**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

# **IDENTITAS PENGUSUL**

Nama : **Tommy Adhyasa Suramardhana**

NRP : **5107 100 144**

Dosen Wali : **Ir. Riyanarto Sarno, SE, MSc, PhD**

1. **JUDUL TUGAS AKHIR**

***Rancang Bangun Aplikasi Account Payable, Account Receivable dan Fix Asset Berorientasi SOAD pada platform JAVA*.**

# **URAIAN SINGKAT**

Pada tugas akhir ini penulis akan membuat sebuah perangkat lunak sistem manajemen akutansi yang khusus menangani domain fungsi *Account Payable*, *Account Receivable* dan *Fixed Asset* berorientasi SOAD (*Service Oriented Analysis and Design*) yang dirancang sesuai dengan model pengembangan SaaS (*Software as a Service*). Dengan digunakannya SOAD sebagai metode perancangan aplikasi berbasiskan SOA (*Service Oriented Architecture*), yang memiliki keterdekatan konsep, yakni konsep IT dengan konsep bisnis diharapkan kerjasama erat antara praktisi TI (*programmer*, *system analyst*, *software architect*) dan praktisi bisnis (sales, marketer, bagian produksi, akuntan, para manajer dan eksekutif) akan menjadi jauh lebih baik. Kerjasama ini sangat dibutuhkan demi terwujudnya realisasi investasi TI untuk kemajuan dunia bisnis khususnya dan peradaban manusia pada umumnya. Selain itu, diharapkan aplikasi ini akan memecahkan masalah perusahaan besar dimana kebutuhan pasar dan pola persaingan yang berubah dengan cepat, kecenderungan untuk merger atau kerjasama antar bisnis yang juga semakin meningkat menjadi kebutuhan yang sangat penting.

Sesuai tugas akhir pendahulu kami, domain bisnis akutansi yang ditentukan telah dirancang dengan menggunakan kerangka kerja COBIT AI1 (domain Mengidentifikasi Solusi Otomatis), AI2 (domain Memperoleh dan Memelihara Perangkat Lunak Aplikasi) dan AI6 (domain Mengelola Perubahan), kemudian servis bisnisnya dirancang dengan pemodelan arsitektur SOAD. Pada tugas akhir ini, rancangan domain tersebut akan kembali dirancang dan disesuaikan dengan model pengembangan SaaS tapi masih menggunakan SOAD sebagai pemodelan arsitekturnya.

# **LATAR BELAKANG**

Kondisi dunia bisnis saat ini telah berkembang menjadi semakin kompleks, semakin kompetitif, bergerak dengan cepat serta semakin sulit untuk diprediksi. Agar dapat bersaing dan sukses, perusahaan perlu memadukan bisnis dan sumber daya IT yang dimiliki agar dapat secara fleksibel mengakomodasi adanya perubahan untuk kemudian dilakukan adaptasi terhadap perubahan tersebut secara cepat dan tepat.

Berbagai tantangan bisnis yang ada menuntut perusahaan untuk memiliki kemampuan respons yang cepat dan fleksibel terhadap setiap peluang, ancaman dari luar, tuntutan pelanggan, langkah-langkah kompetitor, maupun perubahan regulasi. Untuk mewujudkan hal tersebut diperlukan adanya mekanisme untuk dapat memberikan informasi yang tepat pada saat yang tepat dan diberikan kepada orang yang tepat pula sehingga pada akhirnya dapat membantu pengambilan keputusan bisnis yang lebih baik dengan lebih cepat. Untuk dapat menghasilkan informasi tersebut diperlukan dukungan infrastruktur perusahaan yang terintegrasi, yang mampu memanfaatkan sumber daya IT yang telah ada yang juga dapat dengan mudah ditambahkan fitur dan fungsionalitas baru. Infrastruktur tersebut haruslah fleksibel dan tangkas agar dapat menyesuaikan diri terhadap perubahan yang terjadi baik pada sisi bisnis maupun IT.

Perkembangan bisnis terus berubah-ubah mulai dari yang berinteraksi secara vertikal (tahun 1980) lalu berubah menjadi interaksi secara horizontal (tahun 1980 dan 1990-an) yang pada akhirnya saat ini semua bagian dari dunia bisnis berinteraksi dengan pola yang membentuk sebuah ekosistem. Karenanya lingkungan Teknologi Informasi (TI) harus bisa selalu menyesuaikan perubahan yang terjadi pada dunia bisnis ini, salah satu caranya adalah dengan membuat aplikasi dan sistem yang dapat berkomunikasi satu dengan lainnya tapi tetap menjaga integritasnya.

Perangkat lunak saat ini sangat dibutuhkan di dunia bisnis saat ini, dimana perangkat lunak dapan membantu kita mengecek pengiriman barang melewati berbagai negara, mengatur penyimpanan dalam jumlah yang besar, melatih pegawai, bahkan menolong kita untuk membangun relasi yang baik dengan pelanggan. Selama berapa dekade, perusahaan menjalankan aplikasinya di dalam infrastruktur internal atau jaringan komputer. Di beberapa tahun terakhir, sistem pembelian lisensi aplikasi tradisional sudah mulai ketinggalan zaman, karena banyak vendor dan pelanggan yang telah berpindah telah bermigrasi ke sebuah perangkat lunak sebagai model bisnis jasa.

*Software as a Service*, atau “SaaS”, adalah sebuah perangkat lunak pengiriman model dimana vendor perusahaan mengembangkan aplikasi perangkat lunak berbasis web, dan kemudian berperan sebagai host dan mengoperasikan aplikasi tersebut melalui Internet untuk digunakan oleh pelanggan. Pelanggan tidak perlu membeli lisensi perangkat lunak atau peralatan infrastruktur tambahan, dan biasanya hanya membayar biaya bulanan (juga disebut sebagai pembayaran anuitas) untuk menggunakan perangkat lunak. Menurut sebuah lembaga riset terkemuka, pasar SaaS mencapai $ 6.3B pada tahun 2006; masih sebagian kecil dari pendapatan industry software berlisensi yang mencapai hingga $ 300B. Namun, walaupun begitu pertumbuhan SaaS sejak tahun 2000 mencapai 26 CAGR%, sedangkan pertumbuhan perangkat lunak berlisensi tetap relatif datar. Hal ini disebabkan permintaan SaaS yang terus didorong oleh kenyataan kebutuhan bisnis sebenarnya - yaitu kemampuan untuk menurunkan biaya yang berkaitan dengan IT, penurunan waktu pembuatan, dan sangat mendorong untuk terus berinovasi.

Selain menggunakan SaaS sebagai salah satu solusi dari banyak masalah yang telah disebutkan diatas, dalam membangun sebuah sistem kita bisa menggunakan SOAD untuk merancang aplikasi yang dibutuhkan dan melakukan pembangunan aplikasi dengan menggunakan SOA. SOAD (*Service Oriented Analysis and Design*) adalah metode atau pendekatan pengembangan perangkat lunak yang dirancang khusus untuk *Service Oriented Architecture (SOA).* Sedangkan SOA adalah konsep pengembangan perangkat lunak yang melakukan partisi sistemnya menjadi beberapa servis yang berdiri secara independen. SOA digunakan dalam perancangan servis karena dapat menyatukan serbagai sistem yang memiliki platform berbeda dan tahan terhadap perubahan. Dengan mengorganisasikan TI perusahaan lebih pada service dibanding pada aplikasi, SOA memberikan beberapa keuntungan kunci berikut:

* Meningkatkan produktifitas, *agility* (kelincahan), dan kecepatan baik bagi Bisnis maupun TI.
* Menjadikan TI bisa memberikan layanan yang lebih cepat dan lebih sejalan dengan bisnis.
* Menjadikan Bisnis lebih cepat merespon dan memberikan pengalaman pemakai yang lebih optimal.

Pada tugas akhir penulis akan membangun sebuah perangkat lunak sistem manajemen akutansi yang khusus menangani domain fungsi *Account Payable*, *Account Receivable* dan *Fixed Asset* dengan menggunakan metode SOAD yang dirancang sesuai dengan model pengembangan SaaS. *Account Payable* / Hutang Dagang adalah jumlah uang yang harus dibayarkan kepada pemasok, karena perusahaan melakukan pembelian barang atau jasa servis yang telah dilakukan dibayar secara kredit dan memiliki surat perjanjian pembayaran. Sedangkan *Account Receivable* adalah kumpulan dari semua account yang berhubungan dengan penjualan barang atau pemberian servis secara kredit kepada pelanggan. Selain itu pada sebuah perusahaan ada juga bisnis proses *Fix Asset*. *Fix Asset* / Aset Tetap merupakan aktiva tetap yang terdiri dari kelompok bangunan dan bukan bangunan.

Dengan digunakannya SOAD sebagai metode perancangan aplikasi berbasiskan SOA, yang memiliki keterdekatan konsep, yakni konsep IT dengan konsep bisnis diharapkan kerjasama erat antara praktisi TI (*programmer*, *system analyst*, *software architect*) dan praktisi Bisnis (sales, marketer, bagian produksi, akuntan, para manajer dan eksekutif) akan menjadi jauh lebih baik. Kerjasama ini sangat dibutuhkan demi terwujudnya realisasi investasi TI untuk kemajuan dunia bisnis khususnya dan peradaban manusia pada umumnya. Selain itu, diharapkan aplikasi ini akan memecahkan masalah perusahaan besar dimana kebutuhan pasar dan pola persaingan yang berubah dengan cepat, kecenderungan untuk merger atau kerjasama antar bisnis yang juga semakin meningkat menjadi kebutuhan yang sangat penting.

# **RUMUSAN MASALAH**

Permasalahan yang diangkat dalam menyelesaikan tugas akhir ini adalah:

* Bagaimana membuat aplikasi Account Payable, Account Receivable dan Fixed Asset untuk suatu perusahaan untuk menggantikan proses pengolahan data manual dan sebagai penyedia informasi terkait Account Payable, Account Receivable dan Fixed Asset sebagai usaha peningkatan kinerja perusahaan.
* Bagaimana mengimplementasikan pemodelan arsitektur *Service Oriented Analysis and Design* (SOAD) dalam merancang servis dari sistem manajemen akuntansi untuk Account Payable, Account Receivable dan Fixed Asset.
* Bagaimana mengembangkan suatu aplikasi enterprise yang menggunakan konsep SaaS sesuai dengan SOAD.

# **BATASAN MASALAH**

Asumsi dan ruang lingkup permasalahan yang dikerjakan dalam Tugas Akhir ini adalah:

* Hasil dari Tugas Akhir ini adalah menghasilkan sebuah aplikasi dari sistem manajemen akuntansi Account Payable, Account Receivable dan Fix Asset yang servisnya dibangun dengan menggunakan pemodelan arsitektur SOAD dengan SaaS sebagai model deploymentnya.
* ERP yang dibangun merupakan ERP dari perusahaan yang bergerak di bidang manufacturing.
* Aplikasi berupa aplikasi web dengan platform Java dengan menggunakan arsitektur MVC dengan kerangka kerja Java Server Faces dan Hibernate sebagai library Object Relational Mapping.

# **TUJUAN PEMBUATAN TUGAS AKHIR**

Tujuan pembuatan tugas akhir ini adalah memanfaatkan pemodelan arsitektur SOAD untuk mendesain sistem manajemen aset yang reusable, scalable dan maintainable untuk domain fungsi Account Payable, Account Receivable dan Fix Asset. Nantinya hasil akhir dari tugas akhir ini adalah sebuah aplikasi dari proses Account Payable, Account Receivable dan Fix Asset yang menggunakan SaaS sebagai model deploymentnya.

# **MANFAAT PEMBUATAN TUGAS AKHIR**

Manfaat yang dapat diambil tugas akhir ini adalah dapat membangun sebuah aplikasi yang memiliki sistem manajemen akuntansi yang dapat terbagi-bagi menjadi beberapa servis, sehingga nantinya dapat dengan mudah di-*reuse* sesuai dengan manajemen akuntansi lain yang dibutuhkan.

# **RINGKASAN TUGAS AKHIR**

Pada tugas akhir ini penulis akan membuat sebuah perangkat lunak sistem manajemen akutansi yang khusus menangani domain fungsi *Account Payable*, *Account Receivable* dan *Fixed Asset* dengan menggunakan metode SOAD (*Service Oriented Analysis and Design*) yang dirancang sesuai dengan model pengembangan SaaS (*Software as a Service*). Dengan digunakannya SOAD sebagai metode perancangan aplikasi berbasiskan SOA (*Service Oriented Architecture*), yang memiliki keterdekatan konsep, yakni konsep IT dengan konsep bisnis diharapkan kerjasama erat antara praktisi TI (*programmer*, *system analyst*, *software architect*) dan praktisi Bisnis (*sales*, *marketer*, bagian produksi, akuntan, para manajer dan eksekutif) akan menjadi jauh lebih baik.

Sesuai tugas akhir pendahulu kami, domain bisnis akutansi yang ditentukan telah dirancang dengan menggunakan kerangka kerja COBIT AI1 (domain Mengidentifikasi Solusi Otomatis), AI2 (domain Memperoleh dan Memelihara Perangkat Lunak Aplikasi) dan AI6 (domain Mengelola Perubahan), kemudian servis bisnisnya dirancang dengan pemodelan arsitektur SOAD. Pada tugas akhir ini, rancangan domain tersebut akan kembali dirancang dan disesuaikan dengan model pengembangan SaaS tapi masih menggunakan SOAD sebagai pemodelan arsitekturnya.

Rancangan aplikasi yang akan kami bangun dengan menggunakan konsep SaaS nantinya dapat digunakan oleh banyak perusahaan yang memiliki banyak branch dengan satu Head Office. Walaupun tugas akhir ini merupakan pengembangan dari tugas akhir sebelumnya, tetapi terdapat perbedaan dari segi arsitektur pembangunannya, karena digunakaannya model pengembangan SaaS. Bila pada tugas akhir sebelumnya setiap domain fungsi akan memiliki database sendiri di tiap domain fungsinya, maka pada tugas akhir ini database akan digunakan oleh satu perusahaan/satu HO. Selain itu, bila tiap domain fungsi pada tugas akhir sebelumnya saling berkomunikasi dengan menggunakan web service, maka pada tugas akhir ini web service hanya akan digunakan untuk berkomunikasi dengan HO. Komunikasi antara domain fungsi tidak lagi akan dihubungkan dengan web service karena seluruh domain fungsi akan mengakses database yang sama. Berikut ini gambar yang menunjukkan gambaran arsitektur ERP yang akan penulis bangun :



**Gambar 1** Desain ERP dengan *model deployment* SaaS

*Account Payable* adalah hutang dagang yang dimiliki oleh suatu perusahaan karena adanya pembelian barang atau jasa yang dilakukan secara kredit dan memiliki surat perjanjian pembayaran. Dalam hutang dagang ini terdapat beberapa sub modul didalamnya, yaitu :

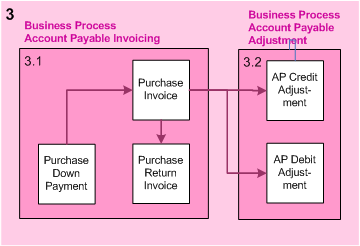
1. *Purchase Down Payment* : adalah proses untuk mencatat dokumen faktur uang muka pembelian yang diterbitkan oleh supplier. Proses ini dilakukan jika mendapatkan faktur dp atas pembelian yang dilakukan perusahaan. Pencatatan faktur pembelian ini nantinya akan digunakan untuk menerbitkan *request payment for purchase dp invoice* yang digunakan untuk melakukan pembayaran *Cash and Bank*.

2. *Purchase Invoice* : adalah proses untuk mencatat dokumen faktur pembelian / *invoice* yang diterbitkan oleh *supplier*. Proses ini dilakukan jika mendapatkan faktur pembelian dari *supplier* atas barang atau jasa yang dipesan untuk dibeli. Pencatatan faktur pembelian ini nantinya akan digunakan untuk menerbitkan *request payment for purchase invoice* yang digunakan untuk melakukan pembayaran pada domain *Cash and Bank*.

3. *Purchase Return Invoice* : adalah proses untuk mencatat dan membuat dokumen faktur pengembalian pembelian yang diterbitkan oleh supplier. Proses ini dilakukan jika terjadi pengembalian barang atau jasa pada bagian inventori, ini yang menyebabkan pengurangan hutang perusahaan kepada *supplier* tersebut, tujuannya agar dapat dikompensasikan dengan faktur pembelian yang sebelumnya telah dicatat. Pencatatan faktur pengembalian pembelian ini nantinya akan digunakan untuk menerbitkan *purchase debit notes*.

4. *Account Payable Credit Adjustment* : adalah proses untuk penyesuaian hutang dengan menambah hutang dagang perusahaan terhadap *supplier* tertentu selain dari *purchase invoice*. Transaksi ini dilakukan jika perusahaan mendapatkan dokumen *request for purchase rebate* dari *supplier* yang menyebabkan hutang perusahaan kepada *supplier* tersebut akan bertambah.

5. *Account Payable Debit Adjustment* : adalah proses untuk penyesuaian hutang dengan mengurangi hutang dagang perusahaan terhadap *supplier* tertentu selain dari pembayaran dan *return*. Proses ini dilakukan jika perusahaan mendapatkan dokumen *request for purchase rebate* dari *supplier* yang menyebabkan hutang perusahaan kepada *supplier* tersebut akan berkurang.



**Gambar 2** Diagram *business process* dari domain fungsi *account payable*

Aplikasi *Account Payable* ini akan memiliki beberapa fitur utama, antara lain :

1. *Purchase DP Invoice* : adalah fitur untuk menangani pencatatan uang muka hutang kepada *supplier*, dan akan menerbitkan *request payment for purchase dp invoice*.
2. *Purchase Invoice* : adalah fitur untuk menangani pencatatan hutang kepada *supplier* baik dengan adanya *purchase order* maupun tidak dari bagian inventori, dan akan menerbitkan *request payment for purchase dp invoice*.
3. *Purchase Return Invoice* : adalah fitur yang akan menangani pencatatan pengembalian pembelian kepada pihak *supplier*, sehingga nantinya catatan hutang perusahaan kepada *supplier* tersebut akan berkurang sesuai dengan nomor faktur yang dicatat pada transaksi *purchase invoice*  sebelumnya, dan akan menerbitkan *purchase debit notes*.
4. *Account Payable Adjustment* : adalah fitur yang akan menangani pencatatan penyesuaian hutang perusahaan baik itu yang akan mengurangi ataupun akan menambah hutang pada *supplier*.
5. *Manajemen Data Transaction Code* : adalah fitur yang akan menangani pencatatan semua data kode transaksi yang dibutuhkan oleh aplikasi yang berisi kode, akun credit, akun debit, keterangan dan tipe transakasi. Data ini akan dipergunakan pada setiap transaksi yang ada pada aplikasi ini.

Sedangkan domain fungsi *Account Receivable* adalah piutang dagang yang dimiliki oleh suatu perusahaan karena adanya penjualan barang atau jasa yang dilakukan secara kredit kepada pelanggannya. Dalam piutang dagang ini ada beberapa sub modul didalamnya, yaitu :

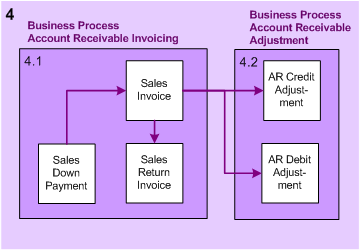
1. *Sales Down Payment* : adalah proses untuk mencatat dan menerbitkan *sales dp invoice* yang akan diterbitkan untuk pelanggan. Proses ini terjadi jika ada *sales order* pada bagian inventori atas barang atau jasa yang perusahaan jual yang disertai dp untuk bisa membelinya.

2. *Sales Invoice* / Faktur Penjualan: adalah proses untuk mencatat dan membuat faktur penjualan / *sales invoice* yang diterbitkan untuk pelanggan sebagai bukti pengeluaran piutang dan penjualan. Proses ini terjadi saat terjadi *sales order* dari pelanggan pada bagian inventori atas barang atau jasa yang dijual oleh perusahaan.

3.  *Sales Return Invoice* : adalah proses untuk mencatat *sales return notes* untuk pelanggan. Proses ini dilakukan jika ada pengembalian barang atau jasa dari pelanggan yang telah memebeli kepada perusahaan di bagian inventori. Pencatatan ini akan mengurangi piutang perusahaan terhadap *supplier* tersebut, tujuannya agar nanti bisa dikompensasikan dengan *sales invoice* yang sudah dicatat sebelumnya.

4. *Account Receivable Credit Adjustment* : adalah proses untuk penyesuaian piutang dengan mengurangi piutang terhadap pembeli tertentu selain dari pembayaran dan *return.* Proses ini dilakukan jika perusahaan mendapatkan catatan penyesuaian piutang dari pelanggan yang menyebabkan piutang perusahaan terhadap pelanggan tersebut akan berkurang dan digunakan untuk menerbitkan *request for sales rebate*.

5. *Account Receivable Debit Adjustment* : adalah proses penyesuaian piutang dengan menambah piutang terhadap pelanggan tertentu selain dari *sales invoice*. Proses ini dilakukan jika perusahaan mendapatkan catatan penyesuaian piutang dari pelanggan yang menyebabkan piutang perusahaan kepada pelanggan tersebut akan bertambah dan digunakan untuk menerbitkan *request for sales rebate*.



**Gambar 3** Diagram *business process* dari domain fungsi *account receivable*

Aplikasi *Account Receivable* ini akan memiliki beberapa fitur utama, antara lain :

1. *Sales DP Invoice* : adalah fitur untuk menangani pencatatan uang muka piutang kepada *customer*, dan akan menerbitkan *sales dp invoice*.
2. *Sales Invoice* : adalah fitur untuk menangani pencatatan piutang kepada *customer* baik dengan adanya *sales order* maupun tidak dari bagian inventori, dan akan menerbitkan *sales invoice*.
3. *Sales Return Invoice* : adalah fitur yang akan menangani pencatatan pengembalian penjualan kepada pihak *customer*, sehingga nantinya catatan piutang perusahaan kepada *customer* tersebut akan berkurang sesuai dengan nomor faktur yang dicatat pada transaksi *sales invoice*  sebelumnya, dan akan menerbitkan *sales credit notes*.
4. *Account Receivable Adjustment* : adalah fitur yang akan menangani pencatatan penyesuaian piutang perusahaan baik itu yang akan mengurangi ataupun akan menambah hutang pada *customer*.
5. *Manajemen Data Transaction Code* : adalah fitur yang akan menangani pencatatan semua data kode transaksi yang dibutuhkan oleh aplikasi yang berisi kode, akun credit, akun debit, keterangan dan tipe transakasi. Data ini akan dipergunakan pada setiap transaksi yang ada pada aplikasi ini.

*Fixed Asset* atau yang sering disebut aktiva tetap adalah harta yang menunjang kegiatan operasional perusahaan yang sifatnya permanen kepemilikannya. Menurut undang-undang perpajakan di Indonesia UU No 36 tahun 2008 Pasal 11 terdapat dua kelompok aktiva tetap baik yang merupakan bangunan maupun yang bukan.

Kelompok aktiva tetap yang bangunan terdiri dari :

1. Kelompok bangunan permanen dengan umur 20 tahun

2. Kelompok bangunan tidak permanen dengan umur 10 tahun

Kelompok aktiva tetap yang bukan bangunan terdiri dari :

1. Golongan 1 dengan umur sampai 4 tahun

2. Golongan 2 dengan umur antara 4 sampai 8 tahun

3. Golongan 3 dengan umur antara 8 sapai 16 tahun

4. Golongan 4 dengan umur 20 tahun

Dalam modul aktiva tetap ini ada beberapa sub modul didalamnya, yaitu :

1.  *Fixed Asset Additional* : adalah proses untuk pencatatan penambahan barang yang dianggap sebagai harta tetap yang dimiliki perusahaan. Proses ini terjadi jika terdapat penerimaan harta tetap di bagian inventori yang menyebabkan catatan harta tetap perusahaan akan bertambah. Proses didalamnya adalah melakukan pengelompokan tipe harta tetap tersebut, dan diasumsikan harta tetapnya masih berada di gudang.

2. *Fixed Asset Transfer* : adalah proses untuk mencatat pemindahan harta tetap dari suatu lokasi ke lokasi lainnya.

3.  *Fixed Asset Depreciation* : adalah proses untuk melakukan depresiasi untuk setiap harta tetap yang dimiliki perusahaan sesuai dengan tipe hartanya masing-masing di tiap bulannya sepanjang perkiraan umur aset tersebut masih ada. Metode depresiasi yang digunakan pada aplikasi ini adalah metode garis lurus (*Straight Line Method*) dan metode saldo menurun (*Double Decling Balance Method*).

4. *Fixed Asset Stock Take* : adalah proses untuk memastikan harta tetap yang tercatat pada sistem dengan keadaan yang sebenarnya itu sesuai atau tidak, jika tidak sesuai maka catatan yang ada pada sistem akan disesuaikan dengan keadaan sebenarnya.

5. *Fixed Asset Revaluation* : adalah proses untuk menilai kembali nilai perolehan dari harta tetap karena dianggap tidak lagi mencerminkan nilai aset tersebut yang sebenarnya. Nilai perolehan harta tetap yang baru pada umumnya lebih besar dari nilai buka harta tersebut pada saat proses ini dilakukan, jika nilai aset yang baru lebih rendah dari nilai aset sebelumnya maka harta tetap perusahaan nilainya akan dikurangi dan begitu pula sebaliknya.

6.  *Fixed Asset Maintenance* : adalah proses untuk mencatat proses pemeliharaan harta tetap perusahaan. Proses ini akan melakukan proses *update* pada kondisi harta tetap dan mencatat berapa biaya pemeliharaannya.

7.  *Fixed Asset Disposal* : adalah proses untuk melakukan penghapusan harta tetap yang dimiliki oleh perusahaan. Proses ini akan menyebabkan aset perusahaan akan berkurang dan tentunya proses depresiasi tidak akan dilakukan.

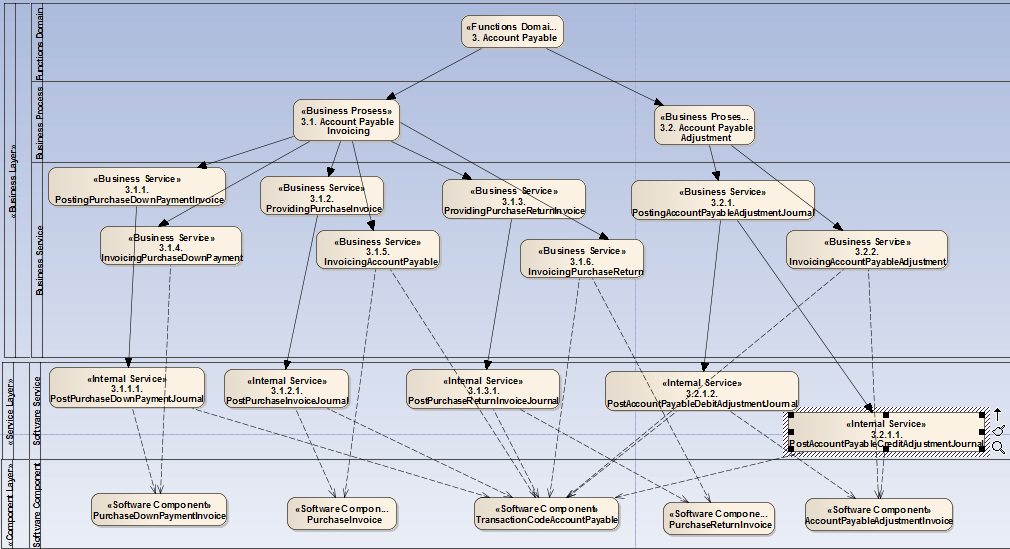


**Gambar 4** Diagram *business process* dari domain fixed asset

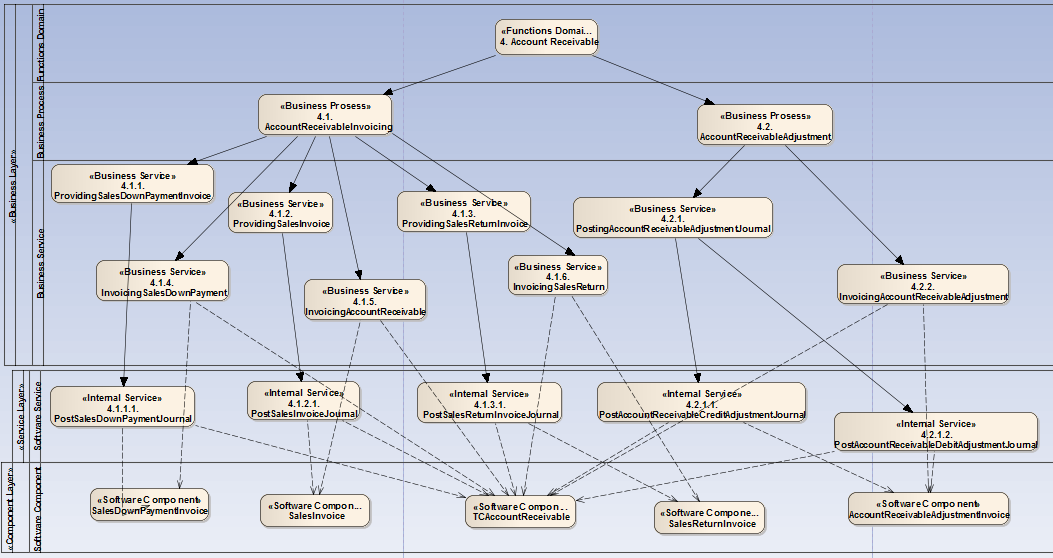
Aplikasi *Fixed Assets* ini akan memiliki beberapa fitur utama, antara lain :

1. *Fixed Asset Additional* : adalah fitur untuk menangani pencatatan penambahan harta tetap perusahaan ketika sebuah aset ditambahkan pada gudang perusahaan.
2. *Fixed Asset Transfer* : adalah fitur untuk menangani pencatatan pemindahan harta tetap perusahaandari suatu lokasi ke lokasi lainnya.
3. *Fixed Asset Depreciation* : adalah fitur untuk menangani pencatatan depresiasi untuk setiap harta tetap yang dimiliki perusahaan sesuai dengan tipe hartanya yang akan dilakukan tiap bulan.
4. *Fixed Asset Maintenance* : adalah fitur untuk menangani pencatatan proses pemeliharaan harta tetap perusahaan.
5. *Fixed Asset Stock Take* : adalah fitur untuk menangani pencatatan data pelakasanaan *stock take* terhadap harta tetap perusahaan.
6. *Fixed Asset Revaluation* : adalah fitur untuk menangani pencatatan data revaluasi terhadap aset perusahaan.
7. *Fixed Asset Disposal* : adalah fitur untuk menangani pencatat data penghapusan terhadap aset perusahaan.
8. Manajemen *Data Fiscal Fixed Asset Type* : adalah fitur yang digunakan untuk menangani pencatatn semua data jenis harta tetap fiskal yang terdapat pada perusahaan.
9. Manajemen *Data Fixed Asset Type* : adalah fitur yang digunakan untuk menangani pencatatan semua data jenis harta tetap yang dimiliki perusahaan, meliputi kode nama, kategori dan keterangan. Data ini akan digunakan untuk menggolongkan aset perusahaan pada saat transaksi *Fixed Asset Additional*.
10. Manajemen *Data Transaction Code* : adalah fitur yang akan menangani pencatatan semua data kode transaksi yang dibutuhkan oleh aplikasi yang berisi kode, akun credit, akun debit, keterangan dan tipe transakasi.

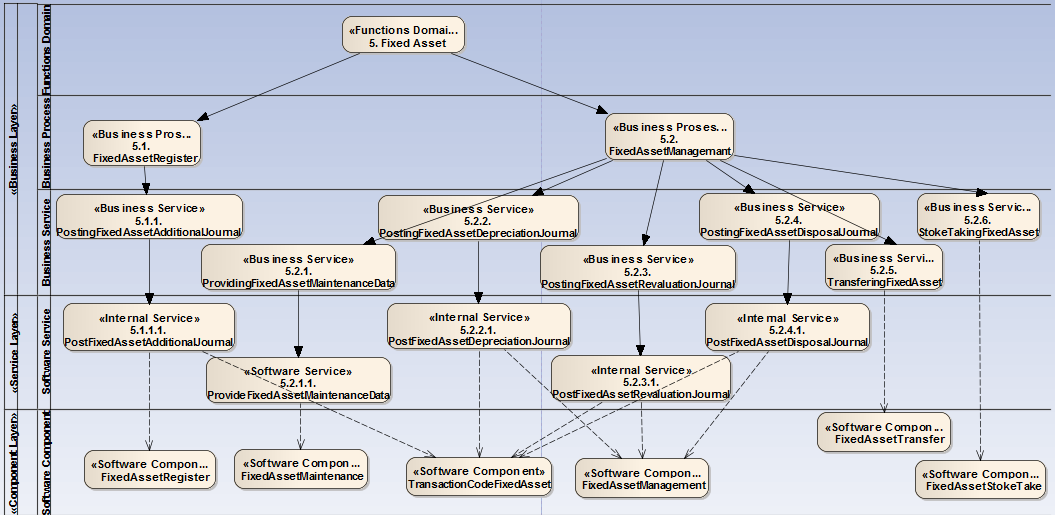
Berikut rancangan *bussiness service* dan *internal service* SOAD domain fungsi *Account Payable*, *Account Receivable* dan *Fixed Asset* :



**Gambar 5** Prototype *Diagram SOAD Domain Fungsi Account Payable*



**Gambar 6** Prototype *Diagram SOAD Domain Fungsi Account Receivable*



**Gambar 7** Prototype *Diagram SOAD Domain Fungsi Fixed Asset*

Seperti yang telah dijelaskan diatas, penulis akan menggunakan orientasi *Service Oriented Architecture* (SOA) dalam pembangunan aplikasi ini. Tujuannya agar setiap domain fungsi yang nantinya akan saling berinteraksi dalam sebuah sistem ERP. SOA menyediakan kemudahan dalam integrasi aplikasi yang dibagun oleh developer yang berbeda, hal ini disebabkan aplikasi-aplikasi tersebut akan terenkapsulasikan ke dalam komponen service layer yang reusable, sehingga masing-masing domain fungsi tersebut tetap akan dapat berinteraksi dengan *service-service* yang telah disediakan.

Pada aplikasi yang penulis bangun, aplikasi ini akan memiliki memiliki 5 buah layer antara lain :

1. *Web Service Layer*

Layer ini bertanggung jawab dalam hal implementasi web service yang disediakan.

1. *Presentation Layer*

Layer ini bertanggung jawab dalam hal user interface dengan pengguna.

1. *Application Service Layer*

Seluruh *control logic* yang ada di dalam perangkat lunak ini dibangun di dalam layer ini.

1. *Domain Model Layer*

Domain model layer berisi implementasi *class-class* yang digunakan dan data transfer object yang nantinya digunakan oleh web service dalam proses tranfer data.

1. *Data Access Layer*

Layer ini bertanggung jawab dalam hal koneksi dengan database yang digunakan.

Secara garis besar, gambaran hubungan antara layer-layer yang telah disebutkan di atas adalah sebagai berikut :



**Gambar 8** *Desain arsitektur aplikasi yang akan dibangun.*

# **TINJAUAN PUSTAKA**

Metodologi yang akan digunakan untuk menyelesaikan Tugas Akhir ini adalah :

1. ***SaaS***

SaaS merupakan perangkat lunak yang berbentuk layanan/service. SaaS adalah suatu model penyampaian aplikasi perangkat lunak oleh suatu vendor perangkat lunak yang mengembangkan aplikasi web yang diinangi dan dioperasikan (baik secara mandiri maupun melalui pihak ketiga) untuk digunakan oleh pelanggannya melalui Internet.

Dalam SaaS, seluruh business process dan data pelanggan ditempatkan di sebuah server data center milik services provider. Server ini melayani sejumlah klien dengan sistem one-to-many, pelanggan tinggal memakai dan membayar jasa sewanya setiap bulan sesuai dengan pemakaian. Sistem tersebut dapat mereduksi atau menghilangkan biaya menginstal dan menjalankan aplikasi pada komputer pengguna, SaaS menghapuskan kerumitan dalam pemeliharaan software, aktifitas operational dan support. Sehingga dapat dikatakan, dengan SaaS, pengeluaran belanja TI konsumen bisa lebih hemat. Selain itu, terdapat keuntungan potensial penerapan SaaS yaitu:

* Kemudahan administrasi.
* Pengelolaan *update* dan *patch* perangkat lunak secara otomatis.
* Kompatibilitas, karena semua pengguna memiliki versi *software* yang sama.
* Kemudahan untuk berkolaborasi.
* Dapat di akses dari seluruh dunia (global), karena menggunakan *browser* internet.

1. ***SOA***

SOA (*Service Oriented Architecture*/Arsitektur Berorientasi Layanan) adalah suatu gaya arsitektur sistem yang membuat dan menggunakan proses bisnis dalam bentuk paket layanan sepanjang siklus hidupnya. SOA juga mendefinisikan dan menentukan arsitektur teknologi informasi (TI) yang dapat menunjang berbagai aplikasi untuk saling bertukar data dan berpartisipasi dalam proses bisnis. Fungsi-fungsi ini tidak terikat dengan sistem operasi dan bahasa pemrograman yang mendasari aplikasi-aplikasi tersebut.

SOA membagi fungsi-fungsi menjadi unit-unit yang berbeda (layanan), yang dapat didistribusikan melalui suatu jaringan dan dikombinasikan serta digunakan ulang untuk membentuk aplikasi bisnis. Layanan-layanan ini saling berkomunikasi dengan mempertukarkan data antar mereka atau dengan mengkoordinasikan aktivitas antara dua atau lebih layanan. Konsep SOA sering dianggap didasari atau berkembang dari konsep-konsep yang lebih lama dari komputasi terdistribusi dan pemrograman modular[6]

Berikut adalah 8 prinsip utama dari arsitektur SOA[7] :

1. *Standarized Service Contract*

Ketika menciptakan sebuah *service* yang akan digunakan, kita perlu untuk mendefinisikan tujuan dan kemampuannya dalam standar yang telah ditentukan. Hal ini dilakukan agar dapat memudahkan penggunaan *service* secara konsisten.

1. *Service Loose Coupling*

*Coupling* dapat diartikan sebagai kebergantungan, maksudnya apakah *service* yang digunakan saling bergantung dengan *service* lainnya. Semakin tinggi tingkat kebergantungan *service* maka aplikasi akan menjadi semakin kurang fleksibel. Dengan prinsip *loose coupling*, maka tingkat kebergantungan tersebut diupayakan untuk serendah mungkin.

1. *Service Abstraction*

*Service* diupayakan untuk menyimpan logika kerjanya kecuali apa yang didefinisikan pada *service contract*, hal ini dilakukan untuk mempermudah proses *loose coupling*.

1. *Service Reusability*

Service harus dapat digunakan kembali, oleh karena itu sangatlah penting adanya *service abstraction* untuk menyembunyikan logika yang dimilikinya.

1. *Service Autonomy*

Service diharuskan memiliki kontrol penuh atas logika kerja terabstraksikan yang ia tangani.

1. *Service Statelessness*

Service diupayakan untuk tidak memiliki *state* sama sekali, hal ini dilakukan agar tidak banyak data yang harus diubah dalam proses kerja aplikasi.

1. *Service Discoverability*.

Servis yang didesign untuk mengeluarkan deskripsi sehingga servis yang dicari dapat ditemukan dan ditafsirkan via ***discovery mechanisms*** yang tersedia.

1. *Service Composability*

Koleksi dari servis-servis dapat dikoordinasikan dan dirakit menjadi gabungan servis.

1. ***SOAD***

SOAD memberikan panduan untuk merancang, membangun, agregasi, dan penyebaran aplikasi seperti layanan web yang dibangun dengan teknologi yang dapat mencakup SOAP, WSDL atau UDDI. SOAD adalah sebuah bidang yang muncul yang berhubungan dengan identifikasi dan membangun service berdasarkan kebutuhan bisnis yang ada. Titik awal dalam SOAD adalah pemodelan bisnis proses. SOAD meliputi seluruh rentang mulai dari analisis proses bisnis sampai ke desain teknologi Informasi dan implementasinya.

1. ***ERP (Enterprise Resource planning)***

ERP / Perencanaan Sumber Daya Perusahaan adalah sistem informasi yang diperuntukkan bagi perusahaan manufakturing maupun jasa yang berperan mengintegrasikan dan mengotomasikan proses bisnis yang berhubungan dengan aspek operasi, produksi maupun distribusi diperusahaan tersebut. Jadi ERP adalah sebuah termologi yang diberikan kepada sistem informasi yang mendukung transaksi atau operasi sehari-hari dalam pengelolaan sumber daya perusahaan meliputi dana, manusia, mesin, suku cadang, waktu, material dan kapasistas.

Keuntungan penggunaan ERP diantaranya adalah integrasi data keuangan, standarisaasi proses operasi, standarisasi data dan informasi, penurunan inventori dan tenaga kerja, peningkatan servis level dan kontrol keuangan dan penurunan waktu yang dibutuhkan untuk mendapatkan informasi.

1. ***Web Service***

Web service mendefinikan cara untuk mengintegrasikan aplikasi berbasis web dengan menggunakan XML, SOAP, WSDL dan UDDI open standar melalui protokol internet. XML digunakan untuk memberi "tag" pada data, SOAP digunakan untuk transfer data, WSDL digunakan untuk mendeskripsikan keberadaan service dan UDDI digunakan untuk mendaftarkan list service yang tersedia. Web Service mengijinkaan suatu organisasi untuk mengkomunikasikan data tanpa harus mengetahui kompleksnya sistem IT dibelakang firewall.

Tidak seperti model client/server, seperti web server/sistem halaman web, web service tidak menyediakan GUI untuk user. Web service hanya berbagi logika bisnis, data dan proses melalui interface dan saling berkomunikasi melewati jaringan komputer. Web service mengijinkan bermacam-macam aplikasi yang berasal dari mana saja untuk berkomunikasi tanpa ada pemborosan waktu karena harus terdapat penyesuaian standar komunikasi, dan karena semua komunikasi melalui XML, Web service tidak diikat oleh salah satu sistem operasi atau bahasa pemrograman[8].

1. ***JSF***

JSF adalah framework untuk membangun user interface untuk aplikasi web. Dibangun berdasar pada konsep-konsep yang diperkenalkan oleh Struts dan membagi bersama keuntungan sebuah arsitektur yang benar-benar memisahkan presentasi layer dari business logic dan sebuah standard komponen user interface yang perangkatnya serupa dengan widget Swing.



**Gambar 9** *JavaServerFaces Framework*

Seperti yang kita lihat, JSF juga mempunyai pemisahan yang jelas antara komponen layer Model, View , dan Controller. Sama seperti Struts, JSF memiliki sebuah controller servlet bagian depan yaitu FacesServlet yang bertanggung jawab untuk menerima permintaan dari client dan kemudian menjalankan action yang dibutuhkan yang dituntun oleh framework. Persamaan lainnya adalah mereka berdua menggunakan action handler yang terpisah dari controller servlet bagian depan. Meskipun demikian handle-handle Faces ini sedikit berbeda dengan Struts.

Faces dan Struts memiliki tujuan yang sama berkaitan dengan layer View. Disini, Struts hanya menyediakan sebuah set libray-libray tag yang ditambahkan pada bagian atas fungsi HTML standard. Sebaliknya, Faces menyediakan set sendiri dari komponen-komponen beserta sebuah set library-library untuk memperlihatkan komponen-komponen ini sebagai tag-tag dan sebuah komponen hasil render yang menterjemahkan komponen UI menjadi HTML[9].

1. **METODOLOGI**

Perencanaan dalam menyelesaikan Tugas Akhir ini ada beberapa tahap, diantaranya sebagai berikut:

1. Penyusunan Proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan tugas akhir adalah penyusunan proposal tugas akhir. Pada proposal ini, penulis mengajukan gagasan perancangan dan pengembangan aplikasi perangkat lunak *Account Payable, Account Receivable dan Fixed Asset* dengan menggunakan SOAD pada platform java.

1. Studi Literatur

Pada tahapan ini akan dilakukan studi literatur mengenai objek kasus dan metode yang digunakan, diantaranya :

* + - * 1. Pemahaman mengenai proses bisnis *Account Payable, Account Receivable dan Fixed Asset*.
        2. Perancangan menggunakan SOAD yang disesuaikan dengan model deployment SaaS.
        3. Pemahaman mengenai framework aplikasi web dan web service pada platform java.

1. Implementasi

Implementasi merupakan tahap untuk merancang dan membangun aplikasi perangkat lunak *Account Payable, Account Receivable dan Fixed Asset*.

1. Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba terhadap aplikasi yang telah dibuat.

Langkah-langkah uji coba yang akan dilakukan antara lain :

1. Menguji apakah aplikasi yang dibuat sesuai dengan service pada level bisnis yang telah ditentukan saat proses desain.
2. Menguji apakah aplikasi yang dibuat setelah diintegrasikan dapat menghasilkan data yang valid.
3. Penyusunan Buku Tugas Akhir

Tahap terakhir merupakan penyusunan laporan yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perancangan dan prototyping yang telah dibuat. Secara garis besar, buku laporan tugas akhir ini terdiri atas beberapa bagian yaitu:

1. Pendahuluan
   1. Latar Belakang
   2. Permasalahan
   3. Batasan Tugas Akhir
   4. Tujuan
   5. Metodologi
   6. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Uji Coba dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

# **JADWAL KEGIATAN**

Tugas akhir ini diharapkan bisa dikerjakan menurut jadwal sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Kegiatan | 2010 | | | | | | | 2011 | | |
| Juni | Juli | Agu | Sep | Okt | Nov | Des | | Jan | Feb |
| 1. | Penyusunan Proposal Tugas Akhir |  |  |  |  |  |  |  | |  |  |
| 2. | Studi Literatur |  |  |  |  |  |  |  | |  |  |
| 3. | Implementasi |  |  |  |  |  |  |  | |  |  |
| 4. | Pengujian dan Evaluasi |  |  |  |  |  |  |  | |  |  |
| 5. | Penyusunan Buku Tugas Akhir |  |  |  |  |  |  |  | |  |  |

# **DAFTAR PUSTAKA**

[1] IBM. (2004). *Patterns: Service-Oriented Architecture and Web Services.*

[2] Brown, Paul C. (2008). *Implementating SOA : Total Architecture in Practice*. Addison Wesley Profesional, Massachusetts.

[3] Bieberstein, et al. (2008). *Executing SOA: A Practical Guide for The Service Oriented Architecture*. IBM Press, Indiana.

[4] Microsoft. 2010. Software as a Service. (Online)

(http://www.microsoft.com/serviceproviders/saas/default.mspx, diakses 22 September 2010)

[5] Wikipedia. 2010. *Software as a Service*. (Online)

# (http://en.wikipedia.org/wiki/Software\_as\_a\_service, diakses 23 September 2010)

[6] Wikipedia. 2010. *Service Oriented Architecture*. (Online)

# (http://id.wikipedia.org/wiki/SOA, diakses 6 November 2010)

[7] SOA System inc. 2009. *SOA Principles*. (Online)

# http://soaprinciples.com/default.php, diakses 6 November 2010)

[8] Webopedia. 2010. *Web Services*. (Online)

# (http://www.webopedia.com/TERM/W/Web\_services.html, diakses 6 November 2010)

[9] J.E.N.I . 2010. *JSF*. (Online)

# (http://poss.ipb.ac.id/files/JENI-Web%20Programming-Bab%209-JSF.pdf, diakses 6 November 2010)

**LEMBAR PENGESAHAN**

###### **Surabaya, 27 September 2010**

Menyetujui,

Pembimbing II

Dwi Sunaryono S.Kom

NIP : 132163671

Pembimbing I

Prof. Drs.Ec.Ir. Riyanarto Sarno,M.Sc,Ph.D

NIP : 131570363