

# SISTEM INFORMASI PENJUALAN DENGAN KONTROL TRANSAKSI SECARA ONLINE MENGGUNAKAN CODEIGNITER

Luqman Affandi<sup>1)</sup>

<sup>1</sup>Teknologi Informasi - Manajemen Informatika, Politeknik Negeri Malang  
email: laffandi@yahoo.com

## Abstrack

*Point Of Sale (POS) is a computer program that is used to assist the process of buying and selling goods. The program includes a warehouse transaction records, the process at the cashier and management report for the owner. The program is not limited to use in the store, but in a few places that provide public services and selling goods, such as pharmacies, clinics and more.*

*In a small scale, POS program is usually run by the owner of the store itself or entrusted to one of its employees. Owners or employees of the store will input the name of goods, quantity of goods, managing stock, serving customers, and to make daily reports as well as income. If this is the case, then the owner will be able to control all activities of transactions.*

*If the owners are not always at the store, then allow to happen fraud committed by employees, in order to avoid this it is necessary to control the transactions that occur in order to minimize the possibility of fraud by having each transaction will always be reported online via email owners, and also the final recap report daily transactions at the time of the transaction employee to log out of the system.*

**Keywords :** POS, online

## 1. PENDAHULUAN

*Point Of Sale (POS)* atau sistem informasi penjualan adalah sebuah program komputer yang digunakan untuk membantu proses jual beli barang. Program tersebut meliputi pencatatan transaksi gudang, proses di kasir dan manajemen laporan untuk pemilik. Program tersebut tidak terbatas digunakan dalam toko, melainkan di beberapa tempat layanan umum yang menyediakan jual beli barang, seperti apotik, klinik dan sebagainya.

Program POS pada umumnya bersifat *offline* di Internet, yaitu program *desktop* atau web yang memanfaatkan sistem *client server*. Bahkan beberapa jenis POS juga masih bersifat *standalone* yaitu program hanya dapat dijalankan di satu komputer saja. Hal ini terjadi karena biaya implementasi POS secara *online* cukup besar. Besarnya biaya ditimbulkan karena harus membeli IP *public* agar dapat diakses secara *online*, penyediaan *server* sendiri atau dapat menggunakan jasa ISP berupa

*Virtual Private Server (VPS)* dan tentunya jalur Internet. Jika menggunakan program POS yang *standalone* atau *client server*, pengguna program hanya menggunakan beberapa komputer dan memanfaatkan jaringan *Local Area Network (LAN)*.

Dalam skala kecil, program POS biasanya dijalankan oleh pemilik toko sendiri atau dipercayakan ke salah satu karyawannya. Pemilik atau karyawan toko akan menginputkan nama barang, jumlah barang, mengelola stok, melayani pembeli, dan sampai membuat laporan harian maupun rugi laba. Jika hal ini terjadi, maka pemilik akan dapat mengontrol seluruh kegiatan transaksi yang terjadi.

Jika pemilik tidak selalu berada di toko, maka semua kegiatan dilakukan oleh pegawai. Setiap hari pegawai tersebut akan membuat laporan dan selanjutnya dilaporkan ke pemilik secara periodik. Selain merekap untuk dibuat laporan, nota-nota yang telah dicetak juga diarsipkan dan diberikan ke pemilik untuk dilakukan pengecekan. Hal ini akan memakan waktu

yang cukup lama baik dari sisi pemilik maupun karyawan, karena harus bekerja dua kali, merekap dan mengecek transaksi.

Sistem yang seperti ini memungkinkan terjadi beberapa kecurangan seperti, menghilangkan nota transaksi penjualan, memanipulasi laporan dan sebagainya. Beberapa cara yang ditempuh oleh para pemilik toko adalah dengan memasang kamera pengintai (CCTV), tetapi hal ini juga sulit dilakukan pengecekan karena harus dilihat ulang semua kejadian oleh pemilik.

Untuk mengatasi hal tersebut, perlu dibuatkan sebuah program POS yang bekerja secara *offline* atau menggunakan jaringan LAN tetapi dapat mengirimkan nota transaksi secara *realtime* ke pemilik agar mendapatkan laporan sedini mungkin.

Hal ini dapat dilakukan dengan cara menambahkan fasilitas kirim *email* secara otomatis ke pemilik. Dengan menambahkan fasilitas tersebut, toko tidak memerlukan biaya yang besar. Toko hanya perlu modem untuk mengirim email yang berisi data transaksi. Sedangkan program POS tetap dijalankan secara *offline* atau menggunakan jaringan LAN dengan sistem *client server*.

Pengembangan POS dengan pengiriman *email* secara otomatis untuk dapat mengirimkan data transaksi secara *online* dapat dibangun menggunakan bahasa pemrograman berbasis *framework* yaitu *CodeIgniter*. *Framework* tersebut selain menerapkan konsep *Model View Controller (MVC)* yang menerapkan konsep *Object Oriented Programming*, juga memiliki vasilitas *library* yang cukup lengkap. Salah satunya adalah *library send email*. Pengolahan data dapat dilakukan dengan membuat program *CRUD (Create, Read, Update, Delete)* dengan bantuan *generator Grocery CRUD*. Sedangkan untuk otomatisasi pengiriman *email* menggunakan *library send email*.

## 2. KAJIAN LITERATUR

### • Sistem

Menurut Sutarman (2012:13), sistem adalah kumpulan elemen yang saling

berhubungan dan saling berinteraksi dalam satu kesatuan untuk menjalankan suatu proses pencapaian suatu tujuan utama.

Sedangkan menurut Raymond McLeod (dalam Al-Bahra Bin Ladjamudin, 2013:3), sistem adalah sekelompok elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan. Dalam buku yang sama, menurut Gordon B. Davis, sistem sebagai bagian-bagian yang saling berkaitan yang beroperasi bersama untuk mencapai beberapa sasaran atau maksud. Jadi sistem merupakan kumpulan elemen yang terintegrasi untuk beroperasi bersama dalam suatu proses untuk mencapai suatu tujuan atau sasaran.

### • Informasi

Menurut Tata Sutabri (2012:22), informasi adalah data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan.

Menurut Gordon B. Davis (dalam Al-Bahra Bin Ladjamudin, 2013:8), informasi adalah data yang telah diolah menjadi sebuah bentuk yang berguna dan nyata atau berupa nilai yang dapat dipahami dalam keputusan sekarang maupun yang akan datang. Jadi informasi adalah data yang telah diolah menjadi suatu bentuk yang dapat digunakan untuk proses pengambilan keputusan.

### • Sistem informasi

Menurut Al-Bahra Bin Ladjamudin (2013:13) mendeskripsikan sistem informasi sebagai sekumpulan prosedur organisasi yang pada saat dilaksanakan akan memberikan informasi bagi pengambil keputusan dan atau untuk mengendalikan organisasi.

Menurut Tata Sutabri (2012:38), sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi organisasi yang bersifat manajerial dalam kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan – laporan yang diperlukan.

Menurut Tata Sutabri (2012), klasifikasi sistem informasi tersebut sebagai

berikut :

- a. Sistem informasi berdasarkan level organisasi, dikelompokkan menjadi level operasional, level fungsional dan level manajerial.
- b. Sistem informasi berdasarkan aktifitas manajemen, dikelompokkan menjadi sistem informasi perbankan, sistem informasi akademik, sistem informasi kesehatan, sistem informasi asuransi dan sistem informasi perhotelan.
- c. Sistem informasi berdasarkan fungsionalitas bisnis, dikelompokkan menjadi sistem informasi akuntansi, sistem informasi keuangan, sistem informasi manufaktur, sistem informasi pemasaran dan sistem informasi sumber daya manusia.

Jadi sistem informasi adalah suatu sistem yang memberikan informasi bagi pengambil keputusan dengan laporan-laporan yang dibutuhkan.

- **Online**

Menurut kamus Oxford, *online* dapat diartikan sebagai “*while connected to a computer or under computer control, by means of the Internet or other computer network*” yang dapat diartikan secara bebas dengan saat terhubung ke komputer atau dibawah kontrol komputer, dalam arti kontrol Internet atau komputer jaringan yang lain.

- **Email**

Menurut kamus oxford, *email* adalah “*Messages distributed by electronic means from one computer user to one or more recipients via a network or The system of sending messages by electronic*” yang dapat diartikan secara bebas yaitu pesan yang didistribusikan secara elektronik dari pengguna satu komputer ke satu atau lebih penerima melalui jaringan atau sistem pengiriman pesan melalui sarana elektronik.

- **CodeIgniter**

Akhir-akhir ini *CodeIgniter* menjadi sebuah *framework* yang hangat dibicarakan di Indonesia. Hampir semua milis dan forum PHP banyak membahas masalah *CodeIgniter*. Terlebih lagi banyak perusahaan-perusahaan ternama di Indonesia (Kompas.com, okezone.com,

urbanesia.com, bejubel.com, dan lain-lain) yang telah menggunakan *CodeIgniter* dalam produk mereka.

*CodeIgniter* adalah sebuah *framework* PHP yang dapat mempercepat pengembang untuk membuat sebuah aplikasi web. Ada banyak *library* dan *helper* yang berguna didalamnya dan tentunya mempermudah proses *development*. Ibarat ingin membangun rumah maka Anda tidak perlu membuat semen, memotong kayu menjadi papan, mengubah batu menjadi porselen dan lain-lain. Anda cukup memilih komponen-komponen tersebut lalu dikombinasikan menjadi rumah yang indah.

Keuntungan yang didapat dalam penggunaan *framework* adalah :

- **Menghemat Waktu Pengembangan**
  - Dengan struktur dan *library* yang telah disediakan oleh *framework* maka tidak perlu lagi memikirkan hal-hal tersebut, jadi Anda hanya fokus ke proses bisnis yang akan dikerjakan.
- **Reuse of code** – Dengan menggunakan *framework* maka pekerjaan kita akan memiliki struktur yang baku, sehingga kita dapat menggunakannya kembali di proyek-proyek lainnya.
- **Bantuan komunitas** - Ada komunitas-komunitas yang siap membantu jika ada permasalahan, selain itu juga bisa berbagi ilmu sehingga dapat meningkatkan kemampuan pemrograman kita.
- **Kumpulan best practice** – sebuah *framework* merupakan kumpulan *best practice* yang sudah teruji. Jadi kita dapat meningkatkan kualitas kode kita.

Sebelum mendalami *CodeIgniter* lebih jauh, sebaiknya dipahami terlebih dahulu apa itu *framework*. *Framework* adalah sebuah struktur konseptual dasar yang digunakan untuk memecahkan sebuah permasalahan, bahkan isu-isu kompleks yang ada.

Sebuah *framework* telah berisi sekumpulan arsitektur/konsep-konsep yang dapat mempermudah dalam pemecahan

sebuah permasalahan. *Framework* bukanlah peralatan/*tools* untuk memecahkan sebuah masalah, tetapi sebagai alat bantu. *Framework* hanya menjadi sebuah konstruksi dasar yang menopang sebuah konsep atau sistem yang bersifat *essential support*.

Salah satu alasan mengapa orang menggunakan *framework* terutama dalam membangun sebuah aplikasi adalah kemudahan yang ditawarkan. Didalam sebuah *framework* biasanya sudah tersedia struktur aplikasi yang baik, *standard coding* (1), *best practice* (2) dan *design pattern* (3), dan *common function* (4). Dengan menggunakan *framework* kita dapat langsung fokus kepada *business process* yang dihadapi tanpa harus berfikir banyak masalah struktur aplikasi, standar *coding* dan lain-lain.

Dengan memanfaatkan *design pattern* dan *common function* yang telah ada di dalam *framework* maka hal tersebut dapat mempercepat proses pengembangan aplikasi. Kita tidak perlu membuat sesuatu fungsionalitas yang bersifat umum. Tanpa disadari ketika kita membangun sebuah aplikasi yang banyak melibatkan banyak fungsionalitas yang telah dibangun itu ternyata sama atau berulang-ulang. Dengan pengelempokkan itulah kita dapat mempercepat pengembangan aplikasi.

Sedangkang *CodeIgniter* adalah sebuah *web application framework* yang bersifat *open source* digunakan untuk membangun aplikasi php dinamis. Tujuan utama pengembangan *CodeIgniter* adalah untuk membantu *developer* untuk mengerjakan aplikasi lebih cepat daripada menulis semua kode dari awal. *CodeIgniter* menyediakan berbagai macam *library* yang dapat mempermudah dalam pengembangan. *CodeIgniter* diperkenalkan kepada publik pada tanggal 28 februari 2006. *CodeIgniter* sendiri dibangun menggunakan konsep *Model-View-Controller development pattern*.

*CodeIgniter* sendiri merupakan salah satu *framework* tercepat dibandingkan dengan *framework* lainnya. Pada acara frOSCon (Agustus 2008), pembuat php

Rasmus Lerdorf mengatakan dia menyukai *CodeIgniter* karena dia lebih ringan dan cepat dibandingkan *framework* lainnya.

*CodeIgniter* sangat ringan, terstruktur, mudah dipelajari, dokumentasi lengkap dan dukungan yang luar biasa dari forum *CodeIgniter*. Selain itu *CodeIgniter* juga memiliki fitur-fitur lainnya yang sangat bermanfaat, antara lain:

- Menggunakan *Pattern MVC*. Dengan menggunakan *pattern MVC* ini, struktur kode yang dihasilkan menjadi lebih terstruktur dan memiliki standar yang jelas.
- *URL Friendly*. URL yang dihasilkan sangat *user friendly*. Pada *CodeIgniter* diminimalisasi penggunaan *\$\_GET* dan di gantikan dengan *URI*.
- Kemudahan. Kemudahan dalam mempelajari, membuat *library* dan *helper*, memodifikasi serta mengintegrasikan *Library* dan *helper*.

Jika kita membandingkan antara *CodeIgniter* dengan *framework* lainnya maka

beberapa poin yang membuat *CodeIgniter* unggul adalah:

- Kecepatan. Berdasarkan hasil *benchmark CodeIgniter* merupakan salah satu *framework* PHP tercepat yang ada saat ini.
- Mudah dimodifikasi dan beradaptasi. Sangat mudah memodifikasi *behavior framework* ini. Tidak membutuhkan *server requirement* yang macam-macam serta mudah mengadopsi *library* lainnya.
- Dokumentasi lengkap dan jelas. Bahkan tanpa buku ini pun *CodeIgniter* sebenarnya telah menyediakan sebuah panduan yang lengkap mengenai

*CodeIgniter*. Semua informasi yang anda butuhkan tentang *CodeIgniter* ada disana.

- **Learning Curve Rendah.** *CodeIgniter* sangat mudah dipelajari. Dalam pemilihan *framework* hal ini sangat penting diperhatikan karena kita juga harus memperhatikan *skill* dari seluruh anggota team. Jika sebuah *framework* sangat sulit dipelajari maka akan beresiko untuk memperlambat tim pengembang.  
([www.codeigniter.com](http://www.codeigniter.com) dan [www.koder.web.id](http://www.koder.web.id))

- **Grocery CRUD**

*Grocery CRUD* adalah sebuah *library open source* yang membuat hidup pengembang menjadi lebih mudah. Hanya dengan menggunakan beberapa baris kode anda dapat membuat *CRUD* yang stabil dengan desain yang bagus. Sebuah sistem yang sepenuhnya komplis bahkan bagi pemrogram PHP pemula dapat dengan mudah menggunakannya.

Dengan *Grocery CRUD* Anda tidak perlu khawatir dengan perbedaan browser karena sistem ini sudah mendukung semua browser terbaru seperti *Mozilla Firefox*, *Google Chrome*, *Safari*, *Internet Explorer*, *Opera* dan lain-lain.

Anda tidak perlu melakukan sesuatu yang khusus untuk mengimplementasikan *Grocery* ini dalam proyek Anda. Cukup ikuti langkah 5 menit instalasi mudah ini dan Anda akan dengan mudah menempatkan *Grocery CRUD* yang handal pada proyek Anda. Tema standar yang diberikan adalah *Flexgrid ajax*.

Sebagai pengembang web biasanya Anda akan bermasalah dengan pembuatan proyek sistem *backoffice* yang sederhana, stabil dan aman. Berapa banyak *CRUD* yang harus Anda siapkan untuk pelanggan Anda, dan berapa banyak waktu yang dibutuhkan untuk membuatnya? Biasanya membutuhkan waktu berminggu-minggu untuk membuat semua logika bisnis model dan desain awal untuk setiap tabel, seperti yang kita ketahui banyak hal sederhana

yang sebenarnya sama yaitu *Create* (membuat serta menambah), *Update* (mengganti) dan *Delete* (menghapus) dan tentu saja tabel untuk menampilkan data.

Dengan *Grocery CRUD* dan *CodeIgniter framework* yang handal, Anda dapat membuat sistem *CRUD* penuh hanya dalam satu menit. Dengan *generator codeigniter CRUD* Anda tidak perlu menyalin semua kode *css*, *javascript*, *tabel*, *form*, *grid*, *function*, *model*, *libraries*, *view* lagi dan lagi dalam sistem *backoffice*. Hanya dengan beberapa baris kode Anda siap untuk memiliki CMS Anda sendiri. *CRUD* bekerja dengan baik dengan *CodeIgniter 2.0.x* dan *2.1.x*. ([www.grocerycrud.com](http://www.grocerycrud.com)).

- **Unified Modelling Language**

*Unified Modeling Language* (UML) adalah himpunan struktur dan teknik untuk pemodelan dan desain program berorientasi objek (OO) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OO dan sekelompok tool untuk mendukung pengembangan sistem tersebut. (David M. Kroenke, 2005:60)

- **Use Case**




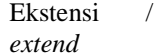
*Use case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

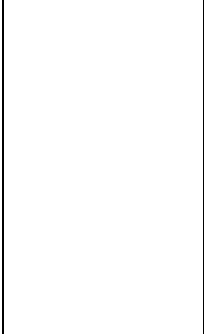

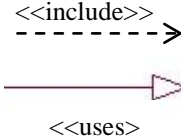
Syarat penamaan *use case* adalah nama didefinisikan sesederhana mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

- Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

- *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Berikut adalah simbol-simbol yang ada pada diagram *use case*:

Tabel 1 Simbol diagram *Use Case*

| Simbol  | Nama                          | Keterangan   |
|---|-------------------------------|--|
|    | <i>Use case</i>               | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama <i>use case</i> |
|  | Aktor / <i>actor</i>          | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri   |
|  | Asosiasi / <i>association</i> | Komunikasi antara aktor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> memiliki interaksi dengan aktor   |
|  | Ekstensi / <i>extend</i>      | Relasi <i>use case</i>   |

|  |                                      |  |
|--|--------------------------------------|--|
|    |                                      | tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu  |
|    | Generalisasi / <i>generalization</i> | Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya  |
|  | Menggunakan / <i>include / uses</i>  | Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> di mana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini |


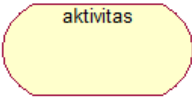
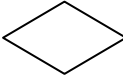

- **Activity Diagram**



Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan adalah diagram aktivitas menggambarkan aktivitas sistem, bukan apa yang dilakukan oleh aktor.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

- Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
  - Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
  - Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
  - Rancangan menu yang ditampilkan pada perangkat lunak.
- Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel 2 Simbol Diagram Aktivitas

| Simbol  | Nama                          | Keterangan   |
|---|-------------------------------|--|
|  | Status awal                   | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal |
|  | Aktivitas                     | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja      |
|  | Percabangan / <i>decision</i> | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu             |
|  | Penggabungan / <i>join</i>    | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan                 |

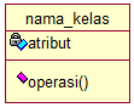

|  |              |  |
|--|--------------|--|
|  | Status akhir | Status akhir yang dilakukan oleh sistem, sebuah diagram aktivitas memiliki sebuah status akhir |
|  | Swimlane     | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi            |


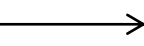

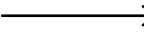

- **Class Diagram**

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

- Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
  - Operasi atau metode adalah fungsi-fungsi yang dimiliki suatu kelas.
- Simbol-simbol yang ada pada diagram kelas:

Tabel 3 Simbol Diagram Kelas

| Simbol  | Nama                         | Keterangan  |
|---|------------------------------|---|
|  | Kelas                        | Kelas pada struktur sistem  |
|  | Antarmuka / <i>interface</i> | Merupakan kumpulan operasi tanpa implementasi dari suatu class. Implementasi operasi dalam interface dijabarkan |

|   |  |  |
|---|--|--|
|   |  | oleh operasi dalam class   |
|    | Asosiasi / <i>association</i>                  | Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i>   |
|    | Asosiasi berarah / <i>directed association</i> | Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i> |
|    | Generalisasi                                   | Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)  |
|  | Kebergantungan / <i>dependency</i>             | Relasi antar kelas dengan makna kebergantungan antar kelas   |
|  | Agregasi / <i>aggregation</i>                  | Relasi antar kelas dengan makna semua-bagian ( <i>whole-part</i> )   |

### 3. METODE PENELITIAN

Dalam penelitian ini dilakukan studi kasus di D'Va Klinik Kecantikan Genteng Banyuwangi. Di klinik tersebut dilakukan penjualan berupa produk-produk kecantikan, serta resep yang dipakai dalam perawatan pasien. Proses pengelolaan dan penjualan barang dilakukan oleh pegawai klinik, sedangkan semua kegiatan diawasi oleh seorang dokter yang sekaligus menjadi pemilik klinik.

Metode yang digunakan dalam penelitian ini adalah metode pengembangan sistem menggunakan *agile development*

*method* yaitu pengembangan perangkat lunak dimana pengembang dan pengguna program saling duduk bersama untuk menyelesaikan sebuah proyek program. Tahapan pengumpulan data, analisis masalah, desain, koding dan implementasi dilakukan secara bersama-sama dan berkesinambungan.

Dari hasil pengumpulan data didapat beberapa fakta yaitu Klinik tersebut sudah menggunakan program berbasis *desktop*, tetapi program tersebut belum bisa menangani stok dengan baik, sehingga berakibat diproses transaksi pembelian barang serta dilaporkan akhir termasuk laporan rugi laba. Dokter sebagai pemilik harus selalu memasukkan barang baru untuk menambah stok, padahal dokter harus memeriksa pasien, sehingga tidak memungkinkan untuk memasukkan stok secara *uptodate*. Selain itu juga terbentur dengan kesibukan dokter tersebut di tempat lain, sehingga tidak selalu ada di klinik. Padahal proses stok barang dan penjualan barang harus berjalan terus. Dokter menginginkan proses stok dilakukan oleh karyawan begitu juga proses penjualan, tetapi dokter sebagai pemilik juga menginginkan dapat melihat setiap saat transaksi dan laporan di kliniknya.

Solusi yang dapat ditawarkan yaitu dengan membuat program POS online agar dokter dapat melihat semua kegiatan transaksi yang terjadi. Tetapi hal ini tidak mungkin dilakukan karena terkendala dengan sarana prasarana. Jika dibangun *POS online*, maka diperlukan *server* atau sewa *server*, serta harus terhubung dengan *IP public*. Koneksi Internet di klinik juga harus terjamin konektifitasnya. Untuk mendapatkan koneksi yang terjamin dapat dilakukan dengan berlangganan Internet *dedicated*. Dua hal ini akan menjadi kendala tersendiri jika akan diterapkan. Selain itu, keberadaan *Internet Service Provider (ISP)* di sekitar klinik juga terbatas.

Klinik tersebut telah memiliki jaringan Internet dengan layanan *up to*. Selain itu juga sudah terdapat jaringan LAN yang menghubungkan antar ruangan. Beberapa



komputer personal juga sudah terkoneksi dengan jaringan tersebut serta sudah mendapatkan layanan Internet.

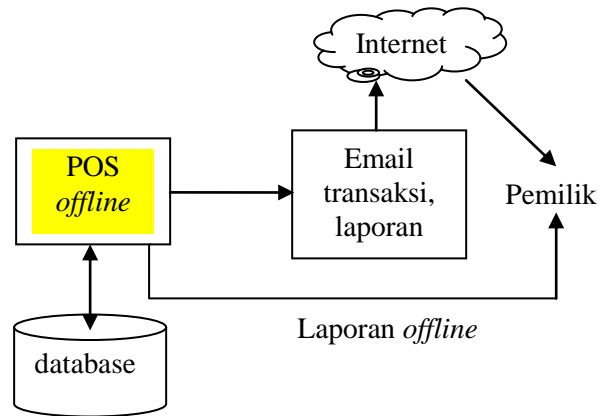
Setelah dilakukan analisis dari kendala dan fasilitas yang ada, maka dapat dilakukan pengembangan program POS berbasis web menggunakan *CodeIgniter*. Hal ini dilakukan agar program dapat dijalankan di jaringan LAN yang ada di klinik dan tidak perlu di-online-kan. Program tersebut dilengkapi dengan pengiriman data transaksi dan laporan menggunakan fasilitas *email*. *Email* tersebut dapat dikirimkan ke alamat *email* dokter selaku pemilik dengan memanfaatkan jaringan Internet yang ada. Sehingga proses kontrol terhadap transaksi yang terjadi dapat dilakukan dimana dan kapan saja.

Dari hasil analisis masalah, diperlukan pengiriman data secara *online* meliputi transaksi stok, transaksi penjualan barang dan perawatan, retur barang, serta laporan pendapatan per hari dan rugi laba. Proses tersebut dapat diletakkan pada:

- Transaksi Stok : proses kirim *email* dapat diletakkan pada saat proses simpan di program transaksi stok barang, karena data yang sudah di simpan pastinya adalah data yang baru dimasukkan oleh petugas gudang, sehingga proses ini dapat menangkal kecurangan.
- Transaksi penjualan : proses kirim *email* dapat diletakkan pada saat proses cetak nota. Data di nota adalah data transaksi penjualan yang sah yang dilakukan oleh petugas kasir dengan pelanggan. Hal ini juga dapat mengurangi kecurangan.
- Retur barang, laporan pendapatan dan rugi laba : proses pengiriman data ini diletakkan pada proses *logout*, karena dapat diasumsikan bahwa jika pegawai selesai melakukan transaksi di akhir jam kerja, mereka melakukan

*logout* dari program. Pada saat proses tersebut diletakkan perintah kirim *email* untuk mengirim data retur dan perhitungan pendapatan serta laporan perhitungan rugi laba.

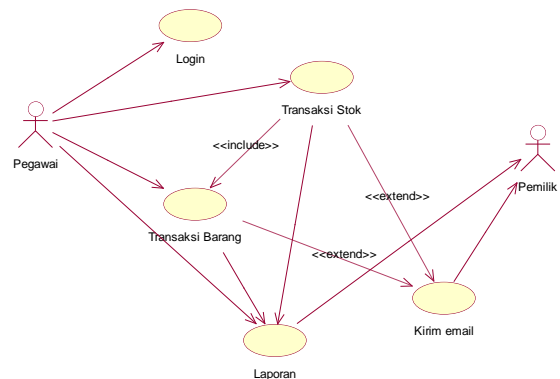
Gambar 1 menunjukkan diagram alur kerja sistem secara keseluruhan :



Gambar 1. Alur Kerja Sistem

Untuk dapat menerapkan alur kerja tersebut diperlukan sebuah rancangan desain program menggunakan konsep *Object Oriented Design (OOD)* yang terdiri dari *use case*, *activity diagram*, *class diagram*.

Gambar 2 berikut menunjukkan *use case* dari sistem yang dibuat:

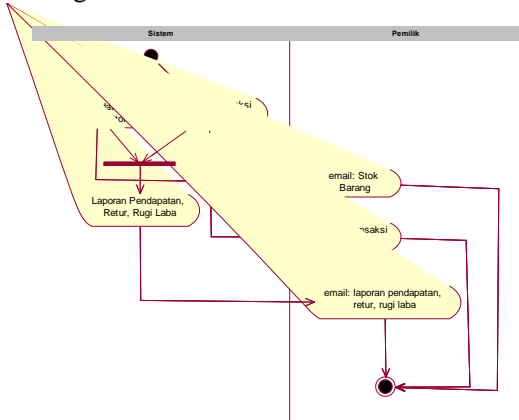


Gambar 2 Diagram *use case*

*Use Case* tersebut menggambarkan

pegawai dapat melakukan *login* kemudian mengelola transaksi stok (pembelian), transaksi barang (penjualan) dan laporan. Dari hasil transaksi stok dan barang akan digunakan untuk proses kirim *email* ke pemilik sehingga setiap saat terjadi transaksi pemilik secara langsung akan mengetahuinya.

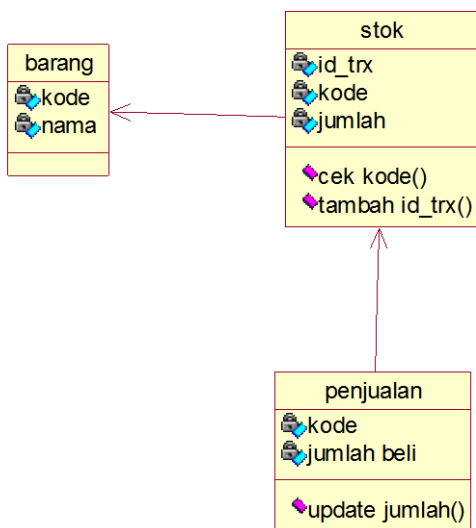
Diagram aktivitas



Gambar 3 Diagram Aktifitas

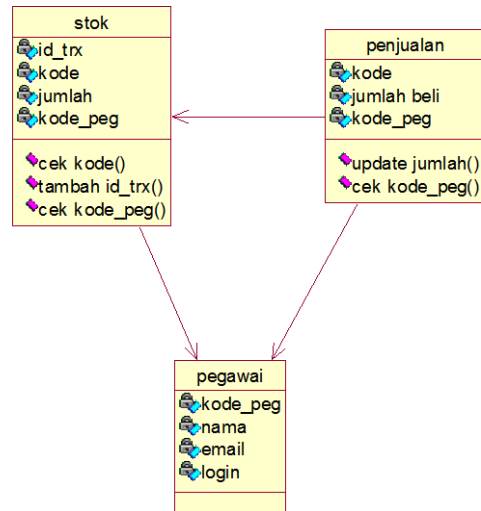
Aktivitas yang dilakukan pada proses pengiriman *email* adalah, setelah terjadi transaksi stok dan transaksi barang akan digunakan sebagai perhitungan laporan pendapatan, retur dan rugi laba. Hasil tersebut akan dikirimkan pada saat pegawai logout sistem.

Kelas diagram transaksi



Gambar 4 Diagram Kelas Transaksi  
Jurnal Teknologi Informasi Vol. 7 No. 1

Kelas diagram kirim *email*

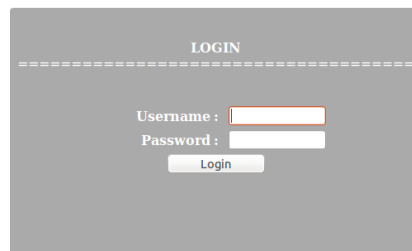


Gambar 5 Diagram Kelas Kirim *email*

#### 4. HASIL DAN PEMBAHASAN

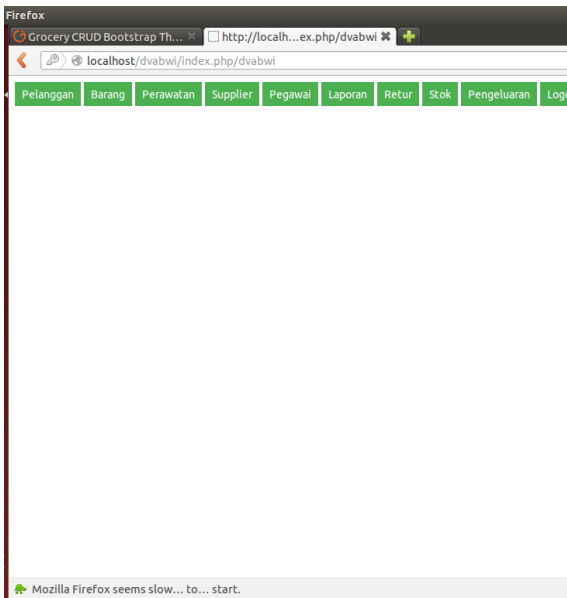
Setelah dilakukan desain dan koding, maka program dapat diimplementasikan di sebuah *server* lokal yang dapat diakses menggunakan jaringan LAN. *Server* dan *client* terhubung dengan jaringan Internet melalui sebuah modem ADSL yang digunakan untuk pengiriman *email* secara otomatis oleh program.

Berikut ini adalah *interface* dari program POS dengan kontrol transaksi secara *online*:



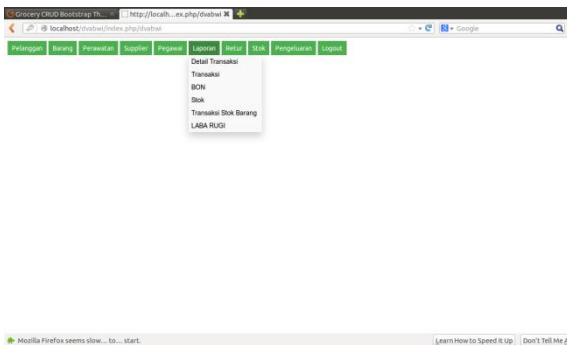
Gambar 6 *Login* Pegawai dan Dokter

*Form login* pegawai dan dokter digunakan untuk memverifikasi siapa dan hak apa yang akan diberikan. Dalam program ini terdapat 3 kelompok pemakai, yaitu pemilik (*owner*), bagian kasir dan gudang.



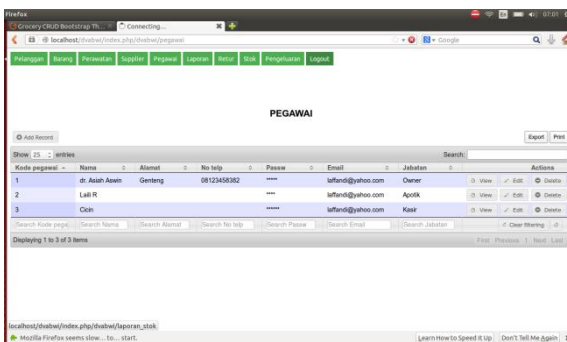
Gambar 7 Menu Pemilik

Pada menu pemilik (*owner*), semua from akan ditampilkan karena pemilik sebagai pemegang hak penuh terhadap aplikasi.



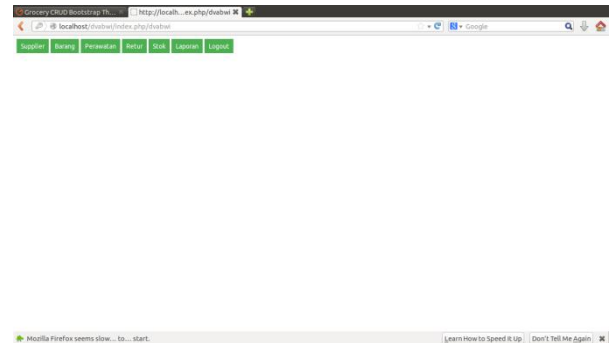
Gambar 8 Laporan secara *offline*

Menu laporan secara *offline* dapat digunakan untuk mengetahui semua laporan kegiatan yang terjadi.



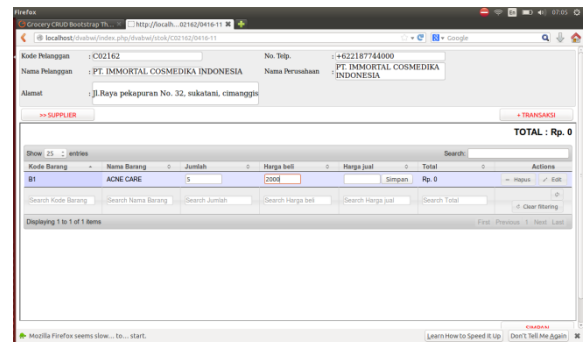
Gambar 9 Setting *email* untuk laporan *online*

Form ini digunakan untuk mendaftarkan *email* pemilik yang digunakan untuk mengirimkan laporan transaksi stok, barang dan pendapatan jika sewaktu-waktu akan diubah alamatnya.



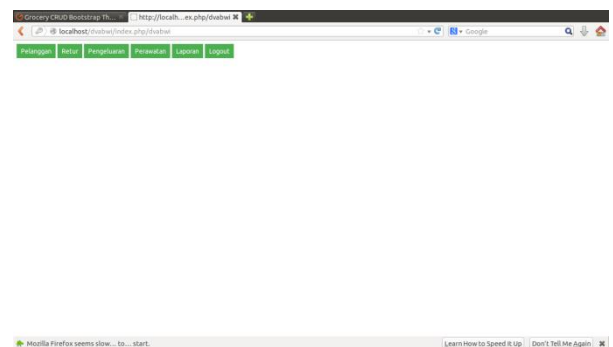
Gambar 10 Menu untuk Pegawai Gudang

Untuk pegawai gudang menu yang ditampilkan adalah untuk transaksi stok barang.



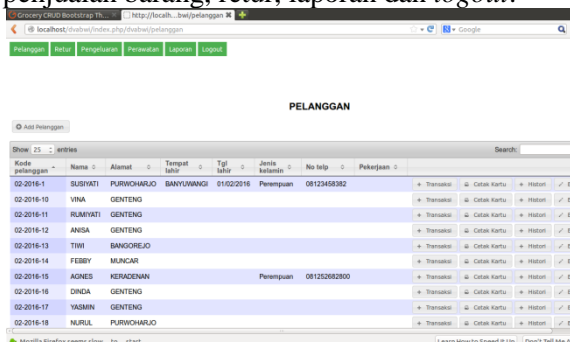
Gambar 11 transaksi stok

Di program transaksi stok, terdapat proses kirim *email*.



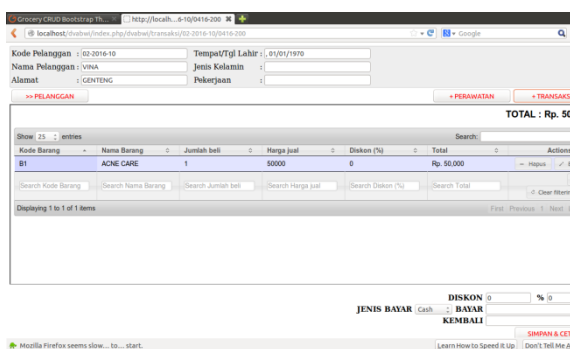
Gambar 12 Menu Kasir

Menu ini digunakan untuk melayani pelanggan mulai dari pendataan pelanggan, penjualan barang, retur, laporan dan *logout*.



Gambar 13 Pengelolaan Pelanggan

Menu pengelolaan pelanggan, pegawai dapat mendata namapelanggan kemudian terdapat fasilitas untuk langsung menuju menu transaksi, dapat juga mencetak kartu pelanggan, histori dan ubah data.



Gambar 14 Transaksi Penjualan

Pada program transaksi pelanggan, setelah pegawai melakukan proses simpan dan cetak secara otomatis sistem akan melakukan proses kirim *email* kepada pemilik, hal ini dapat menjadi bahan pengecekan bagi pemilik pada tiap transaksi yang terjadi.

## 5. KESIMPULAN

Kesimpulan:

1. Program POS dapat dijalankan secara offline, tetapi pengecekan transaksi dapat dilakukan secara *realtime* menggunakan sistem email.
2. Biaya yang diperlukan untuk implementasi sistem lebih murah

Jurnal Teknologi Informasi Vol. 7 No. 1

karena hanya membutuhkan jaringan Internet berupa modem untuk mengirim *email*.

## 6. REFERENSI

Al-BahraBinLadjamudin. 2013. *Analisis dan Desain Sistem Informasi*. Yogyakarta: Graha Ilmu.

A.S. Rosa. M. Shalahuddin. 2013. *“Rekayasa Perangkat Lunak. Terstruktur dan Berorientasi Objek”*. Bandung: Informatika.

Ellislab. 1996. *“CodeIgniter”*. <http://www.codeigniter.com/>. Diakses tanggal 2 Maret 2016 jam 14.00.

Oxforddictionaries. *“online”*. <http://www.oxforddictionaries.com/definition/english/online>: diakses tanggal 2 Maret 2016 jam 11:00

Oxforddictionaries. *“online”*. <http://www.oxforddictionaries.com/definition/english/email>: diakses tanggal 2 Maret 2016 jam 11:15

Skoumbourdis, John. 2016. *“Grocery CRUD”*. <http://www.grocerycrud.com/>. Diakses tanggal 2 Maret 2016 jam 15.00

Sutabri, Tata. 2012. *Analisis Sistem Informasi*. Bandung: Informatika.

Sutarman, 2012. *Pengantar Teknologi Informasi*. Jakarta: PT. Bumi Aksara.

W3snoop. 2011. *“CodeIgniter”*. <http://koder.web.id.w3snoop.com/>. Diakses tanggal 2 Maret 2016 jam 14.30.