ini digunakan untuk menambah, merubah dan menghapus data pengguna yang ada dalam <i>database</i> .
Aktor	<i>Admin</i>
<i>Use Case</i> yang berkaitan	<i>Login</i>

Kondisi awal	Menampilkan menu <i>Admin</i>
Aksi aktor	Reaksi sistem
1. <i>Admin</i> memilih menu master data → data pengguna.	2. Menampilkan tabel data pengguna.
3. Menekan tombol Tambah Data. <i>Admin</i> lalu mengisi data pengguna lalu menekan tombol <i>save</i>	4. Akan menyimpan data pengguna ke dalam <i>database</i> .
5. Menekan tombol <i>edit</i> lalu mengubah data yang diinginkan dan menekan tombol <i>save</i>	6. Akan mengubah data didalam <i>database</i> dan data pengguna pada master data akan berubah.

7. Menekan tombol delete dan muncul pesan lalu pilih yes.	8. Akan menghapus data berdasarkan <i>record</i> didalam <i>database</i> dan data pengguna yang dipilih akan dihapus.
Kondisi akhir	Data pengguna telah berhasil diolah

5. Skenario *Use Case* Kelola Data Nasabah

Identifikasi	
Nama	Kelola Data nasabah
Tujuan	Untuk mengelola data nasabah
Deskripsi	Proses kelola data nasabah ini digunakan untuk menambah, mengubah dan menghapus data pengguna yang ada dalam <i>database</i> .
Aktor	<i>Admin</i>
<i>Use Case</i> yang berkaitan	<i>Login</i>
Kondisi awal	Menampilkan menu <i>Admin</i>
Aksi aktor	Reaksi sistem
1. <i>Admin</i> memilih menu master data → data nasabah.	2. Menampilkan tabel data nasabah.
3. Menekan tombol	4. Akan menyimpan data nasabah ke dalam <i>database</i> .



Tambah Data. <i>Admin</i> lalu mengisi data nasabah lalu menekan tombol <i>save</i>	
5. Menekan tombol <i>edit</i> lalu mengubah data yang diinginkan dan menekan tombol <i>save</i>	6. Akan mengubah data didalam <i>database</i> dan data nasabah pada master data akan berubah.
7. Menekan tombol delete dan muncul pesan lalu pilih yes.	8. Akan menghapus data berdasarkan <i>record</i> didalam <i>database</i> dan data nasabah yang dipilih akan dihapus.
Kondisi akhir	Data nasabah telah berhasil diolah

6. Skenario *Use Case* Kelola Data Transaksi

Identifikasi	
Nama	Kelola Data Transaksi
Tujuan	Untuk mengelola data nasabah

Deskripsi	Proses kelola data nasabah ini digunakan untuk menambah, mengubah dan menghapus data pengguna yang ada dalam <i>database</i> .
Aktor	<i>Admin dan Teller</i>
<i>Use Case</i> yang berkaitan	<i>Login</i>
Kondisi awal	Menampilkan menu <i>Admin dan Teller</i>
Aksi aktor	Reaksi sistem
1. <i>Admin dan Teller</i> memilih menu transaksi	2. Menampilkan tabel data transaksi.
3. Menekan tombol Transaksi Baru. Lalu mengisi data transaksi yang dibutuhkan.	4. Akan menyimpan data transaksi ke dalam <i>database</i> .
Kondisi akhir	Data nasabah telah berhasil diolah

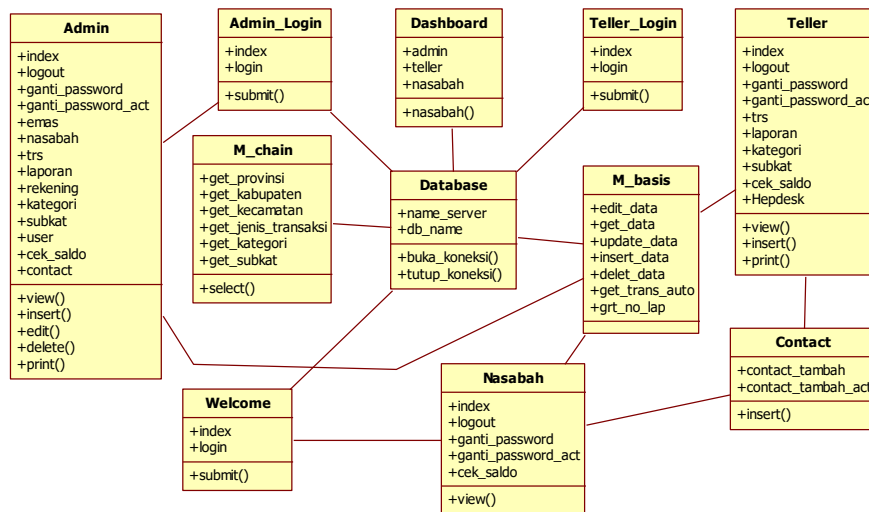
7. Skenario *Use Case* Kelola Data Laporan

Identifikasi	
Nama	Data laporan
Tujuan	Mengelola data laporan pembayaran

Deskripsi	Proses pengolahan data laporan merupakan proses untuk mengatur dan mengolah data laporan pada sistem
Aktor	<i>Admin</i> dan <i>Teller</i>
<i>Use Case</i> yang berkaitan	<i>Login</i>
Kondisi awal	Menampilkan menu <i>Admin</i>
Aksi aktor	Reaksi sistem
1. <i>Admin</i> memilih menu laporan pembayaran	2. Menampilkan pilihan laporan
3. Menekan laporan yang diinginkan lalu pilih tombol <i>view</i> atau bias langsung <i>download</i>	4. Akan memunculkan pengaturan untuk mencetak laporan sesuai dengan yang diinginkan.
Kondisi akhir	Data laporan telah diolah

### 3.1.2 Class Diagram

*Class diagram* merupakan gambaran struktur dan hubungan antar objek-objek yang ada pada *system* yang akan dibangun. Strukturnya meliputi atribut-atribut dan *method-method* yang ada pada masing-masing kelas. Adapun *Class Diagram* pada aplikasi Bank Sampah Istimewa dapat dilihat pada Gambar 3.2.



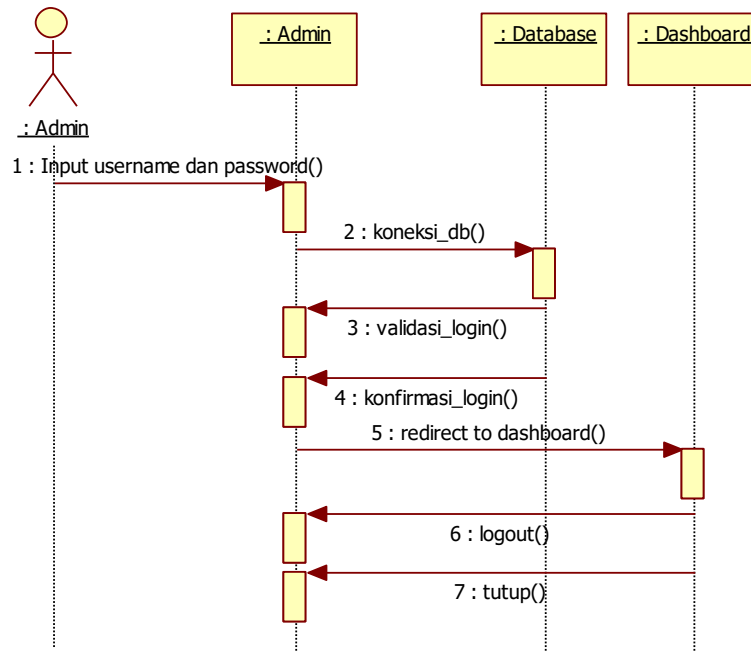
Gambar 3.2 Perancangan *Class Diagram*

### 3.1.3 Sequence Diagram

*Sequence Diagram* pada pembangunan *system* ini menggambarkan interaksi antar masing-masing objek pada setiap *Use Case* dalam urutan waktu. Interaksi ini berupa pengiriman serangkaian data antar objek-objek yang saling berinteraksi. Adapun *Sequence Diagram* pada aplikasi Sistem Bank Sampah ini sebagai berikut:

### 3.1.3.1 Sequence Diagram Login

#### 1. Admin

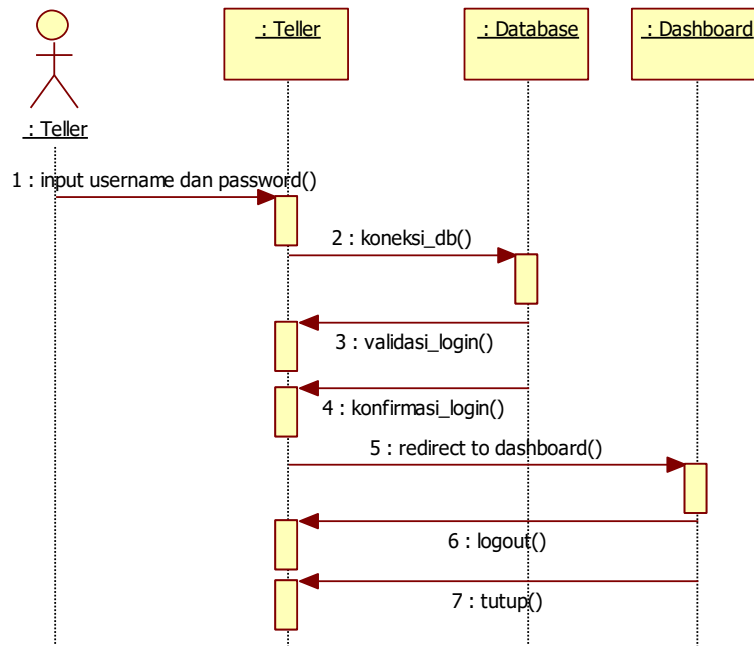


Gambar 3.3 Perancangan *Sequence Diagram Login Admin*

Keterangan:

Pertama *Admin* memasukkan *Username* dan *Password* pada *form login* lalu controller *Admin* akan bertindak dan membuka koneksi ke *database*, setelah itu sistem akan memvalidasi user admin dan mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke dashboard *admin*.

## 2. Teller

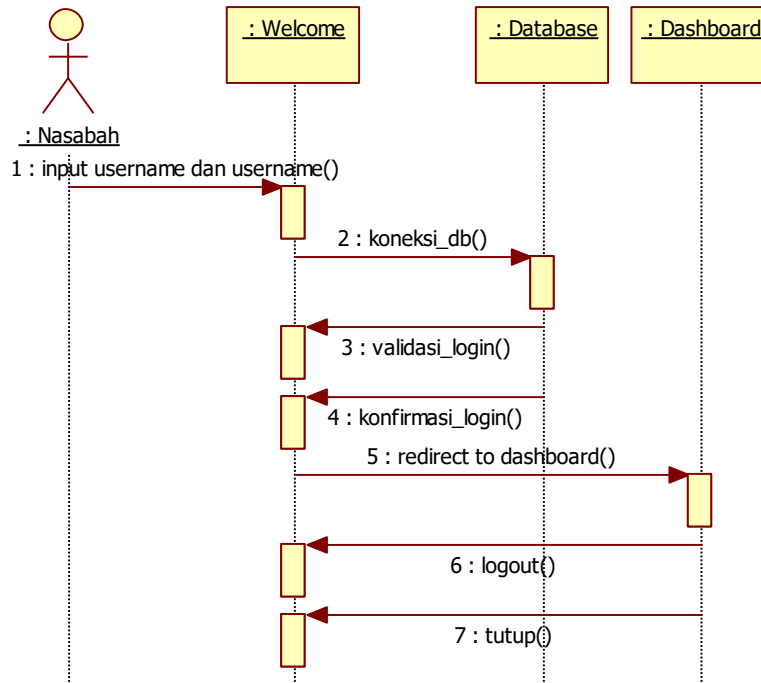


Gambar 3.4 Perancangan *Sequence Diagram Login Teller*

Keterangan:

Pertama *Teller* memasukan *Username* dan *Password* pada *form login* lalu controller *Teller* akan bertindak dan membuka koneksi ke *database*, setelah itu sistem akan memvalidasi *user teller* dan mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke dashboard teller.

### 3. Nasabah

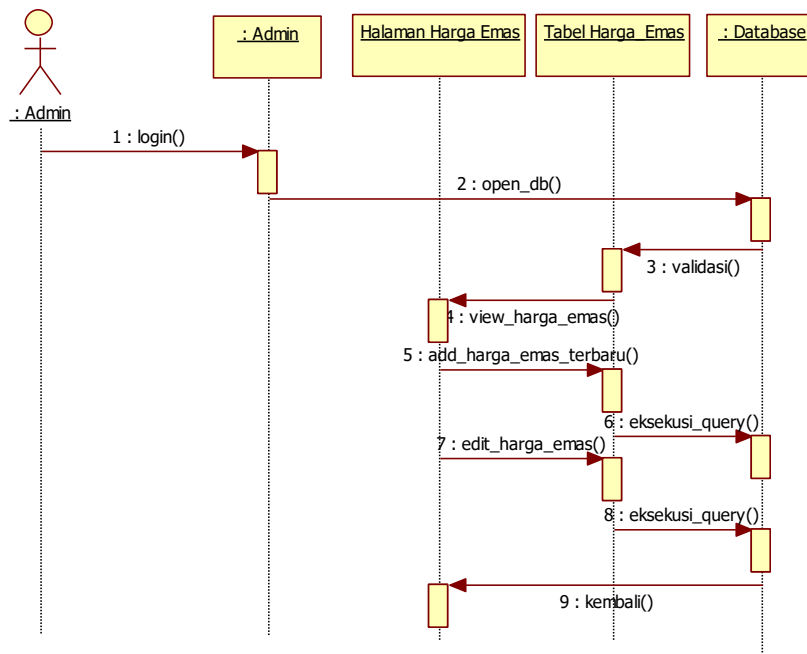


Gambar 3.5 Perancangan *Sequence Diagram Login* Nasabah

Keterangan :

Pertama Nasabah memasukkan *Username* dan *Password* pada *form login* lalu controller Nasabah akan bertindak dan membuka koneksi ke *database*, setelah itu sistem akan memvalidasi *user* nasabah dan mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke dashboard nasabah.

### 3.1.3.2 Sequence Diagram Kelola Data Harga Emas



Gambar 3.6 Perancangan *Sequence Diagram* kelola data harga emas

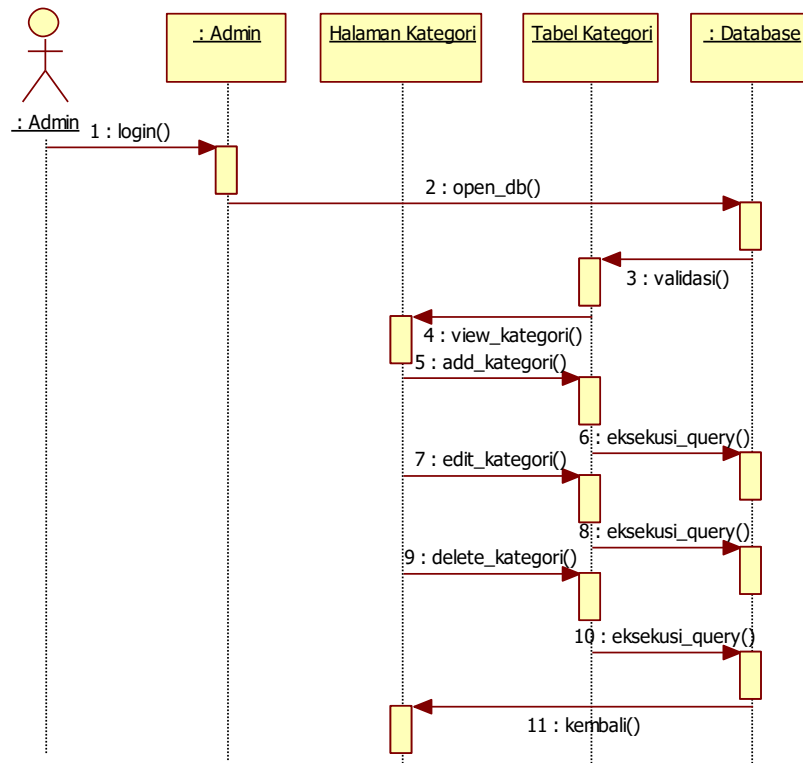
Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman harga emas, sistem membuka database untuk menampilkan data harga emas. Admin memiliki aksi seperti *view*, *add*, dan *edit* data emas harian (hitungannya per-1 *gram*). Jika sudah selesai admin kembali ke halaman harga emas.



### 3.1.3.3 Sequence Diagram Kelola Data Harga Sampah

#### 1. Kategori

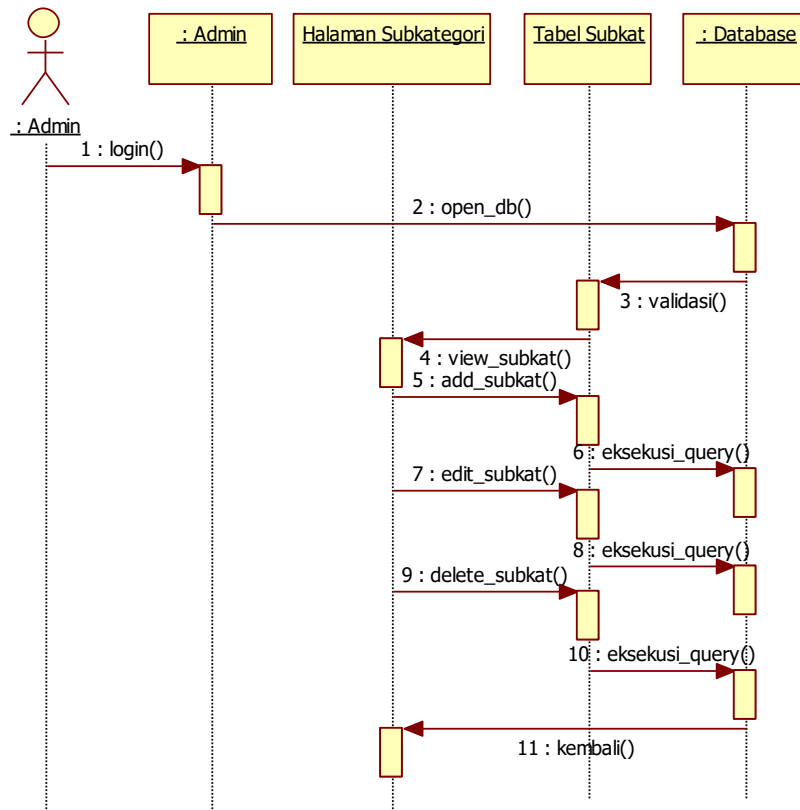


Gambar 3.7 Perancangan *Sequence Diagram* kelola data kategori

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman kategori, sistem membuka database untuk menampilkan data kategori sampah. Admin memiliki aksi seperti *view*, *add*, dan *edit* data kategori sampah sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman kategori.

## 2. Subkategori

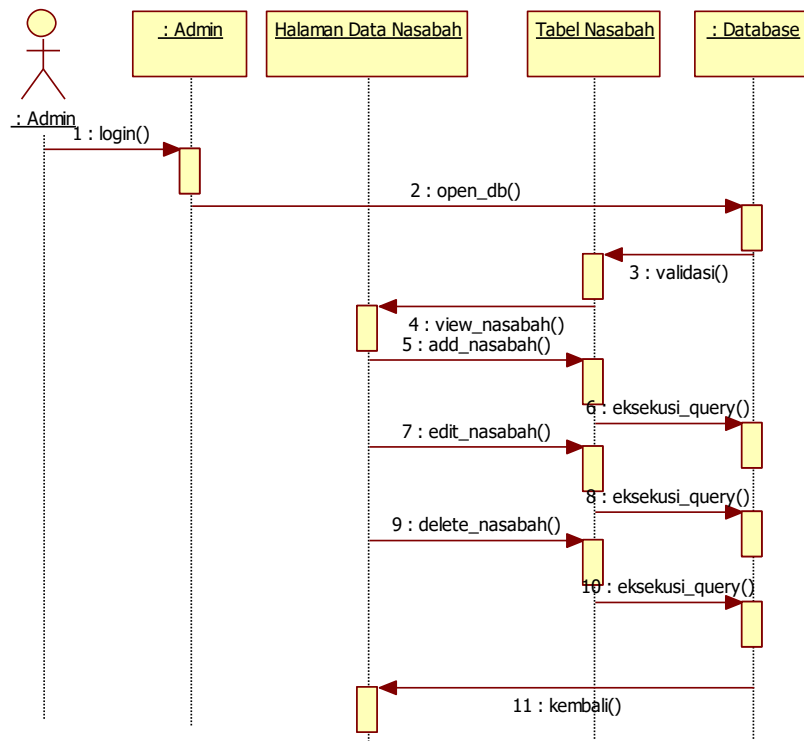


Gambar 3.8 Perancangan *Sequence Diagram* kelola data kategori

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman subkategori, sistem membuka *database* untuk menampilkan data subkategori sampah. Admin memiliki aksi seperti *view*, *add*, dan *edit* data subkategori sampah sewaktu-waktu dibutuhkan. Jika sudah selesai *admin* kembali ke halaman subkategori.

### 3.1.3.4 Sequence Diagram Kelola Data Nasabah

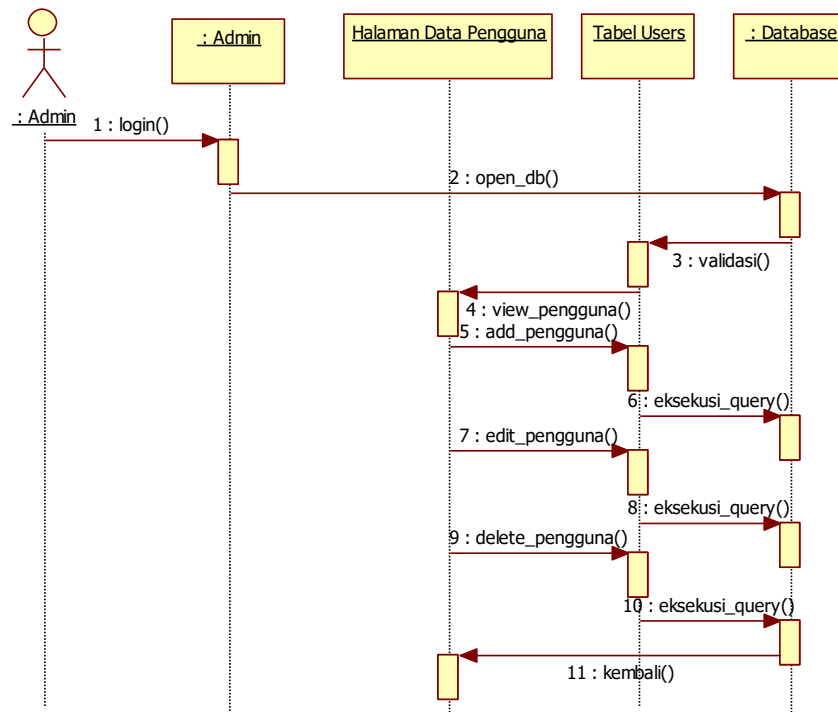


Gambar 3.9 Perancangan *Sequence Diagram* kelola data nasabah

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Data Nasabah, sistem membuka *database* untuk menampilkan data Nasabah. *Admin* memiliki aksi seperti *view*, *add*, dan *edit* data nasabah sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman data nasabah.

### 3.1.3.5 Sequence Diagram Kelola Data Pengguna



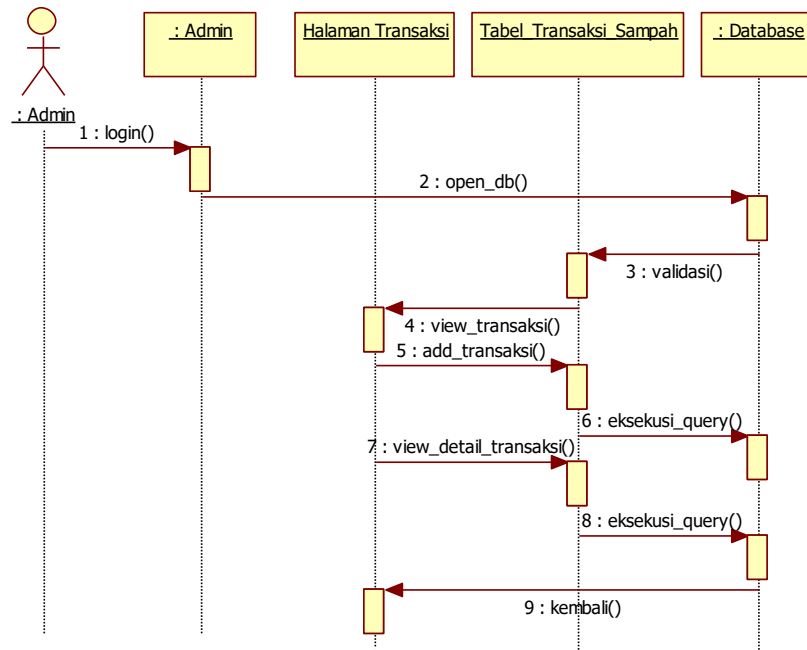
Gambar 3.10 Perancangan *Sequence Diagram* kelola data pengguna

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Data Pengguna, sistem membuka *database* untuk menampilkan data Pengguna sistem. *Admin* memiliki aksi seperti *view*, *add*, dan *edit* data pengguna sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman data pengguna.

### 3.1.3.6 Sequence Diagram Kelola Data Transaksi

#### 1. Admin

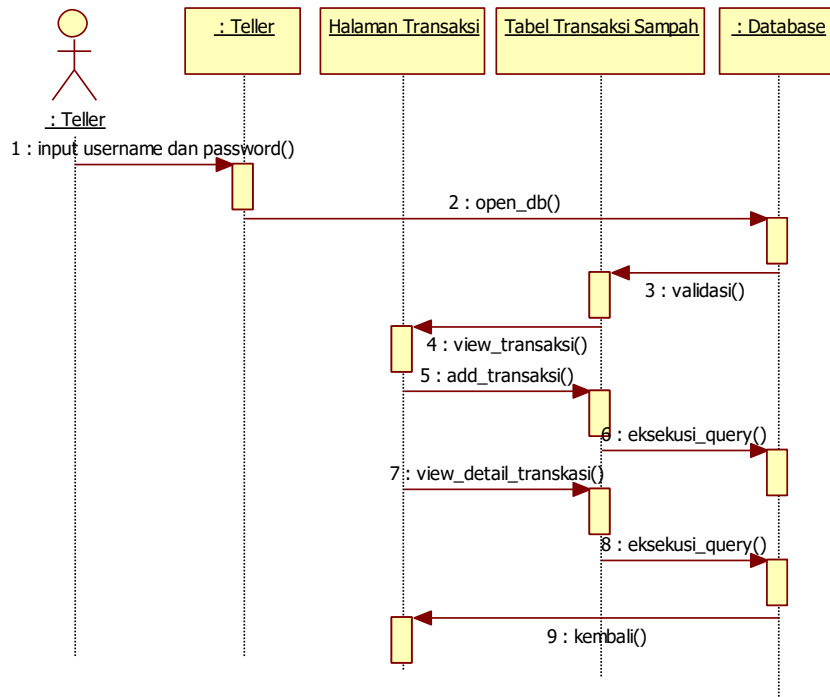


Gambar 3.11 Perancangan *Sequence Diagram* kelola data transaksi (*Admin*)

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Transaksi, sistem membuka *database* untuk menampilkan data Transaksi. *Admin* memiliki aksi seperti *view* dan *add* data transaksi sewaktu-waktu dibutuhkan. Jika sudah selesai *admin* kembali ke halaman data transaksi.

## 2. *Teller*



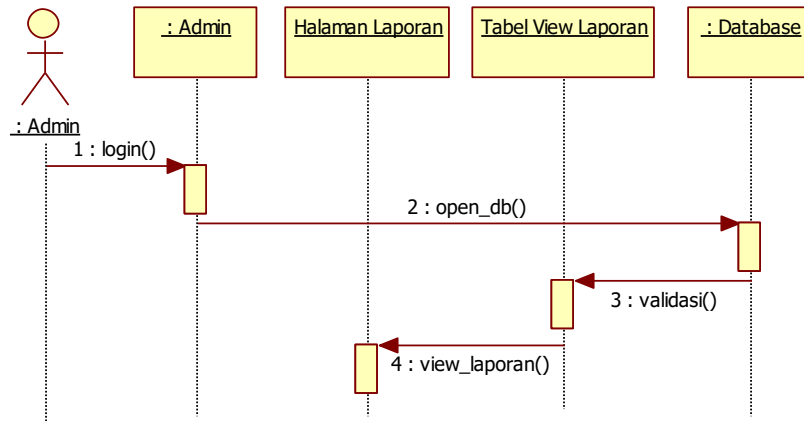
Gambar 3.12 Perancangan *Sequence Diagram* kelola data transaksi (*Teller*)

Keterangan :

*Teller* melakukan *login* untuk masuk ke sistem. Setelah itu *teller* membuka halaman Transaksi, sistem membuka *database* untuk menampilkan data Transaksi. *Teller* memiliki aksi seperti *view* dan *add* data transaksi sewaktu-waktu dibutuhkan. Jika sudah selesai *teller* kembali ke halaman data transaksi.

### 3.1.3.7 Sequence Diagram Kelola Data Laporan

#### 1. *Admin*

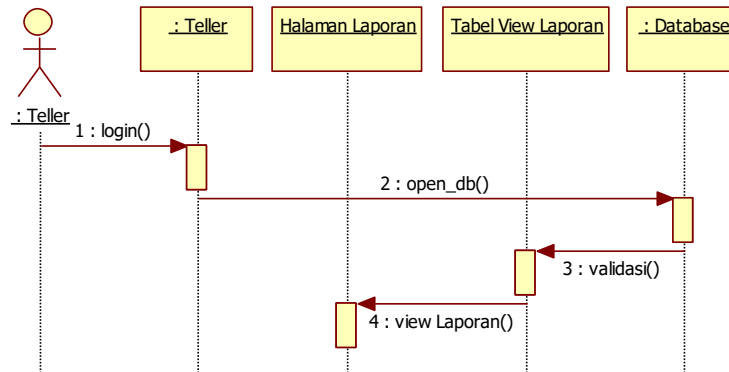


Gambar 3.13 Perancangan *Sequence Diagram* kelola data laporan (*Admin*)

Keterangan :

*Admin* melakukan *login* untuk masuk ke sistem. Setelah itu *admin* membuka halaman Laporan, sistem membuka *database* untuk menampilkan data Laporan. *Admin* memiliki hanya memiliki aksi *view*.

## 2. Teller



Gambar 3.14 Perancangan *Sequence Diagram* kelola data laporan (*Teller*)

Keterangan :

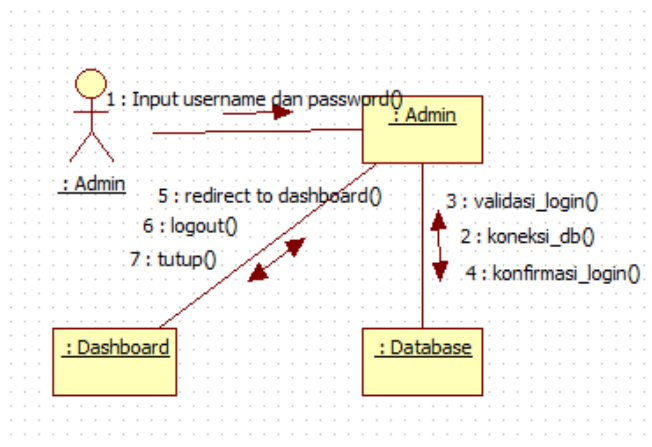
*Teller* melakukan *login* untuk masuk ke sistem. Setelah itu *teller* membuka halaman Laporan, sistem membuka *database* untuk menampilkan data Laporan. *Teller* memiliki hanya memiliki aksi *view*.



### 3.1.4 Collaborative Diagram

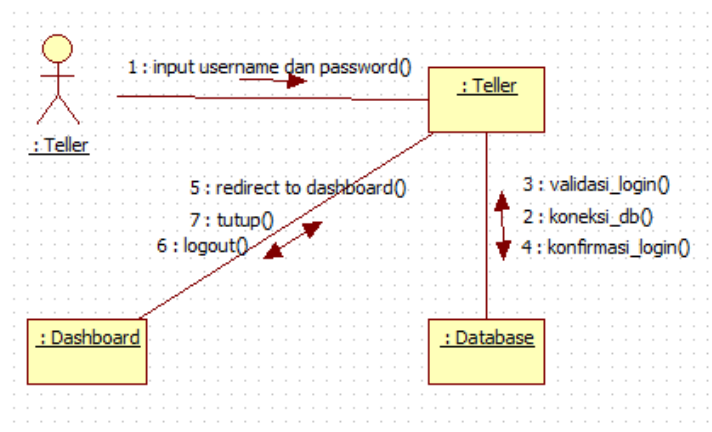
#### 3.1.4.1 Collaborative Diagram Login

##### 1. Admin



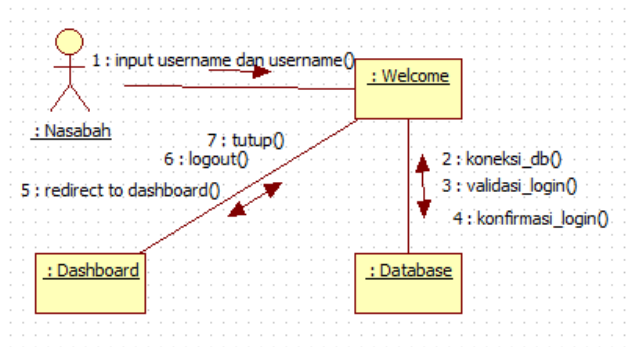
Gambar 3.15 Perancangan *Collaborative Diagram Login (Admin)*

##### 2. Teller



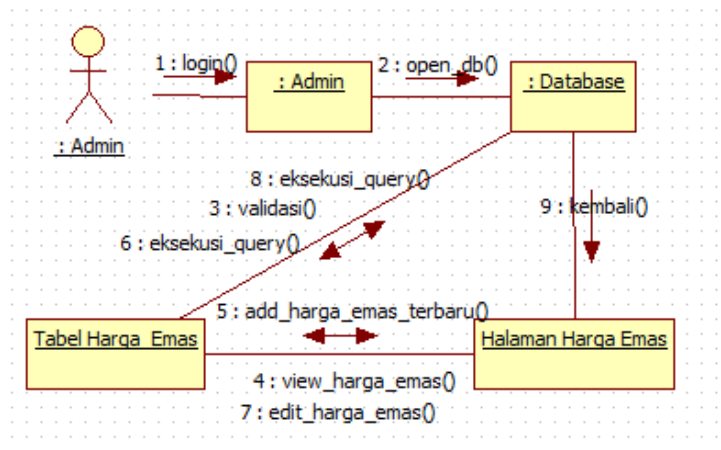
Gambar 3.16 Perancangan *Collaborative Diagram Login (Teller)*

### 3. Nasabah



Gambar 3.17 Perancangan *Collaborative Diagram Login* (Nasabah)

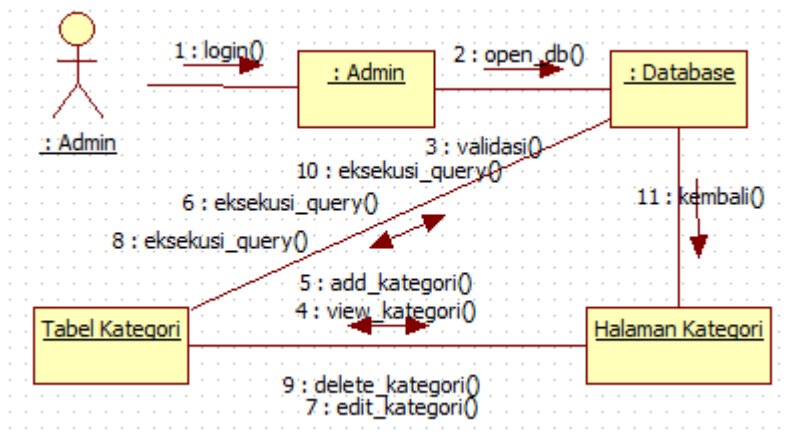
#### 3.1.4.2 Collaborative Diagram Kelola Data Harga Emas



Gambar 3.18 Perancangan *Collaborative Diagram Kelola Data Harga Emas*

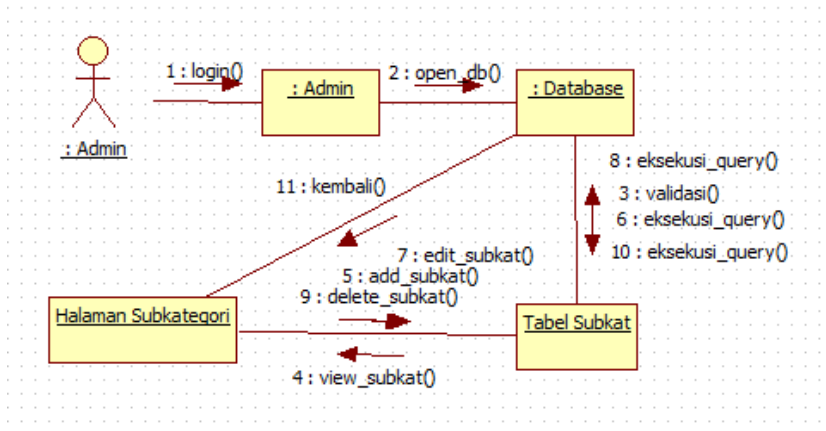
### 3.1.4.3 Collaborative Diagram Kelola Data Sampah

#### 1. Kategori



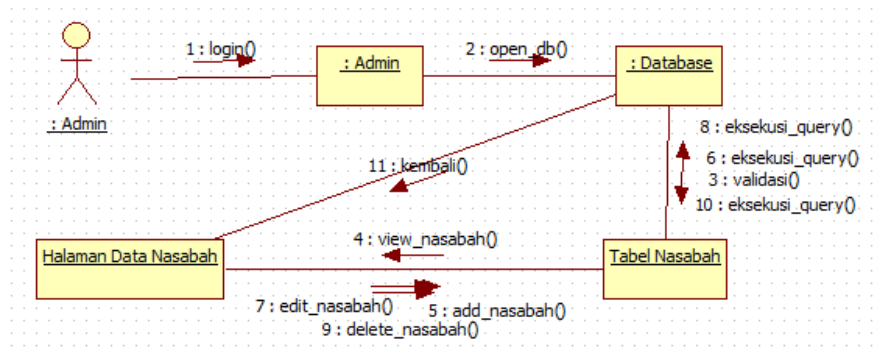
Gambar 3.19 Perancangan *Collaborative Diagram* Kelola Data Kategori

#### 2. Subkategori



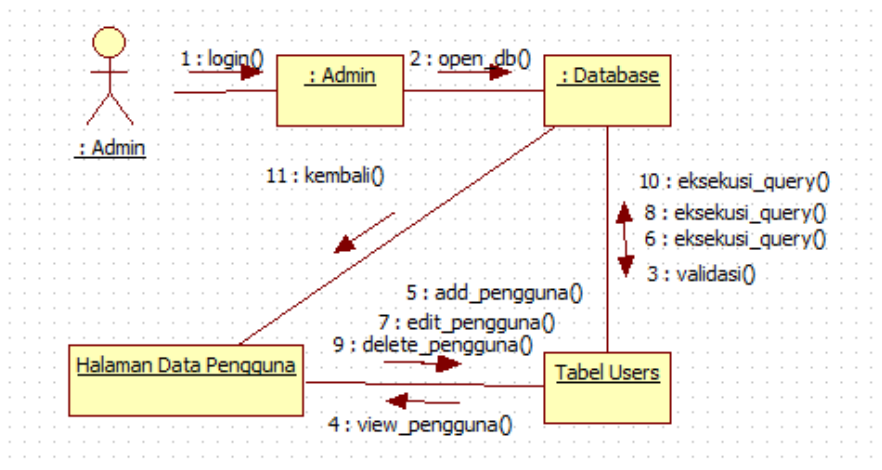
Gambar 3.20 Perancangan *Collaborative Diagram* Kelola Data Subkategori

#### 3.1.4.4 Collaborative Diagram Kelola Data Nasabah



Gambar 3.21 Perancangan *Collaborative Diagram* Kelola Data Nasabah

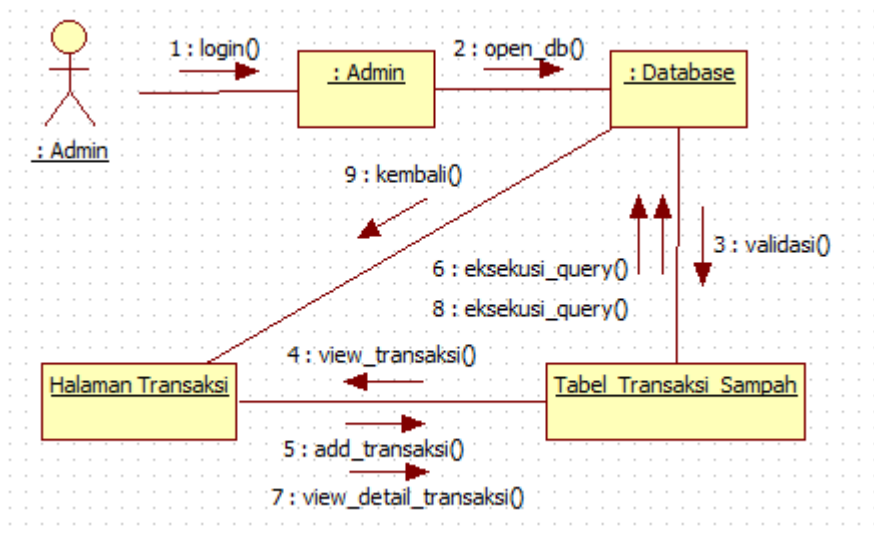
#### 3.1.4.5 Collaborative Diagram Kelola Data Pengguna



Gambar 3.22 Perancangan *Collaborative Diagram* Kelola Data Pengguna

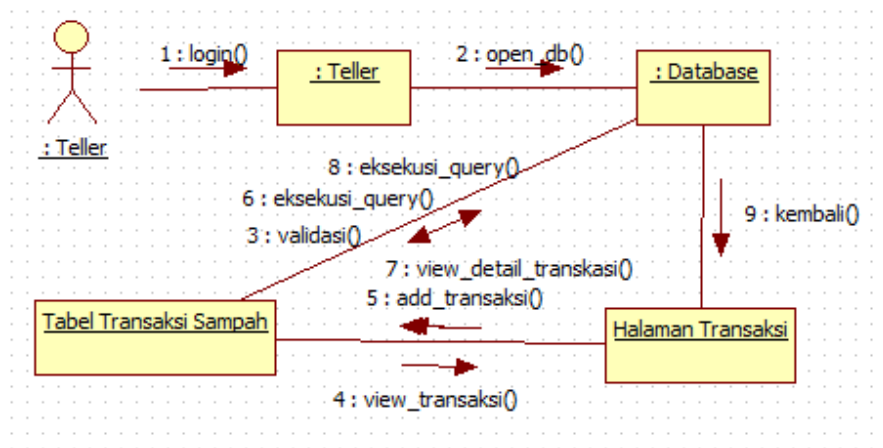
### 3.1.4.6 Collaborative Diagram Kelola Data Transaksi

#### 1. Admin



Gambar 3.23 Perancangan *Collaborative Diagram* Kelola Data Transaksi (Admin)

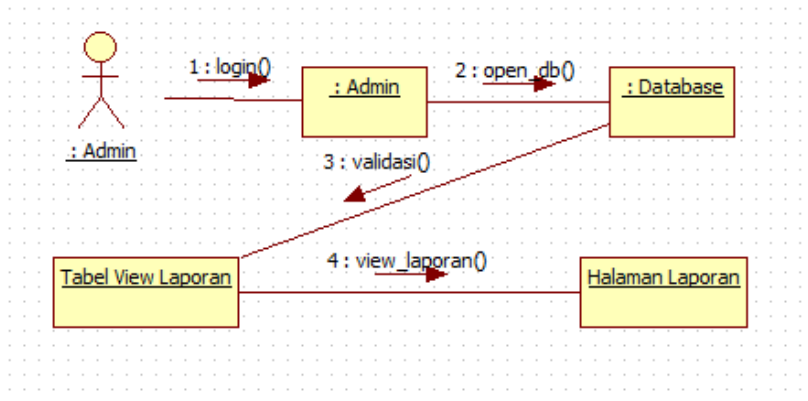
#### 2. Teller



Gambar 3.24 Perancangan *Collaborative Diagram* Kelola Data Transaksi (Teller)

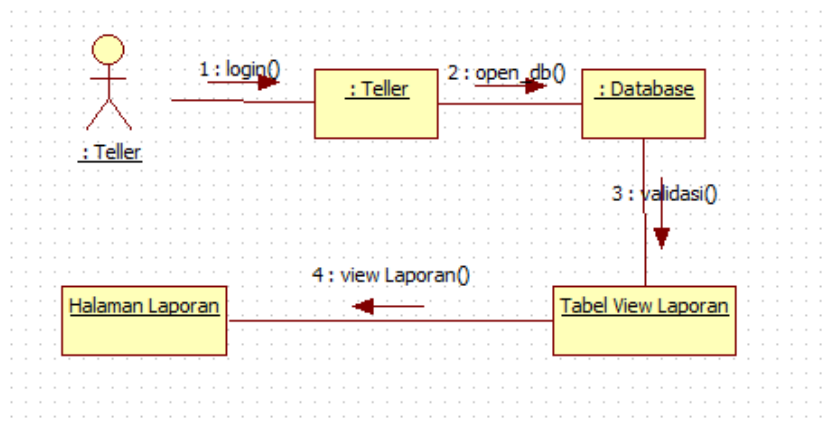
### 3.1.4.7 Collaborative Diagram Kelola Data Laporan

#### 1. Admin



Gambar 3.25 Perancangan *Collaborative Diagram* Kelola Data Laporan (Admin)

#### 2. Teller



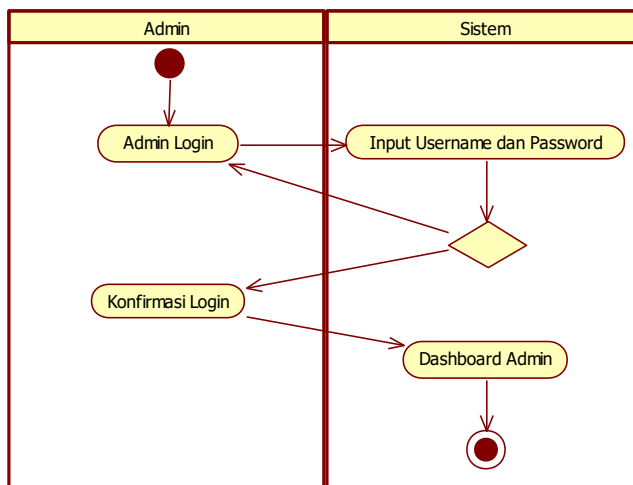
Gambar 3.26 Perancangan *Collaborative Diagram* Kelola Data Laporan (Teller)

### 3.1.5 Activity Diagram

*Activity diagram* digunakan untuk memodelkan aliran kerja atau *workflow* dari urutan aktivitas dalam suatu proses yang mengacu pada *Use Case Diagram* yang ada. Adapun *Activity Diagram* pada aplikasi Bank Sampah Istimewa ini yaitu sebagai berikut:

#### 3.1.5.1 Activity Diagram Login

##### 1. Admin

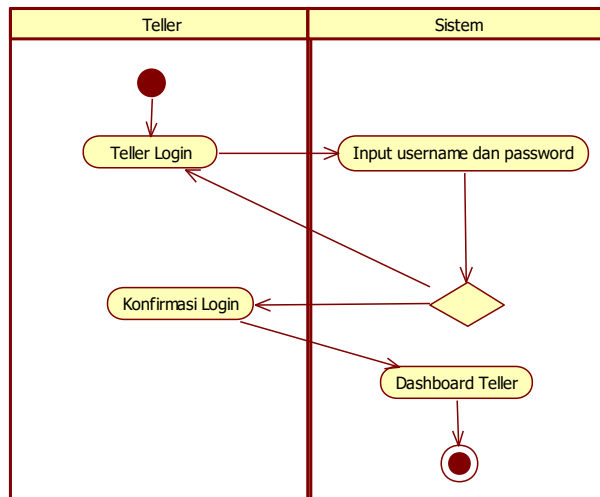


Gambar 3.27 Perancangan *Activity Diagram* Kelola Data Login (*Admin*)

Keterangan :

Pertama *Admin* memasukkan *Username* dan *Password* pada *form login* lalu *controller Admin* akan bertindak dan membuka koneksi ke *database*, setelah itu *sistem* akan memvalidasi user *admin* dan mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke *dashboard admin*.

## 2. Teller



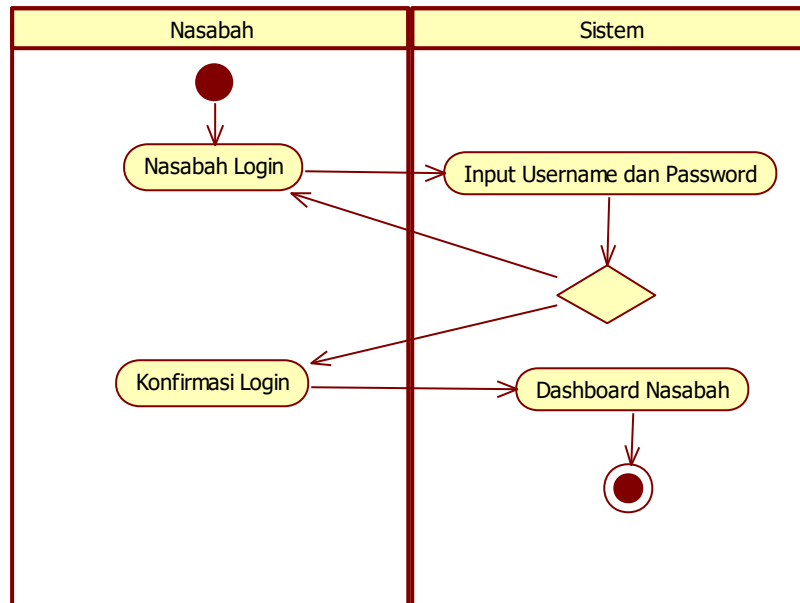
Gambar 3.28 Perancangan *Activity Diagram* Kelola Data Login (Teller)

Keterangan :

Pertama *Teller* memasukan *Username* dan *Password* pada *form login* lalu *controller Teller* akan bertindak dan membuka koneksi ke *database*, setelah itu *sistem* akan memvalidasi *user teller* dan mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke *dashboard teller*.



### 3. Nasabah

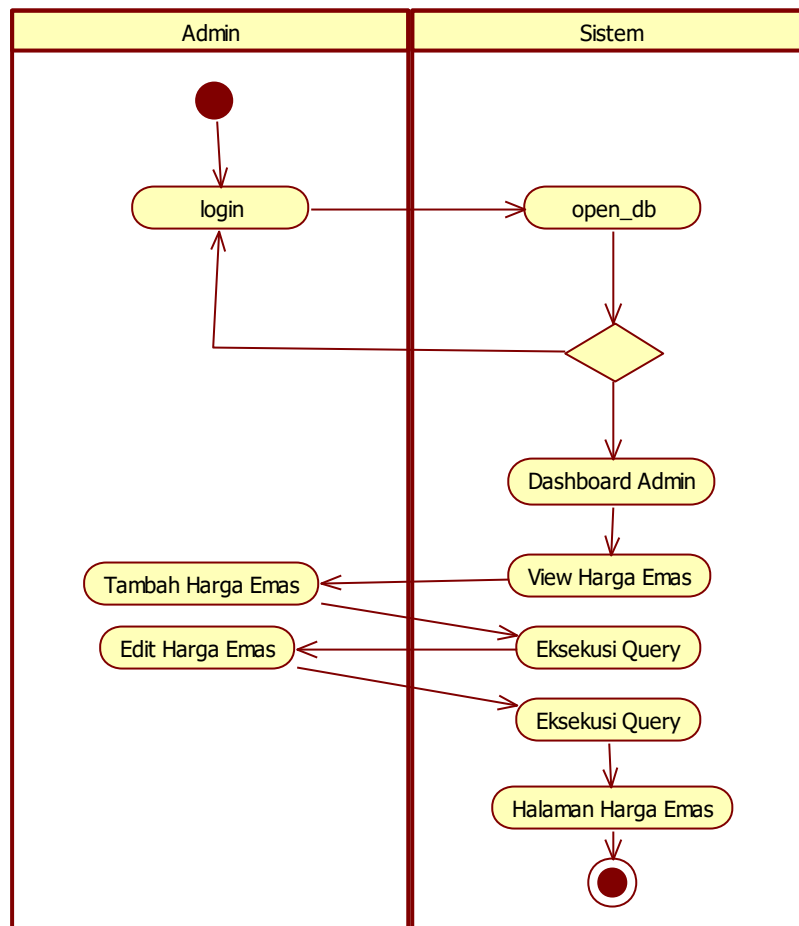


Gambar 3.29 Perancangan *Activity Diagram* Kelola Data Login (Nasabah)

Keterangan :

Pertama Nasabah memasukkan *Username* dan *Password* pada *form login* lalu controller Nasabah akan bertindak dan membuka koneksi ke *database*, setelah itu sistem akan memvalidasi *user* nasabah dan mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke dashboard nasabah.

### 3.1.5.2 Activity Diagram Kelola Data Harga Emas



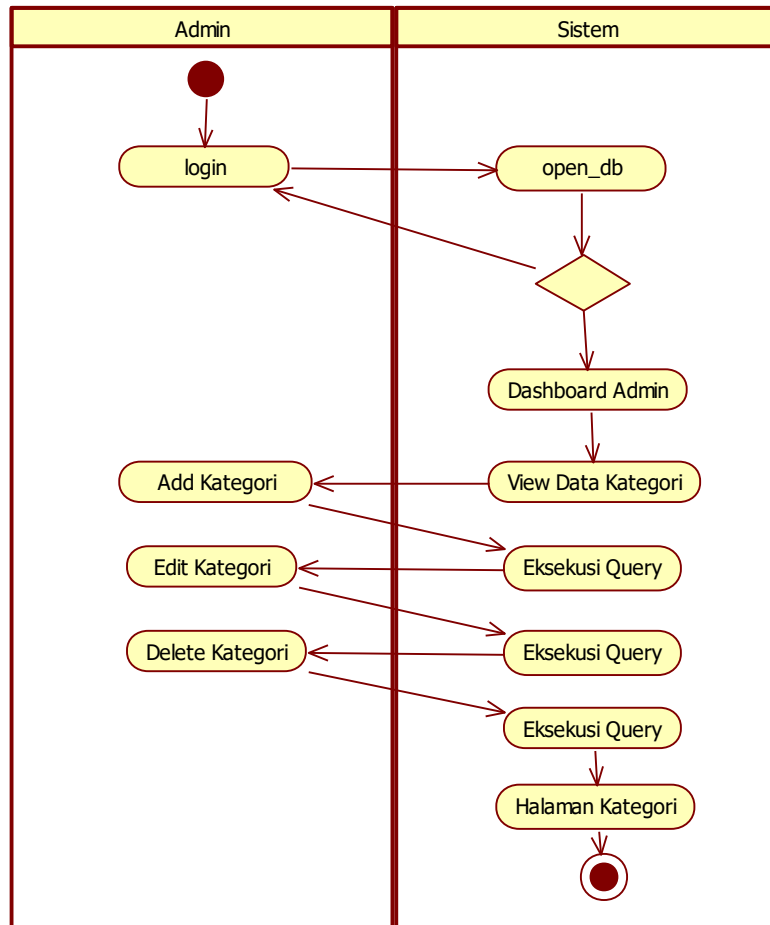
Gambar 3.30 Perancangan *Activity Diagram* Kelola Data Harga Emas

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman harga emas, sistem membuka database untuk menampilkan data harga emas. Admin memiliki aksi seperti *view*, *add*, dan *edit* data emas harian (hitungannya per-1 gram). Jika sudah selesai admin kembali ke halaman harga emas.

### 3.1.5.3 Activity Diagram Kelola Data Sampah

#### 1. Kategori

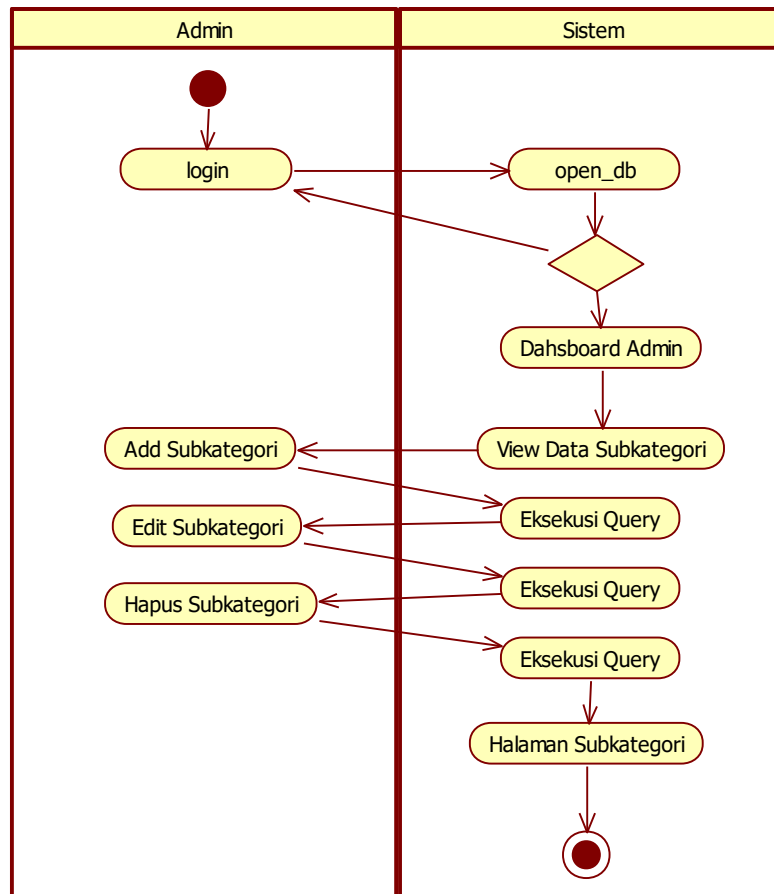


Gambar 3.31 Perancangan *Activity Diagram* Kelola Data Kategori

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman kategori, sistem membuka database untuk menampilkan data kategori sampah. Admin memiliki aksi seperti *view*, *add*, dan *edit* data kategori sampah sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman kategori.

## 2. Subkategori

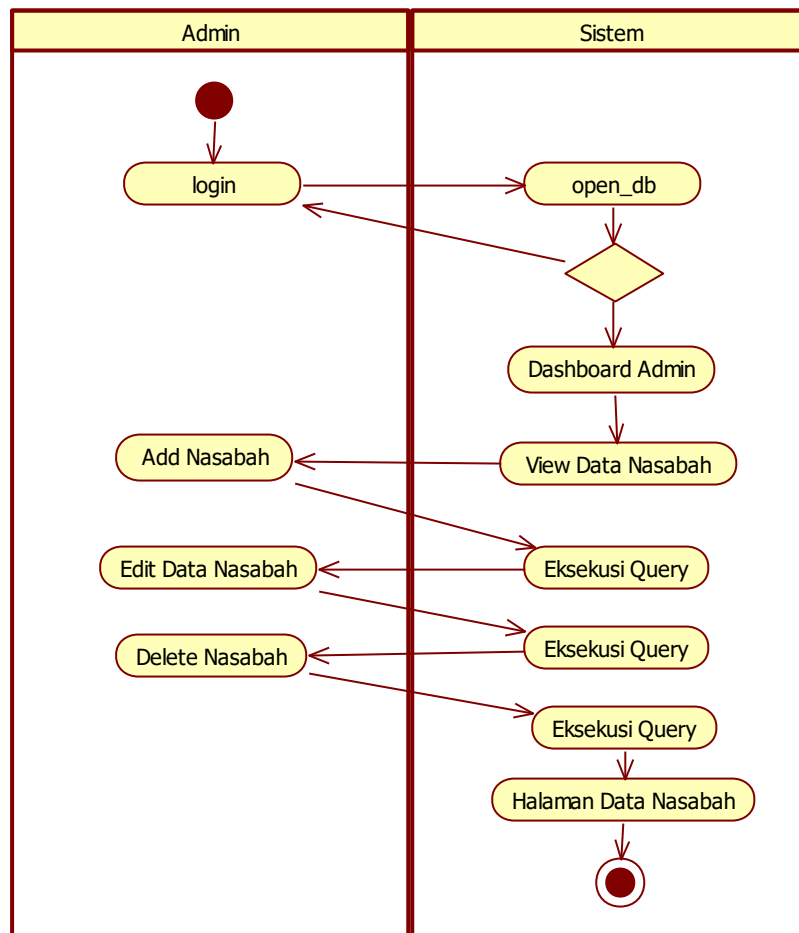


Gambar 3.32 Perancangan Activity Diagram Kelola Data Subkategori

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman subkategori, sistem membuka *database* untuk menampilkan data subkategori sampah. Admin memiliki aksi seperti *view*, *add*, dan *edit* data subkategori sampah sewaktu-waktu dibutuhkan. Jika sudah selesai *admin* kembali ke halaman subkategori.

### 3.1.5.4 Activity Diagram Kelola Data Nasabah

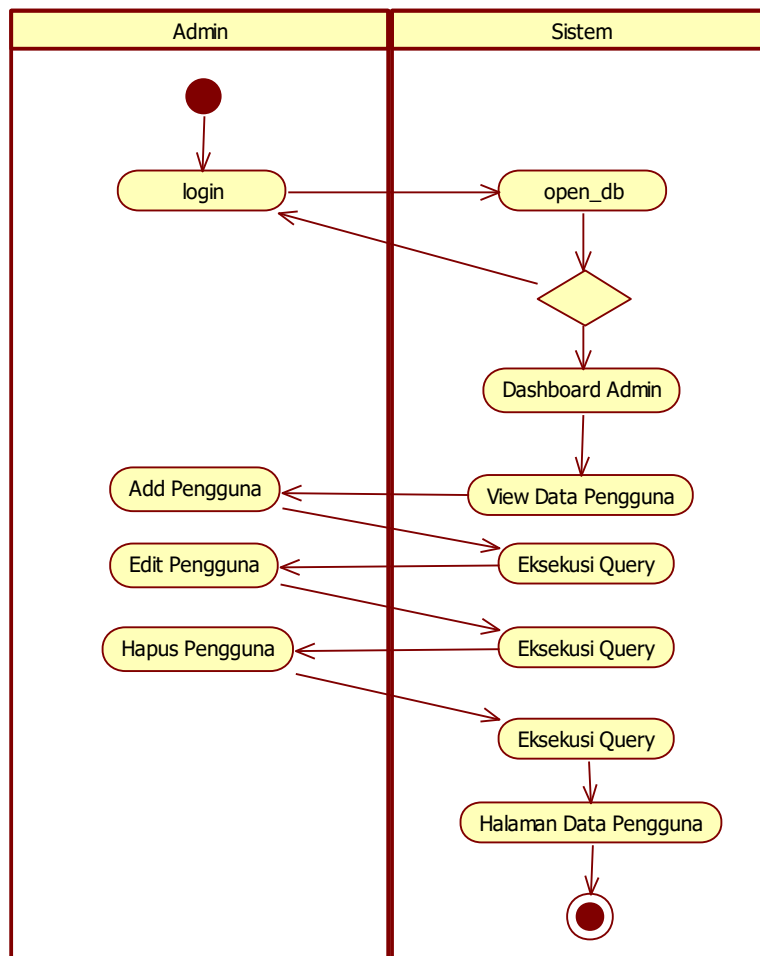


Gambar 3.33 Perancangan *Activity Diagram* Kelola Data Nasabah

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Data Nasabah, sistem membuka *database* untuk menampilkan data Nasabah. *Admin* memiliki aksi seperti *view*, *add*, dan *edit* data nasabah sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman data nasabah.

### 3.1.5.5 Activity Diagram Kelola Data Pengguna



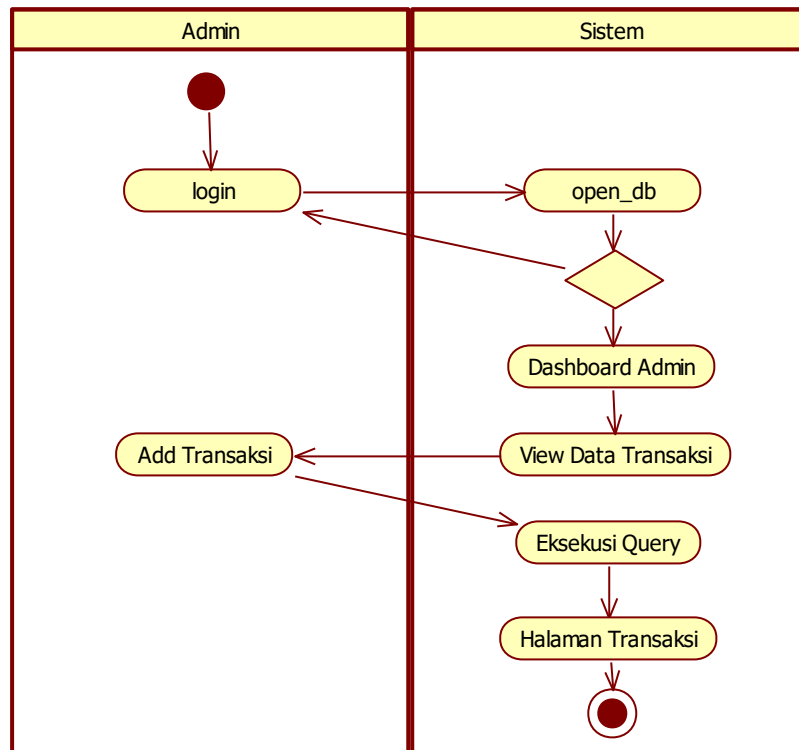
Gambar 3.34 Perancangan *Activity Diagram* Kelola Data Pengguna

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Data Pengguna, sistem membuka *database* untuk menampilkan data Pengguna sistem. *Admin* memiliki aksi seperti *view*, *add*, dan *edit* data pengguna sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman data pengguna.

### 3.1.5.6 Activity Diagram Kelola Data Transaksi

#### 1. Admin

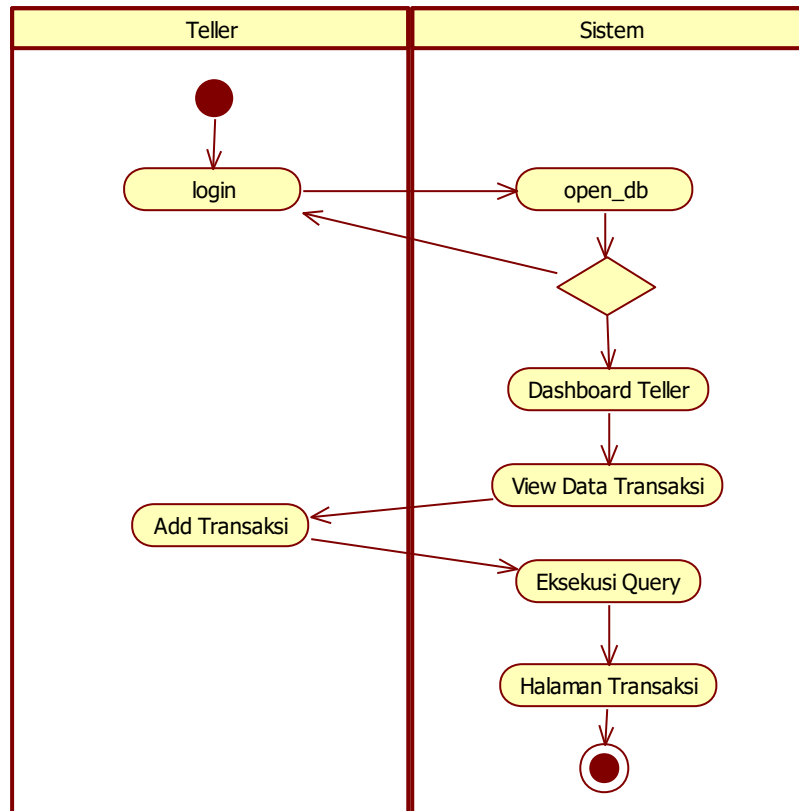


Gambar 3.35 Perancangan *Activity Diagram* Kelola Data Transaksi (*Admin*)

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Transaksi, sistem membuka *database* untuk menampilkan data Transaksi. *Admin* memiliki aksi seperti *view* dan *add* data transaksi sewaktu-waktu dibutuhkan. Jika sudah selesai *admin* kembali ke halaman data transaksi.

## 2. Teller



Gambar 3.36 Perancangan *Activity Diagram* Kelola Data Transaksi (Teller)

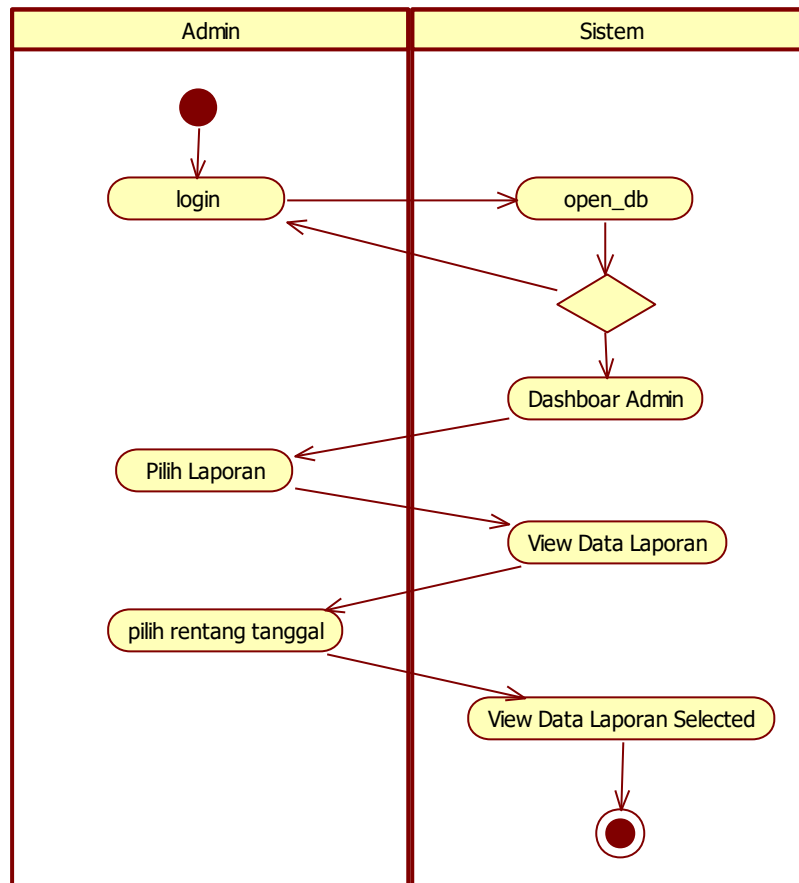
Keterangan :

*Teller* melakukan *login* untuk masuk ke sistem. Setelah itu *teller* membuka halaman Transaksi, sistem membuka *database* untuk menampilkan data Transaksi. *Teller* memiliki aksi seperti *view* dan *add* data transaksi sewaktu-waktu dibutuhkan. Jika sudah selesai *teller* kembali ke halaman data transaksi.



### 3.1.5.7 Activity Diagram Kelola Data Laporan

#### 1. Admin

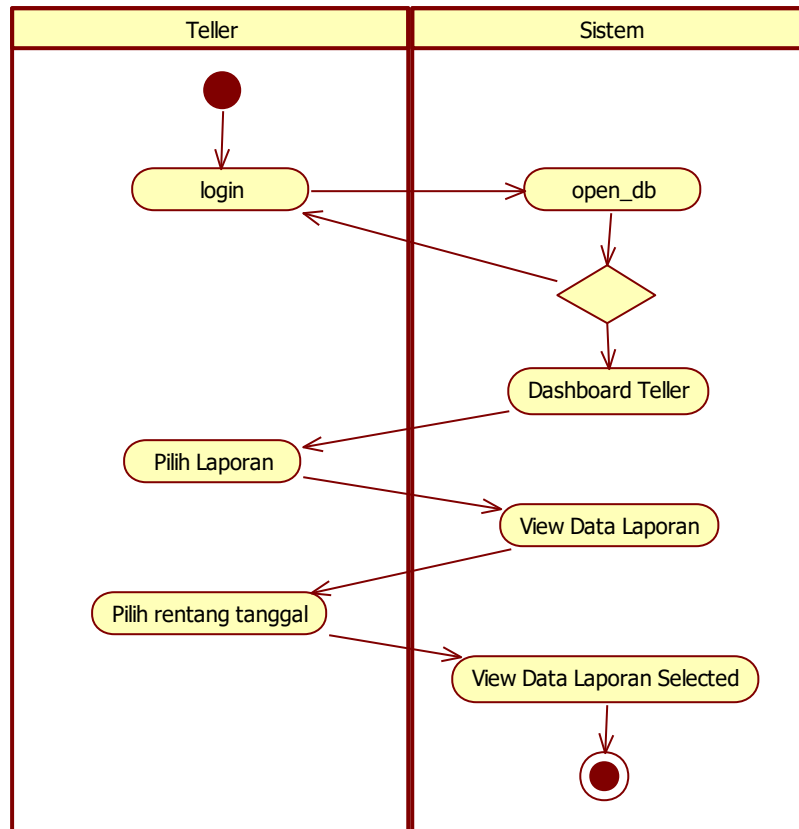


Gambar 3.37 Perancangan *Activity Diagram* Kelola Data Laporan (*Admin*)

Keterangan :

*Admin* melakukan *login* untuk masuk ke sistem. Setelah itu *admin* membuka halaman Laporan, sistem membuka *database* untuk menampilkan data Laporan. *Admin* memiliki hanya memiliki aksi *view*.

## 2. Teller



Gambar 3.38 Perancangan *Activity Diagram* Kelola Data Laporan (Teller)

Keterangan :

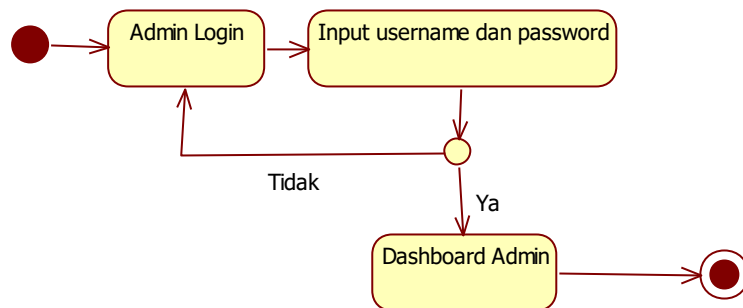
*Teller* melakukan *login* untuk masuk ke sistem. Setelah itu *teller* membuka halaman Laporan, sistem membuka *database* untuk menampilkan data Laporan. *Teller* memiliki hanya memiliki aksi *view*.

### 3.1.6 Statechart Diagram

*Statechart diagram* adalah suatu diagram yang menggambarkan daur hidup (*behavior pattern*) dari sebuah objek, dari awal objek tersebut diinisialisasi sampai di *destroy*. Menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimulasi yang diterima. Adapun *Statechart Diagram* pada aplikasi Bank Sampah Istimewa yaitu sebagai berikut :

#### 3.1.6.1 Statechart Diagram Login

##### 1. Admin

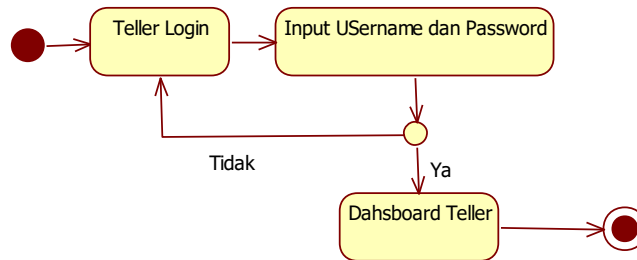


Gambar 3.39 Perancangan *Statechart Diagram Login (Admin)*

Keterangan :

Pertama *Admin* memasukan *Username* dan *Password* pada *form login* lalu *controller Admin* akan bertindak dan membuka koneksi ke *database*, setelah itu sistem akan memvalidasi user admin dan mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke *dashboard admin*.

## 2. Teller

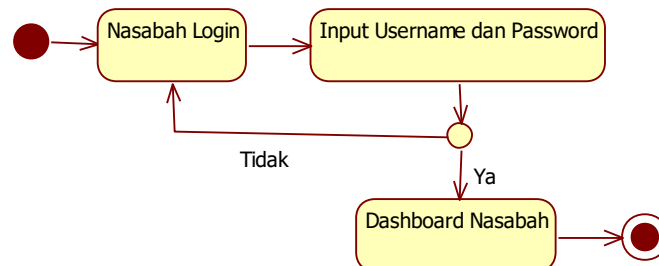


Gambar 3.40 Perancangan *Statechart Diagram Login (Teller)*

Keterangan :

Pertama *Teller* memasukan *Username* dan *Password* pada *form login* lalu *controller Teller* akan bertindak dan membuka koneksi ke *database*, setelah itu sistem akan memvalidasi *user teller* dan mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke *dashboard teller*.

## 3. Nasabah



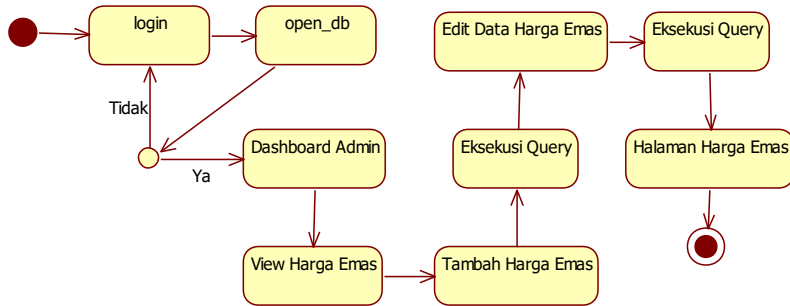
Gambar 3.41 Perancangan *Statechart Diagram Login (Nasabah)*

Keterangan :

Pertama *Nasabah* memasukan *Username* dan *Password* pada *form login* lalu *controller Nasabah* akan bertindak dan membuka koneksi ke *database*, setelah itu sistem akan memvalidasi *user nasabah* dan

mencocokkan dengan *database*, jika berhasil maka akan diarahkan ke dashboard nasabah.

### 3.1.6.2 Statechart Diagram Kelola Data Harga Emas



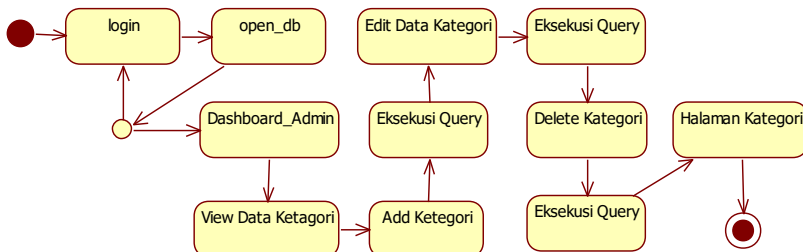
Gambar 3.42 Perancangan *Statechart Diagram* Kelola Data Harga Emas

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman harga emas, sistem membuka database untuk menampilkan data harga emas. Admin memiliki aksi seperti *view*, *add*, dan *edit* data emas harian (hitungannya per-1 *gram*). Jika sudah selesai admin kembali ke halaman harga emas.

### 3.1.6.3 Statechart Diagram Kelola Data Sampah

#### 1. Kategori

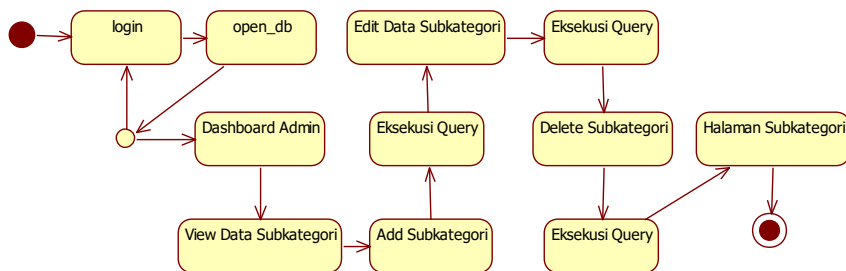


Gambar 3.43 Perancangan *Statechart Diagram* Kelola Data Kategori

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman kategori, sistem membuka database untuk menampilkan data kategori sampah. Admin memiliki aksi seperti *view*, *add*, dan *edit* data kategori sampah sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman kategori.

## 2. Subkategori

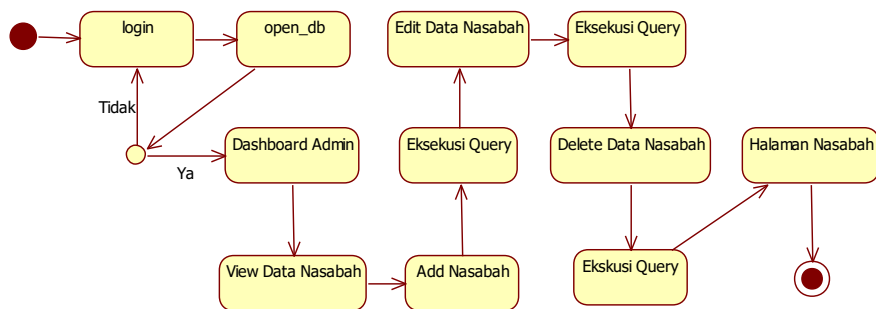


Gambar 3.44 Perancangan *Statechart Diagram* Kelola Data Kategori

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman subkategori, sistem membuka *database* untuk menampilkan data subkategori sampah. Admin memiliki aksi seperti *view*, *add*, dan *edit* data subkategori sampah sewaktu-waktu dibutuhkan. Jika sudah selesai *admin* kembali ke halaman subkategori.

### 3.1.6.4 Statechart Diagram Kelola Data Nasabah

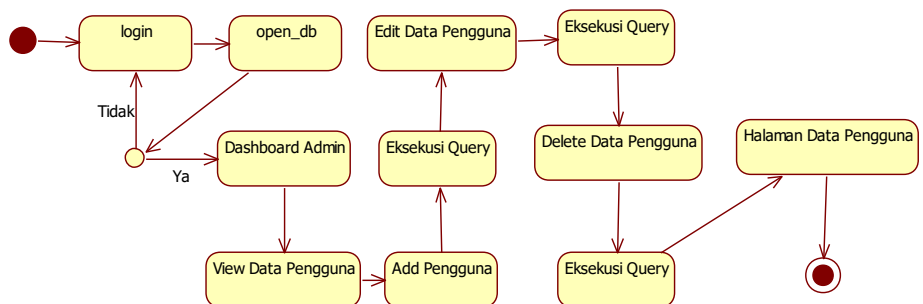


Gambar 3.45 Perancangan *Statechart Diagram* Kelola Data Subkategori

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Data Nasabah, sistem membuka *database* untuk menampilkan data Nasabah. *Admin* memiliki aksi seperti *view*, *add*, dan *edit* data nasabah sewaktu-waktu dibutuhkan. Jika sudah selesai *admin* kembali ke halaman data nasabah.

### 3.1.6.5 Statechart Diagram Kelola Data Pengguna



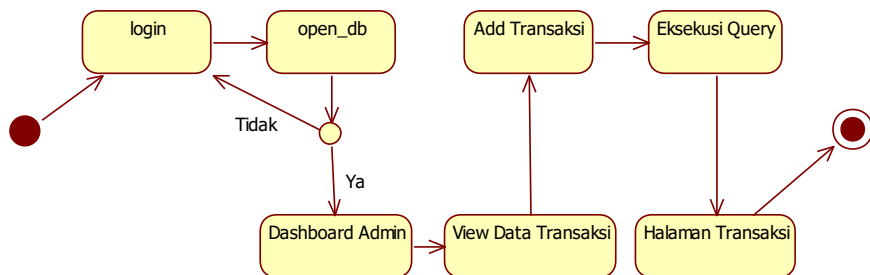
Gambar 3.46 Perancangan *Statechart Diagram* Kelola Data Pengguna

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Data Pengguna, sistem membuka *database* untuk menampilkan data Pengguna sistem. *Admin* memiliki aksi seperti *view*, *add*, dan *edit* data pengguna sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman data pengguna.

### 3.1.6.6 Statechart Diagram Kelola Data Transaksi

#### 1. *Admin*



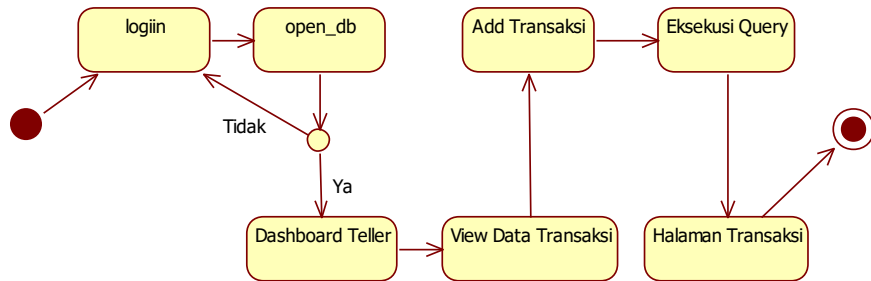
Gambar 3.47 Perancangan *Statechart Diagram* Kelola Data Transaksi (*Admin*)

Keterangan :

*Admin* melakukan *login* untuk masuk ke system. Setelah itu *admin* membuka halaman Transaksi, sistem membuka *database* untuk menampilkan data Transaksi. *Admin* memiliki aksi seperti *view* dan *add* data transaksi sewaktu-waktu dibutuhkan. Jika sudah selesai admin kembali ke halaman data transaksi.



## 2. Teller



Gambar 3.48 Perancangan *Statechart Diagram* Kelola Data Transaksi (*Teller*)

Keterangan :

*Teller* melakukan *login* untuk masuk ke sistem. Setelah itu *teller* membuka halaman Transaksi, sistem membuka *database* untuk menampilkan data Transaksi. *Teller* memiliki aksi seperti *view* dan *add* data transaksi sewaktu-waktu dibutuhkan. Jika sudah selesai *teller* kembali ke halaman data transaksi.

### 3.1.6.7 Statechart Diagram Kelola Data Laporan

#### 3.1.7