SISTEM PENJUALAN *SPARE PART* TOKO AJM MOTOR MENGGUNAKAN CI BERBASIS ARSITEKTUR MVC

Ade Setiadi
Jurusan Magister Ilmu Komputer
Universitas Budi Luhur
Email: adesetiadi.ubl@gmail.com

Fifit Alfiah
Jurusan Magister Ilmu Komputer
Universitas Budi Luhur
Email: fifitalfiah.ubl@gmail.com

ABSTRAK

Saat ini penggunaan komputer dan perangkat lunak semakin banyak hingga ke berbagai kehidupan, bahkan sampai pada bidang ekonomi. Sistem Informasi merupakan peranan sangat penting dalam kegiatan bisnis di suatu perusahaan. Toko AJM Motor bergerak dalam bidang penjualan spare part mobil dan masih menggunakan cara yang konvensional dalam mengolah data tentang transaksi penjualan, untuk itu Toko AJM Motor membutuhkan sistem informasi yang dapat menunjang kelancaran dalam melakukan transaksi penjualannya. Dalam dunia teknologi khususnya pemrograman saat ini, baik itu *desktop* maupun *web base* semakin marak pengerjaannya menggunakan *framework* dan salah satu *framework* berbasis PHP yang banyak digunakan yaitu *CodeIgniter* (CI). Framework CI memang dikembangkan untuk memudahkan dalam developing aplikasi dengan struktur file *source code*-nya menggunakan pendekatan arsitektur *Models-Views-Controller* (MVC) dan pemrograman berorientasi objek. Oleh sebab itu, kami menggunakan CI dalam *developing* aplikasi ini dengan metode *Object Oriented Analysis and Design* sebagai metode pengembangan system. Dengan dirancangnya sistem ini telah mempermudah pemilik toko dalam mengelola data pelanggan, *supplier* dan barang yang di beli dan di jual serta pembuatan laporan yang diperlukan untuk perhitungan penjualan dan dapat memberikan informasi yang berguna bagi pemilik toko secara *up to date*.

Kata kunci: *codeigniter* (CI), *models-views-controller* (MVC), o*bject oriented*, penjualan.

ABSTRACT

Currently who using of computers and software increasingly to the various of life, even in the economic field. System Information is a very important role in business activities in a company. Shop AJM Motor is engaged in the sale of spare parts of cars and still use the conventional way in data processing of sales transactions, to the shop AJM Motor need information systems that can support the smooth running perform sales transactions. In the world of technology, especially programming at this time, be it a desktop or web base increasingly widespread use of the process framework and a PHP based framework that is widely used is CodeIgniter (CI). CI Framework was developed to facilitate in developing applications with source code file structure its approach Models architecture-Views-Controller (MVC) and object-oriented programming. Therefore, we use the CI in developing this application with the method of Object Oriented Analysis and Design as a method of system development. With this system he designed, enables shop owners to manage the data of customers, suppliers and goods that are bought and sold as well as the preparation of reports required for the calculation of sales and can provide useful information for the store owners are up to date.

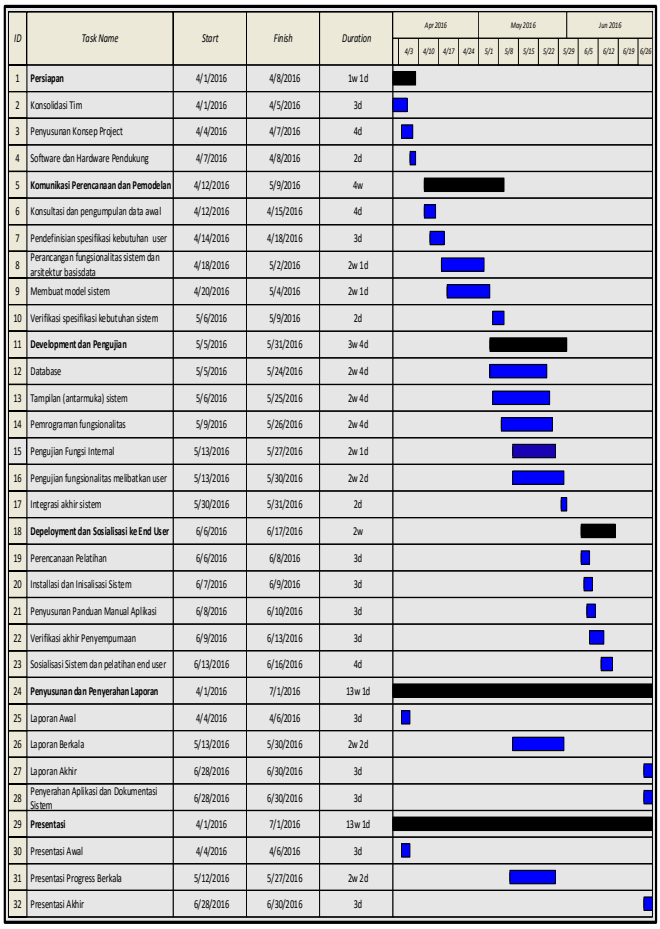
Keywords: *codeigniter (CI), models-view-controller (MVC), object oriented, sales.*

1. PENDAHULUAN

Cara Peran sistem informasi terhadap kemajuan organisasi sudah tidak diragukan lagi. Dengan dukungan sistem informasi yang baik maka sebuah perusahaan akan memiliki berbagai keunggulan kompetitif sehingga mampu bersaing dengan perusahaan lain. Pemanfaatan komputer sebagai alat kerja bantu, khususnya sebagai media pengolah data, baik yang berskala besar maupun skala kecil terus berkembang dengan pesat. Itu semua berkat kemajuan teknologi yang didorong oleh keinginan manusia untuk dapat melakukan pekerjaan yang cepat, tepat dan aman.

Bagi suatu perusahaan yang terus berkembang seperti pada AJM *Auto Parts*. Momentum Teknik sebagai suatu perusahaan yang bergerak di bidang penjualan *sparepart* mobil, tentunya memiki suatu sistem penjualan yang

berguna untuk mempermudah pemilik dalam mengatasi masalah-masalah yang selalu timbul terutama dalam laporan barang masuk, laporan barang keluar serta laporan stok ketersediaan barang agar berjalan sesuai dengan prosedurnya sehingga hasilnya dapat dengan mudah untuk mengontrol data dan dalam pengambilan keputusan penjual.



ID	Task Name	Start	Finish	Duration
1	**Persiapan**	4/1/2016	4/8/2016	1w 1d
2	Konsolidasi Tim	4/1/2016	4/5/2016	3d
3	Penyusunan Konsep Project	4/4/2016	4/7/2016	4d
4	Software dan Hardware Pendukung	4/7/2016	4/8/2016	2d
5	**Komunikasi Perencanaan dan Pemodelan**	4/12/2016	5/9/2016	4w
6	Konsultasi dan pengumpulan data awal	4/12/2016	4/15/2016	4d
7	Pendefinisian spesifikasi kebutuhan user	4/14/2016	4/18/2016	3d
8	Perancangan fungsionalitas sistem dan arsitektur basisdata	4/18/2016	5/2/2016	2w 1d
9	Membuat model sistem	4/20/2016	5/4/2016	2w 1d
10	Verifikasi spesifikasi kebutuhan sistem	5/6/2016	5/9/2016	2d
11	**Development dan Pengujian**	5/5/2016	5/31/2016	3w 4d
12	Database	5/5/2016	5/24/2016	2w 4d
13	Tampilan (antarmuka) sistem	5/6/2016	5/25/2016	2w 4d
14	Pemrograman fungsionalitas	5/9/2016	5/26/2016	2w 4d
15	Pengujian Fungsi Internal	5/13/2016	5/27/2016	2w 1d
16	Pengujian fungsionalitas melibatkan user	5/13/2016	5/30/2016	2w 2d
17	Integrasi akhir sistem	5/30/2016	5/31/2016	2d
18	**Depeloyment dan Sosialisasi ke End User**	6/6/2016	6/17/2016	2w
19	Perencanaan Pelatihan	6/6/2016	6/8/2016	3d
20	Installasi dan Inisalisasi Sistem	6/7/2016	6/9/2016	3d
21	Penyusunan Panduan Manual Aplikasi	6/8/2016	6/10/2016	3d
22	Verifikasi akhir Penyempurnaan	6/9/2016	6/13/2016	3d
23	Sosialisasi Sistem dan pelatihan end user	6/13/2016	6/16/2016	4d
24	**Penyusunan dan Penyerahan Laporan**	4/1/2016	7/1/2016	13w 1d
25	Laporan Awal	4/4/2016	4/6/2016	3d
26	Laporan Berkala	5/13/2016	5/30/2016	2w 2d
27	Laporan Akhir	6/28/2016	6/30/2016	3d
28	Penyerahan Aplikasi dan Dokumentasi Sistem	6/28/2016	6/30/2016	3d
29	**Presentasi**	4/1/2016	7/1/2016	13w 1d
30	Presentasi Awal	4/4/2016	4/6/2016	3d
31	Presentasi Progress Berkala	5/12/2016	5/27/2016	2w 2d
32	Presentasi Akhir	6/28/2016	6/30/2016	3d

Gambar 1. *Gantt Chart*

Diagram perencanaan yang digunakan untuk penjadwalan sumber daya dan alokasi waktu dalam pengembangan *system. Gantt Chart* juga dapat membantu penggunanya untuk memastikan bahwa semua kegiatan telah direncakan, urutan kinerja telah diperhitungkan, perkiraan waktu kegiatan telah tercatat, dan keseluruhan waktu proyek telah dibuat [1].

Program kerja yang akan dilakukan oleh tim pengembang sistem dalam proyek ini disesuaikan dengan metodologi *RAD* (*Rapid Application Development*) menggunakan *prototyping*. Dalam pengembangan sistem informasi pun khususnya berbasis *website* saat ini sudah menggunakan konsep *MVC* (*Model View Controller*) yang membuat para *developer* (pembangun aplikasi) menjadi mudah dan penulisan *source code* yang menjadi lebih rapi sehingga mudah untuk kegiatan maintenance dan pengembangan. Konsep tersebut dikembangkan atas dasar *OOP* (*Object Oriented Programming*) atau Pemrograman Berorientasi Objek dan kemudian dikembangkan lagi dalam bentuk *MVC Framework*, Salah-satu *MVC Framework* yang populer digunakan saat ini adalah *Framework CodeIgniter* yang dikembangkan oleh *Eliss Lab [9]*.

Dalam penelitian ini kami menggunakan teknik pengembangan *prototyping*, diterapkan dalam kerangka kerja penelitian tahap demi tahap dan menghasilkan sebuah *website e-commerce*[10] berbentuk *prototype* berbasis OOP, dibangun menggunakan arsitektur MVC dan diterjemahkan menggunakan *framework Codeigniter*. Pada tahap analisis dan desain pada fase *prototyping* menggunakan alat analisis- desain *Unified Modelling Language* (UML), dan aplikasi diuji dengan menggunakan teknik *black box* untuk memastikan secara fungsional *prototype* dapat berjalan dengan baik.

2. METODOLOGI

2.1 Teknik Pengembangan Prototyping

Metode *prototyping [2]* sering digunakan pada dunia nyata. Karena metode ini secara keseluruhan akan mengacu kepada kepuasan pengguna. Bisa dikatakan bahwa metode ini merupakan metode *waterfall* yang dilakukan secara berulang-ulang.

2.2 Tahapan Metode Prototyping

1) Pemilihan Fungsi**.** Mengacu pada pemilahan fungsi yang harus ditampilkan oleh *prototyping*. Pemilahan harus selalu dilakukan berdasarkan pada tugas-tugas yang relevan yang sesuai dengan contoh kasus yang akan diperagakan.
2) Penyusunan Sistem Informasi**.** Bertujuan memenuhi permintaan kebutuhan akan tersedianya *prototype*.
3) Evaluasi**.**
4) Penggunaan selanjutnya**.**

2.3 Jenis-Jenis Prototyping

1) *Feasibility prototyping*. Digunakan untuk menguji kelayakan dari teknologi yang akan digunakan untuk sistem informasi yang akan disusun.
2) *Requirement prototyping*. Digunakan untuk mengetahui kebutuhan aktivitas bisnis pengguna. Misalnya dalam sebuah perusahaan terdapat penggunadirektur, manajer, dan karyawan. Maka penggunaan sistem dapat dibedakan berdasarkan penggunatersebut sesuai dengan kebutuhannya.
3) *Designprototyping*. Digunakan untuk mendorong perancangan sistem informasi yang akan digunakan.
4) *Implementation prototyping*. Merupakan lanjutan dari rancangan *protype*, *prototype* ini langsung disusun sebagai suatu sistem informasi yang akan digunakan.

2.4 Keunggulan Metode Prototyping

1) Adanya komunikasi baik antara pengembang dengan pelanggan.
2) Pengembang dapat bekerja lebih baik untuk memenuhi kebutuhan pelanggan.
3) Pelanggan berperan aktif dalam pengembangan sistem.
4) Menghemat waktu dalam pengembangannya.
5) Penerapan lebih mudah karena pemakai akan mengetahui apa yang diharapkan.

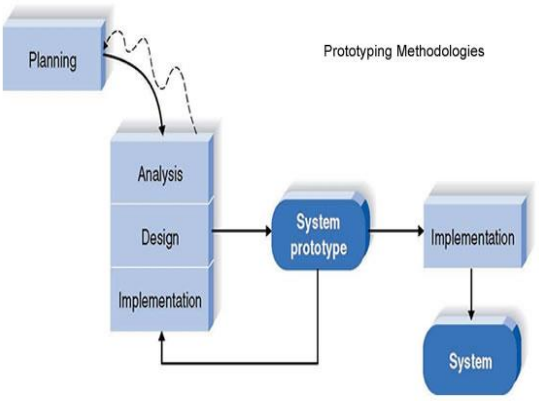
2.5 Kelemahan Metode Prototyping

1) Kualitas sistem kurang baik karena hanya mengedepankan aspek kenyamanan pengguna.
2) Pengembang kadang-kadang menggunakan implementasi yang sembarangan.
3) Tidak mencerminkan proses perancangan yang baik.

2.6 Metode Prototyping

Metode *Prototype* merupakan suatu paradigma baru dalam metode pengembangan perangkat lunak dimana metode ini tidak hanya sekedar evolusi dalam dunia pengembangan perangkat lunak, tetapi juga merevolusi metode pengembangan perangkat lunak yang lama yaitu sistem sekuensial yang biasa dikenal dengan nama SDLC atau *waterfall development* model. Metodelogi *prototyping* ditunjukkan pada gambar 2. Tahapan dalam gambar 2 metodologi *prototyping*, yaitu:

1) Perencaan dan komunikasi terlebih dahulu yang dilakukan antara pelanggan dengan tim pemgembang perangkat lunak mengenai spesifikasi kebutuhan yang diinginkan.
2) Akan dilakukan perencanaan dan pemodelan secara cepat berupa rancangan cepat (*quick design*) dan kemudian akan memulai konstruksi pembuatan *prototype*.
3) *Prototype* kemudian akan diserahkan kepada para *stakeholder* untuk dilakukan evaluasi lebih lanjut sebelum diserahkan kepada para pembuat *software*.
4) Pembuatan *software* sesuai dengan *prototype* yang telah dievaluasi yang kemudian akan diserahkan kepada pelanggan.
5) Jika belum memenuhi kebutuhan dari pelanggan maka akan kembali ke proses awal sampai dengan kebutuhan dari pelanggan telah terpenuhi.



Gambar 2. Metodologi *Prototyping [2]*

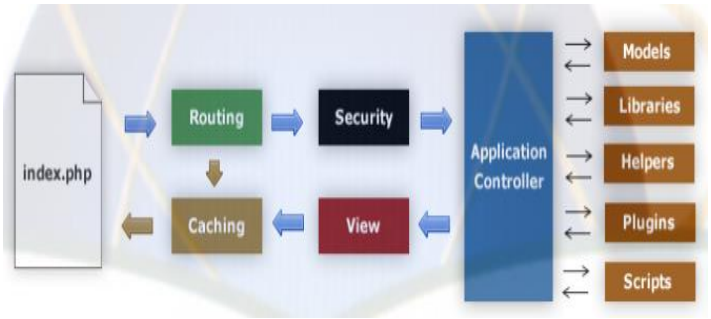
3. LANDASAN TEORI

3.1 CodeIgniter Framework

CodeIgniter adalah *framework* aplikasi *web* yang *open source* untuk bahasa pemrograman PHP. *CodeIgniter* memiliki banyak fitur yang membuatnya berbeda dengan *framework* lainnya. Tidak seperti beberapa *framework* PHP lainnya, dokumentasi untuk *framework* ini sangat lengkap, yang mencakup seluruh aspek dalam *framework*. *CoeIgniter* juga mampu berjalan pada lingkungan *Shared Hosting* karena memiliki ukuran yang sangat kecil, namun memiliki kinerja yang sangat luar biasa [3].

Dari segi pemrograman, *CodeIgniter* kompatibel dengan PHP4 dan PHP5, sehingga akan berjalan dengan baik pada *web host* yang banyak dipakai saat ini. *CodeIgniter* menggunakan pola *design Model-View-Controllwe (MVC)*, yang merupakan cara untuk mengatur aplikasi web ke dalam 3 bagian yang berbeda, yaitu *Model*- lapisan abstraksi *database*, *Views*- file-file tampilan *template* depan, dan *Controller*- logika bisnis dari aplikasi. Pada intinya *CodeIgniter* juga membuat penggunaan ekstensif dari pola *design Singleton*. Maksudnya adalah cara untuk me-*load class* sehingga jika *class* itu di panggil dalam beberapa kali, kejadian yang sama pada *class* tersebut akan digunakan kembali. Hal ini sangat berguna dalam koneksi *database*, karena kita hanya ingin menggunakan satu koneksi setiap kali *class* itu digunakan.

CodeIgniter dikembangkan oleh *Rick Ellis*, dengan versi awal yang dirilis pada tanggal 28 februari 2006. Dari tahun itulah hingga sekarang telah muncul banyak versi *CodeIgniter* yang terus berkembang dengan penambahan fitur baru dari sebelumnya. Bagaimana suatu proses data mengalir pada system yang menggunakan *CodeIgniter Framework* dapat diilustrasikan pada gambar berikut:



Gambar 3. *Application Flow Chart CodeIgniter FrameWork [9]*

Keterangan:

1) Index.php berfungsi sebagai *front controller*, menginisialisasi *base resource* untuk menjalankan *CodeIgniter*.
2) *Router* memeriksa HTTP *request* untuk memecahkan apa yang harus dilakukan dengannya.
3) Jika *Cache* aktif, maka hasilnya akan langsung dikirimkan ke *browser* dengan mengabaikan aliran data normal.
4) *Security*, sebelum *controller* dimuat HTTP *request* dan data yang dikirmkan *user* akan difilter untuk keamanan.

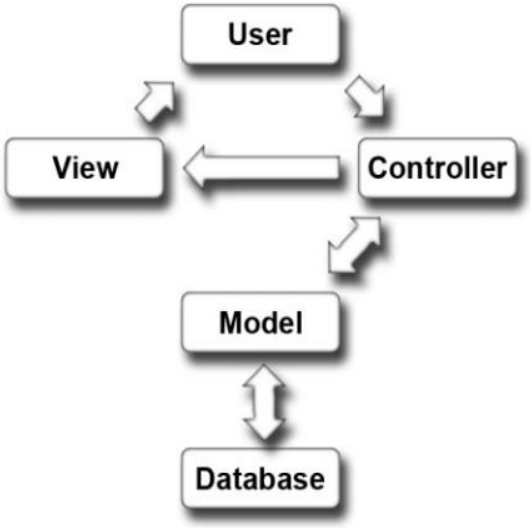
5) *Controller*, memuat *model, core libraries, plugins, helpers*, dan semua *resource* yang diperlukan untuk memproses *request*.
6) Akhirnya *view* yang dihasilkan akan dikirim ke *browser*. Jika *cache* aktif, maka *view* akan disimpan sebagai *cache* dahulu sehingga pada *request* berikutnya langsung ditampilkan.

3.2 MVC (Model View Controller)

Model-View-Control (MVC) pertama kali diperkenalkan peneliti *Xerox PARC* yang bekerja pada bahasa pemrograman *Smalltalk* di akhir 1970-an dan awal 1980-an. *Smalltalk* adalah bahasa pemrograman yang berorientasi objek, bertipe dinamis, dan reflektif. *Smalltalk* pertama kali digunakan dalam pembelajaran edukasi dan hal ini berbeda dari data *mainframe* dan struktur *control* dalam program *Smalltalk* yang terlibat pada *Windowed User Interfaces*, konsep pemrograman berorientasi objek, pengantar pesan antara komponen-komponen objek, dan kemampuan untuk memonitor dan memodifikasi struktur dan perilakunya sendiri [3].

MVC merupakan sebuah *pattern* atau teknik pemrograman yang memisahkan antar pengembang aplikasi berdasarkan komponen utama pada sebuah aplikasi, seperti manipulasi data, *user interface* dan bagian yang menjadi kontrol aplikasi [4].

Secara sederhana dapat dikatakan bahwa antara desain dan proses data berada pada tempat yang terpisah. Saat ini *MVC* merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi *web*, yang berawal pada bahasa *Smalltalk*.



Gambar 4. *Model-View-Contro [11]*

Terdapat tiga jenis komponen yang membangun suatu *MVC pattern* yang terdapat pada gambar 4, yaitu:

1) *Model- Model* berhubungan dengan data dan interaksi ke *database atau webservice*. Model juga mempresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks, XML maupun *webservice*. Biasanya didalam model akan berisi *class* dan fungsi untuk melakukan manipulasi data seperti *insert, update, delete dan search*, namun tidak dapat berhubungan dengan bagian *view* secara langsung, aplikasi *website* biasanya menggunakan *database* untuk menyimpan data, oleh karena itu model biasanya akan berhubungan dengan perintah-perintah *query SQL*.
2) *View- View* berhubungan dengan segala sesuatu yang akan ditempatkan ke *end-user*, biasa berupa halaman *web, RSS, Javascript* dan lain-lain. Programmer harus menghindari adanya logika pemrosesan data di *view*. Di dalam *view* hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. *View* dapat dikatakan sebagai halaman *website* yang dibuat menggunakan HTML dengan bantuan *CSS, Javascript dan Jquery*. Didalam *view* juga harus dihindari adanya kode untuk melakukan koneksi ke *database*. View hanya dikhususkan untuk menampilkan data-data hasil dari *model dan controller*. Bagian ini tidak memiliki akses secara langsung terhadap bagian model.
3) *Controller- Controller* merupakan penghubung antara *model dan view*. Didalam *controller* inilah terdapat *class* dan fungsi-fungsi yang memproses permintaan dari *view* kedalam struktur data didalam model. *Controller* juga tidak boleh berisi kode untuk mengakses basis data. Tugas

controller adalah menyediakan berbagai variabel yang akan ditampilkan di *view*, memanggil *model* untuk melakukan akses ke d*atabase*, menyediakan *validasi* atau pengecekan terhadap *input*.

3.3 Pemrograman Beorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya [5]. Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1) Meningkatkan produktivitas, karena kelas dan objek yan ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reuseable*)
2) Kecepatan pengembangan, karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengolahan.
3) Kemudahan pemeliharaan, Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
4) Konsistensi, karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
5) Kualitas perangkat lunak, karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan. Beberapa contoh bahasa pemrograman yang mendukung pemrograman berorientasi objek: bahasa pemrograman *Smalltalk, Eiffel, C++, PHP, Java.*

3.4 MYSQL

MySQL merupakan *database* yang paling digemari dikalangan *programmer website*, dengan alasan bahwa *program* ini merupakan *database* yang sangat kuat dan cukup stabil untuk digunakan sebagai media penyimpanan data. Sebagai sebuah *database server* yang mampu untuk memanajemen *database* dengan baik, *MySQL* terhitung merupakan *database* yang paling digemari dan paling banyak digunakan dibanding *database* lainya [6].

4. PEMBAHASAN

4.1 Analisis Perancangan Sistem

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahapan untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek OOA biasanya menggunakan kartu CRC (*Component, Responsibility, Collaborator*) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (*Unified Modeling Language*) pada bagian *diagram use case, diagram* kelas, dan diagram objek [5].

Pemodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML. Kendala dan permasalahan pembangunan sistem berorientasi objek biasanya dapat dikenali dalam tahap ini. OOA dan OOD dalam proses yang berulang-ulang seringnya memiliki batasan yang samar, sehingga terdapat dua tahapan yang sering juga disebut OOAD (*Object Oriented Analysis and Design*) atau dalam bahasa Indonesia berarti Analisis dan desain berorientasi objek.

4.1.1 Kebutuhan Fungsional (Functional Requirements)

Kebutuhan fungsional adalah jenis kebutuhan yang berisi proses-proses apa saja yang nantinya akan dilakukan oleh sistem. Sistem harus sesuai, baik dari segi perangkat lunak maupun perangkat keras untuk memudahkan user:

1) Mengelola otoritas *login* atau *passcode*, yaitu data *passcode* yang dimasukan oleh *seller/admin* dengan benar, dan selanjutnya memilih menu untuk malakukan aktifitas pekerjaan,
2) Mengelola *input, edit dan delete* untuk *customer dan supplier*.
3) Mengelola barang yang masuk ke toko melalui pembelian dari *supplier*, mengelola barang yang keluar dari toko karena adanya transaksi penjualan barang kepada *customer*.
4) Mengelola *system* untuk menyediakan *report*, seperti *report* produk masuk, *report* produk keluar dan *report* stok produk untuk dihasilkan tiap bulannya.

5) Menyediakan fungsi *logout*.

4.1.2 Kebutuhan Non-Fungsional (Non-Functional Requirements)

Kebutuhan *Non Fungsiona*l adalah jenis kebutuhan yang berupa *property* perilaku yang dimiliki oleh sistem, yang meliputi teknologi, operasional, kinerja pegawai, serta keamanan dalam perusahaan. Bisa berupa:

1) Sistem yang dibuat lebih bersahabat *(user friendly)* untuk tampilan maupun kemudahan pengolahan data.
2) Sistem yang digunakan harus kompatibel, baik dari perangkat lunak *(software)* maupun perangkat keras *(hardware)* agar memudahkan *admin* dalam pengolahan data.

4.2 Unified Modeling Language

Unified Modeling Language (UML) adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. *Artifact* dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya [7].

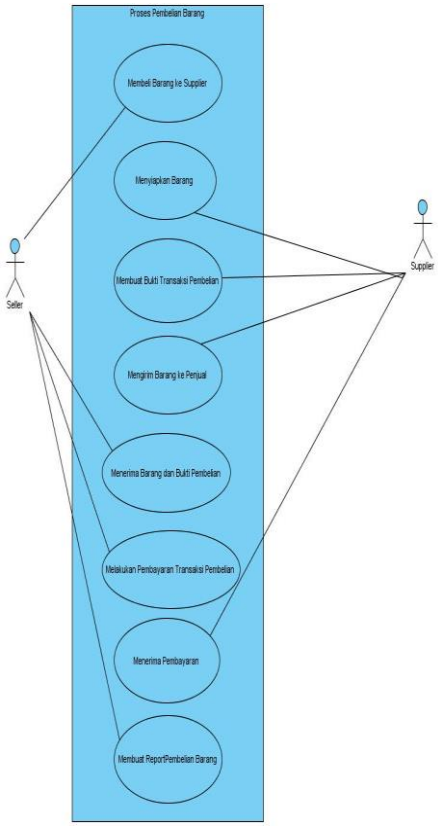
UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

UML singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standar [8]. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti, bagaimana elemen pada model-model yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk:

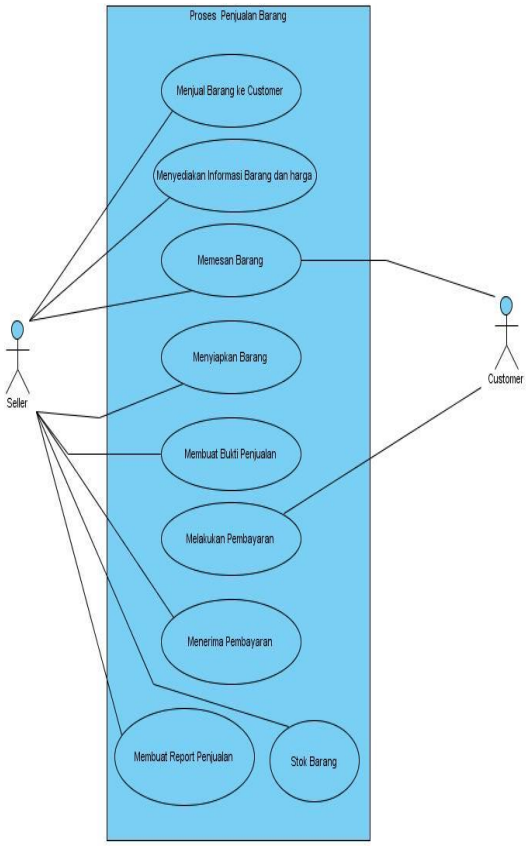
1) Merancang perangkat lunak.
2) Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3) Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4) Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Jenis-jenis diagram yang dimiliki oleh UML adalah sebagai berikut:

1) Diagram kelas (*class diagram*) Diagram kelas adalah inti dari proses pemodelan objek. Baik *forward engineering* (proses perubahan model menjadi kode program) atau *reverse engineering* (proses perubahan kode program menjadi model) memanfaatkan diagram ini.
2) Diagram paket (*package diagram*)
3) Diagram *use-case Diagram* ini digunakan untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna. *Use case* menggambarkan *external view* dari sistem yang akan dibuat modelnya.
4) Diagram interaksi dan *squence*
5) Diagram komunikasi *(communication diagram*)
6) Diagram *statechart* (*statechart diagram*) *Diagram statechart* dalam UML kadang disebut dengan istilah diagram *state machine*. Diagram ini menggambarkan perilaku sistem perangkat lunak yang kita buat dan perilaku kelas, subsistem dan seluruh aplikasi. Selain itu diagram state *machine* bermanfaat juga untuk menyediakan cara yang baik dalam memodelkan komunikasi yang terjadi dengan entitas luar via protokol atau sistem dasarnya. UML memiliki dua *state machine* antara lain: (a) *Behavioral State Machine*, mesin ini digunakan untuk menunjukkan perilaku elemen yang dimodelkan, misalnya suatu objek, (b) *Protocol State Machine*, mesin ini digunakan untuk menunjukkan perilaku protokol.
7) Diagram aktivitas (*activity diagram*) Diagram aktifitas lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem itu dirakit. Diagram ini tidak hanya memodelkan *software* melainkan memodelkan model bisnis juga. Diagram aktivitas menunjukkan aktivitas sistem dalam bentuk kumpulan aksi-aksi.
8) Diagram komponen (*component diagram*)
9) Diagram deployment (*deployment diagram*)



Gambar 5. ***Use Case Diagram*** **Proses Pembelian Barang (*Product In*)**

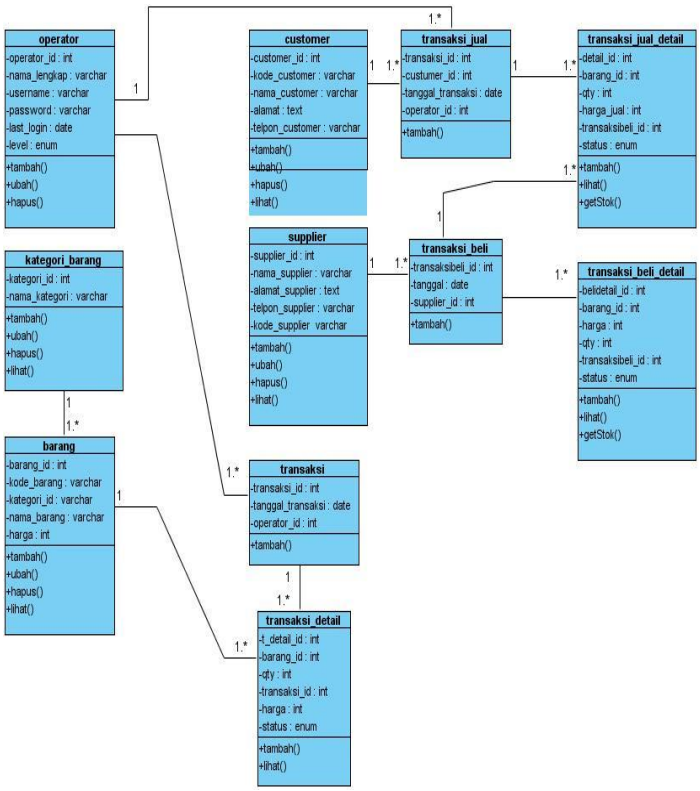


Gambar 6. ***Use Case Diagram*** **Proses Penjualan Barang (*Product Out*)**

4.3 Perancangan Sistem

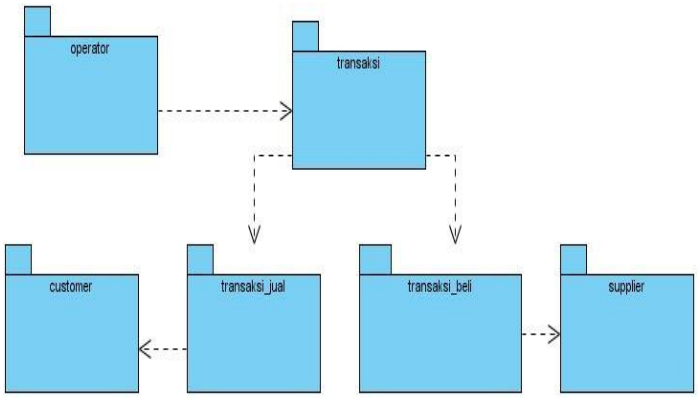
4.3.1 Class Diagram

Class diagram merupakan diagram yang menggambarkan jenis – jenis objek dalam sistem dengan berbagai macam relasi yang dimiliki. *Class diagram* menjelaskan hubungan antar *class* dalam sebuah *system* yang sedang dibuat dan bagaimana caranya agar *class – class* tersebut saling berkolaborasi untuk mencapai sebuah tujuan. *Class diagram* memiliki 3 area pokok (utama) yaitu: nama, atribut, dan operasi. Adapun *class diagram* Toko *Spare Part* AJM Motor dapat dilihat pada gambar dibawah ini:



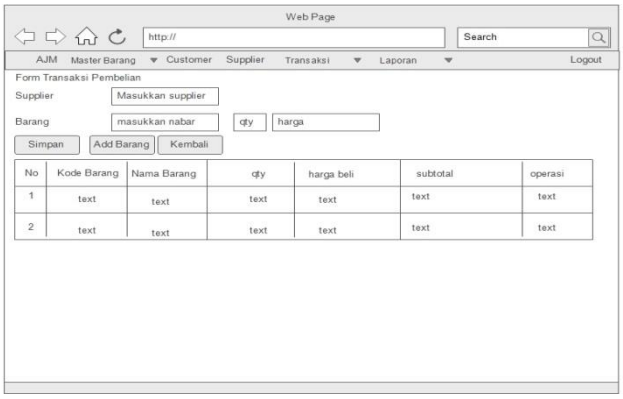
Gambar 7. *Class Diagram* Sistem

4.3.2 Package Database



Gambar 8. *Package Database*

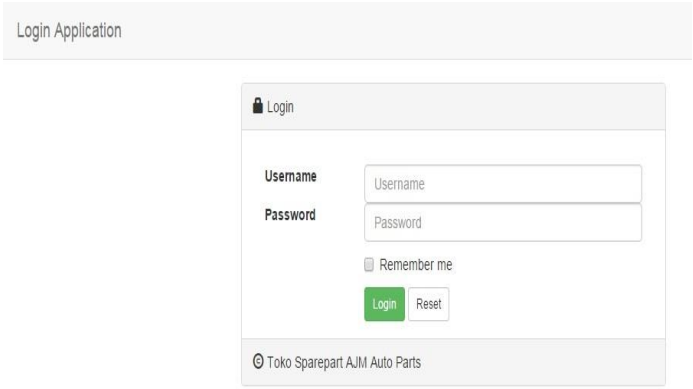
4.3.3 Rancangan User Interface



Gambar 9. Rancangan Tampilan *Form* Transaksi Pembelian

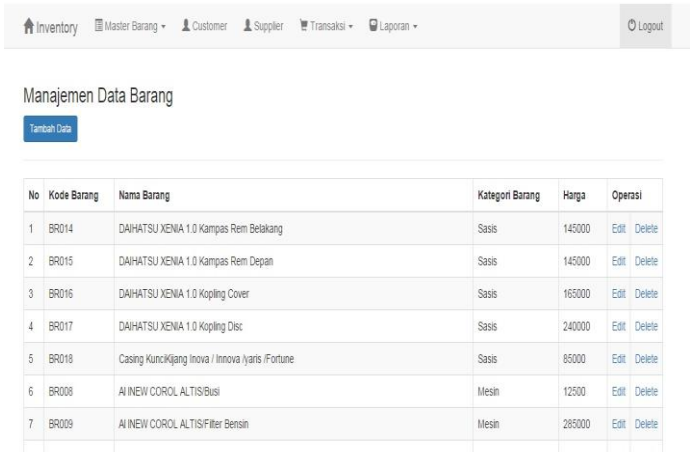
5. HASIL PENELITIAN

a) Halaman *Login*
Halaman ini dibuat merupakan tampilan pertama yang muncul pada saat program dijalankan, dimana pada *form* ini para pengguna atau operator akan diminta untuk memasukkan *User ID* beserta *Password.*



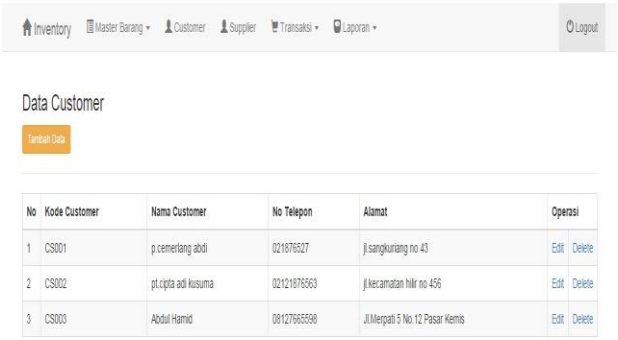
Gambar 10. Halaman *Login*

b) Menu *Master* Barang -> Data Barang



Gambar 11. Halaman *Edit, Add and Delete* Data Barang

c) Menu *Customer*

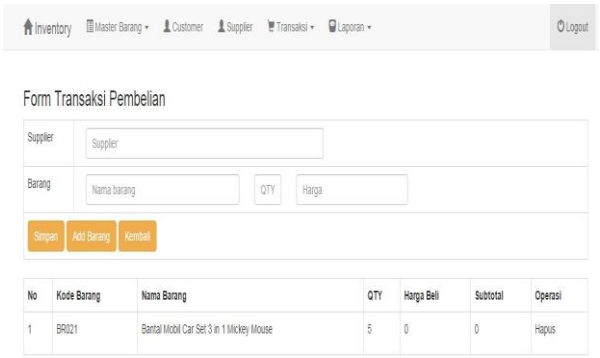


Gambar 12. Halaman *Add, Edit and Delete* Data *Customer*

d) Menu *Supplier*

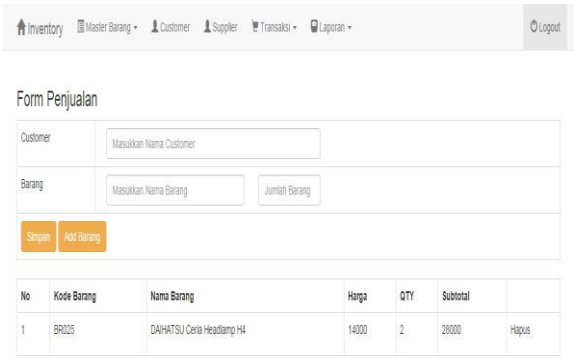
Gambar 13. Halaman *Add, Edit and Delete* Data *Supplier*

e) Menu transaksi Pembelian



Gambar 14. Halaman Transaksi Pembelian Barang Ke *Supplier*

f) Menu Transaksi Penjualan



Gambar 15. Halaman Transaksi Penjualan Ke *Customer*

6. KESIMPULAN

Berdasarkan dari penelitian atas sistem yang dilakukan oleh penulis pada toko *spare part* AJM motor, maka penulis menyimpulkan beberapa kesimpulan diantaranya adalah:

1) Bahwa dengan dirancangnya sistem ini akan mempermudah pemilik toko dalam mengelola data pelanggan, *supplier* dan barang yang di beli dan di jual serta pembuatan laporan yang diperlukan untuk perhitungan penjualan dan dapat memberikan informasi yang berguna bagi pemilik toko secara *up to date*.
2) Dengan adanya pembuatan sistem informasi penjualan pada AJM Motor, dari sistem yang masih konvensional menjadi sistem yang terkomputerisasi dapat digunakan dan dimengerti sesuai yang diharapkan terhadap pemilik toko AJM Motor.
3) Dengan adanya sistem informasi penjualan pada toko AJM Motor yang terkomputerisasi semua data dokumen penjualan yang sebelumnya masih konvensional sekarang menjadi tersimpan dengan baik dalam perangkat komputer.
4) Dalam membuat *website system* penjualan pada Toko AJM Motor ini lebih efisien dalam biaya, *efektif* dalam waktu yang tidak terlalu lama dan juga memudahkan peneliti karena menggunakan *framework* yang baik yaitu *CodeIgniter* dengan berbasis arsitektur *MVC* (*Model- View-Controler).*
5) Diharapkan kedepannya mampu mengembangkan atau melanjutkan dalam pembuatan sistem, dikarenaka terdapat menu yang belum diselesaikan dengan rapih dan belum sesuai dengan keinginan *user*.

DAFTAR PUSTAKA

[1] Heizer, J., and Render, B. (2005). *Operations Management*. *7th Edition*. New Jersey: Prentice-Hall.
[2] Pressman, R. (2002). *Rekayasa Perangkat Lunak Pendekatan Praktisi*. Yogyakarta: Andi.
[3] Andika, R. 2011. "Penerapan Ci (Codeigniter) Dalam Pengembangan System Informasi Manajemen Surat Dan Pengarsipan (Studi Kasus: Pt. Semen Padang)*"*. Jakarta: UIN Syarif Hidayatullah.
[4] Ardhana, Y K. (2013). *Pemrograman Php: Codeigniter Black Box*. Jakarta: Jasakom.
[5] Shalahuddin, M., dan Rosa, A S. (2011). *Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur Dan Berorientasi Objek)*. Bandung: Modula.
[6] Nugroho, B. (2004). *Aplikasi Pemprograman Web Dinamis Dengan PHP Dan Mysql*. Yogyakarta: Gava Media.
[7] Koespradono, S., dan Yuliana, R K. 2013. "Sistem Informasi Pengolahan Data Pertumbuhan Ekonomi Dan Ketimpangan Di Kabupaten Klaten (Tahun 2003-2012) Menggunakan Framework Codeigniter". ISSN:2338-6304 Institut Sains & Teknologi Akprind Yogyakarta: Jurnal Script Vol. 1 No. 1 Desember 2013.
[8] Widodo, P P., dan Herlawati. (2011). *Menggunakan Uml*. Bandung: Informatika.
[9] Octafian, D T. 2015. "Web Multi E-Commerce Berbasis Framework Codeigniter. Jurnal Teknologi Dan Informatika (Teknomatika) Multi E-Commerce Berbasis Framework Codeigniter". Vol. 5 No. 1 Jan 2015.
[10] Indah, I N. 2013. "Pembuatan Sistem Informasi Penjualan Pada Toko Sehat Jaya Elektronik". Pacitan: Indonesian Jurnal On Computer Science - Speed (Ijcss) 16 Fti Unsa Vol 10 No 2 – Mei 2013 - Ijcss.Unsa.Ac.Id. ISSN: 1979-9330 (Print) - 2088-0154 (Online) - 2088-0162 (Cdrom).
[11] Rosmala D., Ichwan M., dan Gandalisha M I. 2011. "Komparasi Framework Mvc (Codeigniter, Dan Cakephp) Pada Aplikasi Berbasis Web (Studikasus: Sistem Informasi Perwalian di Jurusan Informatika Institut Teknologi Nasional)". Institut Teknologi Nasional Bandung: Jurnal Informatika. No.2, Vol. 2, Mei – Agustus 2011.