**SISTEM INFORMASI AKUNTANSI BERBASIS WEB   
DENGAN PENDEKATAN AJAX**

**PADA PT. BUMI SENTOSA**

**LAPORAN KERJA PRAKTEK**

Diajukan Untuk Memenuhi Salah Satu Syarat Kelulusan Sarjana

Pada Teknik Informatika STMIK BANDUNG BALI

Program Pendidikan Strata - 1 (S1)

Disusun Oleh :

**I Nengah Adi Sayoga 1208004**



**JURUSAN TEKNIK INFORMATIKA**

**SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER**

**BANDUNG BALI**

**DENPASAR**

**2011**

**SISTEM INFORMASI AKUNTANSI BERBASIS WEB   
DENGAN PENDEKATAN AJAX   
PADA PT. BUMI SENTOSA**

# LEMBAR PENGESAHAN

Oleh:

I Nengah Adi Sayoga 1208004

Telah diperiksa dan disetujui sebagai Laporan Kerja Praktek   
Tanggal \_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2011

(Putu Yuli Mardini, SE) (Ir. I Gede Made Karma, MT)

Pembimbing Perusahaan Dosen Pembimbing

Mengetahui,

(Ni Nym. Emang Smrti, S.Kom)

Ketua Jurusan Teknik Informatika

# PERNYATAAN KEASLIAN

Saya sebagai penulis menyatakan dengan sesungguhnya bahwa laporan kerja praktek ini merupakan hasil pemikiran dan pemaparan asli. Jikalau terdapat referensi terhadap karya orang lain, penulis lakukan dengan menyebut sumber secara jelas.

Demikian pernyataan ini penulis buat secara sadar dan sebenar-benarnya.

Penulis

(I Nengah Adi Sayoga)

1208004

# KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa (Ida Sang Hyang Widhi Wasa) sehingga Penulis dapat menyelesaikan kerja prakter dengan judul **“Sitem Informasi Akuntansi Berbasis Web dengan Pendekatan Ajax pada PT. Bumi Sentosa”** sebagai salah satu syarat dalam menyelesaikan program sarjana pada jurusan Teknik Informatika STMIK BANDUNG BALI.

Dalam penyusunan laporan ini, penulis telah banyak menerima dukungan, bantuan dan bimbingan dari berbagai pihak. Pada kesempatan ini Penulis ingin terima kasih yang sebesar-besarnya kepada:

1. Bapak Ir. I Gede Made Karma, MT, selaku dosen pembimbing, atas arahan dan bimbingannya kepada Penulis.
2. Bapak I Made Alit Sastrawan, SE selaku direktur PT. Bumi Sentosa yang telah memberikan kesempatan untuk melaksanakan kerja praktek ini.
3. Ibu Putu Yuli Mardini, SE selaku manajer internal PT. Bumi Sentosa yang telah memberikan informasi-informasi perusahaan yang diperlukan penulis.
4. Seluruh staf dosen di STMIK BANDUNG BALI yang telah mengajari penulisan selama ini.
5. Teman-teman dan semua pihak yang tidak dapat Penulis sebutkan satu-persatu.

Penulis menyadari bahwa dalam penyusunan laporan ini masih terdapat banyak kekurangan dan jauh dari sempurna. Mengingat kemampuan dan pengalaman Penulis yang masih terbatas dalam penelitian di lapangan, teknik pengolahan data dan teknik penyajian. Sehingga dengan rendah hati Penulis mengharapkan kritik dan saran yang bersifat membangun dari semua pihak.

Akhir kata, semoga laporan ini dapat bermanfaat bagi penulis khususnya dan bagi semua pihak yang berkepentingan pada umumnya.

Denpasar, \_\_\_\_ Mei 2011

Penulis

# DAFTAR ISI

**HALAMAN JUDUL i**

[LEMBAR PENGESAHAN ii](#_Toc293952002)

[PERNYATAAN KEASLIAN iii](#_Toc293952003)

[KATA PENGANTAR iv](#_Toc293952004)

[DAFTAR ISI vi](#_Toc293952005)

[DAFTAR TABEL xi](#_Toc293952006)

[DAFTAR GAMBAR xii](#_Toc293952007)

[BAB I PENDAHULUAN 1](#_Toc293952008)

[1.1 Latar Belakang 1](#_Toc293952010)

[1.2 Identifikasi Masalah 2](#_Toc293952011)

[1.3 Batasan Masalah 3](#_Toc293952012)

[1.4 Tujuan Dan Manfaat 3](#_Toc293952013)

[1.5 Metode Penelitian 4](#_Toc293952014)

[1.5.1 Metode Pengumpulan Data 4](#_Toc293952015)

[1.5.2 Metode Pengembangan Perangkat Lunak 4](#_Toc293952016)

[BAB II LANDASAN TEORI 8](#_Toc293952017)

[2.1 Sistem Informasi 8](#_Toc293952019)

[2.1.1 Pengertian Sistem 8](#_Toc293952020)

[2.1.2 Pengertian Informasi 8](#_Toc293952021)

[2.1.3 Pengertian Sistem Informasi 9](#_Toc293952022)

[2.1.4 Pengertian Sistem Informasi Akuntansi 10](#_Toc293952023)

[2.1.5 Komponen-Komponen Sistem Informasi Akuntansi 11](#_Toc293952024)

[2.2 Analisa dan Desain Sistem 12](#_Toc293952025)

[2.2.1 Analisa Sistem 12](#_Toc293952026)

[2.2.2 Desain Sistem 13](#_Toc293952027)

[2.2.2.1 *Unified Modeling Language* (UML) 14](#_Toc293952028)

[2.2.2.2 *Entity Relationship Diagram* (ERD) 15](#_Toc293952029)

[2.3 Basis Data 16](#_Toc293952030)

[2.4 Perangkat Lunak 18](#_Toc293952031)

[2.4.1 *Web* 18](#_Toc293952032)

[2.4.2 PHP (*Hypertext Preprocessor*) 19](#_Toc293952033)

[2.4.3 Zend Framework 19](#_Toc293952034)

[2.4.4 Ajax 20](#_Toc293952035)

[2.4.5 JQuery 20](#_Toc293952036)

[2.4.6 MySQL 21](#_Toc293952037)

[BAB III GAMBARAN UMUM PERUSAHAAN 24](#_Toc293952038)

[3.1 Proyek 24](#_Toc293952040)

[3.1.1 Penyedia Tenaga Kerja 24](#_Toc293952041)

[3.1.2 Instalasi dan Pemeliharaan Jaringan Listrik 24](#_Toc293952042)

[3.1.3 Konsultasi Teknologi Informasi 25](#_Toc293952043)

[3.1.4 Pengadaan Barang 25](#_Toc293952044)

[3.1.5 Pelayanan Teknik 25](#_Toc293952045)

[3.1.6 Pemeliharaan AMR (*Automatic Meter Reading*) 26](#_Toc293952046)

[3.1.7 PPTL (Penertiban Pemakaian Tenaga Listrik) 26](#_Toc293952047)

[3.1.8 Survey Pelanggan 27](#_Toc293952048)

[3.1.9 Pengembangan dan *Updating* GIS (*Geographics Infor-mation System*) 27](#_Toc293952049)

[3.1.10 Penyediaan Tenaga Administrasi 27](#_Toc293952050)

[3.2 Struktur Organisasi 28](#_Toc293952051)

[*3.3* *Job Description* 29](#_Toc293952052)

[3.3.1 Direktur 29](#_Toc293952053)

[3.3.2 Marketing 30](#_Toc293952054)

[3.3.3 Operasional 30](#_Toc293952055)

[3.3.4 Internal 30](#_Toc293952056)

[BAB IV ANALISA DAN PERANCANGAN SISTEM 32](#_Toc293952057)

[4.1 Mekanisme Sistem Yang Sedang Berjalan 32](#_Toc293952059)

[4.1.1 *Use Case* Diagram 32](#_Toc293952060)

[4.1.2 *Activity* Diagram 33](#_Toc293952061)

[4.2 Mekanisme Sistem Yang Diusulkan 33](#_Toc293952062)

[4.2.1 *Use Case* Diagram 34](#_Toc293952063)

[4.2.2 *Activity* Diagram 34](#_Toc293952064)

[4.2.3 *Class* Diagram 35](#_Toc293952065)

[4.2.4 *Sequence* Diagram 37](#_Toc293952066)

[4.3 *Entity Relationship Diagram* (ERD) 44](#_Toc293952067)

[4.4 Tabel Relasi 45](#_Toc293952068)

[*4.5* Struktur *Database* 46](#_Toc293952069)

[4.6 Perancangan Sistem 48](#_Toc293952070)

[4.6.1 Perancangan Input 50](#_Toc293952071)

[4.6.1.1 Rancangan Login 50](#_Toc293952072)

[4.6.1.2 Rancangan Menu Utama 50](#_Toc293952073)

[4.6.1.3 Rancangan Pengolahan Data *User* 51](#_Toc293952074)

[4.6.1.4 Rancangan Pengolahan Data Akun 52](#_Toc293952075)

[4.6.1.5 Rancangan Pengolahan Transaksi Jurnal 53](#_Toc293952076)

[4.6.1.6 Rancangan Pengolahan Jurnal Penyesuaian 54](#_Toc293952077)

[4.6.2 Perancangan Output 56](#_Toc293952078)

[4.6.2.1 Rancangan Laporan Akun 56](#_Toc293952079)

[4.6.2.2 Rancangan Laporan Jurnal Umum 57](#_Toc293952080)

[4.6.2.3 Rancangan Laporan Buku Besar 58](#_Toc293952081)

[4.6.2.4 Rancangan Laporan Laba/Rugi 59](#_Toc293952082)

[4.6.2.5 Rancangan Laporan Neraca 60](#_Toc293952083)

[BAB V IMPLEMENTASI SISTEM 61](#_Toc293952084)

[5.1 Komponen Sistem 61](#_Toc293952086)

[5.1.1 Login *User* 61](#_Toc293952087)

[5.1.2 Menu Utama 62](#_Toc293952088)

[5.1.3 Pengolahan Data *User* 63](#_Toc293952089)

[5.1.4 Pengolahan Data Akun 66](#_Toc293952090)

[5.1.5 Pengolahan Transaksi Juranl 70](#_Toc293952091)

[5.1.6 Pengolahan Jurnal Penyesuaian 72](#_Toc293952092)

[5.1.7 Laporan Data Akun 74](#_Toc293952093)

[5.1.8 Laporan Jurnal Umum 75](#_Toc293952094)

[5.1.9 Laporan Buku Besar 76](#_Toc293952095)

[5.1.10 Laporan Laba/Rugi 77](#_Toc293952096)

[5.1.11 Laporan Neraca 78](#_Toc293952097)

[5.2 Spesifikasi Kebutuhan Perangkat Lunak 80](#_Toc293952098)

[5.2.1 Antar Muka Sistem 80](#_Toc293952099)

[5.2.2 Proses Sistem Informasi Akuntansi 80](#_Toc293952100)

[5.3 Kebutuhan Perangkat Keras 81](#_Toc293952101)

[5.4 Kegiatan Implementasi 82](#_Toc293952102)

[5.4.1 Pemrograman 82](#_Toc293952103)

[5.5 Pengujian Sistem 82](#_Toc293952104)

[BAB VI PENUTUP 83](#_Toc293952105)

[6.1 Kesimpulan 83](#_Toc293952107)

[6.2 Saran 83](#_Toc293952108)

[DAFTAR PUSTAKA 84](#_Toc293952109)

[LAMPIRAN KODE PROGRAM 86](#_Toc293952110)

# DAFTAR TABEL

[Tabel 4.1 Tabel User 46](#_Toc292635605)

[Tabel 4.2 Tabel Account 47](#_Toc292635606)

[Tabel 4.3 Tabel Bukti Transaksi 48](#_Toc292635607)

[Tabel 4.4 Tabel Journal 48](#_Toc292635608)

# DAFTAR GAMBAR

[Gambar 1.1 Model *Waterfall* (*Classic Life Cycle*) 5](#_Toc293842822)

[Gambar 2.1 Kruchten 4+1 *model view* 14](#_Toc293842823)

[Gambar 3.1 Struktur Organisasi PT. Bumi Sentosa 29](#_Toc293842824)

[Gambar 4.1 *Use Case* Diagram yang Sedang Berjalan 32](#_Toc293842825)

[Gambar 4.2 *Activity* Diagram yang Sedanga Berjalan 33](#_Toc293842826)

[Gambar 4.3 *Use Case* Diagram yang Diusulkan 34](#_Toc293842827)

[Gambar 4.4 *Activity* Diagram yang Diusulkan 34](#_Toc293842828)

[Gambar 4.5 *Class* Diagram *(Model)* 35](#_Toc293842829)

[Gambar 4.6 *Class* Diagram (*Controller*) 36](#_Toc293842830)

[Gambar 4.7 *Class* Diagram *(Form)* 36](#_Toc293842831)

[Gambar 4.8 *Sequence* Diagram (Login) 37](#_Toc293842832)

[Gambar 4.9 *Sequence* Diagram (Daftar Akun) 38](#_Toc293842833)

[Gambar 4.10 *Sequence* Diagram (Tambah Akun) 38](#_Toc293842834)

[Gambar 4.11 *Sequence* Diagram (Tambah Jurnal) 39](#_Toc293842835)

[Gambar 4.12 *Sequence* Diagram (Laporan Akun) 40](#_Toc293842836)

[Gambar 4.13 *Sequence* Diagram (Laporan Jurnal Umum) 40](#_Toc293842837)

[Gambar 4.14 *Sequence* Diagram (Laporan Buku Besar) 41](#_Toc293842838)

[Gambar 4.15 *Sequence* Diagram (Laporan Laba/Rugi) 42](#_Toc293842839)

[Gambar 4.16 *Sequence* Diagram (Laporan Neraca) 43](#_Toc293842840)

[Gambar 4.17 *Sequence* Diagram (Transaksi Periode) 44](#_Toc293842841)

[Gambar 4.18 *Entity Relationship Diagram* 45](#_Toc293842842)

[Gambar 4.19 Tabel Relasi 46](#_Toc293842843)

[Gambar 4.20 Rancangan Form *Login* 50](#_Toc293842844)

[Gambar 4.21 Rancangan Menu Utama 50](#_Toc293842845)

[Gambar 4.22 Rancangan Daftar *User* 51](#_Toc293842846)

[Gambar 4.23 Rancangan Form Tambah Data *User* 51](#_Toc293842847)

[Gambar 4.24 Rancangan Form Ubah Data *User* 51](#_Toc293842848)

[Gambar 4.25 Rancangan Form Ubah *Password* 52](#_Toc293842849)

[Gambar 4.26 Rancangan Daftar Akun 52](#_Toc293842850)

[Gambar 4.27 Rancangan Form Tambah Data Akun 52](#_Toc293842851)

[Gambar 4.28 Rancangan Form Ubah Data Akun 53](#_Toc293842852)

[Gambar 4.29 Rancangan Daftar Data Jurnal 53](#_Toc293842853)

[Gambar 4.30 Rancangan Tambah/Ubah Data Jurnal 54](#_Toc293842854)

[Gambar 4.31 Rancangan Daftar Data Jurnal Penyesuaian 54](#_Toc293842855)

[Gambar 4.32 Rancangan Tambah/Ubah Data Jurnal Penyesuaian 55](#_Toc293842856)

[Gambar 4.33 Rancangan Laporan Data Akun 56](#_Toc293842857)

[Gambar 4.34 Rancangan Laporan Jurnal Umum 57](#_Toc293842858)

[Gambar 4.35 Rancangan Laporan Buku Besar 58](#_Toc293842859)

[Gambar 4.36 Rancangan Laporan Laba/Rugi 59](#_Toc293842860)

[Gambar 4.37 Rancangan Laporan Neraca 60](#_Toc293842861)

[Gambar 5.1 Implementasi Form Login 62](#_Toc293842862)

[Gambar 5.2 Implementasi Menu Utama 63](#_Toc293842863)

[Gambar 5.3 Implementasi Daftar User 64](#_Toc293842864)

[Gambar 5.4 Implementasi Form Tambah Data User 65](#_Toc293842865)

[Gambar 5.5 Implementasi Form Ubah Data User 65](#_Toc293842866)

[Gambar 5.6 Implementasi Form Ubah Password User 66](#_Toc293842867)

[Gambar 5.7 Implementasi Daftar Akun 68](#_Toc293842868)

[Gambar 5.8 Implementasi Form Tambah Data Akun 68](#_Toc293842869)

[Gambar 5.9 Implementasi Form Ubah Data Akun 69](#_Toc293842870)

[Gambar 5.10 Implementasi Daftar Jurnal 70](#_Toc293842871)

[Gambar 5.11 Implementasi Form Tambah Data Jurnal 71](#_Toc293842872)

[Gambar 5.12 Implementasi Form Ubah Data Jurnal 71](#_Toc293842873)

[Gambar 5.13 Implementasi Daftar Jurnal Penyesuaian 73](#_Toc293842874)

[Gambar 5.14 Implementasi Form Tambah Data Jurnal Penyesuaian 73](#_Toc293842875)

[Gambar 5.15 Implementasi Form Ubah Jurnal Penyesuaian 74](#_Toc293842876)

[Gambar 5.16 Implementasi Laporan Data Akun 75](#_Toc293842877)

[Gambar 5.17 Implementasi Laporan Jurnal Umum 76](#_Toc293842878)

[Gambar 5.18 Implementasi Laporan Buku Besar 77](#_Toc293842879)

[Gambar 5.19 Implementasi Laporan Laba/Rugi 78](#_Toc293842880)

[Gambar 5.20 Implementasi Laporan Neraca 79](#_Toc293842881)

# BAB I PENDAHULUAN

# 

## Latar Belakang

Kemajuan teknologi informasi telah mengubah cara perusahaan dalam mengumpulkan data, memproses dan melaporkan informasi keuangan. Oleh karena itu auditor akan banyak menemukan lingkungan dimana data tersimpan lebih banyak dalam media elektronik dibanding media kertas. Sebenarnya tidak ada perbedaan konsep audit yang berlaku untuk sistem yang kompleks dan sistem manual, yang berbeda hanyalah metode-metode spesifik yang cocok dengan situasi sistem informasi akuntansi yang ada.

PT. Bumi Sentosa merupakan perusahaan mitra PLN (Perusahaan Listrik Negara) yang mengerjakan proyek-proyek PLN seperti pelayanan teknik, survey, penyediaan satpam dan sebagainya. Visi dari perusahaan ini adalah menjadi perusahaan skala nasional yang memberikan pelayanan yang lebih baik bagi setiap pelanggan.

Walaupun PT. Bumi Sentosa sudah memiliki aplikasi program akuntansi, tetapi program ini hanya baru di-*install* terbatas hanya dua komputer saja, jika ingin men-*install* di komputer lainnya atau komputernya bermasalah sehinga program ini hilang, maka harus menghubungi pihak pengembang aplikasi dan tentu saja akan memerlukan tambahan biaya, sehingga menjadi tidak efisien dari segi waktu dan biaya. Tidak hanya dari sisi pengguna, dari sisi pengembang juga akan kerepotan bolak-balik untuk memeriksa aplikasinya apakah sudah berjalan dengan baik atau belum.

Melalui pembuatan kerja praktek ini, penulis menawarkan aplikasi berbasis *web*. Dengan berbasis *web*, aplikasi bisa diakses dari mana saja dan tidak perlu di-*install* pada sisi *client*, cukup dengan membuka *browser*. perawatannya pun bisa dilakukan dari jarak jauh. Dan dengan pendekatan Ajax, aplikasi ini akan menjadi lebih responsif. Kerja praktek yang akan dibuat berjudul “Sistem Informasi Akuntansi Berbasis Web dengan Pendekatan Ajax pada PT. Bumi Sentosa”.

## Identifikasi Masalah

Dengan mengacu pada latar belakang masalah, maka masalah dapat diidentifikasikan sebagai berikut:

1. Aplikasi yang terpasang belum cukup banyak.
2. Instalasi aplikasi baru pada sisi *client* lebih merepotkan daripada berbasis web yang hanya memerlukan *browser*.
3. Sistem lama hanya dapat diakses dari dalam gedung perusahaan.
4. Perawatan aplikasi harus mendatangkan pembuat aplikasi tersebut, dan setiap kedatangan akan dikenakan biaya.

## Batasan Masalah

Dengan luasnya cakupan sistem informasi akuntansi, dalam penulisan kerja praktek ini penulis membatasi pembahasan masalah yang ada yaitu:

* 1. Masukan terdiri dari pemasukan data *account*, transaksi jurnal (jurnal umum dan jurnal penyesuaian).
  2. Laporan terdiri dari laporan account, jurnal umum, buku besar, rugi/laba dan neraca.
  3. Pendekatan Ajax yang digunakan terdiri dari banyak halaman dan setiap halaman menerapkan Ajax.
  4. Pada sistem ini diusulkan untuk menggunakan Zend *Framework*, JQuery dan dengan *database* MySQL.

## Tujuan Dan Manfaat

Adapun tujuan dan manfaat penulisan kerja praktek ini yaitu:

1. Tidak ada program khusus yang di-*install* pada *client*.
2. Tidak tergantung pada sistem operasi tertentu.
3. Dapat diakses dari mana saja, selama terkoneksi internet.
4. Perawatan aplikasi dapat dilakukan dari jarak jauh.
5. Untuk memenuhi salah satu syarat kelulusan program sarjana (S-1) di STMIK Bandung Bali.

## Metode Penelitian

Metode penelitian adalah suatu metode yang menggambarkan rencana yang akan digunakan dalam suatu penelitian. Metode penelitian yang akan diterapkan, diantaranya:

### Metode Pengumpulan Data

Dalam penyusunan kerja praktek ini digunakan teknik atau metode-metode pengumpulan data, dengan tujuan agar data yang dibutuhkan dapat diperoleh dengan lengkap, sehingga dapat memperoleh data dan informasi untuk pengambilan kesimpulan dalam penyusunan laporan.

Metode pengumpulan data yang digunakan adalah:

1. Wawancara: proses tanya-jawab dan tatap muka langsung dengan narasumber.

2. Observasi: melakukan pengumpulan data dan pengamatan langsung pada objek penelitian.

3. Studi pustaka: teknik pengumpulan data dengan mempelajari dari sejumlah buku dan media elektronik untuk teori yang diperlukan dalam pembuatan kerja praktek.

### Metode Pengembangan Perangkat Lunak

Pengembangan perangkat lunak menggunakan metode *Waterfall Model* (*Classic Life Cycle*) yang dilakukan dari tahap definisi sampai pemeliharaan. Model *Waterfall*, membutuhkan pendekatan yang sistematis dalam pengembangan sebuah perangkat lunak.



Gambar 1.1 Model *Waterfall* (*Classic Life Cycle*)

Sumber: Dan Brandon

Keterangan:

1. *Definition* (definisi).

Langkah definisi melibatkan pembuatan tujuan yang jelas, mengidentifikasi mengapa dan bagaimana sistem yang diusulkan akan lebih baik, lebih murah, lebih cepat daripada yang digantikan, dan biasanya secara keseluruhan/kasar analisis biaya-manfaat.

1. *Specification* (spesifikasi/kebutuhan).

Langkah dari spesifikasi melibatkan lingkup pernyataan lengkap (kebutuhan), skenario *use case*, penyusunana *user manual* (spesifikasi desain eksternal), rincian rencana proyek, spesifikasi sumber daya yang dibutuhkan, menyaring perkiraan waktu dan biaya, menyaring analisis biaya-manfaat, dan menandatangani persetujuan persyaratan dan *user manual* oleh para pemangku kepentingan.

1. *Design* (desain).

Langkah desain melibatkan resolusi dari masalah teknis yang kritis, pemilihan arsitektur dan *platform*, adopsi standar, penempatan staf, penyelesaian desain eksternal (desain *user interface*), desain struktur data penting dan *database*, desain internal dari algoritma dan proses, persyaratan matrix ketertelusuran, naskah uji pendahuluan, waktu final dan perkiraan biaya, dan analisis biaya-manfaat akhir.

1. *Construction* (konstruksi/pemrograman dan *unit testing*)

Langkah konstruksi melibatkan pelaksanaan desain (misalnya, melalui *coding*), *unit testing*, integrasi sistem, draf dokumentasi internal, dan penyelesaian uji skrip.

1. *Testing* (pengujian/sistem dan integrasi).

Langkah pengujian melibatkan integrasi skala penuh dan pengujian sistem, penyelesaian dokumentasi pengguna, penyelesaian material pelatihan, penerapan prosedur pengendalian perubahan formal, penyelesaian dokumentasi internal.

1. *Installation* (Instalasi).

Langkah instalasi melibatkan produk jadi, pelatihan pengguna akhir, memproduksi dokumentasi pelajaran, dan mendefinisikan prosedur untuk operasi penanganan, dukungan pengguna, dan manajemen konfigurasi.

1. *Maintenance* (operasi dan pemeliharaan).

Langkah pemeliharaan melibatkan mengikuti dan merevisi prosedur untuk menyelesaikan masalah dan eskalasi masalah, operasi, *backup* keamanan, kontrol konfigurasi, dan kualitas/pemantauan kerja.

# BAB II LANDASAN TEORI

# 

## Sistem Informasi

### Pengertian Sistem

Konsisten, seperangkat komponen yang terkoordinasi bertindak bersama sebagai satu kesatuan menuju fungsi umum atau tujuan merupakan suatu sistem. Komponen sistem harus bekerja bersama. Sistem yang tidak konsisten bekerja terhadap dirinya sendiri. (Lee Ratzan, 2004)

Pengertian tersebut mencerminkan adanya beberapa bagian dan hubungan antarbagian, ini menunjukkan kompleksitas dari sistem yang meliputi kerja sama antar bagian yang interdependen satu sama lain. Selain itu dilihat bahwa sistem berusaha mencapai tujuan. Pencapaian tujuan ini menyebabkan timbulnya dinamika, perubahan yang terus menerus perlu dikembangkan dan dikendalikan. Definisi tersebut menunjukkan bahwa sistem sebagai gugus dari elemen-elemen yang saling berinteraksi secara teratur dalam rangka mencapai tujuan atau subtujuan.

### Pengertian Informasi

Kebanyakan orang tahu apa informasi itu... atau setidaknya mereka pikir begitu. Mereka ragu. Mereka berusaha menemukan kata yang tepat. Beberapa orang mungkin mendefinisikan informasi sebagai data. Data bukan informasi untuk dua alasan. Pertama, data pada hakekatnya tidak tertanam dengan pengertian maknanya, seperti informasi. Data menggunakan simbol, nomor, atau tulisan untuk proses menunjukkan informasi. Dan kedua, kata “data” adalah kata benda jamak. (Lee Ratzan, 2004)

Menurut Kusrini (2007), Informasi adalah data yang sudah diolah menjadi sebuah bentuk berarti bagi pengguna, yang bermanfaat dalam pengambilan keputusan saat ini atau mendukung sumber infomasi.

### Pengertian Sistem Informasi

Menurut N. Kumar dan R. Mittal (2004), Sistem informasi adalah suatu kombinasi terorganisir dari orang, *hardware*, *software*, jaringan komunikasi, dan *resouce* data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam suatu organisasi. Orang-orang telah mengandalkan pada sistem informasi untuk berkomunikasi satu sama lain dengan menggunakan berbagai perangkat fisik (*hardware*), instruksi pemrosesan informasi dan prosedur (perangkat lunak), saluran komunikasi (jaringan), dan disimpan data (sumber data) sejak ditarik peradaban.

### Pengertian Sistem Informasi Akuntansi

Menurut Teguh Wahyono dan Leonard Pujiatmoko (2008), akuntansi merupakan suatu seni mencatat, mengklasifikasi, dan meringkas data keuangan dengan cara yang tertentu untuk mendapatkan informasi yang bersifat keuangan, yang dibutuhkan suatu organisasi agar dapat beroperasi secara efisien, mengetahui hasil kerja selama ini, dan merencanakan kegiatan ke depannya agar lebih baik.

Akuntansi kadang-kadang disebut reaktif - bereaksi terhadap perkembangan bisnis dan teknologi. Menariknya, akuntansi berada di ujung tombak dari revolusi teknologi informasi pada awalnya. Akar akuntansi digital dapat ditelusuri ke era depresi dan Perang Dunia II. Peraturan pajak, pada saat itu, menjadi kompleks, dan pada Perang Dunia II, diperkenalkan berbagai masalah logistik dan manajemen data. (Ashutosh Deshmukh, 2006)

Menurut Samiaji Sarosa (2009), Sistem Informasi Akuntansi (SIA) adalah sebuah sistem yang mengumpulkan, mencatat, menyimpan, dan memproses data sehingga menghasilkan informasi yang berguna dalam membuat keputusan. SIA bisa berupa kertas dan alat tulis (manual) maupun terkomputerisasi penuh (serba otomatis) atau kondisi di antara keduanya (gabungan manual dan komputerisasi).

### Komponen-Komponen Sistem Informasi Akuntansi

Komponen-komponen yang menjadi bagian dari sistem informasi akuntansi antara lain:

1. Manajemen Data

Subsistem manajemen data memasukkan satu *database* yang berisi data yang relevan untuk suatu situasi dan dikelola oleh perangkat yang disebut Sistem Manajemen *Database* (*Database Management System*).

1. Manajemen Model

Merupakan paket perangkat lunak yang memasukkan model keuangan, statistik, ilmu manajemen, atau model kuantitatif lainnya yang memberikan kapasitas analitik dan manajemen perangkat lunak yang tepat. Perangkat lunak ini sering disebut Sistem Manajemen Basis Model (MBMS).

1. Antarmuka Pengguna (*User Interface*)

Pengguna berinteraksi dan memerintahkan Sistem Informasi Akuntansi melalui subsistem ini. Pengguna adalah bagian yang dipertimbangkan dari sistem.

1. Manajemen Basis Pengetahuan

Subsistem ini dapat mendukung semua subsistem lain atau bertindak sebagai suatu komponen independen. Subsistem ini dapat diinterkoneksikan dengan repositori pengetahuan perusahaan yang kadang-kadang disebut basis pengetahuan organisasional.

## Analisa dan Desain Sistem

### Analisa Sistem

Analisa sistem merupakan tahap melakukan investigasi awal terhadap sistem yang sudah berjalan. Pada tahapan ini, dilakukan kajian secara menyeluruh serta mendalam terhadap kegiatan sistem pengolahan data dan sistem saat ini dan juga pemahaman secara tepat mengenai bentuk sistem informasi yang bagaimana dikehendaki oleh manajemen.

Untuk mendapatkan hasil analisa yang komprehensif, langkah-langkah yang perlu dilakukan adalah melihat mengamati dokumen-dokumen sistem yang telah ada pada perusahaan, yang bisa tergambar dalam bentuk bagan alir sistem (*system flowchart*). Bagan alir sistem (*system flowchart*) merupakan seperangkat simbol yang menunjukkan suatu aktivitas tertentu yang berisi tentang aliran data yang berlangsung dalam kegiatan organisasi, hubungan dan keterkaitan serta pekerjaan antar bagian di dalam organisasi.

Simbol-simbol yang digunakan dalam bagan alir sistem adalah:

|  |  |  |
| --- | --- | --- |
| 1. |  | Simbol Terminal. Menunjukkan mulai atau berakhir-nya suatu sistem. |
| 2. |  | Simbol Dokumen. Menunjukkan dokumen input atau output baik untuk proses manual, mekanik atau komputer. |
| 3. |  | Simbol *Database*. Menunjukkan penyimpanan data ke *database*. |
| 4. |  | Simbol Proses. Menunjukkan proses dari operasi program komputer. |
| 5. |  | Simbol Storage. Penyimpanan input maupun output. |
| 6. |  | Simbol Arus. Menunjukkan arus proses dari suatu proses satu ke proses lain. |
| 7. |  | Simbol Proses Manual. Menunjukkan pekerjaan manual. |
| 8. |  | Simbol Penghubung. Menunjukkan penghubung ke halaman yang sama. |
| 9. |  | Simbol Keputusan Bersyarat (*Decision*). Simbol yang digunakan jika mengandung persyaratan ya atau tidak. |

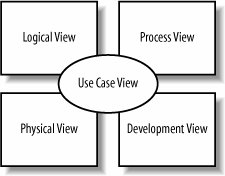
### Desain Sistem

Desain sistem bertujuan memberikan gambaran secara umum kepada *user* tentang sistem serta mengidentifikasi komponen-komponen sistem informasi yang dedesain secara rinci. Alat bantu yang digunakan dalam pembuatan desain sistem adalah:

#### *Unified Modeling Language* (UML)

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi.

Ada beberapa cara untuk memecah model diagram UML, salah satunya adalah Kruchten 4+1 *model view* (Kim Hamilton dan Russell Miles, 2006):



Gambar 2.1 Kruchten 4+1 *model view*

1. *Logical view*. Menjelaskan deskripsi abstrak dari bagian suatu sistem. Jenis-jenis diagram UML ini termasuk *class*, *object*, *state machine*, dan *interaction diagrams*.
2. *Process view*. Menjelaskan proses dalam sistem. Hal ini sangat berguna ketika memvisualisasikan apa yang harus terjadi dalam sistem. *View* ini biasanya berisi *activity diagram*.
3. *Development view*. Menjelaskan bagaimana bagian-bagian sistem diatur dalam modul dan komponen. Hal ini sangat berguna untuk mengelola lapisan dalam arsitektur sistem. *View* ini biasanya berisi *package* dan *component diagrams*.
4. *Physical view*. Menjelaskan bagaimana desain sistem, seperti yang dijelaskan dalam tiga *view* sebelumnya, kemudian dibawa hidup sebagai satu set entitas dunia nyata. *View* ini biasanya berisi *deployment diagram*.
5. *Use case view*. Menjelaskan fungsi sistem yang dimodelkan dari perspektif dunia luar, diperlukan untuk menjelaskan sistem apa yang seharusnya dilakukan. *View* ini biasanya berisi *use case diagram*, *description*, dan *overview diagram*.

#### *Entity Relationship Diagram* (ERD)

ERD merupakan sebuah model data, yaitu kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, makna data dan batasan data. ERD dibentuk oleh dua komponen utama yaiut *entity* dan relasi yang dideskripsikan lebih jauh melalui sejumlah atribut/properti. *Entitiy* merupakan individu yang merepresentasikan suatu objek yang nyata eksistensinya dan dapat dibedakan dari sesuatu yang lain. Setiap *entity* memiliki atribut yang mendeskripsikan karakteristik dari *entity* tersebut. Sedangkan relasi adalah hubungan yang terjadi antar *entity*.

Simbol-simbol yang digunakan pada ERD adalah:

|  |  |  |
| --- | --- | --- |
| 1. |  | Persegi. Menyatakan himpunan *entity*/entitas. |
| 2. |  | Belah Ketupat. Menyatakan himpunan relasi. |
| 3. |  | Garis. Sebagai penghubung antara himpunan relasi dengan *entity* dan himpunan entitas dengan atributnya |

## Basis Data

Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di simpanan luar komputer dan digunakan perangkat lunak tertentu untuk mengolah/ memanipulasinya. Prinsip utama dari sistem basis data ini adalah pengaturan data atau *file* dengan tujuan untuk memudahkan dan mempercepat dalam pengambilan kembali data atau arsip. Yang diunggulkan dalam basis data adalah pengaturan, pemeliharaan, pengelompokkan atau pengorganisasian data yang akan kita simpan sesuai fungsi atau jenisnya. Pembagian ini dapat berbentuk sejumlah *file* atau tabel terpisah atau dalam bentuk pendefinisian kolom-kolom data setiap *file* atau tabel.

Beberapa definisi basis data adalah sebagai berikut:

1. Sekumpulan data *store* (bisa dalam jumlah besar) yang tersimpan dalam *magnetic disc*, *optical disk*, dan media penyimpanan sekunder lainnya.
2. Sekumpulan program-program aplikasi umum yang sifatnya “*batch*” yang mengeksekusi dan memproses data secara umum (hapus, cari, *update*, dan lainnya).
3. Basis data terdiri dari data yang di-*share* bagi banyak *user* dan memungkinkan penggunaan data yang sama pada waktu bersamaan oleh banyak *user*.
4. Koleksi terpadu dari data-data yang saling berkaitan dari suatu *enterprise*. Misalkan basis data inventori akan terdiri dari data-data seperti data barang, jumlah stok, vendor dan data barang masuk dan barang keluar.

Konsep dasar basis data:

1. *Field*
   1. *Field* merupakan implementasi dari suatu atribut data.
   2. *Field* merupakan unit terkecil dari data yang berarti (*meaningful data*) yang disimpan dalam suatu *file* atau basis data.
2. *Record*
   1. *Field-field* tersebut diorganisasikan dalam *record-record*.
   2. *Record* merupakan koleksi dari *field-field* yang disusun dalam format yang telah ditentukan.
   3. Selama desain sistem, *record* akan diklasifikasikan sebagai *fixed-length record* atau *variable-length record*.
      1. *Fixed-length record*: tiap *instance record* punya *field,* jumlah *field*, dan ukuran logik yang sama.
      2. *Variable-length record*: mengijinkan *record-record* yang berbeda-beda dalam *file* yang sama memiliki panjang yang berbeda-beda.
3. *File* dan Tabel
   1. *Record-record* yang serupa diorganisasikan dalam grup-grup yang disebut *file*. Jadi *file* merupakan kumpulan semua kejadian dari struktur *record record* yang diberikan.
   2. Tabel merupakan ekivalen basis data relasional dari sebuah *file*.

## Perangkat Lunak

### *Web*

*World Wide Web* sering disingkat www atau *web* adalah suatu metode untuk menampilkan informasi di internet, baik berupa teks, gambar, suara maupun video yang interaktif dan mempunyai kelebihan untuk menghubungkan (*link*) satu dokumen dengan dokumen lainnya (*hypertext*) yang dapat diakses melalui sebuah *browser*. Informasi yang disajikan melalui *browser* dibangun dengan bahasa semi pemrograman HTML (*Hypertext Markup Language*), dan kemudian ditingkatkan fungsinya dengan menyisipkan kode-kode bahasa pemrograman *web*, seperti PHP (*Hypertext Preprocessor*), ASP (*Active Server Page*), JSP (*Java Server Page*) dan lain-lain, sehingga mampu menampilkan informasi yang lebih interaktif dan dinamis serta terhubung dengan *database*. (Yuhefizar, 2008)

### PHP (*Hypertext Preprocessor*)

PHP awalnya dikenal PHP dengan *Personal Home Page*. Saat ini, diartikan PHP *Hypertext Preprocessor*. PHP adalah sebuah bahasa pemrograman yang didesain untuk menghasilkan halaman *web* secara interaktif. Pada komputer yang melayaninya, dinamakan *server web*. Tidak seperti HTML (*Hypertext Markup Language*), yang mana browser *web* menggunakan *tag* dan *markup* untuk menghasilkan sebuah halaman, kode PHP dijalankan diantara halaman yang di-*request* dan *server web*, menambahkan dan mengubah *output* HTML dasar. Contohnya, kode PHP dapat digunakan untuk menampilkan sebuah penghitung dari pengunjung.

Semua kode PHP diawali dengan tag <?php dan diakhiri ?>. Bisa juga digunakan tag <? ... ?> atau tag ASP <% ... %>. Tetapi yang paling disarankan adalah tag <?php ... ?>. Tag penutup PHP juga dapat dihilangkan, seperti yang disarankan oleh Zend Framework.

### Zend Framework

Zend *Framework* adalah aplikasi *web* framework berorientasi objek sumber terbuka yang diimplementasikan dalam PHP. Zend *Framework* didesain dan dibangun untuk meningkatkan produktivitas pengembang. Tidak seperti *framework* lainnya yang memerlukan file konfigurasi untuk dapat berjalan, kebanyakan aspek dari aplikasi Zend *Framework* dapat didefinisikan pada *runtime* menggunakan *command* PHP sederhana. Ini menyelamatkan waktu pengembang karena daripada file konfigurasi yang kompleks mengontrol semua aspek dari aplikasi, kita hanya mengkonfigurasi bagian yang berbeda dari yang normal.

Perbedaan Zend *Framework* dari banyak basis kode PHP lain adalah implementasi dari *model-view-controller* (MVC). MVC adalah sebuah pola yang aplikasinya dibagi menjadi 3 bagian:

1. Logika bisnis: *Controller* menangani logika dibelakang aplikasi.
2. Data: *Model* menangani pengelolaan data aplikasi.
3. Presentasi: *View* menangani dalam menampilkan halaman dinamis.

### Ajax

Ajax adalah singkatan dari *Asynchronous JavaScript and XML*. Elemen kunci disini adalah *Asynchronous*. Secara sederhana, Ajax menawarkan sebuah teknik untuk memanggil *server* pada *background* via JavaScript dan memperoleh kembali data tambahan seperlunya, meng-*update* sebagian halaman tanpa meyebabkan halaman dimuat ulang secara keseluruhan.

### JQuery

JQuery adalah pustaka JavaScript yang cepat dan ringkas yang menyederhanakan pen-*traverse*-an, penanganan *event*, animasi, dan interaksi Ajax untuk pengembangan *web* yang cepat. JQuery dirancang untuk mengubah cara kita menulis JavaScript.

Framework JavaScript JQuery telah menjadi *rising star* di dunia pengembangan web. Misinya sebagai pustaka JavaScript sederhana - berusaha untuk membuat kehidupan pengembang web lebih mudah dengan *patch* pada bagian-bagian tertentu dari perkembangan lintas-browser dan dengan membuat tugas-tugas lain yang biasanya diperlukan jauh lebih lama oleh pengembang.

### MySQL

MySQL adalah sebuah *full-featured relational database* yang bebas. MySQL telah dikembangkan pada tahun 1990an untuk memenuhi kebutuhan perkembangan komputer untuk mengatur informasi secara cerdas. Inti asli dari pengembangan MySQL yaitu mencoba untuk memecahkan masalah dari kebutuhan mereka untuk *database* dengan menggunakan mSQL, sebuah *database* yang kecil dan sederhana. Itu menjadi jelas bahwa mSQL tidak dapat memecahkan semua masalah yang mereka inginkan, jadi mereka membuat *database* yang lebih kuat yang menjadi MySQL. (Michele Davis and Jon Phillips, 2007).

MySQL memiliki beberapa keistimewaan, antara lain:

1. Portabilitas. MySQL dapat berjalan stabil pada berbagai sistem operasi.
2. *Open Source*. MySQL didistribusikan secara bebas, dibawah lisensi GPL (*General Public Licencese*) sehingga dapat digunakan secara cuma-cuma.
3. *Multiuser*. MySQL dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. *Performance tuning*. MySQL memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. Jenis Kolom. MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed*/*unsigned* integer, float, double, char, text, date, timestamp, dan lain-lain.
6. Perintah dan Fungsi. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah SELECT dan WHERE dalam perintah (query).
7. Keamanan. MySQL memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, dan izin akses user dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. Skalabilitas dan Pembatasan. MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (records) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. Konektivitas. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix soket (UNIX), atau *Named Pipes* (NT).
10. Lokalisasi. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa.
11. Antar Muka. MySQL memiliki interface (antar muka) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
12. Klien dan Peralatan. MySQL dilengkapi dengan berbagai peralatan (*tools*) yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk online.
13. Struktur tabel. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE.

# BAB III GAMBARAN UMUM PERUSAHAAN

# 

PT. Bumi Sentosa yang beralamat di Jalan Suli No. 93 Denpasar merupakan perusahaan yang berisikan personil-personil yang dinamis, enerjik dan meiliki bakat enterpreneur yang tinggi dengan pengalaman pengelolaan dan manajemen outsourcing yang sukses. Perusahaan ini memiliki visi dan misi menjadi perusahaan skala nasional yang memberikan pelayanan yang lebih baik bagi setiap pelanggan.

## Proyek

Adapun proyek yang sedang ditangani yaitu:

### Penyedia Tenaga Kerja

Merupakan produk layanan yang menyediakan kebutuhan tenaga kerja bagi perusahaan-perusahaan atau pelanggan sehingga perusahaan atau pelanggan yang membutuhkan tenaga kerja tersebut dapat fokus pada pengembangan bisnis masing-masing, dan kegiatan yang tidak bersifat esensial dapat diserahkan kepada Bumi Sentosa dan Bumi Sentosa akan mengelolanya dengan lebih baik.

### Instalasi dan Pemeliharaan Jaringan Listrik

Merupakan jenis kegiatan perusahaan yang melayani pekerjaan instalasi dan pemeliharaan jaringan listrik.

### Konsultasi Teknologi Informasi

Teknologi Informasi merupakan gabungan antara Teknologi komputer dan Teknologi komunikasi. Teknologi komputer diantaranya adalah perangkat keras (*hardware*) dan perangkat lunak (*software*) sedangkan Teknologi Komunikasi adalah proses penyampaian informasi jarak jauh. Sehingga diharapkan dengan adanya Teknologi Informasi proses pengelolaan dan penyampaian informasi lebih mudah dan cepat untuk dilakukan.

### Pengadaan Barang

Merupakan jenis kegiatan perusahaan yang melayani permintaan pengadaan perangkat komputer beserta perangkat pendukungnya dan pengadaan material jaringan listrik.

Disamping pengadaan, perusahaan ini juga melayani penyewaan perangkat komputer. Dalam kegiatan penyewaan, perusahaan ini sekaligus memberikan pemeliharaan dan garansi terhadap perangkat yang disewakan sehingga perangkat tersebut selalu dapat dioperasionalkan oleh *user* dengan optimal dan merupakan komitmen perusahaan untuk mengganti perangkat tersebut dengan yang baru jika mengalami kerusakan.

### Pelayanan Teknik

Merupakan pekerjaan penyediaan jasa tenaga kerja di bidang teknik listrik dan teknik komputer. Lingkup pekerjaan meliputi penanganan gangguan listrik, inspeksi jaringan dan pemeliharaan. Pekerjaan ini melakukan penanganan gangguan yang personilnya ditempatkan di masing-masing unit PLN (Perusahaan Listrik Negara), melakukan inspeksi secara rutin berdasarkan jenis komponen kelistrikannya seperti JTM (Jaringan Tegangan Menengah), JTR (Jaringan Tegangan Rendah) dan Gardu Distribusi, melakukan pemeliharaan jaringan listrik beserta pekerjaan pemangkasan dahan pohon.

### Pemeliharaan AMR (*Automatic Meter Reading*)

Merupakan pekerjaan penyediaan jasa tenaga kerja di bidang teknik listrik dan teknik komputer. Lingkup pekerjaan meliputi pemeliharaan komponen dan komunikasi AMR. Pekerjaan ini melakukan pemeliharaan rutin terhadap komponen dan jaringan komunikasi data AMR serta menangani dan memperbaiki seandainya ada gangguan pada komponen maupun jaringan komunikasi data AMR. Sehingga komponen AMR selalu bekerja dengan baik dan komunikasi data tetap terjaga dan sistem AMR tersebut berjalan dengan baik pula.

### PPTL (Penertiban Pemakaian Tenaga Listrik)

Merupakan pekerjaan penyediaan jasa tenaga kerja di bidang teknik listrik dan teknik komputer. Lingkup pekerjaan meliputi pemeriksaan kepada pelanggan terhadap kemungkinan-kemungkinan terjadinya pencurian atau penyalahgunaan tenaga listrik. Pekerjaan ini melakukan pemeriksaan kepada pelanggan atas dasar target operasi atau rutin yang dilakukaan oleh petugas yang telah berpengalaman dalam memeriksa, mengetahui modus operandi serta melakukan eksekusi dilapangan sesuai dengan aturan yang berlaku.

### Survey Pelanggan

Merupakan pekerjaan penyediaan jasa tenaga kerja di bidang survey dan teknik komputer. Pekerjaan ini meliputi survey pelanggan dan pekerjaan penggambaran atau digitasi peta hasil survey pada komputer dengan menggunakan *software* GIS (*Geographics Information System*). Pekerjaan ini selain menghasilkan data survey juga menghasilkan gambar berupa digitasi peta GIS. Sehingga data GIS yang dimiliki PLN akan selalu ter-*update* dan baik pelanggan maupun PLN akan senantiasa dapat mengetahui posisi geografisnya.

### Pengembangan dan *Updating* GIS (*Geographics Infor-mation System*)

Merupakan pekerjaan penyediaan jasa tenaga kerja di bidang survey dan teknik komputer. Lingkup pekerjaan meliputi survey penyambungan pelanggan dan pekerjaan penggambaran atau digitasi peta hasil survey penyambungan atau perluasan pada komputer dengan menggunakan software GIS. Pekerjaan ini melakukan updating data GIS berdasarkan kegiatan penyambungan pelanggan, perluasan jaringan yang dilakukan oleh PLN. Sehingga data pelanggan dan jaringan tersambung selalu *up to date*.

### Penyediaan Tenaga Administrasi

Merupakan penyediaan tenaga kerja di bidang:

1. Administrasi, yaitu tenaga kerja dengan kemampuan administrasi dan penguasaan komputer untuk aplikasi perkantoran (*office*) yang lengkap dan terampil serta dididik memiliki kepribadian yang bagus dan *communication skill* yang baik terutama pada posisi sebagai *front liner*.
2. Pengemudi, yaitu tenaga kerja dengan kemampuan mengemudikan kendaraan dengan ijin yang lengkap serta teampil dalam bekerja.
3. Satpam, yaitu tenaga kerja dengan kemampuan sebagai tenaga keamanan dan memiliki sertifikat dan keanggotaan dari pihak kepolisian setempat, serta dididik memiliki kepribadian yang bagus, jujur dan memberikan rasa keamanan yang tinggi kepada pelanggan.
4. *Cleaning service*, yaitu tenaga kerja dengan kemampuan sebagai tenaga kebersihan baik pada kebersihan di ruangan maupun outdoor dengan cara kerja dan kemampuan penggunaan peralatan yang baik dan tepat.

## Struktur Organisasi

Struktur organisasi PT. Bumi Sentosa sebagai berikut:

**Direktur**

**Manajer   
Marketing**

**Manajer Operasional**

**Manajer**

**Internal**

Admin Proyek

Pelaksana Operasional

Administrasi & Keuangan

Kepegawaian

Customer Service

Technical Support

Umum

Gambar 3.1 Struktur Organisasi PT. Bumi Sentosa

## *Job Description*

*Job Description* berisi uraian tugas dari masing-masing bagian sebagai berikut:

### Direktur

Tugas dan wewenang direktur yaitu: merumuskan strategi perusahaan; menetapkan metode, teknologi, sumber daya untuk mencapai strategi; menyiapkan rencana dan program jangka panjang; mengadakan pengawasan dan evaluasi terhadap jalannya program dan rencana yang telah digariskan.

### Marketing

Tugas dan wewenang manajer marketing yaitu: mengikuti dan memenangkan tender; mempromosikan perusahaan ke pihak luar. Tugas dan wewenang admin proyek yaitu: mengurus administrasi-administrasi pada proyek yang ada; melakukan penagihan ke penyedia proyek.

### Operasional

Tugas dan wewenang manajer operasional yaitu: mengurus proyek yang telah berjalan; mengawasi pekerjaan di lapangan. Tugas dan wewenang pelaksana operasional yaitu: melaksanakan pekerjaan-pekerjaan operasional perusahaan.

### Internal

Tugas dan wewenang manajer internal yaitu: mengerjakan hal-hal pada internal perusahaan dan bertanggung jawab atas pekerjaan bawahannya.

Tugas dan wewenang administrasi dan keuangan yaitu: merencanakan dan menganalisa pembelanjaan perusahaan; mengatur struktur aktiva (struktur kekayaan perusahaan); mengatur struktur finansial; mengatur struktur modal. Tugas lainnya adalah menyediakan Laporan keuangan (Neraca, Laporan Rugi/Laba).

Tugas dan wewenang kepegawaian yaitu: mengendalikan dan menyelenggarakan kegiatan dibidang administrasi kepegawaian; melaksanakan proses kegiatan penggajian, kenaikan pangkat, kenaikan berkala, mutasi, kesejahteraan pegawai dan pembinaan pegawai; mengurus proses Askes, Astek, Taspen dan proses pegawai yang telah mencapai usia pensiun dan penghargaan.

Tugas dan wewenang *customer service* yaitu: menerima telephone; melayani tamu yang datang.

Tugas dan wewenang *technical support* yaitu: sebagai tenaga *support* teknis seperti merawat dan menangani masalah-masalah pada program aplikasi khusus perusahaan, sistem operasi, menangani virus maupun masalah aplikasi perkantoran, serta masalah-masalah *hardware* yang terjadi.

Tugas dan wewenang umum yaitu: menyelenggarakan kegiatan dibidang kerumah-tanggaan; mengadakan pembelian barang-barang yang diperlukan perusahaan.

# BAB IV ANALISA DAN PERANCANGAN SISTEM

# 

## Mekanisme Sistem Yang Sedang Berjalan

Bagian keuangan menerima bukti nota pembelian atau bukti transaksi keuangan dan mencatatnya. Data kemudian dicek apakah sudah benar atau tidak? Jika masih ada kesalahan maka data di-*edit* kembali. Pada akhir periode akuntansi dilakukan penginputan jurnal penyesuaian untuk menyesuaikan saldo rekening-rekening ke saldo yang sebenarnya, atau untuk memisahkan antara pendapatan dan beban dari suatu periode dengan periode yang lain.

Setiap akhir bulan bagian keuangan membuat laporan akuntansi (jurnal umum, buku besar, laba/rugi, neraca) dan menyerahkannya ke bagian internal.

### *Use Case* Diagram



Gambar 4.1 *Use Case* Diagram yang Sedang Berjalan

### *Activity* Diagram



Gambar 4.2 *Activity* Diagram yang Sedanga Berjalan

## Mekanisme Sistem Yang Diusulkan

Mekanisme sistem yang diusulkan hampir sama dengan mekanisme sistem yang sedang berjalan, bedanya sistem yang diusulkan berbasis web.

### *Use Case* Diagram



Gambar 4.3 *Use Case* Diagram yang Diusulkan

### *Activity* Diagram



Gambar 4.4 *Activity* Diagram yang Diusulkan

### *Class* Diagram



Gambar 4.5 *Class* Diagram *(Model)*



Gambar 4.6 *Class* Diagram (*Controller*)



Gambar 4.7 *Class* Diagram *(Form)*

### *Sequence* Diagram



Gambar 4.8 *Sequence* Diagram (Login)



Gambar 4.9 *Sequence* Diagram (Daftar Akun)



Gambar 4.10 *Sequence* Diagram (Tambah Akun)



Gambar 4.11 *Sequence* Diagram (Tambah Jurnal)



Gambar 4.12 *Sequence* Diagram (Laporan Akun)



Gambar 4.13 *Sequence* Diagram (Laporan Jurnal Umum)



Gambar 4.14 *Sequence* Diagram (Laporan Buku Besar)



Gambar 4.15 *Sequence* Diagram (Laporan Laba/Rugi)



Gambar 4.16 *Sequence* Diagram (Laporan Neraca)



Gambar 4.17 *Sequence* Diagram (Transaksi Periode)

## *Entity Relationship Diagram* (ERD)

ERD merupakan sebuah model data, yaitu kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, makna data dan batasan data. ERD dibentuk oleh dua komponen utama yakni *entity* dan relasi yang dideskripsikan lebih jauh melalui sejumlah atribut/ properti. *Entity* merupakan individu yang merepresentasikan suatu obyek yang nyata eksistensinya dan dapat dibedakan dari sesuatu yang lain. Setiap *entity* memiliki atribut yang mendeskripsikan karakteristik dari entity tersebut. Sedangkan Relasi adalah hubungan yang terjadi antar *entity*.

Berikut gambar *entity relationship diagram* pada sistem informasi akuntansi:



Gambar 4.18 *Entity Relationship Diagram*

## Tabel Relasi

Berdasarkan desain struktur database tersebut diatas, selanjutnya akan dibuatkan sebuah table relasi yang akan menghubungkan tabel yang satu dengan table yang lainnya. Relasi antar tabel dalam *database* dibuat agar tidak terjadi kerangkapan dan ketidakkonsistenan data sehingga informasi yang diberikan oleh sistem tidak salah. Berikut gambar dari tabel relasinya:



Gambar 4.19 Tabel Relasi

## Struktur *Database*

Pada tahap ini akan dilakukan pembuatan struktur tabel beserta tipe datanya dengan menggunakan MySQL 5.1. Pembuatan struktur database merupakan langkah awal dalam pembuatan program aplikasi. Adapun tabel-tabel dan tipe datanya adalah sebagai berikut:

1. Nama Tabel: **users**

Fungsi: Untuk memanipulasi data-data pengguna aplikasi.

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Field** | **Type** | **Description** |
| PK | id | int(11) | id dari tabel ini |
|  | username | varchar(25) | nama user untuk login |
|  | displayNama | varchar(50) | nama user yang ditampilkan |
|  | password | varchar(50) | password lgon |
|  | role | varchar(25) | peran/tugas user |

Tabel 4.1 Tabel User

1. Nama Tabel: **account**

Fungsi: Untuk memanipulasi data-data account.

Pada tabel account ini, data akan ditampilkan dalam bentuk hirarki. Algoritma yang digunakan dalam pembentukan *tree* ini adalah *modified preorder tree traversal algorithm* yang memerlukan dua field tambahan yaitu lft dan rgt.

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Field** | **Type** | **Description** |
| PK | id | int(11) | id dari tabel ini |
|  | lft | int(11) | penomoran sebelah kiri |
|  | rgt | int(11) | penomoran sebelah kanan |
|  | kodeAccount | varchar(20) | kode account |
|  | account | varchar(100) | nama account |
|  | normalPos | tinyint(4) | normal pos (debit/kredit) |
|  | kelompok | varchar(1) | kelompok (neraca/laba-rugi) |

Tabel 4.2 Tabel Account

1. Nama Tabel: **bukti\_transaksi**

Fungsi: Untuk memanipulasi data-data bukti transaksi.

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Field** | **Type** | **Description** |
| PK | id | int(11) | id dari tabel ini |
|  | noBukti | varchar(30) | nomor bukti |
|  | tipeJournal | tinyint(4) | tipe journal (journal umum/journal penyesuaian) |
|  | tanggal | int(11) | tanggal transaksi (*timestamp*) |
|  | keterangan | varchar(125) | keterangan |
|  | validatedBy | int(11) | yang memvalidasi |
|  | validatedDate | int(11) | tanggal divalidasi |

Tabel 4.3 Tabel Bukti Transaksi

1. Nama Tabel: **journal**

Fungsi: Untuk memanipulasi data-data journal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Field** | **Type** | **Description** |
| PK | id | int(11) | id dari tabel ini |
| FK1 | idBukti | int(11) | id tabel bukti transaksi |
| FK2 | idAccount | int(11) | id tabel account |
|  | debit | decimal(12,2) | jumlah debit |
|  | kredit | decimal(12,2) | jumlah kredit |

Tabel 4.4 Tabel Journal

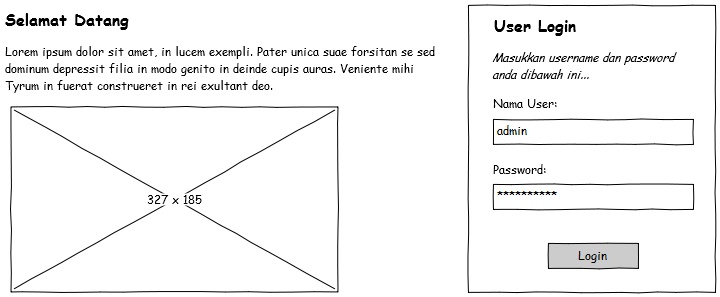
## Perancangan Sistem

Rancangan sistem terdiri dari dua bagian yaitu perancangan input yang merupakan perancangan form pengisian data dan perancangan output yang merupakan tampilan data dan laporan. Dalam rancangan ini menampilkan fasilitas atau komponen yang akan digunakan dalam pembuatan program sebagai berikut:

1. <table>: Untuk menampilkan data dalam tabel.
2. Zend\_Form (*render*: <form>): Untuk menampung semua elemen data di dalam form yang akan di-*submit* ke server.
3. Zend\_Form\_Element\_Hidden (*render:* <input type=”hidden”>): Sebagai penampung data yang tidak perlu ditampilkan, seperti id dari tabel.
4. Zend\_Form\_Element\_Text (*render*: <input type=”text”>): Untuk inputan data teks.
5. Zend\_Form\_Element\_Textarea (*render*: <textarea>): Sama seperti Zend\_Form\_Element\_Text tetapi elemen ini dapat terdiri dari banyak baris.
6. Zend\_Form\_Element\_Password (*render*: <input type=”password”>): Khusus untuk inputan password.
7. Zend\_Form\_Element\_Select (*render*: <select>): Untuk menampilkan pilihan data.
8. Zend\_Form\_Element\_Submit (*render*: <input type=”submit”>): Tombol untuk *submit* ke *server*.
9. ZendX\_Jquery\_Form\_Element\_Datepicker (*render*: <input type= ”text”> dengan tambahan kode javascript): Untuk menampilkan pilihan tanggal.

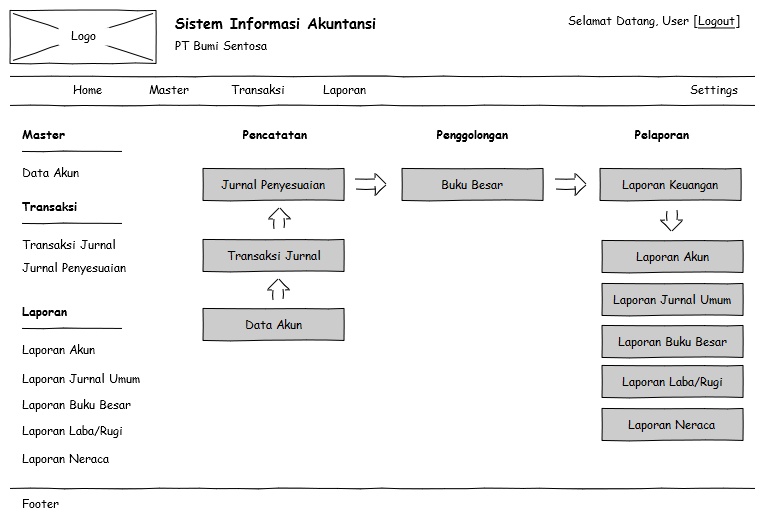
### Perancangan Input

#### Rancangan Login



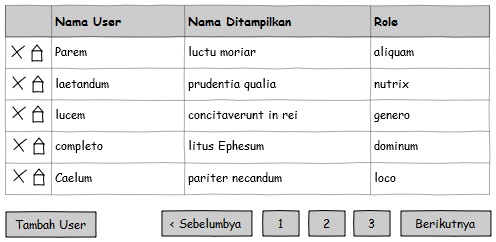
Gambar 4.20 Rancangan Form *Login*

#### Rancangan Menu Utama

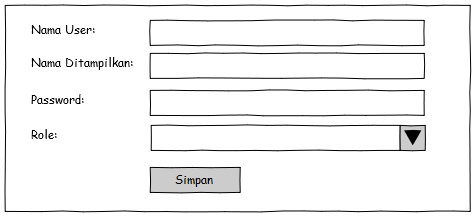


Gambar 4.21 Rancangan Menu Utama

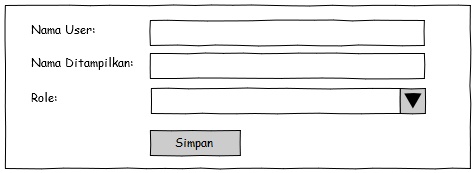
#### Rancangan Pengolahan Data *User*



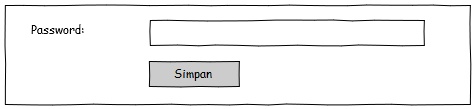
Gambar 4.22 Rancangan Daftar *User*



Gambar 4.23 Rancangan Form Tambah Data *User*

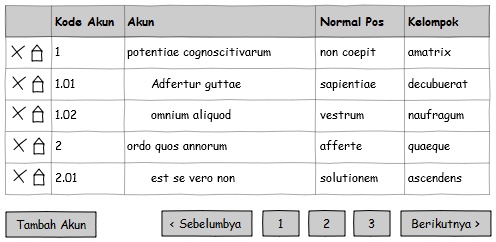


Gambar 4.24 Rancangan Form Ubah Data *User*

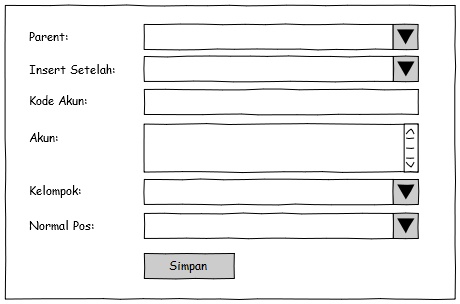


Gambar 4.25 Rancangan Form Ubah *Password*

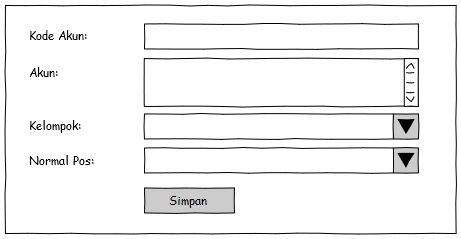
#### Rancangan Pengolahan Data Akun



Gambar 4.26 Rancangan Daftar Akun

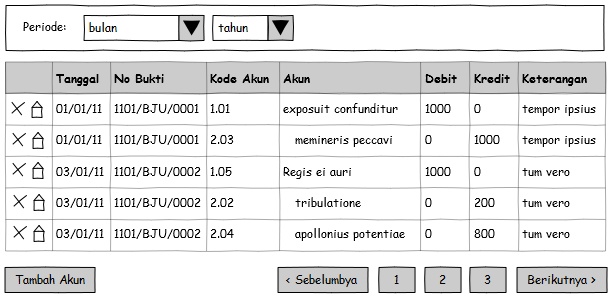


Gambar 4.27 Rancangan Form Tambah Data Akun

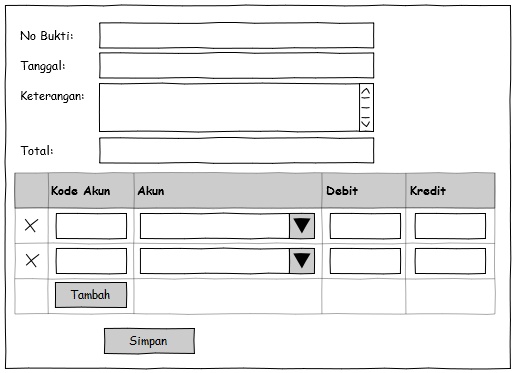


Gambar 4.28 Rancangan Form Ubah Data Akun

#### Rancangan Pengolahan Transaksi Jurnal

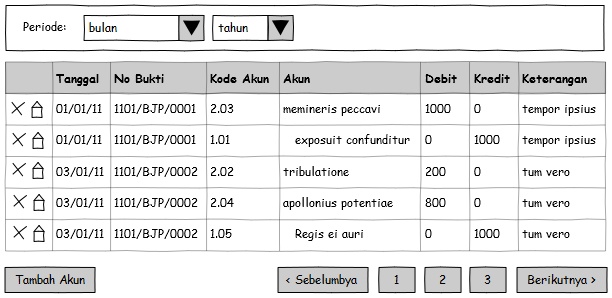


Gambar 4.29 Rancangan Daftar Data Jurnal

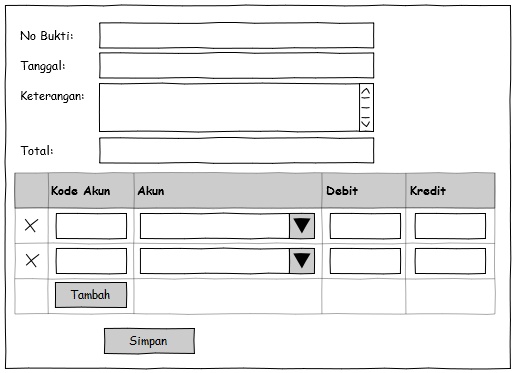


Gambar 4.30 Rancangan Tambah/Ubah Data Jurnal

#### Rancangan Pengolahan Jurnal Penyesuaian



Gambar 4.31 Rancangan Daftar Data Jurnal Penyesuaian



Gambar 4.32 Rancangan Tambah/Ubah Data Jurnal Penyesuaian

### Perancangan Output

#### Rancangan Laporan Akun



Gambar 4.33 Rancangan Laporan Data Akun

#### Rancangan Laporan Jurnal Umum



Gambar 4.34 Rancangan Laporan Jurnal Umum

#### Rancangan Laporan Buku Besar



Gambar 4.35 Rancangan Laporan Buku Besar

#### Rancangan Laporan Laba/Rugi



Gambar 4.36 Rancangan Laporan Laba/Rugi

#### Rancangan Laporan Neraca



Gambar 4.37 Rancangan Laporan Neraca

# BAB V IMPLEMENTASI SISTEM

# 

Setelah dilakukan analisis dan perancangan terhadap sistem, maka sistem tersebut siap diimplementasikan. Tahap implementasi terhadap sistem ini merupakan tahap meletakkan rancangan sistem ke dalam bentuk *coding* bahasa pemrograman.

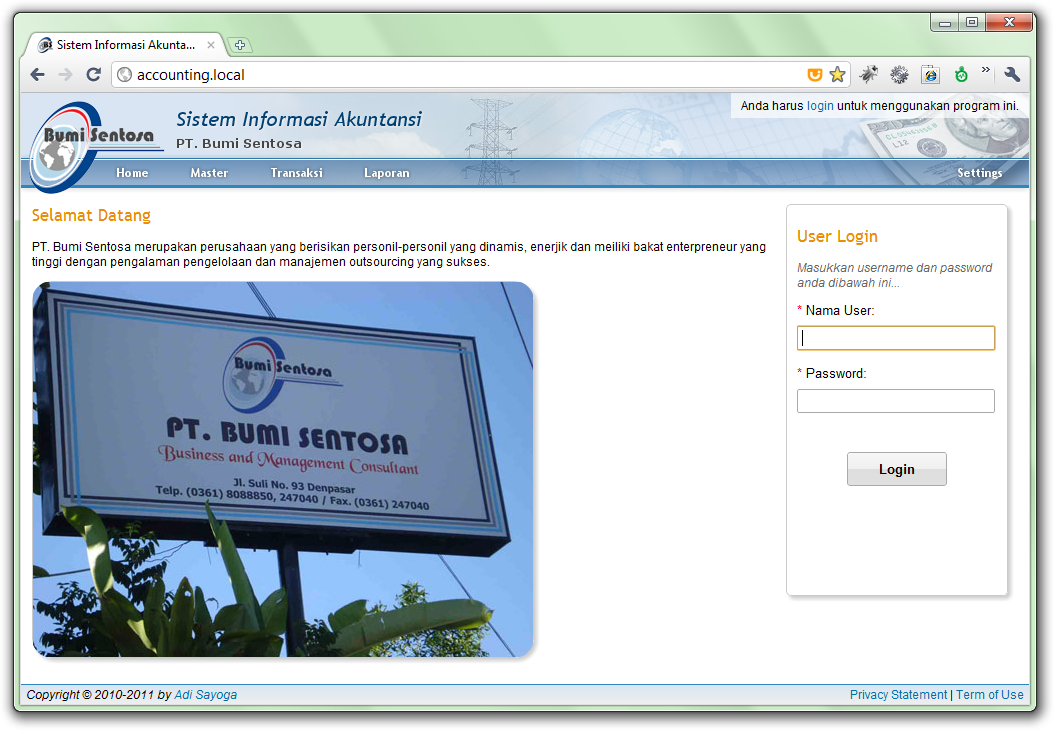
## Komponen Sistem

Komponen sistem berisi tentang deskripsi dan cara kerja tiap modul pada Sistem Informasi Akuntansi.

### Login *User*

|  |  |
| --- | --- |
| **Fungsi** | Login ke sistem |
| **Antarmuka Tampilan** | Tampilan form login |
| **Pemrosesan** | *User* memasukkan nama *user* dan *password* kemudian sistem akan memverifikasi apakah terdaftar dan *password*-nya sesuai, jika ya maka akan dialihkan ke menu utama. |

Pada saat *user* mulai menggunakan aplikasi, *user* harus memasukkan nama *user* dan *password*. Fungsi dari fasilitas ini adalah membatasi *user* supaya tidak sembarang orang bisa mengakses dan menggunakan aplikasi. Masing-masing *user* yang sudah terdaftar akan memiliki hak akses terhadap program yang berbeda-beda sesuai dengan *role* yang diberikan.

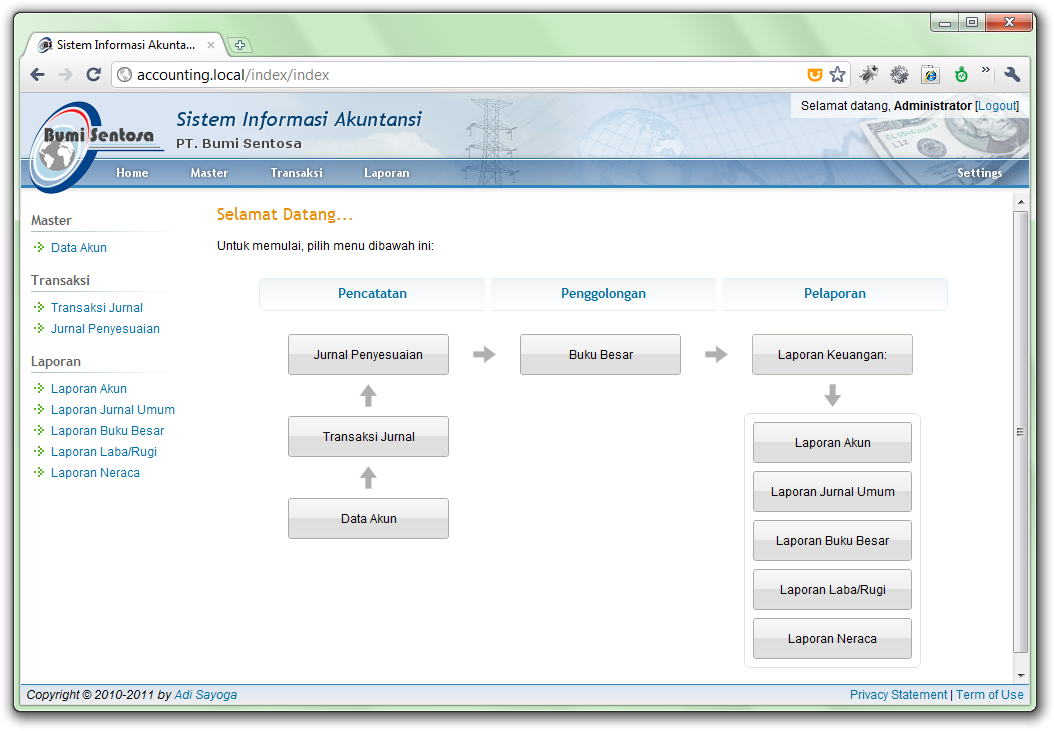


Gambar 5.1 Implementasi Form Login

### Menu Utama

|  |  |
| --- | --- |
| **Fungsi** | Menampilkan menu utama |
| **Antarmuka Tampilan** | Tampilan menu utama |

Menu utama muncul setelah melewati proses login. Pada menu ini dibuat tampilan grafis, sehingga *user* dapat mengakses informasi secara cepat. Disini terdapat tiga bagian yaitu pencatatan, penggolongan dan pelaporan.

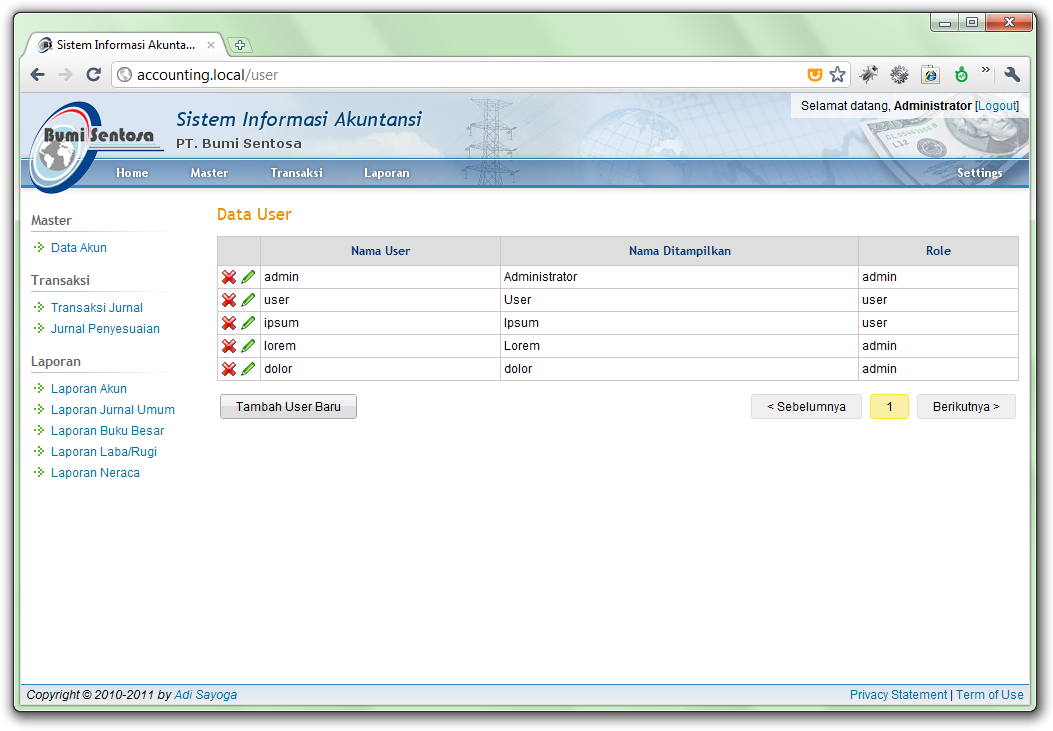


Gambar 5.2 Implementasi Menu Utama

### Pengolahan Data *User*

|  |  |
| --- | --- |
| **Fungsi** | Mengolah data *user* |
| **Antarmuka Tampilan** | Tampilan masukan data *user* |
| **Pemrosesan** | Penambahan, pengubahan, dan penghapusan data *user* |

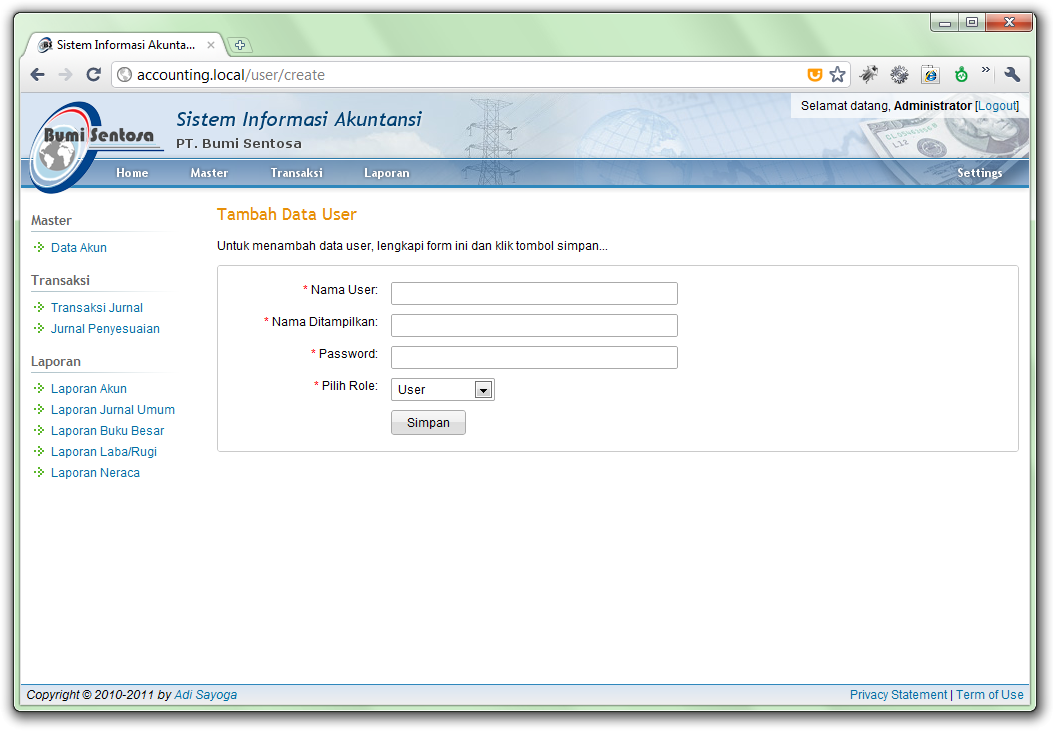
Halaman yang digunakan untuk mengelola data *user*. Data *user* ini digunakan pada saat kita login ke aplikasi. Data *user* ini terdiri dari *field* nama *user*, nama yang ditampilkan, *password* dan *role*.



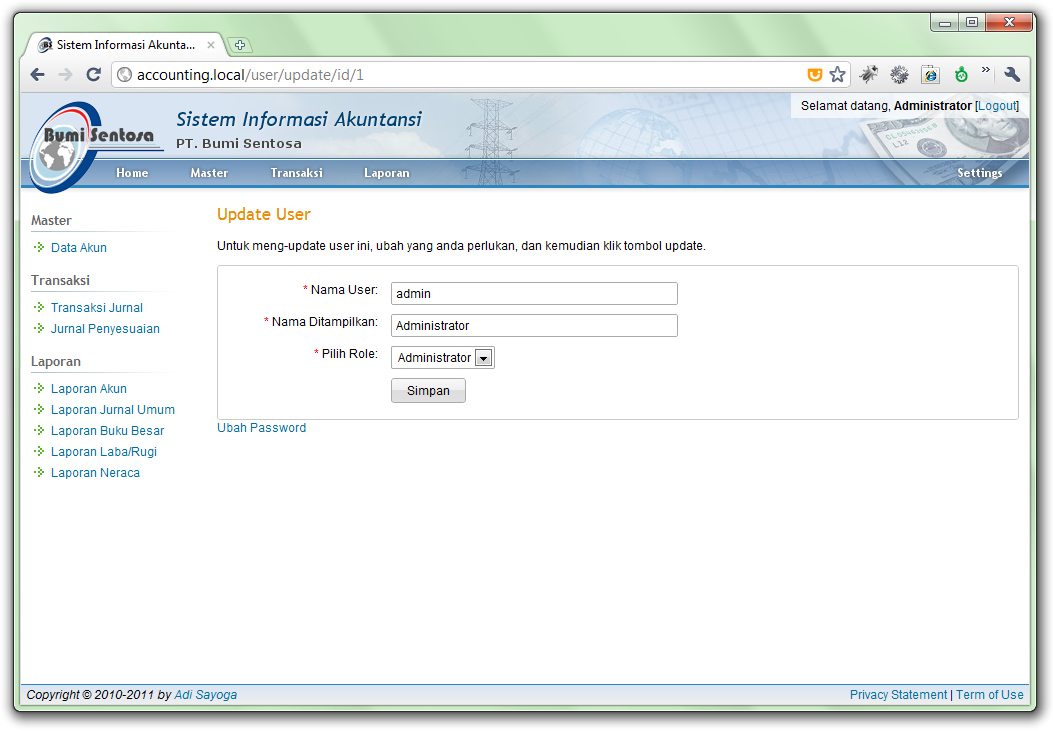
Gambar 5.3 Implementasi Daftar User

Seperti dijelaskan pada bagian login diatas, *user* memiliki hak akses terhadap program yang berbeda-beda sesuai dengan *role* yang diberikan. Pengaturan role ini menggunakan *plugin* Zend\_Acl yaitu pada *library* Acc\_Controller\_Plugin\_Acl. *Front controller* menggunakan sistem *plug-in* untuk dapat menambah fungsionalitas kustom tanpa mengubah inti dari *libary* Zend itu sendiri.

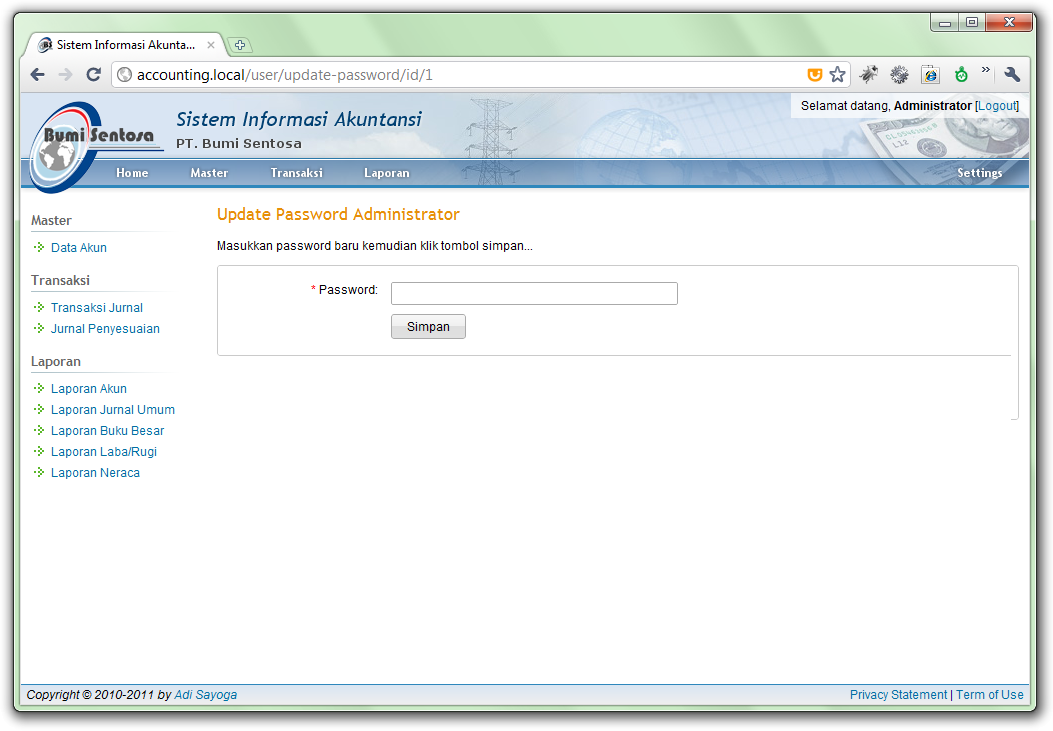
*Role* terdiri dari tiga yaitu *guest*, *user* dan *admin*. *Guest* hanya dapat mengakses halaman login. *user* dapat mengakses data akun, jurnal, dan laporan dan tidak dapat mengakses halaman user ini. sedangkan *admin* mempunyai hak akses penuh.



Gambar 5.4 Implementasi Form Tambah Data User



Gambar 5.5 Implementasi Form Ubah Data User



Gambar 5.6 Implementasi Form Ubah Password User

### Pengolahan Data Akun

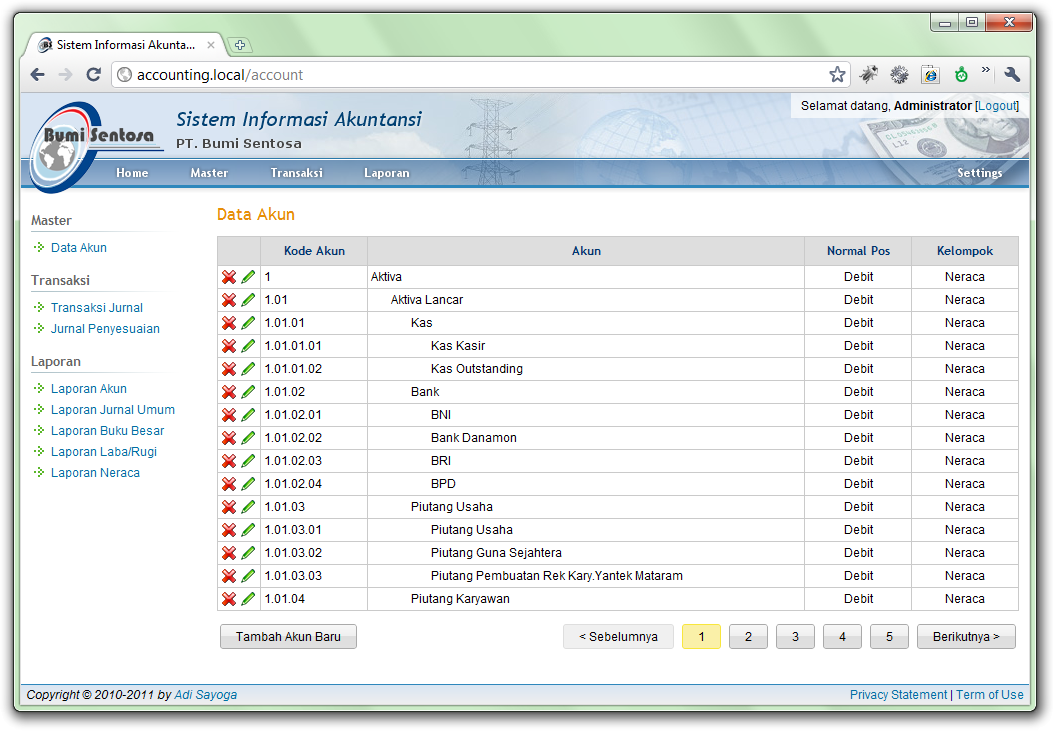
|  |  |
| --- | --- |
| **Fungsi** | Mengolah data akun |
| **Antarmuka Tampilan** | Tampilan masukan data akun |
| **Pemrosesan** | Penambahan, pengubahan, dan penghapusan data akun |

Terdapat enam bagian (*root*) yaitu aktiva, pasiva, modal, pendapatan, biaya, dan pendapatan dan biaya non operasional. Pendapatan dan biaya non operasional dibagi lagi menjadi dua yaitu pendapatan non operasional dan biaya non operasional. Setiap akun memiliki pos normalnya (debit atau kredit) dan pada kelompoknya (neraca atau laba/rugi).

Data akun ditampilkan berhirarki yaitu mempunyai relasi *parent-child* yang setiap *item* memiliki satu *parent* dan kosong atau lebih anak (kecuali *root* yang tidak memiliki *parent*). Pada *database* relasi seperti MySQL tidak dijumpai adanya data berhirarki tetapi hanya *list* datar. Untuk mendapatkan data hirarki, terdapat beberapa model yang digunakan seperti *adjacency list model* dan *nested set model*.

*Adjacency list model* sangat sederhana dan mudah dipahami, yaitu setiap *item* memiliki *pointer* ke orang tuanya, tetapi menjadi sangat lambat jika datanya banyak karena menggunakan fungsi rekursi saat mengambil data. Model ini tidak cocok diterapkan untuk data akun.

Pendekatan lain, *nested set model* (yang digunakan pada aplikasi ini) yaitu dengan menambahkan penomoran mulai dari sisi sebelah kiri *node* dan dilanjutkan dengan di sebelah kanan (*field* lft dan rgt). Pendekatan ini dinamakan *modified preorder tree traversal algorithm*. Dengan pendekatan ini memungkinkan mendapatkan data hirarki langsung melalui SQL murni dan mendapatkan data lebih cepat karena tidak adanya fungsi rekursi.

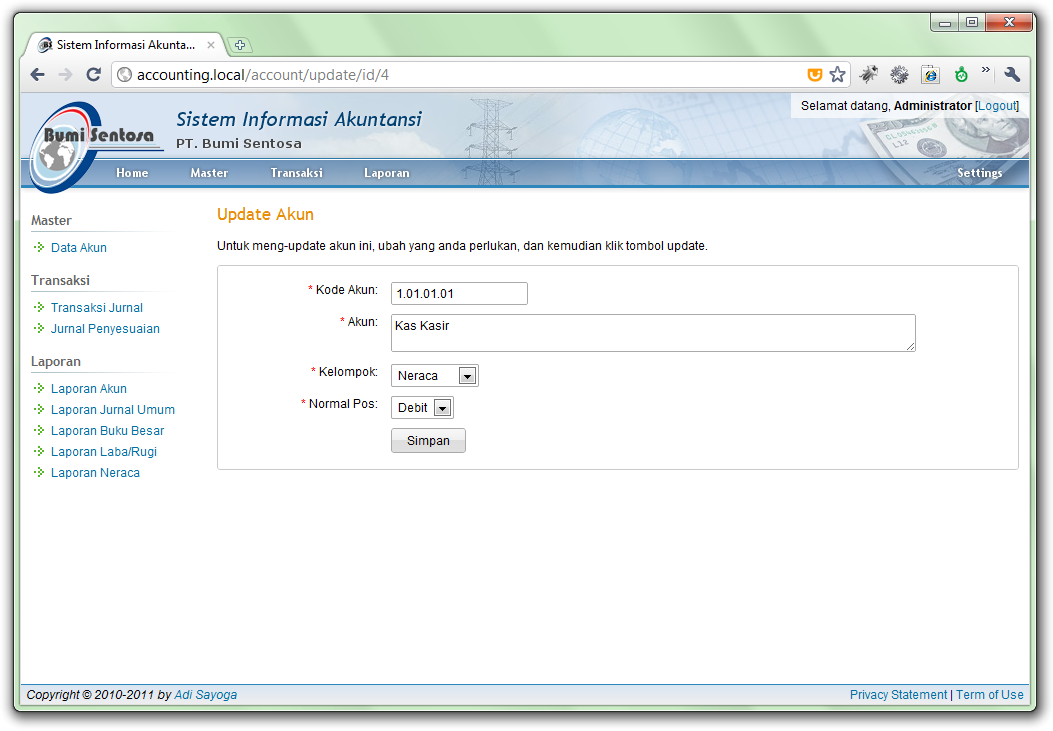


Gambar 5.7 Implementasi Daftar Akun



Gambar 5.8 Implementasi Form Tambah Data Akun

Dalam menambah data akun terdapat dua *field* yang perlu mendapat perhatian, “*parent*” dan “*insert* setelah”, yaitu untuk menentukan orang tua dan urutan setelah anak ke berapa dari orang tua akun ini. Saat memilih “*parent*” maka *field* “*insert setelah*” akan otomatis berisi hanya daftar anak dari “*parent*” yang dipilih (tidak termasuk anak dari anaknya).

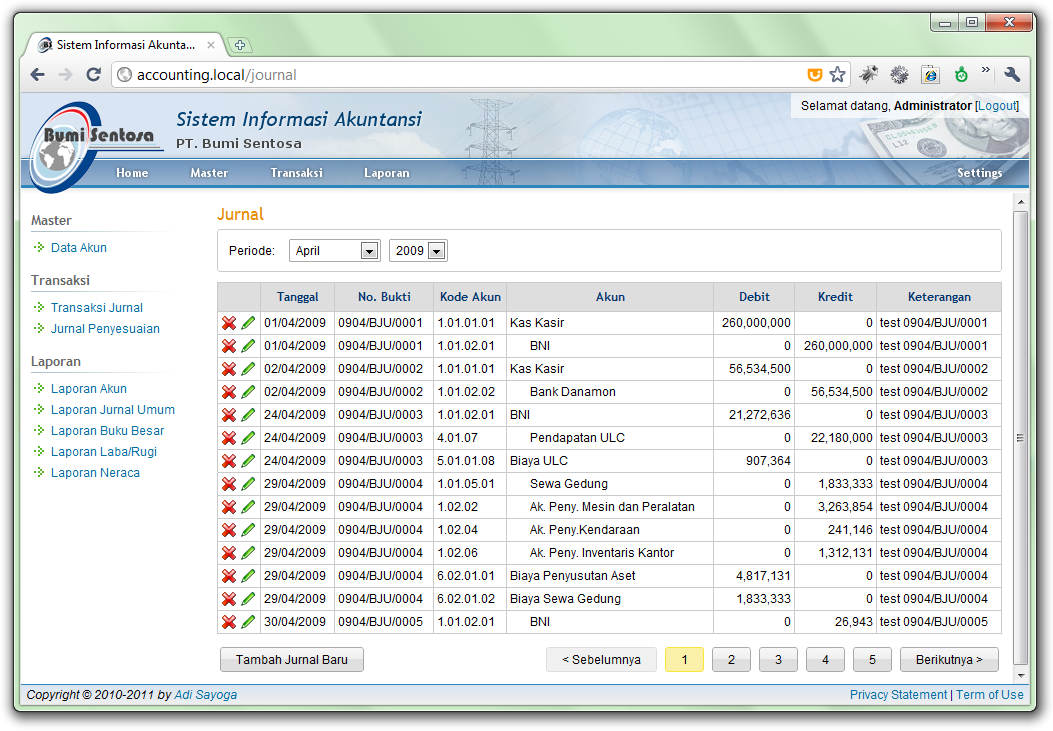


Gambar 5.9 Implementasi Form Ubah Data Akun

Pada saat mengubah data akun, tidak terdapat lagi *field* untuk menentukan hirarki data. Ini bukan tidak mungin untuk mengubahnya, tetapi memerlukan proses yang lebih panjang yaitu proses hapus kemudian tambah (bukan menghapus data akun kemudian menambahnya, tetapi meng-*update* penomoran *node* seperti saat menghapus dan dan saat mengubah data akun). Karena keterbatasan waktu, maka pada aplikasi ini tidak terdapat fasilitas mengubah hirarki.

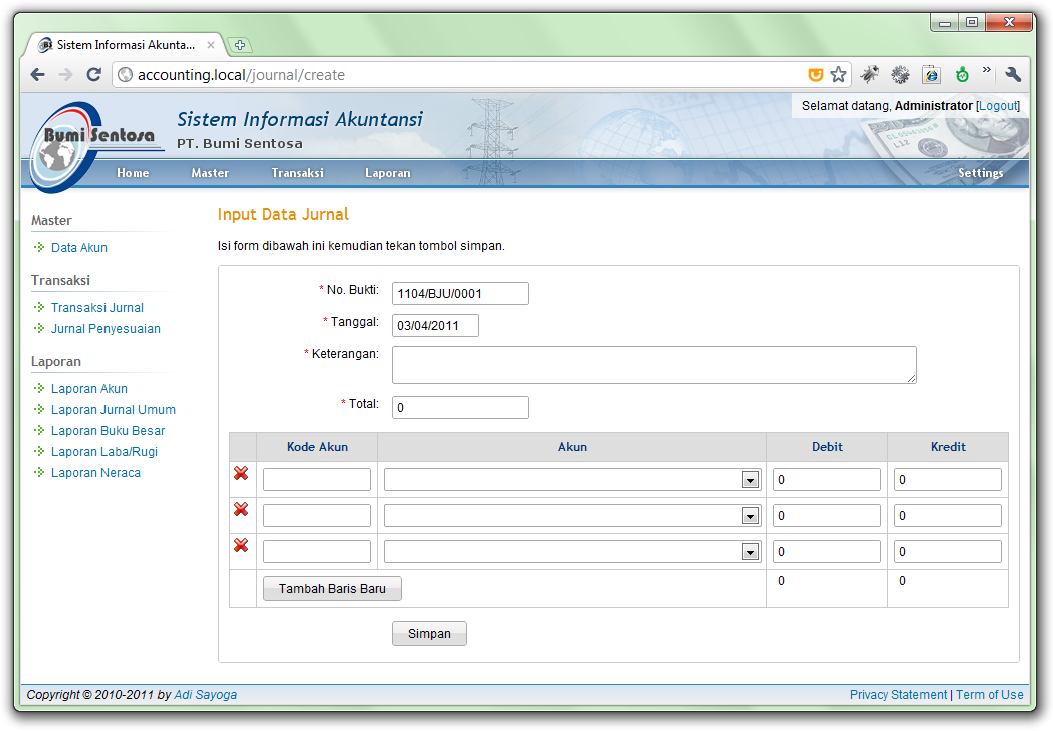
### Pengolahan Transaksi Juranl

|  |  |
| --- | --- |
| **Fungsi** | Mengolah data jurnal |
| **Antarmuka Tampilan** | Tampilan masukan data jurnal |
| **Pemrosesan** | Penambahan, pengubahan, dan penghapusan data jurnal |

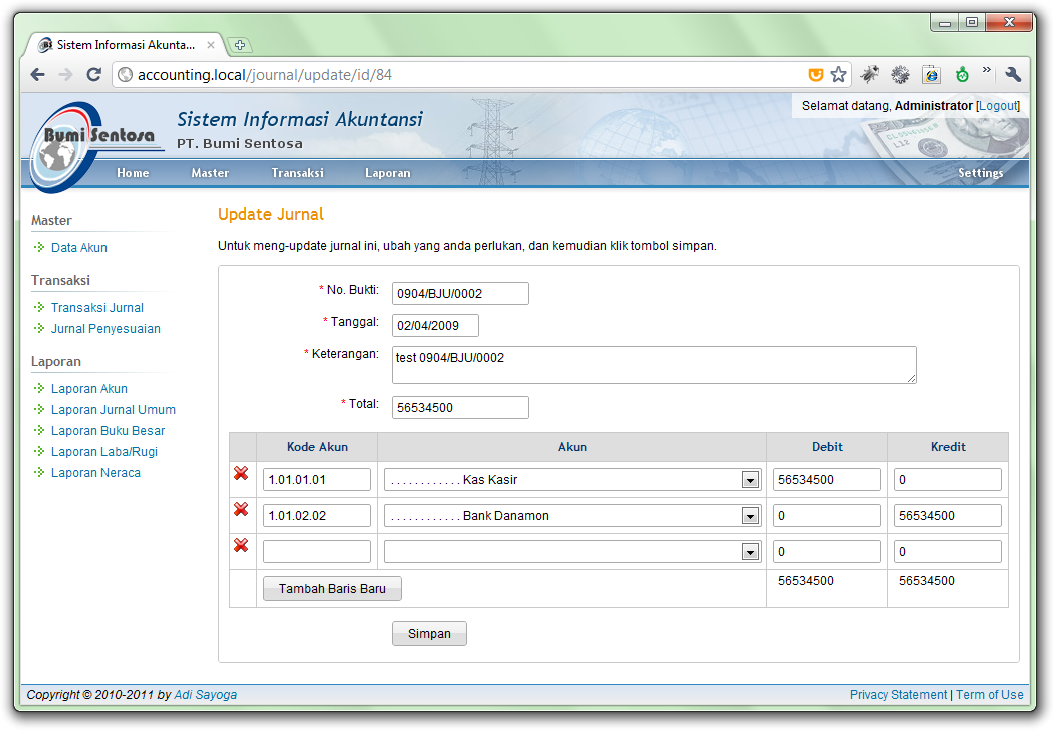


Gambar 5.10 Implementasi Daftar Jurnal

Setiap transaksi mempengaruhi minimal dua akun. Berapapun jumlah akun yang terlibat dalam sebuah transaksi, jumlah debit dan kredit akan selalu sama dalam setiap transaksi.



Gambar 5.11 Implementasi Form Tambah Data Jurnal



Gambar 5.12 Implementasi Form Ubah Data Jurnal

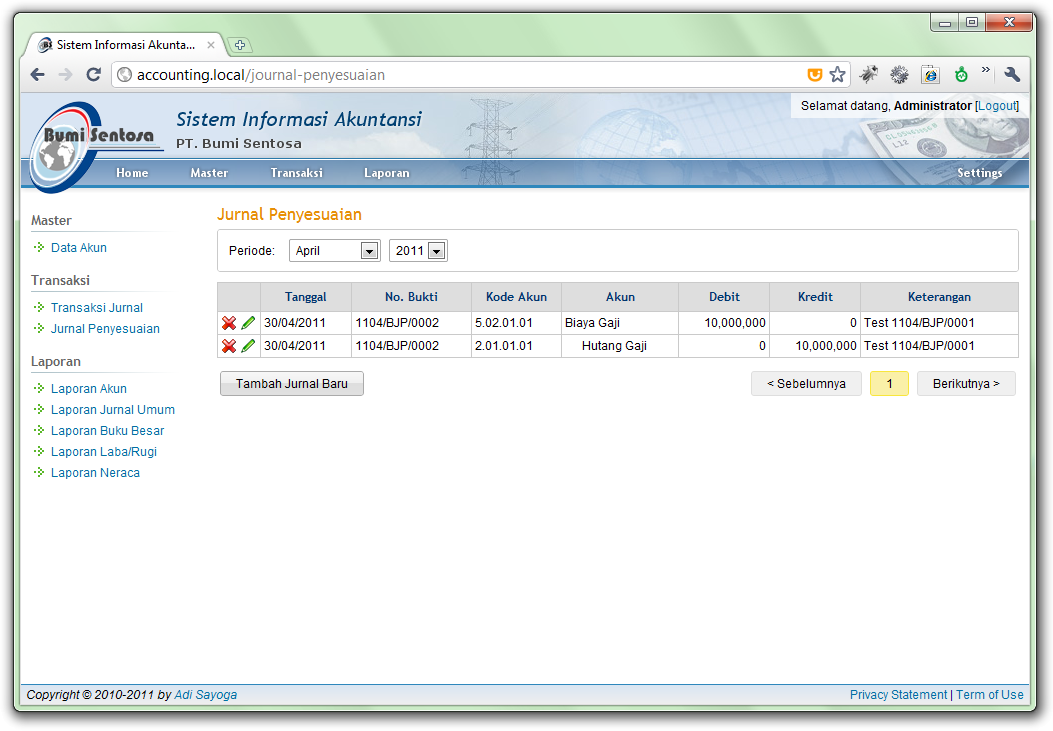
Untuk menginputkan data, inputkan keterangan dan total transaksi (no bukti akan ditampilkan secara otomatis dan tanggal otomatis tanggal saat ini). Total disini merupakan nilai total dari transaksi dan juga berfungsi untuk memverifikasi bahwa jumlah debit/kredit jurnal harus sama dengan total transaksi, dan untuk meminimalkan kesalahan penginputan data.

Kemudian ketikkan kode akun atau pilih akun pada *combo box*. Jika menginputkan kode akun, maka pada *combo box* akun akan otomatis berubah sesuai dengan kode yang diinputkan, sebaliknya jika *combo box* akun yang dipilih maka kode akun akan berubah sesuai dengan akun yang dipilih. Untuk dapat menyimpan, total debit dan total kredit pada setiap akun harus sama begitu juga dengan total transaksi.

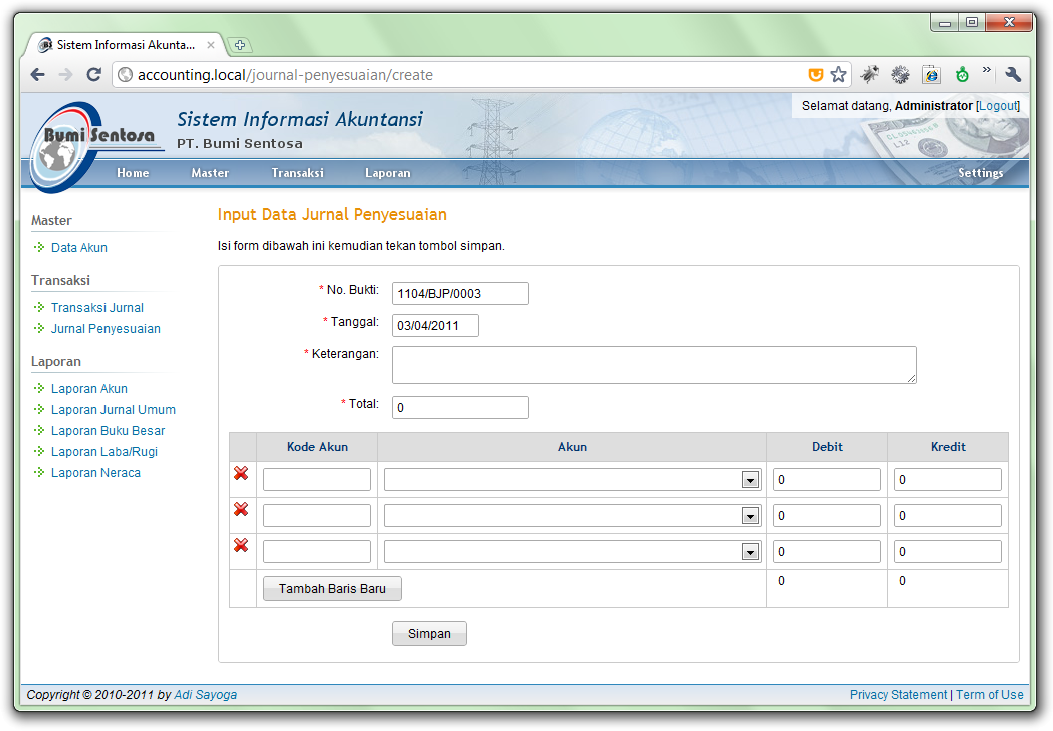
### Pengolahan Jurnal Penyesuaian

|  |  |
| --- | --- |
| **Fungsi** | Mengolah data jurnal penyesuaian |
| **Antarmuka Tampilan** | Tampilan masukan data jurnal penyesuaian |
| **Pemrosesan** | Penambahan, pengubahan, dan penghapusan data juran pemyesuaian |

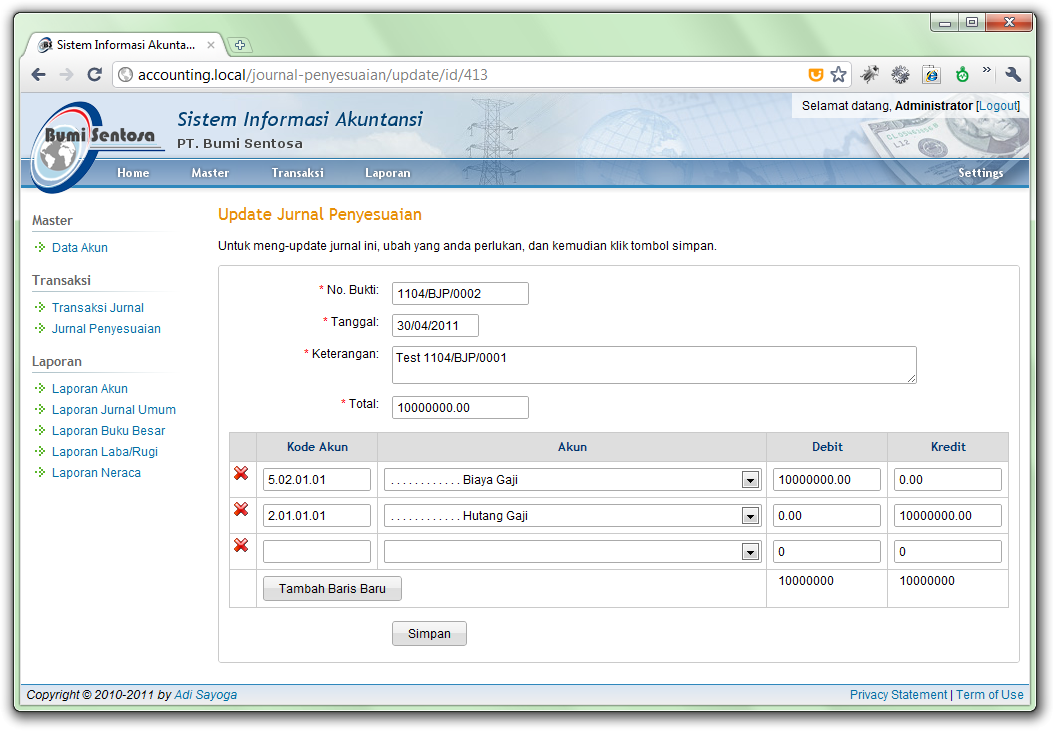
Ayat jurnal yang memutakhirkan akun pada akhir periode disebut dengan ayat jurnal penyesuaian. Suatu ayat jurnal penyesuaian akan selalu melibatkan akun pendapatan atau beban dan akun aset atau kewajiban. Proses penyusunan jurnal penyesuaian dilakukan sesuai dengan penyusunan jurnal umum.



Gambar 5.13 Implementasi Daftar Jurnal Penyesuaian



Gambar 5.14 Implementasi Form Tambah Data Jurnal Penyesuaian

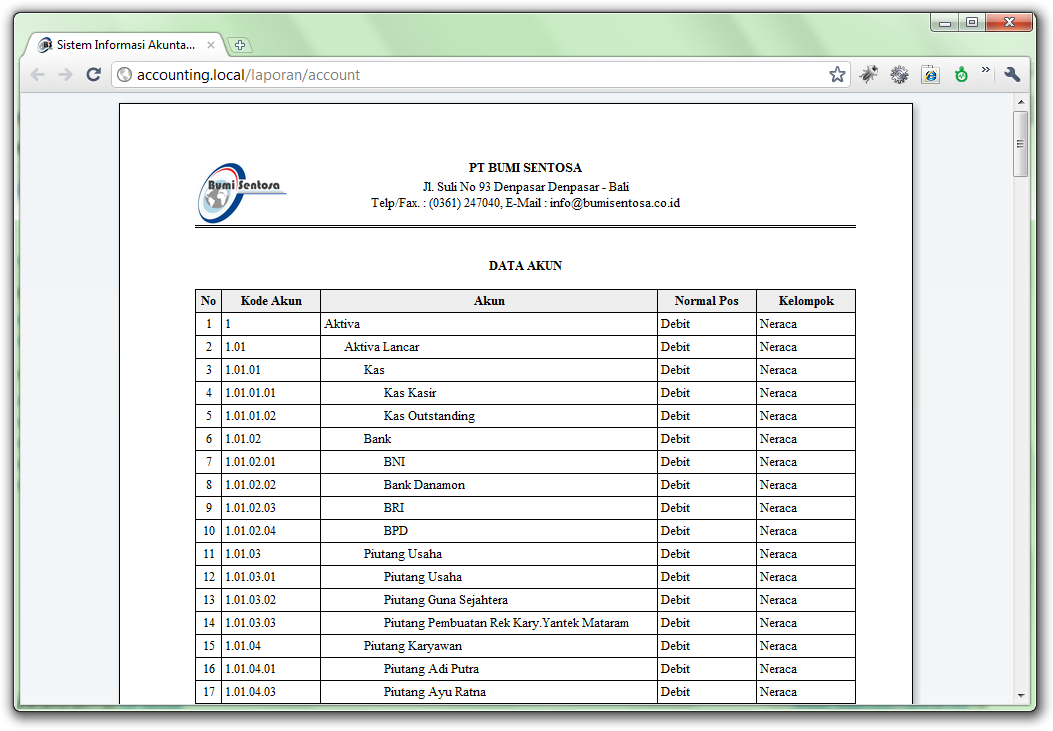


Gambar 5.15 Implementasi Form Ubah Jurnal Penyesuaian

### Laporan Data Akun

|  |  |
| --- | --- |
| **Fungsi** | Menampilkan semua data akun |
| **Antarmuka Tampilan** | Tampilan laporan |
| **Pemrosesan** | Menampilkan data |

Laporan data akun ini menampilkan semua data akun. Laporan ini berguna jika ingin mengetahui keseluruhan daftar akun yang ada.

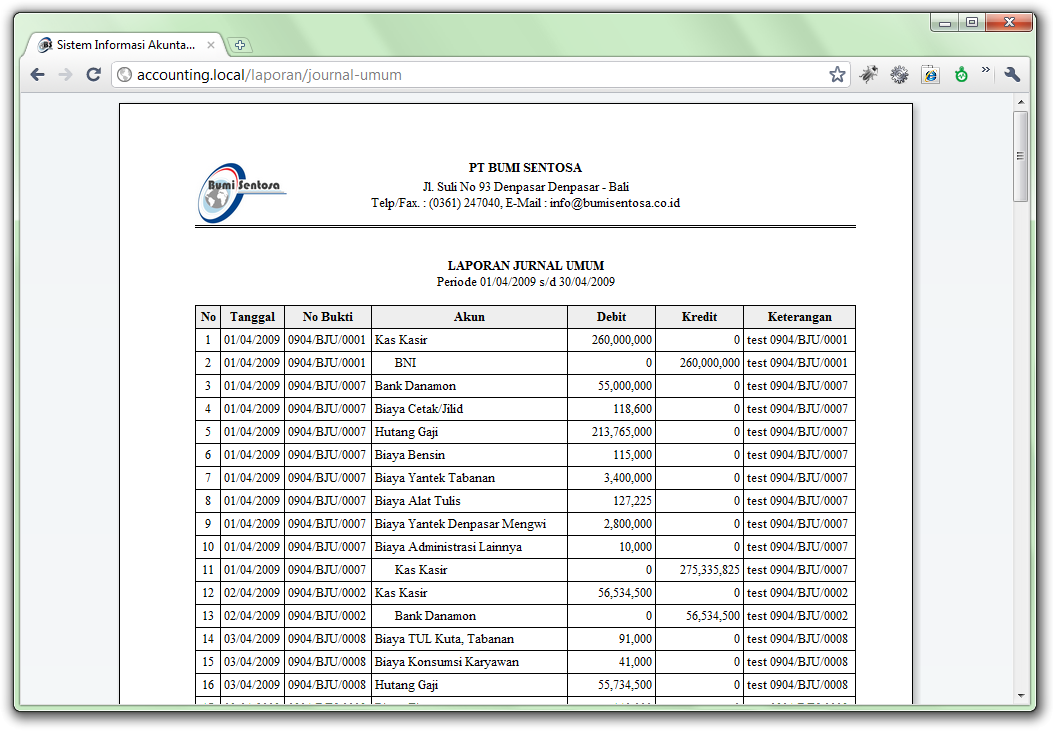


Gambar 5.16 Implementasi Laporan Data Akun

### Laporan Jurnal Umum

|  |  |
| --- | --- |
| **Fungsi** | Menampilkan laporan jurnal umum |
| **Antarmuka Tampilan** | Tampilan laporan |
| **Pemrosesan** | Menampilkan data berdasarkan tanggal awal dan tanggal akhir yang ditentukan oleh *user* |

Saat menampilkan laporan jurnal umum, maka terlebih dahulu akan terdapat pilihan untuk memilih periode tanggal awal dan tanggal akhir. Setelah periode dipilih maka akan ditampilkan laporan jurnal umum. Laporan jurnal umum ini mengambil semua data yang diinput pada saat penjurnalan (jurnal umum maupun jurnal penyesuaian) yang berada diantara parameter tanggal awal dan tanggal akhir.

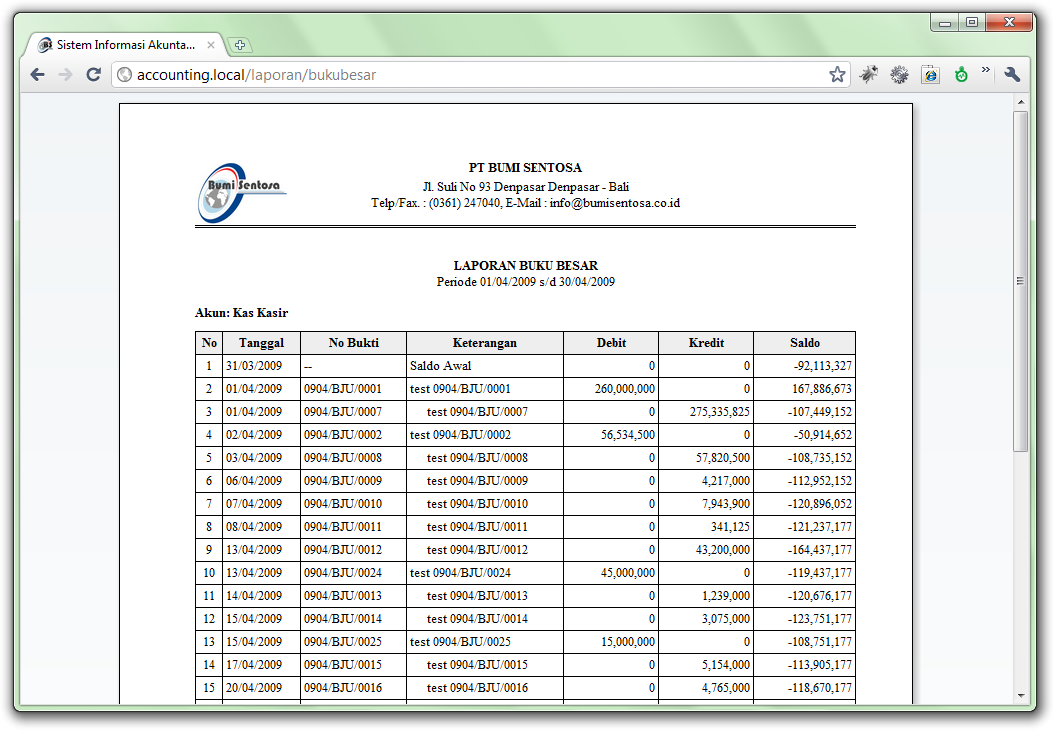


Gambar 5.17 Implementasi Laporan Jurnal Umum

### Laporan Buku Besar

|  |  |
| --- | --- |
| **Fungsi** | Menampilkan laporan buku besar |
| **Antarmuka Tampilan** | Tampilan laporan |
| **Pemrosesan** | Menampilkan data berdasarkan tanggal awal dan tanggal akhir yang ditentukan oleh *user* |

Laporan buku besar terlebih dahulu akan ditanyakan untuk memilih periode tanggal awal dan tanggal akhir serta akun mana yang ditampilkan.

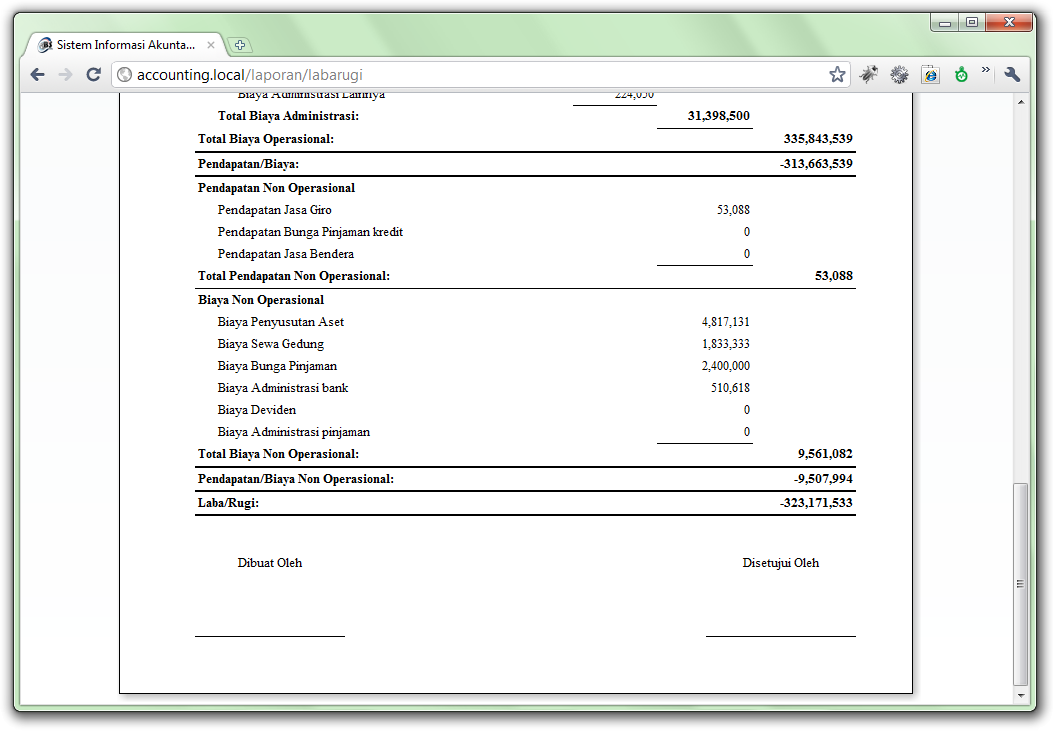


Gambar 5.18 Implementasi Laporan Buku Besar

Saldo awal merupakan total saldo dari sebelum tanggal awal. Kemudian mengambil semua data jurnal per akun periode tanggal awal dan tanggal akhir. Dan saldo akan ditampilkan berakumulasi sesuai dengan debit/kredit jurnal. Nilai saldo paling akhir merupakan nilai saldo awal periode berikutnya.

### Laporan Laba/Rugi

|  |  |
| --- | --- |
| **Fungsi** | Menampilkan laporan laba/rugi |
| **Antarmuka Tampilan** | Tampilan laporan |
| **Pemrosesan** | Menampilkan data berdasarkan tanggal awal dan tanggal akhir yang ditentukan oleh *user* |



Gambar 5.19 Implementasi Laporan Laba/Rugi

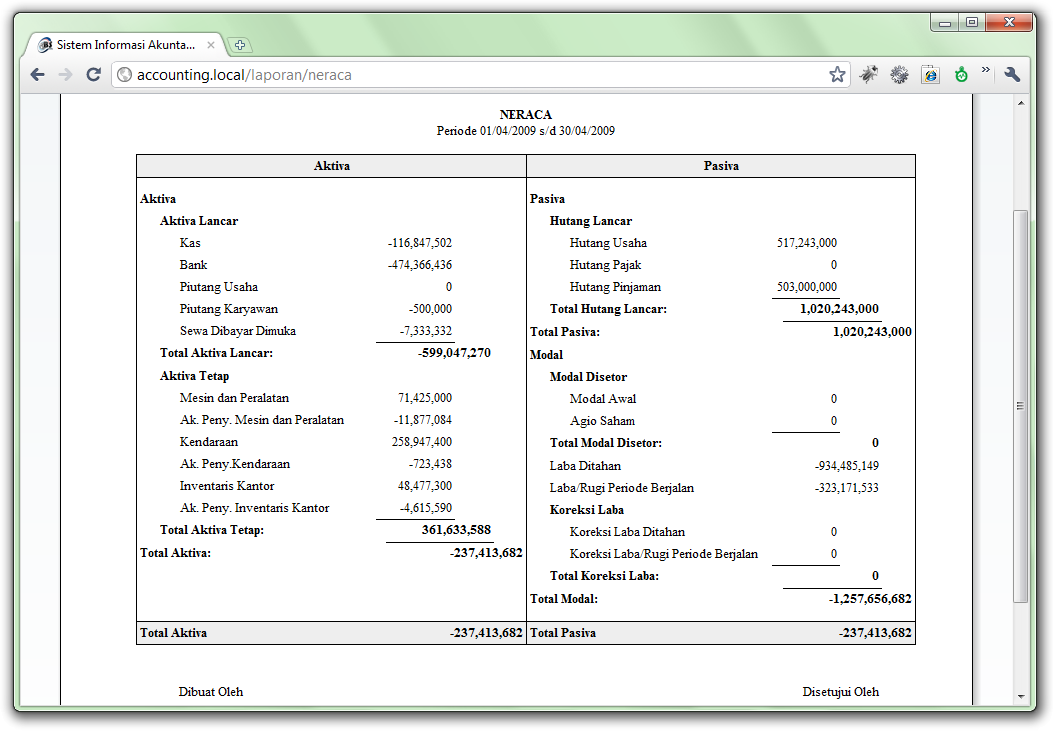
Menampilkan laporan laba/rugi terlebih dahulu terdapat pilihan untuk menentukan periode tanggal awal dan tanggal akhir. Setelah menentukan periode, baru kemudian ditampilkan laporan laba/rugi.

Pada laporan laba/rugi ini, terdiri dari empat bagian yaitu pendapatan, biaya, pendapatan non operasional dan biaya non operasional. Nilainya diambil dari total transaksi jurnal dari periode tanggal awal sampai tanggal akhir.

### Laporan Neraca

|  |  |
| --- | --- |
| **Fungsi** | Menampilkan laporan neraca |
| **Antarmuka Tampilan** | Tampilan laporan |
| **Pemrosesan** | Menampilkan data berdasarkan tanggal awal dan tanggal akhir yang ditentukan oleh *user* |

Sama seperti laporan jurnal umum dan laporan buku besar, terlebih dahulu terdapat pilihan untuk menentukan tanggal awal dan tanggal akhir. Pada laporan neraca ini, total aktiva dan pasiva akan selalu seimbang.



Gambar 5.20 Implementasi Laporan Neraca

Pada akun modal terdapat akun-akun laba ditahan dan labarugi periode berjalan, inilah yang membuat total aktiva dan pasiva selalu seimbang. Untuk mendapatkan nilai laba ditahan dengan cara mendapatkan total dari laba/rugi sampai sebelum periode awal. Sedangkan laba/rugi periode berjalan merupakan nilai laba/rugi dari periode tanggal awal sampai tanggal akhir. Sedangkan untuk akun lainnya didapatkan dari total transaksi masing-masing akun sampai tanggal akhir.

## Spesifikasi Kebutuhan Perangkat Lunak

Tahapan dalam mengimplementasikan Sistem Informasi Akuntansi ini meliputi perancangan antarmuka sistem dan implementasi proses input data akun dan penjurnalan sehingga dapat menghasilkan *output* berupa laporan data akun, buku besar, laba/rugi dan neraca.

Sistem ini dikembangkan dan diimplementasikan menggunakan bahasa pemrograman PHP 5.3 dan Zend *Framework* 1.10 yang berjalan pada *server* Apache 2.2 dengan *database* MySQL 5.1.

### Antar Muka Sistem

Implementasi perangkat lunak ini dilakukan dengan membuat *project* PHP dengan menggunakan zend *framework tool* di direktori /htdocs apache. *Project* ini diorganisasikan menjadi tiga subfolder yaitu: public (semua file yang dapat diakses publik), library (berisi *library framework* dan *custom library* jika diperlukan), dan application (jantung dari project, dimana *model*, *view* dan *controller* berada).

### Proses Sistem Informasi Akuntansi

Sistem ini diimplementasikan berdasarkan hasil analisis sistem yang telah dibuat, dimana sistem ini memiliki beberapa bagian utama yang merupakan fungsi utama dari perangkat lunak yang dikembangkan.

Berikut halaman yang telah dirancang:

1. Proses *input*, halaman proses *input* data terdiri dari data akun, jurnal, dan jurnal penyesuaian.
2. Proses *output*, implementasi sistem untuk proses *output* data akan menghasilkan laporan data akun, jurnal umum, buku besar, laba/rugi, dan neraca.
3. Juga terdapat halaman lain untuk melakukan login.

## Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan untuk membangun sistem ini pada intinya tidak memerlukan spesifikasi hardware khusus dan dapat berjalan pada *flatform* berbeda-beda (*cross flatform*).

Apache memerlukan 50 MB *temporary disk space*, setelah ter-*install* apache menempati kira-kira 10 MB *disk space*. PHP dan Zend *Framework* tidak perlu di-*install*, cukup di-*copy* ke *hardisk* dan mengatur file konfigurasi untuk dapat berhubungan dengan apache, php dan Zend. PHP dan Zend sendiri memerlukan *disk space* masing-masing kira-kira 35 MB dan 86 MB. Untuk *library* JavaScript, besar file JQuery *core* 70,4 KB dan JQuery UI sebesar 210 KB.

Pada *client* tidak ada kebutuhan khusus, hanya memerlukan *browser* dan setiap sistem operasi biasanya sudah ter-*install* *browser* secara *default*.

## Kegiatan Implementasi

### Pemrograman

Pemrograman merupakan kegiatan yang sangat menunjang terselenggaranya sistem yang baru karena program yang baik dan dapat menghasilkan suatu informasi yang sesuai dengan kebutuhan. Setelah pemrograman selesai dilakukan maka harus diuji terlebih dahulu agar dapat menemukan kesalahan-kesalahan pemrograman.

## Pengujian Sistem

Pengujian merupakan serangkaian aktivitas untuk mengintegrasikan model desain *test case* perangkat lunak kedalam sederetan langkah-langkah yang direncanakan dengan baik, dan hasilnya merupakan konstruksi perangkat lunak yang berhasil.

Tahap pertama pengujian dilakukan untuk proses *input* data. Setelah dilakukan *input* data, maka akan dilihat *output* yang dihasilkan apakah sistem telah menghasilkan *output* yang diinginkan dan apakah tabel dapat berelasi dengan baik. Tahap pengujian ini akan meliputi verifikasi untuk memastikan apakah program telah terbebas dari kesalahan. Pengujian ini dilakukan dengan proses *debugging* dan validasi untuk memastikan bahwa sistem sesuai dengan kebutuhan pemakai, pengujian ini dilakukan dengan melakukan uji penerimaan pemakaian.

# BAB VI PENUTUP

# 

## Kesimpulan

Berdasarkan tujuan dari pengembangan Sistem Informasi Akuntansi yang telah dilakukan andalisis dan perancangan terhadap sistem tersebut, maka sistem ini diharapkan dapat:

1. Mempermudah bagian keuangan dalam menangani masalah keuangan perusahaan. // TODO Kesimpulan
2. Membantu bagian internal dalam menganalisa keuntungan dan kerugian perusahaan.

## Saran

Untuk menyempurnakan sistem ini, maka penulis dapat menyarankan sistem ini dapat dikembangkan dengan mengintegrasikan ke modul pembelian peralatan dan perlengkapan, penggajian, dan lainnya yang berhubungan dengan aliran keuangan perusahaan.

Masalah keamanan, aplikasi ini hanya terdiri dari tiga role yaitu *guest*, *user* dan *admin* dan dibuat langsung pada kode program (*hard code*). Kedepannya dapat dikembangkan dengan membuat tabel tambahan sehingga hak akses dapat dikustomisasi pada halaman khusus “siapa dapat mengakses apa”.

# DAFTAR PUSTAKA

Brandon, Dan. 2006. *Project Management for Modern Information Systems*. United States of America: IRM Press

Brinzarea-lamadi, Bogdan, and Audra Hendrix. 2009. *AJAX and PHP: Building Modern Web Applications – Second Edition*. Birmingham: Pack Publishing

Celko, Joe. 2004. *Trees And Hierarchies In SQL For Smarties*. United States of America: Morgan Kaufmann Publishers

Davis, Michele, and Jon Phillips. 2007. *Learning PHP and MySQL - Second Edition*. Sebastopol: O'Reilly

Deshmukh, Ashutosh. 2006. *Digital Accounting: The Effects of the Internet and ERP on Accounting*. United States of America: IRM Press

Evans, Call. 2008. *php|architect’s Guide to Programming with Zend Framework*. United States: Marco Tabini & Associates

Kumar, N., and R. Mittal. 2004. *Management Information System*. New Delhi: Anmol Publications

Kusrini, dan Andri Koniyo. 2007. *Tuntunan Praktis Membangun Sistem Informasi Akuntansi dengan Visual Basic dan Microsoft SQL Server*. Yogyakarta: Andi

Lyman, Forrest. 2009. *Pro Zend Framework Techniques: Build a Full CMS Project*. United States of America: Apress

Ratzan, Lee. 2004. *Understanding Information System: What They Do and Why We Need Them*. United States of America: ALA Editions

Sarosa, Samiaji. 2009. *Sistem Informasi Akuntansi*. Jakarta: Grasindi

Wahyono, Teguh, dan Leonard Pujiatmoko. 2008. *Pengembangan Aplikasi Akuntansi Berbasis Microsoft Visual Basic .NET*. Jakarta: Elex Media

York, Richard. 2009. *Beginning JavaScript and CSS Development with JQuery*. Canada: Wiley Publishing

Yuhefizar. 2008. *10 Jam Menguasai Internet, Teknologi dan Aplikasinya*. Jakarta: Elex Media

http://framework.zend.com, 27 September 2010 pukul 10.00 wita

http://jquery.com, 27 September 2010 pukul 10.20 wita

http://dev.mysql.com/tech-resources/articles/hierarchical-data.html, 27 September 2010 pukuk 11.23 wita

# LAMPIRAN KODE PROGRAM

## Struktur File

+ **Accounting**

+ **public**

| + api

| | + v1.0

| | + jsonrpc.php

| + css

| | + report-screen.css

| | + report.css

| + images

| | *[daftar gambar]*

| + include

| | + report-header.php

| + js

| | + common.js

| | + dateFormat.js\*

| | + journal.js

| | + jquery-1.4.2.min.js\*

| | + jquery-ui-1.8.custom.min.js\*

| | + jquery.formValidation.js

| | + jquery.layout.min.js\*

| | + jquery.pagination.js

| | + jquery.zend.jsonrpc.js\*

| | + json2.js\*

| + skins

| | + default

| | + css

| | | + form.css

| | | + jquery.css

| | | + layout.css

| | | + text.css

| | + images

| | | *[daftar gambar]*

| | + skin.xml

| + index.php

+ **application**

| + configs

| | + application.ini

| + controllers

| | + AccountController.php

| | + ErrorController.php

| | + IndexController.php

| | + JournalController.php

| | + JournalPenyesuaianController.php

| | + LaporanController.php

| | + UserController.php

| + forms

| | + Account.php

| | + Journal.php

| | + Laporan.php

| | + Login.php

| | + User.php

| + layouts

| | + scripts

| | + layout.phtml

| + models

| | + Account.php

| | + Crud.php

| | + Journal.php

| | + JournalPenyesuaian.php

| | + JournalUmum.php

| | + JsonRpc.php

| | + Laporan.php

| | + User.php

| + modules

| | + settings

| | + controllers

| | | + AccountController.php

| | | + IndexController.php

| | + views

| | | + scripts

| | | | + account

| | | | | + index.phtml

| | | | | + rebuild-tree.phtml

| | | | + index

| | | | + index.phtml

| | + Bootstrap.php

| + views

| | + helpers

| | | + LoadSkin.php

| | + scripts

| | + \_partials

| | | + pagination-control.phtml

| | + account

| | | + \_table-row.phtml

| | | + create.phtml

| | | + index.phtml

| | | + update.phtml

| | + error

| | | + error.phtml

| | | + noauth.phtml

| | + index

| | | + index.phtml

| | + journal

| | | + \_table-row.phtml

| | | + create.phtml

| | | + index.phtml

| | | + update.phtml

| | + journal-penyesuaian

| | | + \_table-row.phtml

| | | + create.phtml

| | | + index.phtml

| | | + update.phtml

| | + laporan

| | | + account.phtml

| | | + bukubesar.phtml

| | | + journal-umum.phtml

| | | + labarugi.phtml

| | | + neraca.phtml

| | | + view-bukubesar.phtml

| | | + view-journal-umum.phtml

| | | + view-labarugi.phtml

| | | + neraca.phtml

| | + user

| | + \_table-row.phtml

| | + create.phtml

| | + index.phtml

| | + login.phtml

| | + logout.phtml

| | + state.phtml

| | + update-password.phtml

| | + update.phtml

| + bootstrap.php

+ **library**

+ Acc

+ Controller

+ Plugin

+ Acl.php

\*) framework pihak ketiga:

/public/js/dateFormat.js

Date Format 1.2.3

(c) 2007-2009 Steven Levithan ([stevenlevithan.com](http://www.stevenlevithan.com)) MIT license

Includes enhancements by

Scott Trenda ([scott.trenda.net](http://scott.trenda.net)) and

Kris Kowal ([cixar.com/~kris.kowal/](http://cixar.com/~kris.kowal/))

/public/js/jquery-1.4.2.min.js

jQuery JavaScript Library v1.4.2 (<http://jquery.com/>)

Copyright 2010, John Resig

Dual licensed under the MIT or GPL Version 2 licenses.

<http://jquery.org/license>

Includes Sizzle.js (<http://sizzlejs.com/>)

Copyright 2010, The Dojo Foundation

Released under the MIT, BSD, and GPL Licenses.

/public/js/jquery-ui-1.8.custom.min.js

jQuery UI 1.8

Copyright (c) 2010 AUTHORS.txt (<http://jqueryui.com/about>)

Dual licensed under the MIT (MIT-LICENSE.txt and GPL (GPL-LICENSE.txt) licenses.

<http://docs.jquery.com/UI>

/public/js/jquery.layout.min.js

jquery.layout 1.2.0

Copyright (c) 2008

Fabrizio Balliano (<http://www.fabrizioballiano.net>)

Kevin Dalman (<http://allpro.net>)

Dual licensed under the GPL (<http://www.gnu.org/licenses/gpl.html>) and MIT (<http://www.opensource.org/licenses/mit-license.php>) licenses.

/public/js/jquery.zend.jsonrpc.js

Copyright (c) 2010 - 2011 Tanabicom, LLC (<http://www.tanabi.com>)

Contributions by Mike Bevz (<http://mikebevz.com>)

Released under the MIT license

/public/js/json2.js

<http://www.JSON.org/json2.js>

Public Domain.

NO WARRANTY EXPRESSED OR IMPLIED. USE AT YOUR OWN RISK.

See <http://www.JSON.org/js.html>

## File: /public/js/common.js

/\*\*

\* Copyright 2010-2011, Adi Sayoga

\*/

/\*\*

\* Repeat suatu string

\*/

String.prototype.repeat = function(num) {

return new Array((num \* 1) + 1).join(this);

};

/\*\*

\* Menampilkan pesan informasi pada header

\*

\* @param message - pesan yang ditampilkan

\* @param state - state message (info | alert | error)

\*/

function displayMessage(message, state) {

*// default*

var iconClass = "ui-icon-info";

var backgroundClass = "ui-state-highlight";

switch (state) {

case "info":

iconClass = "ui-icon-info";

backgroundClass = "ui-state-highlight";

break;

case "alert":

iconClass = "ui-icon-alert";

backgroundClass = "ui-state-highlight";

break;

case "error":

iconClass = "ui-icon-circle-close";

backgroundClass = "ui-state-error";

break;

}

*// Icon*

var $icon = $("<div>").addClass("ui-icon " + iconClass).css({

"float" : "left",

"margin-right" : "10px"

});

*// Message*

var $messageDiv = $("<div>").text(message).css({

"overflow" : "hidden"

});

*// Container*

var $container = $("<div>").addClass(backgroundClass

+ " ui-corner-all").append($icon).append($messageDiv).css({

"padding" : "5px 10px"

});

$(".header .message").empty().append($container).fadeIn("slow")

.delay(5000)

.fadeOut(1000);

*// Tampilkan di tengah*

var marginLeft = Math.floor(($(".header").width() - $container.width())

/ 2);

$(".header .message").css({ "left" : marginLeft });

}

## File: /public/js/journal.js

/\*\*

\* Copyright 2011, Adi Sayoga

\*/

var api, listAccount = "";

*// No bukti ini akan di-set melalui kode php (melalui controller), disini tidak bisa disisipkan*

*// melalui kode php.*

*//* **TODO** *Ada cara lebih baik?*

var lastNoBukti;

$(document).ready(function() {

updateUI();

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

*// Validasi form*

$(".form").validate({

smd: api,

functionName: "validateJournalForm",

toValidate: ".form #noBukti, .form #tanggal, .form #keterangan,

.form #total",

exclude: lastNoBukti,

onSubmit: function() {

*// Jika form di-submit dan data valid, maka update data*

validateTotal() && updateDataJournal($(".form #id").val(),

generateData());

}

});

*// Update no bukti otomatis jika tanggal transaksi diubah*

$("#tanggal").change(function() { autoNum(); });

});

/\*\*

\* No bukti otomatis

\*/

function autoNum() {

var prefix = ($(".form #tipeJournal").val() == "1")? "BJU": "BJP";

api.journalAutoNum($("#tanggal").val(), { prefix: prefix }, {

success : function(result, id, method) {

$("#noBukti").val(result);

}

});

}

/\*\*

\* Update UI saat javasipt enable

\*/

function updateUI() {

*// Tombol tambah baru hanya diperlukan jika javascript di-disable*

$(".noJavascript").remove();

*// Tambahkan tombol untuk menambah baris detail*

$(".addRow").html('<input type="button" onClick="addRow()"

value="Tambah Baris Baru" />');

*// detail*

$(".form table tbody tr").each(function() {

*// Action - Hapus checkbox dan tambahkan tombol delete*

var $deleteButton = $('<a href="#" class="delete"></a>');

$(this).find(".include").remove();

$(this).find(".action").append($deleteButton);

$deleteButton.click(function() { deleteRow(this); });

*// Account*

var $kodeAccount = $(this).find(".kodeAccount");

$kodeAccount.attr("disabled", false);

var $account = $(this).find(".account");

if (!$kodeAccount.val()) $account.val(null);

$kodeAccount.blur(function() { displayAccount($(this), $account); });

$account.change(function() { displayKodeAccount($(this),

$kodeAccount); });

*// List account hanya diisi sekali*

if (!listAccount) listAccount = $account.html();

*// Debit/kredit*

$(this).find(".debit").blur(function() { updateTotal(); });

$(this).find(".kredit").blur(function() { updateTotal(); });

});

}

/\*\*

\* Mengosongkan UI untuk menambah data baru

\*/

function clearUI() {

autoNum();

$(".form #id").val("");

$(".form #keterangan").val("");

$(".form #total").val("0");

$(".form table tbody tr").empty();

addRow();

updateTotal();

}

/\*\*

\* Menampilkan data account pada comboBox account

\* @param object $\_self

\* @param object $\_toDisplay

\*/

function displayAccount($\_self, $toDisplay) {

if (!$\_self.val()) {

*// Jika kode account kosong, maka kosongkan juga comboBox account*

$toDisplay.val(null);

return;

}

api.getAccountByKode($\_self.val(), { success: function(data, id,

method) {

if (data[0]) {

$toDisplay.val(data[0].id);

} else {

alert("Kode Akun tidak ditemukan.");

$\_self.val("");

$toDisplay.val(null);

}

}});

}

/\*\*

\* Menampilkan kode account berdasarkan id account

\* @param object $\_self

\* @param object $\_toDisplay

\*/

function displayKodeAccount($\_self, $toDisplay) {

api.getAccountById($\_self.val(), { success:

function(data, id, method) {

if (data[0]) {

$toDisplay.val(data[0].kodeAccount);

} else {

$toDisplay.val("");

}

}});

}

/\*\*

\* Menghapus baris detail

\* @param object \_self

\*/

function deleteRow(\_self) {

if (!$(\_self).parent().next().find(".kodeAccount").val()

|| confirm("Yakin menghapus data ini?")) {

$(\_self).parent().parent().remove();

updateTotal();

}

}

/\*\*

\* Menambah baris baru

\*/

function addRow() {

*// Tombol delete*

var $deleteButton = $('<a href="#" class="delete"></a>');

$deleteButton.click(function() { deleteRow(this); });

*// Account*

var $kodeAccount = $('<input type="text" name="kodeAccount[]"

class="kodeAccount" style="width: 100px;" />');

var $account = $('<select name="account[]" class="account"

style="width: 99%; "></select>');

$kodeAccount.blur(function() { displayAccount($(this), $account); });

$account.change(function() { displayKodeAccount($(this),

$kodeAccount); });

$account.append(listAccount);

*// Debit/kredit*

var $debit = $('<input type="text" name="debit[]" class="debit"

value="0" style="width: 100px;" />');

var $kredit = $('<input type="text" name="kredit[]" class="kredit"

value="0" style="width: 100px;" />');

$debit.blur(function() { updateTotal(); });

$kredit.blur(function() { updateTotal(); });

var $tdAction = $('<td class="action"></td>');

$tdAction.append($deleteButton);

var $tdKodeAccount = $("<td>");

$tdKodeAccount.append($kodeAccount);

var $tdAccount = $("<td>");

$tdAccount.append($account);

var $tdDebit = $("<td>");

$tdDebit.append($debit);

var $tdKredit = $("<td>");

$tdKredit.append($kredit);

var $row = $("<tr>");

$row.append($tdAction).append($tdKodeAccount).append($tdAccount)

.append($tdDebit).append($tdKredit);

$(".form table tbody").append($row);

$account.val(null); *// Kosongkan comboBox account*

}

/\*\*

\* Validasi total transaksi

\*/

function validateTotal() {

var total = parseFloat($("#total").val());

var totalDetails = updateTotal();

$("#total").removeClass("errors-element");

*// Total debit dan kredit harus sama (balance)*

if (totalDetails.debit != totalDetails.kredit) {

displayMessage("Total debit/kredit tidak balance!");

$("#total").addClass("errors-element");

return false;

}

*// Total transaksi harus sama dengan total debit/kredit*

if (total != totalDetails.debit) {

displayMessage("Total tidak sama dengan total debit/kredit");

$("#total").addClass("errors-element");

return false;

}

return true;

}

/\*\*

\* Update total debit/kredit

\*/

function updateTotal() {

var totalDebit = 0, totalKredit = 0;

$(".form table tbody tr").each(function() {

*// Total hanya dihitung jika kode account tidak kosong*

if ($(this).find(".kodeAccount").val()) {

totalDebit += parseFloat($(this).find(".debit").val());

totalKredit += parseFloat($(this).find(".kredit").val());

}

});

*// Update total*

$(".form table tfoot .totalDebit").text(totalDebit);

$(".form table tfoot .totalKredit").text(totalKredit);

return { debit: totalDebit, kredit: totalKredit };

}

/\*\*

\* Mendapatkan data

\*/

function generateData() {

var details = new Array();

$(".form table tbody tr").each(function() {

*// Account yang kosong dilewati*

if (!$(this).find(".kodeAccount").val()) return;

details.push({

"kodeAccount": $(this).find(".kodeAccount").val(),

"idAccount": $(this).find(".account").val(),

"debit": $(this).find(".debit").val(),

"kredit": $(this).find(".kredit").val()

});

});

var data = {

"id": $("#id").val(),

"tipeJournal": $("#tipeJournal").val(),

"noBukti": $("#noBukti").val(),

"tanggal": $("#tanggal").val(),

"keterangan": $("#keterangan").val(),

"total": $("#total").val(),

"details": details

};

console.log(data);

return data;

}

/\*\*

\* Menyimpan data ke database

\* @param json data

\*/

function updateDataJournal(id, data) {

api.updateJournal(id, data, { success: function(result) {

if (result) {

displayMessage("Data berhasil di-update.", "info");

if (!id) { *// Id tidak ditentukan, diasumsikan menambah data baru*

clearUI(); *// Kosongkan untuk menambah data lainnya*

} else {

*// redirect ke daftar journal*

*//* **FIXME** *Hard code, seharusnya ada cara yang lebih balik*

var url = "";

switch ($("#tipeJournal").val()) {

case "1": url = "/journal"; break;

case "2": url = "/journal-penyesuaian"; break;

}

window.location = url;

}

} else {

displayMessage("Data gagal di-update", "error");

}

}});

}

## File: /public/js/jquery.formValidation.js

/\*\*

\* JQuery form validation Plugin 1.2.0

\*

\* Copyright 2010-2011, Adi Sayoga

\* Dual licensed under the MIT or GPL Version 2 licenses.

\*/

(function($) {

/\*\* Validasi form \*/

$.fn.validate = function(options) {

var options = $.extend($.fn.validate.defaults, options);

options.smd.setAsync(true); *// pastikan async callback*

return this.each(function() {

*// validasi form saat submit*

$(this).submit(function(e) {

e.preventDefault();

var data = {};

$(this).find("input:hidden").each(function() {

data[$(this).attr("name")] = $(this).val();

});

$(options.toValidate).each(function() {

data[$(this).attr("name")] = $(this).val();

});

*// hapus error message*

$(this).find(".errors-element").removeClass(".errors-element");

$(this).find(".errors-icon").remove();

validate(this, data, options);

});

*// validasi masing-masing input control*

var \_form = this;

$(options.toValidate).each(function() {

$(this).blur(function() {

var data = {};

$(\_form).find("input:hidden").each(function() {

data[$(this).attr("name")] = $(this).val();

});

data[$(this).attr("name")] = $(this).val();

*// hapus error message*

$(this).removeClass("errors-element");

$(this).parent().find(".errors-icon").remove();

validate($(this), data, options);

});

});

});

};

/\*\* Default options \*/

$.fn.validate.defaults = {

smd: null, *// smd jsonRPC*

functionName: "", *// nama fungsi validasi jsonRPC*

toValidate: "", *// yang divalidasi*

submitRef: "", *// lokasi redirect jika data valid*

exclude: null, *// value yang tidak diikut sertakan pada validasi*

onSubmit: function() {},

onValid: function(\_self) {},

onInvalid: function(\_self) { displayMessage("Data yang dimasukkan

tidak valid, periksa kembali inputan."); }

};

/\*\*

\* validasi ke server

\* @param object \_self

\* @param json data

\* @param json options

\*/

function validate(\_self, data, options) {

*// panggil fungsi json-rpc*

options.smd[options.functionName](data, options.exclude, { success:

function(respons) {

if (!$.isEmptyObject(respons)) {

*// invalid*

for (var elementId in respons) {

var errorsData = respons[elementId];

$("#" + elementId).addClass("errors-element");

$("#" + elementId).parent().append(errorIcon(elementId,

errorsData));

}

$.isFunction(options.onInvalid) && options.onInvalid(\_self);

} else {

*// valid*

$.isFunction(options.onSubmit) && $(\_self).is("form")

&& options.onSubmit();

$.isFunction(options.onValid) && options.onValid(\_self);

if (options.submitRef) {

window.location = options.submitRef;

}

}

}});

};

/\*\*

\* Generate error icon

\* @param string elementId

\* @param json errorData

\*/

function errorIcon(elementId, errorsData) {

*// message*

var message = '<ul class="ui-state-highlight ui-corner-all '

+ 'errors-message" style="position: absolute; display: '

+ 'inline-block; margin: 0 25px; padding: 3px 5px; '

+ 'color: #f00; list-style: none; font-style: italic; '

+ '-webkit-box-shadow: 3px 3px 3px rgba(0, 0, 0, .2); '

+ '-moz-box-shadow: 3px 3px 3px rgba(0, 0, 0, .2); ">';

for (var errorKey in errorsData) {

message += '<li>' + errorsData[errorKey] + '</li>';

}

message += '</ul>';

*// error icon*

var $errorIcon = $('<span class="ui-icon ui-icon-alert errors-icon" '

+ 'style="position: absolute; display: inline-block; '

+ 'margin: 5px; cursor: help; "></span>');

*// tampilkan error message saat icon di-hover*

$errorIcon.hover(function() {

$(this).parent().append(message);

},

function() {

$(this).parent().find(".errors-message").remove();

});

return $errorIcon;

}

})(jQuery);

## File: /public/js/jquery.pagination.js

/\*\*

\* jQuery pagination Plugin 1.0.0

\*

\* Copyright 2010-2011, Adi Sayoga

\* Dual licensed under the MIT or GPL Version 2 licenses.

\*/

(function($) {

/\*\* Menampilkan data per halaman \*/

$.fn.displayPagedData = function(options) {

options = $.extend($.fn.displayPagedData.defaults, options);

return this.each(function() {

var $tbody = $(this).find("tbody");

$tbody.empty();

$.each(options.data, function() {

var $row = $("<tr></tr>");

filterBulan = ((options.tahun)? '/tahun/' + options.tahun: '')

+ ((options.bulan)? '/bulan/' + options.bulan: '');

*// Action (update/delete)*

$row.append('<td class="action">'

+ '<a class="delete" href="' + options.url + '/delete/id/'

+ this.id + filterBulan + '"></a>'

+ '<a class="update" href="' + options.url + '/update/id/'

+ this.id + filterBulan + '"></a>'

+ '</td>');

*// Field*

for (var idx in options.fields) {

*// Konstanta*

var CELL\_PADDING = 3;

var DEPTH\_ALIGN = 20;

var POS\_DEBIT = 1;

var POS\_KREDIT = -1;

var DEFAULT\_ALIGN = "left";

var field = options.fields[idx];

var depth = (field.depth)? this[field.depth] \* DEPTH\_ALIGN

+ CELL\_PADDING: CELL\_PADDING;

var pos = (field.pos)? (this[field.pos] == POS\_KREDIT)?

DEPTH\_ALIGN + CELL\_PADDING: CELL\_PADDING: CELL\_PADDING;

var align = (field.align)? field.align: DEFAULT\_ALIGN;

$row.append('<td style="padding-left: ' + Math.max(depth, pos)

+ 'px; text-align: ' + align + '; ">' + this[field.name]

+ '</td>');

}

$tbody.append($row);

});

});

};

/\*\* Default options untuk data per halaman \*/

$.fn.displayPagedData.defaults = {

data: null, *// Data yang didapatkan dari server*

url: "", *// url ke server (controller)*

tahun: 0, *// filter tahun*

bulan: 0, *// filter tahun*

fields: null *// Daftar field yang disertakan yang dapat berupa:*

*// name, depth, pos, align*

};

/\*\* Control navigasi pada halaman \*/

$.fn.paginationControl = function(options) {

options = $.extend($.fn.paginationControl.defaults, options);

return this.each(function() {

$(this).empty();

filterBulan = ((options.tahun)? '/tahun/' + options.tahun: '')

+ ((options.bulan)? '/bulan/' + options.bulan: '');

*// Halaman sebelumnya*

var prev;

if (options.pagination.previous) {

prev = '<a href="' + options.url + '/page/'

+ options.pagination.previous + filterBulan

+ '" class="button">&lt Sebelumnya</a>';

} else {

prev = '<span class="button button-disabled">&lt

Sebelumnya</span>';

}

*// Daftar halaman*

var pages = "";

$.each(options.pagination.pagesInRange, function() {

if (this == options.pagination.current) {

pages += '<span class="button button-current">' + this

+ '</span>';

} else {

pages += '<a href="' + options.url + '/page/' + this

+ filterBulan + '" class="button">' + this + '</a>';

}

});

*// Halaman berikutnya*

var next;

if (options.pagination.next) {

next = '<a href="' + options.url + '/page/'

+ options.pagination.next + filterBulan

+ '" class="button">Berikutnya &gt;</a>';

} else {

next = '<span class="button button-disabled">Berikutnya &gt;

</span>';

}

$(this).append(prev).append(pages).append(next);

});

};

/\*\* Default options control navigasi \*/

$.fn.paginationControl.defaults = {

pagination: null, *// Objek (json) yang didapatkan dari server*

url: "", *// url ke server*

tahun: 0, *// filter tahun*

bulan: 0 *// filter bulan*

};

})(jQuery);

## File: /application/Bootstrap.php

<?php

class Bootstrap extends Zend\_Application\_Bootstrap\_Bootstrap

{

protected function \_initView()

{

*// Initialize view*

$view = new Zend\_View();

$view->doctype("XHTML1\_STRICT");

$view->headTitle("Sistem Informasi Akuntansi");

$view->headTitle()->setSeparator(" - ");

$view->skin = "default";

$view->addHelperPath(APPLICATION\_PATH . "/views/helpers",

"Zend\_View\_Helper\_");

*// JQuery*

$view->addHelperPath("ZendX/JQuery/View/Helper",

"ZendX\_JQuery\_View\_Helper");

$view->jQuery()->setLocalPath("/js/jquery-1.4.2.min.js")

->setUiLocalPath("/js/jquery-ui-1.8.custom.min.js")

->addJavascriptFile("/js/jquery.layout.min")

->addJavascriptFile("/js/jquery.zend.jsonrpc.js")

->addJavascriptFile("/js/json2.js")

->addJavascriptFile("/js/common.js")

->enable()->uiEnable();

$view->addHelperPath("Acc/View/Helper", "Acc\_View\_Helper");

*// Tambahkan ke ViewRenderer*

$viewRenderer = Zend\_Controller\_Action\_HelperBroker

::*getStaticHelper*("viewRenderer");

$viewRenderer->setView($view);

Zend\_Paginator::*setDefaultItemCountPerPage*(14);

*// Kembalikan nilainya, sehingga bisa disimpan oleh bootstrap*

return $view;

}

protected function \_initAutoLoad()

{

*// Tambahkan autoloader*

$autoLoader = Zend\_Loader\_Autoloader::*getInstance*();

$autoLoader->registerNamespace("Acc\_");

$resourceLoader = new Zend\_Loader\_Autoloader\_Resource(array(

"basePath" => APPLICATION\_PATH,

"namespace" => "",

"resourceTypes" => array(

"form" => array("path" => "forms/", "namespace" => "Form\_"),

"model" => array("path" => "models/", "namespace" => "Model\_")

),

));

return $autoLoader;

}

}

## File: /application/configs/application.ini

[production]

phpSettings.display\_startup\_errors = 0

phpSettings.display\_errors = 0

includePaths.library = APPLICATION\_PATH "/../library"

bootstrap.path = APPLICATION\_PATH "/Bootstrap.php"

bootstrap.class = "Bootstrap"

appnamespace = "Application"

resources.frontController.controllerDirectory = APPLICATION\_PATH

"/controllers"

resources.frontController.plugins.acl = "Acc\_Controller\_Plugin\_Acl"

resources.frontController.params.displayExceptions = 0

resources.frontController.moduleDirectory = APPLICATION\_PATH "/modules"

resources.layout.layoutPath = APPLICATION\_PATH "/layouts/scripts"

resources.modules[] = ""

resources.db.adapter = "pdo\_mysql"

resources.db.params.host = "localhost"

resources.db.params.username = "root"

resources.db.params.password = "s3cret"

resources.db.params.dbname = "accounting"

resources.db.isDefaultTableAdapter = true

[staging : production]

[testing : production]

phpSettings.display\_startup\_errors = 1

phpSettings.display\_errors = 1

[development : production]

phpSettings.display\_startup\_errors = 1

phpSettings.display\_errors = 1

resources.frontController.params.displayExceptions = 1

## File: /application/controllers/AccountController.php

<?php

class AccountController extends Zend\_Controller\_Action

{

public function init()

{

}

/\*\*

\* Index.

\*/

public function indexAction()

{

*// Tambahkan link javascript*

$this->view->JQuery()->addJavascriptFile("/js/jquery.pagination.js");

$accountModel = new Model\_Account();

$adapter = $accountModel->fetchPaginatorAdapter();

$paginator = new Zend\_Paginator($adapter);

$paginator->setPageRange(5);

$page = $this->getRequest()->getParam("page", 1);

$paginator->setCurrentPageNumber($page);

$this->view->paginator = $paginator;

}

/\*\*

\* Halaman untuk menambah data account.

\*/

public function createAction()

{

*// Tambahkan link javascript*

$this->view->JQuery()

->addJavascriptFile("/js/jquery.formValidation.js");

$accountForm = new Form\_Account();

$accountForm->setAction("/account/create");

$accountForm->setMethod("post");

if ($this->getRequest()->isPost() && $accountForm->isValid($\_POST)) {

$accountModel = new Model\_Account();

$result = $accountModel->createNew(array(

"parent" => $accountForm->getValue("parent"),

"after" => $accountForm->getValue("after"),

"kodeAccount" => $accountForm->getValue("kodeAccount"),

"account" => $accountForm->getValue("account"),

"normalPos" => $accountForm->getValue("normalPos"),

"kelompok" => $accountForm->getValue("kelompok")

));

*// Setelah disimpan, kosongkan kembali formnya*

if ($result) {

*// Data account telah berubah, update list account*

$accountData = $accountModel

->fetchKeyValue("id", "account", "kodeAccount");

if ($accountData) {

$accountForm->parent->clearMultiOptions();

$accountForm->parent->addMultiOption(0, "(Tidak Ada)");

$accountForm->parent->addMultiOptions($accountData);

$accountForm->after->clearMultiOptions();

$accountForm->after->addMultiOption(0, "(Paling Awal)");

$accountForm->after->addMultiOptions($accountData);

}

$accountForm->kodeAccount->setValue("");

$accountForm->account->setValue("");

$this->view->message = "Data akun berhasil ditambahkan.";

} else {

$this->view->message

= "Data akun gagal ditambahkan, cek kembali inputan!";

}

}

$this->view->form = $accountForm;

}

/\*\*

\* Halaman untuk mengubah data account.

\*/

public function updateAction()

{

*// Tambahkan link javascript*

$this->view->JQuery()

->addJavascriptFile("/js/jquery.formValidation.js");

$id = $this->getRequest()->getParam("id");

$accountModel = new Model\_Account();

$account = $accountModel->find($id)->current();

if (!$account) throw new Zend\_Exception("User tidak ditemukan.");

$accountForm = new Form\_Account();

$accountForm->setAction("/account/update");

$accountForm->setMethod("post");

$accountForm->removeElement("parent"); *// Parent tidak bisa di-update*

$accountForm->removeElement("after"); *// After tidak diperlukan*

$accountForm->setExclude($account->kodeAccount);

if ($this->getRequest()->isPost()) {

if ($accountForm->isValid($\_POST)) {

$result = $accountModel->updateData(

$accountForm->getValue("id"), array(

"kodeAccount" => $accountForm->getValue("kodeAccount"),

"account" => $accountForm->getValue("account"),

"normalPos" => $accountForm->getValue("normalPos"),

"kelompok" => $accountForm->getValue("kelompok")

));

return $this->\_forward("index");

}

} else {

$accountForm->populate($account->toArray());

}

$this->view->form = $accountForm;

$this->view->lastKodeAccount = $account->kodeAccount;

}

/\*\*

\* Halaman untuk menghapus data account.

\*/

public function deleteAction()

{

$accountModel = new Model\_Account();

$id = $this->getRequest()->getParam("id");

$accountModel->deleteData($id);

return $this->\_forward("index");

}

}

## File: /application/controllers/JournalController.php

<?php

class JournalController extends Zend\_Controller\_Action

{

protected $\_tipeJournal = Model\_Journal::*TIPE\_JOURNAL\_UMUM*;

protected $\_prefixAutoNum = "BJU";

/\*\*

\* Data account untuk comboBox

\* **@return** array

\*/

private function \_getAccountList()

{

$accountModel = new Model\_Account();

$accountList = $accountModel->fetchKeyValue("id", "account");

return $accountList;

}

/\*\*

\* Action untuk create/update

\*/

private function \_actionCreateUpdate()

{

*// Tambahkan link javascript*

$this->view->JQuery()->addJavascriptFile("/js/dateFormat.js");

$this->view->JQuery()

->addJavascriptFile("/js/jquery.formValidation.js");

$this->view->JQuery()->addJavascriptFile("/js/journal.js");

$journalModel = new Model\_Journal();

$dateNow = new Zend\_Date();

$id = $this->getRequest()->getParam("id");

$bukti = null;

$id && $bukti = $journalModel->fetchBukti($id);

if ($bukti) {

$lastNoBukti = $bukti["noBukti"];

} else {

$lastNoBukti = $journalModel->autoNum(

$dateNow, array("prefix" => $this->\_prefixAutoNum));

}

$journalForm = new Form\_Journal();

$journalForm->setExclude($lastNoBukti);

*// detail*

$details = array();

$totalDebit = 0;

$totalKredit = 0;

if ($this->getRequest()->isPost()) {

$kodeAccount = array();

$idAccount = $this->getRequest()->getParam("account");

$debit = $this->getRequest()->getParam("debit");

$kredit = $this->getRequest()->getParam("kredit");

$accountModel = new Model\_Account();

for ($i = 0; $i < count($idAccount); $i++) {

if (!($debit[$i] || $kredit[$i])

|| !$this->getRequest()->getParam("include$i")) continue;

*// Tampilkan kode account berdasarkan id account*

$account = $accountModel->find($idAccount[$i])->current();

if ($account) $kodeAccount[$i] = $account->kodeAccount;

$details[] = array(

"idAccount" => $idAccount[$i],

"kodeAccount" => $kodeAccount[$i],

"debit" => ($debit[$i])? $debit[$i]: 0,

"kredit" => ($kredit[$i])? $kredit[$i]: 0,

);

$totalDebit += $debit[$i];

$totalKredit += $kredit[$i];

*// validasi debit dan kredit harus balance, total harus sama*

*// dengan debit/kredit*

$isValid = true;

$uiState = "ui-state-error";

$uiIcon = "ui-icon-alert";

$message = "";

if ($totalDebit != $totalKredit) {

$message .= "<li>Debit dan kredit tidak balance</li>";

$isValid = false;

}

if ($this->getRequest()->getParam("total") != $totalDebit

|| $this->getRequest()->getParam("total") != $totalKredit) {

$message .= "<li>Total transaksi tidak sama dengan total

debit/kredit</li>";

$isValid = false;

}

*// form di-submit*

if ($journalForm->isValid($\_POST)) {

if ($isValid) {

*// generate data*

$data = array(

"noBukti"

=> $this->getRequest()->getParam("noBukti"),

"tipeJournal"

=> $this->getRequest()->getParam("tipeJournal"),

"tanggal"

=> $this->getRequest()->getParam("tanggal"),

"keterangan"

=> $this->getRequest()->getParam("keterangan"),

"total"

=> $this->getRequest()->getParam("total"),

"details"

=> $details,

);

*// Simpan perubahan*

if ($journalModel->updateData($this->getRequest()

->getParam("id"), $data)) {

$uiState = "ui-state-highlight";

$uiIcon = "ui-icon-info";

$message = "disimpan";

}

}

} else {

$message .= "<li>Data tidak valid, Periksa kembali

inputan!</li>";

}

*// tampilkan message*

$message && $message = "<ul style='list-style: none;

margin: 0; padding: 0; '>$message</ul>";

$this->view->uiState = $uiState;

$this->view->uiIcon = $uiIcon;

$this->view->message = $message;

}

} else {

if (!$bukti) { *// id tidak ada, diasumsikan menambah data baru*

*// set data default*

$journalForm->getElement("noBukti")->setValue($lastNoBukti);

$journalForm->getElement("tipeJournal")

->setValue($this->\_tipeJournal);

$journalForm->getElement("tanggal")

->setValue($dateNow->toString("dd/MM/yyyy"));

} else {

$journalForm->populate($bukti);

$details = $journalModel->fetchDetails($id);

foreach ($details as $detail) {

$totalDebit += $detail["debit"];

$totalKredit += $detail["kredit"];

}

}

}

$this->view->form = $journalForm;

$this->view->details = $details;

$this->view->totalDebit = $totalDebit;

$this->view->totalKredit = $totalKredit;

*// data account untuk comboBox*

$this->view->accountList = $this->\_getAccountList();

*//* **TODO** *Ada cara lebih baik?*

$this->view->headScript()->captureStart(); ?>

*lastNoBukti = "*<?php echo $lastNoBukti; ?>*";*

<?php $this->view->headScript()->captureEnd();

}

public function init()

{

}

/\*\*

\* Index.

\*/

public function indexAction()

{

*// Tambahkan link javascript*

$this->view->JQuery()->addJavascriptFile("/js/jquery.pagination.js");

*// filter bulan*

$tahun = $this->getRequest()->getParam("tahun");

$bulan = $this->getRequest()->getParam("bulan");

$tanggal = getdate();

$tahun || $tahun = $tanggal["year"];

$bulan || $bulan = $tanggal["mon"];

$journalModel = new Model\_JournalUmum();

$journalModel->setTipeJournal($this->\_tipeJournal);

$journalModel->setPeriode($tahun, $bulan);

$adapter = $journalModel->fetchPaginatorAdapter();

$paginator = new Zend\_Paginator($adapter);

$paginator->setPageRange(5);

$page = $this->getRequest()->getParam("page", 1);

$paginator->setCurrentPageNumber($page);

$this->view->paginator = $paginator;

$this->view->bulan = $bulan;

$this->view->tahun = $tahun;

}

/\*\*

\* Halaman untuk menambah data journal.

\*/

public function createAction() {

$this->\_actionCreateUpdate();

}

/\*\*

\* Halaman untuk mengubah data journal.

\*/

public function updateAction()

{

$this->\_actionCreateUpdate();

}

/\*\*

\* Halaman untuk menghapus data journal.

\*/

public function deleteAction()

{

$journalModel = new Model\_JournalUmum();

$id = $this->getRequest()->getParam("id");

$journalModel->deleteData($id);

return $this->\_forward("index");

}

}

## File: /application/controllers/JournalPenyesuaianController.php

<?php

/\*\*

\* Journal penyesuaian ini sama dengan journal umum dengan

\* perbedaan tipe journalnya

\*/

include\_once APPLICATION\_PATH . "/controllers/JournalController.php";

class JournalPenyesuaianController extends JournalController

{

protected $\_tipeJournal = Model\_Journal::*TIPE\_JOURNAL\_PENYESUAIAN*;

protected $\_prefixAutoNum = "BJP";

}

## File: /application/controllers/LaporanController.php

<?php

class LaporanController extends Zend\_Controller\_Action

{

public function init() {}

public function accountAction()

{

*// Laporan tidak memerlukan layout*

$this->view->layout()->disableLayout();

$accountModel = new Model\_Account();

$account = $accountModel->fetchArray();

$this->view->account = $account;

}

public function journalUmumAction()

{

$laporanForm = new Form\_Laporan();

$laporanForm->setAction("/laporan/journal-umum");

$laporanForm->setMethod("post");

$laporanForm->removeElement("idAccount");

if ($this->getRequest()->isPost() && $laporanForm->isValid($\_POST)) {

$this->view->layout()->disableLayout();

return $this->\_forward("view-journal-umum");

} else {

$this->view->form = $laporanForm;

}

}

public function viewJournalUmumAction()

{

$tglAwal = new Zend\_Date($this->\_request->getParam("tglAwal"));

$tglAkhir = new Zend\_Date($this->\_request->getParam("tglAkhir"));

$lapModel = new Model\_Laporan();

$journal = $lapModel->fetchJournalUmum($tglAwal

->get(Zend\_Date::*TIMESTAMP*),

$tglAkhir->get(Zend\_Date::*TIMESTAMP*));

$this->view->tglAwal = $tglAwal;

$this->view->tglAkhir = $tglAkhir;

$this->view->journal = $journal;

}

public function bukubesarAction()

{

$laporanForm = new Form\_Laporan();

$laporanForm->setAction("/laporan/bukubesar");

$laporanForm->setMethod("post");

if ($this->getRequest()->isPost() && $laporanForm->isValid($\_POST)) {

$this->view->layout()->disableLayout();

return $this->\_forward("view-bukubesar");

} else {

$this->view->form = $laporanForm;

}

}

public function viewBukubesarAction()

{

$tglAwal = new Zend\_Date($this->\_request->getParam("tglAwal"));

$tglAkhir = new Zend\_Date($this->\_request->getParam("tglAkhir"));

$idAccount = $this->\_request->getParam("idAccount");

$accountModel = new Model\_Account();

$accountRow = $accountModel->fetchRow("id = $idAccount");

$lapModel = new Model\_Laporan();

$bukubesar = $lapModel->fetchBukubesar($tglAwal

->get(Zend\_Date::*TIMESTAMP*),

$tglAkhir->get(Zend\_Date::*TIMESTAMP*), $idAccount);

$this->view->tglAwal = $tglAwal;

$this->view->tglAkhir = $tglAkhir;

$this->view->account = $accountRow;

$this->view->bukubesar = $bukubesar;

}

public function labarugiAction()

{

$laporanForm = new Form\_Laporan();

$laporanForm->removeElement("idAccount");

$laporanForm->setAction("/laporan/labarugi");

$laporanForm->setMethod("post");

if ($this->getRequest()->isPost() && $laporanForm->isValid($\_POST)) {

$this->view->layout()->disableLayout();

return $this->\_forward("view-labarugi");

} else {

$this->view->form = $laporanForm;

}

}

public function viewLabarugiAction()

{

$tglAwal = new Zend\_Date($this->\_request->getParam("tglAwal"));

$tglAkhir = new Zend\_Date($this->\_request->getParam("tglAkhir"));

$lapModel = new Model\_Laporan();

$labarugi = $lapModel->fetchLabarugi($tglAwal

->get(Zend\_Date::*TIMESTAMP*),

$tglAkhir->get(Zend\_Date::*TIMESTAMP*));

$this->view->tglAwal = $tglAwal;

$this->view->tglAkhir = $tglAkhir;

$this->view->labarugi = $labarugi;

}

public function neracaAction()

{

$laporanForm = new Form\_Laporan();

$laporanForm->removeElement("idAccount");

$laporanForm->setAction("/laporan/neraca");

$laporanForm->setMethod("post");

if ($this->getRequest()->isPost() && $laporanForm->isValid($\_POST)) {

$this->view->layout()->disableLayout();

return $this->\_forward("view-neraca");

} else {

$this->view->form = $laporanForm;

}

}

public function viewNeracaAction()

{

$tglAwal = new Zend\_Date($this->\_request->getParam("tglAwal"));

$tglAkhir = new Zend\_Date($this->\_request->getParam("tglAkhir"));

$lapModel = new Model\_Laporan();

$neraca = $lapModel->fetchNeraca($tglAwal->get(Zend\_Date::*TIMESTAMP*),

$tglAkhir->get(Zend\_Date::*TIMESTAMP*));

$this->view->tglAwal = $tglAwal;

$this->view->tglAkhir = $tglAkhir;

$this->view->neraca = $neraca;

}

}

## File: /application/controllers/UserController.php

<?php

class UserController extends Zend\_Controller\_Action

{

public function init() {}

/\*\*

\* Index

\*/

public function indexAction()

{

*// Tambahkan link javascript*

$this->view->JQuery()->addJavascriptFile("/js/jquery.pagination.js");

$userModel = new Model\_User();

$adapter = $userModel->fetchPaginatorAdapter();

$paginator = new Zend\_Paginator($adapter);

$paginator->setPageRange(5);

$page = $this->getRequest()->getParam("page", 1);

$paginator->setCurrentPageNumber($page);

$this->view->paginator = $paginator;

}

/\*\*

\* Halaman untuk menambah user data baru.

\*/

public function createAction()

{

*// Tambahkan link javascript*

$this->view->JQuery()

->addJavascriptFile("/js/jquery.formValidation.js");

$userForm = new Form\_User();

$userForm->setAction("/user/create");

$userForm->setMethod("post");

if ($this->getRequest()->isPost() && $userForm->isValid($\_POST)) {

$userModel = new Model\_User();

$result = $userModel->createNew(array(

"username" => $userForm->getValue("username"),

"password" => $userForm->getValue("password"),

"displayName" => $userForm->getValue("displayName"),

"role" => $userForm->getValue("role")

));

if ($result) {

*// Setelah disimpan, kosongkan kembali formnya*

$userForm->username->setValue("");

$userForm->password->setValue("");

$userForm->displayName->setValue("");

$userForm->role->setValue("");

$this->view->message = "Data user berhasil ditambahkan.";

} else {

$this->view->message = "Data user gagal ditambahkan,

cek kembali inputan!";

}

}

$this->view->form = $userForm;

}

/\*\*

\* Halaman untuk mengubah data user.

\*/

public function updateAction()

{

*// Tambahkan link javascript*

$this->view->JQuery()

->addJavascriptFile("/js/jquery.formValidation.js");

$id = $this->getRequest()->getParam("id");

$userModel = new Model\_User();

$user = $userModel->find($id)->current();

if (!$user) throw new Zend\_Exception("User tidak ditemukan.");

$userForm = new Form\_User();

$userForm->setAction("/user/update");

$userForm->setMethod("post");

$userForm->removeElement("password");

$userForm->setExclude($user->username);

if ($this->getRequest()->isPost()) {

if ($userForm->isValid($\_POST)) {

$result = $userModel->updateData($userForm->getValue("id"),

array(

"username" => $userForm->getValue("username"),

"password" => $userForm->getValue("password"),

"displayName" => $userForm->getValue("displayName"),

"role" => $userForm->getValue("role")

));

return $this->\_forward("index");

}

} else {

$userForm->populate($user->toArray());

}

$this->view->form = $userForm;

$this->view->lastUsername = $user->username;

}

/\*\*

\* Halaman untuk mengubah password.

\*/

public function updatePasswordAction()

{

*// Tambahkan link javascript*

$this->view->JQuery()

->addJavascriptFile("/js/jquery.formValidation.js");

$userModel = new Model\_User();

$userForm = new Form\_User();

$userForm->setAction("/user/update-password");

*// Hapus field yang tidak diperlukan*

$userForm->removeElement("username");

$userForm->removeElement("displayName");

$userForm->removeElement("role");

if ($this->getRequest()->isPost() && $userForm->isValid($\_POST)) {

$result = $userModel->updatePassword(

$userForm->getvalue("id"),

$userForm->getValue("password")

);

return $this->\_forward("index");

} else {

$id = $this->getRequest()->getParam("id");

$user = $userModel->find($id)->current();

if (!$user) throw new Zend\_Exception("User tidak ditemukan.");

$userForm->populate($user->toArray());

$this->view->user = $user;

}

$this->view->form = $userForm;

}

/\*\*

\* Halaman untuk menghapus data user.

\*/

public function deleteAction()

{

$userModel = new Model\_User();

$id = $this->getRequest()->getParam("id");

$userModel->deleteData($id);

return $this->\_forward("index");

}

/\*\*

\* Keadaan user, apakah sudah login atau tidak.

\*/

public function stateAction()

{

$auth = Zend\_Auth::*getInstance*();

if ($auth->hasIdentity()) {

$this->view->identity = $auth->getIdentity();

}

}

/\*\*

\* Login

\*/

public function loginAction()

{

*// Halaman yang akan diarahkan jika login berhasil*

$ref = $this->getRequest()->getParam("ref");

if (!$ref) $ref = "index";

$ref = "/$ref";

$loginForm = new Form\_Login();

$loginForm->setAction("/user/login");

if ($this->getRequest()->isPost() && $loginForm->isValid($\_POST)) {

$data = $loginForm->getValues();

$userModel = new Model\_User();

if ($userModel->login($data["username"], $data["password"])) {

return $this->\_redirect($ref);

} else {

$this->view->message = "Maaf, username dan/atau password

anda salah!";

}

}

$this->view->ref = $ref;

$this->view->form = $loginForm;

}

/\*\*

\* Logout

\*/

public function logoutAction()

{

$userModel = new Model\_User();

$userModel->logout();

}

}

## File: /application/forms/Account.php

<?php

/\*\*

\* Class untuk form data account.

\* **@author** Adi Sayoga

\*/

class Form\_Account extends Zend\_Form

{

public function \_\_construct($exclude = null, $options = null)

{

parent::*\_\_construct*($options);

$this->setExclude($exclude);

}

/\*\*

\* Exclude untuk validator NoRecordExists

\* **@param** string $exclude - Field value yang di-exclude pada

\* validator kodeAccount

\*/

public function setExclude($exclude)

{

if (!$exclude) return;

$kodeAccount = $this->getElement("kodeAccount");

if (!$kodeAccount) return;

$kodeAccount->removeValidator("Zend\_Validate\_Db\_NoRecordExists");

$kodeAccountValidator = new Zend\_Validate\_Db\_NoRecordExists(array(

"table" => "account", "field" => "kodeAccount",

"exclude" => array("field" => "kodeAccount", "value" => $exclude)

));

$kodeAccount->addValidator($kodeAccountValidator);

}

public function init()

{

$this->setAttrib("class", "form");

$accountModel = new Model\_Account();

$dataAccount = $accountModel->fetchKeyValue("id", "account",

"kodeAccount");

$id = $this->createElement('hidden', 'id');

$id->setDecorators(array("ViewHelper"));

*// Parent*

$parent = new Zend\_Form\_Element\_Select("parent");

$parent->setLabel("Parent:")->setRequired();

$parent->addMultiOption(0, "(Tidak Ada)");

$dataAccount && $parent->addMultiOptions($dataAccount);

*// Insert Setelah...*

$after = new Zend\_Form\_Element\_Select("after");

$after->setLabel("Insert Setelah:")->setRequired();

$after->addMultiOption(0, "(Paling Awal)");

$dataAccount && $after->addMultiOptions($dataAccount);

*// Kode Account*

$kodeAccount = new Zend\_Form\_Element\_Text("kodeAccount");

$kodeAccount->setLabel("Kode Akun:")->setRequired()

->setAttrib("size", "20");

$kodeAccount->addValidator(new Zend\_Validate\_Db\_NoRecordExists(

array("table" => "account", "field" => "kodeAccount")));

*// Account*

$account = new Zend\_Form\_Element\_Textarea("account");

$account->setLabel("Akun:")->setRequired();

$account->setAttribs(array("cols" => 100, "rows" => 2));

*// Normal Pos*

$normalPos = new Zend\_Form\_Element\_Select("normalPos");

$normalPos->setLabel("Normal Pos:")->setRequired();

$normalPos->addMultiOptions(array(1 => "Debit", -1 => "Kredit"));

*// Kelompok*

$kelompok = new Zend\_Form\_Element\_Select("kelompok");

$kelompok->setLabel("Kelompok:")->setRequired();

$kelompok->addMultiOptions(array(

"N" => "Neraca", "L" => "Laba/Rugi"));

*// Submit*

$submit = new Zend\_Form\_Element\_Submit("submit");

$submit->setLabel("Simpan");

$this->addElements(array($id, $parent, $after, $kodeAccount,

$account, $kelompok, $normalPos, $submit));

}

}

## File: /application/forms/Journal.php

<?php

/\*\*

\* Class untuk form journal.

\* **@author** Adi Sayoga

\*/

class Form\_Journal extends Zend\_Form

{

public function \_\_construct($exclude = null, $options = null)

{

parent::*\_\_construct*($options);

$this->setExclude($exclude);

}

/\*\*

\* Exclude untuk validator

\* **@param** string $exclude - Field value yang di-exclude pada validator

\* no bukti

\*/

public function setExclude($exclude)

{

if (!$exclude) return;

$noBukti = $this->getElement("noBukti");

if (!$noBukti) return;

$noBukti->removeValidator("Zend\_Validate\_Db\_NoRecordExists");

$noBuktiValidator = new Zend\_Validate\_Db\_NoRecordExists(array(

"table" => "bukti\_transaksi", "field" => "noBukti",

"exclude" => array("field" => "noBukti", "value" => $exclude)

));

$noBukti->addValidator($noBuktiValidator);

}

public function init()

{

$this->setAttrib("class", "form");

*// ID*

$id = new Zend\_Form\_Element\_Hidden("id");

$id->setDecorators(array("ViewHelper"));

*// Tipe Journal*

$tipeJournal = new Zend\_Form\_Element\_Hidden("tipeJournal");

$tipeJournal->setDecorators(array("ViewHelper"));

*// No. Bukti*

$noBukti = new Zend\_Form\_Element\_Text("noBukti");

$noBukti->setLabel("No. Bukti:")->setRequired()

->setAttrib("size", "20");

$noBukti->addValidator(new Zend\_Validate\_Db\_NoRecordExists(array(

"table" => "bukti\_transaksi", "field" => "noBukti")));

*// Tanggal*

$tanggal = new ZendX\_JQuery\_Form\_Element\_DatePicker("tanggal",

array('jQueryParams' => array("dateFormat" => "dd/mm/yy")));

$tanggal->addValidator(new Zend\_Validate\_Date(array(

"locale" => "id")));

$tanggal->setLabel("Tanggal:")->setRequired()

->setAttrib("size", "10");

*// Keterangan*

$keterangan = new Zend\_Form\_Element\_Textarea("keterangan");

$keterangan->setLabel("Keterangan:")->setRequired();

$keterangan->setAttribs(array("cols" => 100, "rows" => 2));

*// Total*

$total = new Zend\_Form\_Element\_Text("total");

$total->setLabel("Total:")->setRequired()->setValue("0");

*//* **TODO** *bermasalah dengan localization*

*//$total->addValidator(new Zend\_Validate\_Float());*

$this->addElements(array($id, $tipeJournal, $noBukti, $tanggal,

$keterangan, $total));

}

}

## File: /application/forms/Laporan.php

<?php

/\*\*

\* Class untuk form laporan.

\* **@author** Adi Sayoga

\*/

class Form\_Laporan extends Zend\_Form

{

public function init()

{

$this->setAttrib("class", "form");

*// Periode dalam 1 bulan*

$tglAwalValue = new Zend\_Date();

$tglAwalValue->set("1", Zend\_Date::*DAY*);

$tglAkhirValue = new Zend\_Date();

$tglAkhirValue->set("1", Zend\_Date::*DAY*);

$tglAkhirValue->add("1", Zend\_Date::*MONTH*);

$tglAkhirValue->sub("1", Zend\_Date::*DAY*);

*// Tanggal awal*

$tglAwal = new ZendX\_JQuery\_Form\_Element\_DatePicker("tglAwal",

array('jQueryParams' => array("dateFormat" => "dd/mm/yy")));

$tglAwal->addValidator(new Zend\_Validate\_Date(array(

"locale" => "id")));

$tglAwal->setLabel("Tanggal Awal:")->setRequired()

->setAttrib("size", "15");

$tglAwal->setValue($tglAwalValue->get("dd/MM/yyyy"));

*// Tanggal akhir*

$tglAkhir = new ZendX\_JQuery\_Form\_Element\_DatePicker("tglAkhir",

array('jQueryParams' => array("dateFormat" => "dd/mm/yy")));

$tglAkhir->addValidator(new Zend\_Validate\_Date(array(

"locale" => "id")));

$tglAkhir->setLabel("Tanggal Akhir:")->setRequired()

->setAttrib("size", "15");

$tglAkhir->setValue($tglAkhirValue->get("dd/MM/yyyy"));

*// Account*

$account = new Zend\_Form\_Element\_Select("idAccount");

$account->setLabel("Akun:")->setRequired();

$account->setTranslator();

$accountModel = new Model\_Account();

$dataAccount = $accountModel->fetchKeyValue("id", "account",

"kodeAccount");

if ($dataAccount) $account->addMultiOptions($dataAccount);

*// Submit*

$submit = new Zend\_Form\_Element\_Submit("submit");

$submit->setLabel("OK");

$this->addElements(array($tglAwal, $tglAkhir, $account, $submit));

}

}

## File: /application/forms/Login.php

<?php

/\*\*

\* Class untuk login user.

\* **@author** Adi Sayoga

\*/

class Form\_Login extends Zend\_Form

{

public function init()

{

$this->setAttrib("class", "form");

*// Buat elemen baru*

*// Username*

$username = new Zend\_Form\_Element\_Text("username");

$username->setLabel("Nama User:")->setRequired()

->setAttrib("size", "40");

$username->addFilter("StripTags");

*// Password*

$password = new Zend\_Form\_Element\_Password("password");

$password->setLabel("Password:")->setRequired()

->setAttrib("size", "40");

*// Submit*

$submit = new Zend\_Form\_Element\_Submit("submit");

$submit->setLabel("Login");

$this->addElements(array($username, $password, $submit));

}

}

## File: /application/forms/User.php

<?php

/\*\*

\* Class untuk form data user.

\* **@author** Adi Sayoga

\*/

class Form\_User extends Zend\_Form

{

public function \_\_construct($exclude = null, $options = null)

{

parent::*\_\_construct*($options);

$this->setExclude($exclude);

}

/\*\*

\* Exclude untuk validator

\* **@param** string $exclude - Field value yang di-exclude pada validator

\* username

\*/

public function setExclude($exclude)

{

if (!$exclude) return;

$username = $this->getElement("username");

if (!$username) return;

$username->removeValidator("Zend\_Validate\_Db\_NoRecordExists");

$usernameValidator = new Zend\_Validate\_Db\_NoRecordExists(array(

"table" => "users", "field" => "username",

"exclude" => array("field" => "username", "value" => $exclude)

));

$username->addValidator($usernameValidator);

}

public function init()

{

$this->setAttrib("class", "form");

*// Buat elemen baru*

*// Id*

$id = new Zend\_Form\_Element\_Hidden("id");

$id->setDecorators(array("ViewHelper"));

*// Username*

$username = new Zend\_Form\_Element\_Text("username");

$username->setLabel("Nama User:")->setRequired()

->setAttrib("size", "50");

$username->addFilter("StripTags");

$username->addValidator(new Zend\_Validate\_Db\_NoRecordExists(array(

"table" => "users", "field" => "username")));

*// Display name*

$displayName = new Zend\_Form\_Element\_Text("displayName");

$displayName->setLabel("Nama Ditampilkan:")->setRequired()

->setAttrib("size", "50");

$displayName->addFilter("StripTags");

*// Password*

$password = new Zend\_Form\_Element\_Password("password");

$password->setLabel("Password:")->setRequired()

->setAttrib("size", "50");

*// Role*

$role = new Zend\_Form\_Element\_Select("role");

$role->setLabel("Pilih Role:")->setRequired();

$role->addMultiOptions(array("user" => "User", "admin" =>

"Administrator"));

*// Submit*

$submit = new Zend\_Form\_Element\_Submit("submit");

$submit->setLabel("Simpan");

$this->addElements(array($id, $username, $displayName, $password,

$role, $submit));

}

}

## File: /application/layouts/scripts/Layout.phtml

<?php echo '<?xml version="1.0" encoding="UTF-8" ?>';

echo $this->doctype();

$this->layout()->userLogin = $this->action("state", "user");

?>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html;

charset=ISO-8859-1">

<link rel="icon" href="/images/favicon.ico" />

<link rel="icon" href="/images/favicon\_16x16.png" sizes="16x16" />

<link rel="icon" href="/images/favicon\_32x32.png" sizes="32x32" />

<link rel="icon" href="/images/favicon\_48x48.png" sizes="48x48" />

<?php

echo $this->headTitle();

echo $this->loadSkin($this->skin);

echo $this->headLink();

echo $this->jQuery();

echo $this->headScript();

?>

<script type="text/javascript">

*// Layout*

var layout;

$(function() {

layout = $("body").layout({

north: { resizable: false, closable: false, spacing\_open: 0 },

west: { size: 180, closable: false },

south: { resizable: false, closable: false, spacing\_open: 0 }

});

layout.allowOverflow("north");

});

</script>

</head>

<body>

*<!-- Header & Menu -->*

<div class="ui-layout-north">

*<!-- Header -->*

<div class="header">

<div class="message" style="position: *absolute*;

max-width: *600px*; ">*&nbsp;*</div>

<div class="header-title">

<div class="logo1"><a href="/"></a></div><div class="logo2">

<a href="/"></a></div>

<div class="title"><h1>*Sistem Informasi Akuntansi*</h1>

<h2>*PT. Bumi Sentosa*</h2></div>

<div class="user"><?php echo $this->layout()->userLogin; ?></div>

</div>

*<!-- Menu -->*

<div class="nav">

<ul class="menu">

*<!-- Home -->*

<li><a href="/" class="current">*Home*</a></li>

*<!-- Master -->*

<li><a href="#">*Master*</a>

<ul><li><a href="/account">*Data Akun*</a></li></ul>

</li>

*<!-- Transaksi -->*

<li><a href="#">*Transaksi*</a>

<ul>

<li><a href="/journal">*Transaksi Jurnal*</a></li>

<li><a href="/journal-penyesuaian">

*Jurnal Penyesuaian*</a></li>

</ul>

</li>

*<!-- Laporan -->*

<li><a href="#" class="last">*Laporan*</a>

<ul>

<li><a href="/laporan/account" target="blank">

*Laporan Akun*</a></li>

<li><a href="/laporan/journal-umum" target="blank">

*Laporan Jurnal Umum*</a></li>

<li><a href="/laporan/bukubesar" target="blank">

*Laporan Buku Besar*</a></li>

<li><a href="/laporan/labarugi" target="blank">

*Laporan Laba/Rugi*</a></li>

<li><a href="/laporan/neraca" target="blank">

*Laporan Neraca*</a></li>

</ul>

</li>

</ul>

<ul class="menu menu-right">

<li><a href="/settings">*Settings*</a></li>

</ul>

</div>

</div>

</div>

*<!-- Sidebar -->*

<div class="ui-layout-west">

*<!-- Master -->*

<h1>*Master*</h1>

<ul class="master">

<li><a href="/account">*Data Akun*</a></li>

</ul>

*<!-- Transaksi -->*

<h1>*Transaksi*</h1>

<ul class="transaksi">

<li><a href="/journal">*Transaksi Jurnal*</a></li>

<li><a href="/journal-penyesuaian">*Jurnal Penyesuaian*</a></li>

</ul>

*<!-- Laporan -->*

<h1>*Laporan*</h1>

<ul class="laporan">

<li><a href="/laporan/account" target="blank">

*Laporan Akun*</a></li>

<li><a href="/laporan/journal-umum" target="blank">

*Laporan Jurnal Umum*</a></li>

<li><a href="/laporan/bukubesar" target="blank">

*Laporan Buku Besar*</a></li>

<li><a href="/laporan/labarugi" target="blank">

*Laporan Laba/Rugi*</a></li>

<li><a href="/laporan/neraca" target="blank">

*Laporan Neraca*</a></li>

</ul>

</div>

*<!-- Content -->*

<div class="ui-layout-center">

<?php echo $this->layout()->content; ?>

</div>

*<!-- Footer -->*

<div class="ui-layout-south">

<div class="menu-footer">

<a href="#">*Privacy Statement*</a> *|*

<a href="#">*Term of Use*</a>

</div>

<div class="copyright"><em>*Copyright &copy; 2010-2011 by*

<a href="http://adisayoga.appspot.com" target="blank">

*Adi Sayoga*</a></em></div>

</div>

</body>

</html>

## File: /application/models/Account.php

<?php

/\*\*

\* Class untuk manage data account.

\* **@author** Adi Sayoga

\*/

class Model\_Account extends Zend\_Db\_Table\_Abstract implements Model\_Crud

{

const DELETE\_ALL\_CHILD = 1;

const ALL\_CHILD\_TO\_PARENT = 2;

const FIRSTCHILD\_TO\_PARENT = 3;

protected $\_name = "account";

private $\_deleteMethod = self::*DELETE\_ALL\_CHILD*;

public function fetchData($params = null) {

$select = $this->select();

*// Filter*

if (isset($params["filters"])) {

foreach ($params["filters"] as $field => $filter) {

if ($filter == null) {

$select->where("$field is null");

} else {

$select->where("$field = ?", $filter);

}

}

}

*// Sort*

isset($params["sortFields"])

&& $select->order($params["sortFields"]);

return $this->fetchAll($select);

}

/\*\*

\* Mendapatkan semua data dalam bentuk tree, disini menghasilkan

\* tambahan field yaitu <code>depth</code>: kedalaman dari node, dan

\* <code>hasChild</code>: ada atau tidaknya child.

\* **@param** array $params OPTIONAL - Parameter yang diterima:

\* <code>array filters</code> - filter pada where clause yang

\* berupa <code>key => value</code>.

\* **@return** array

\*/

public function fetchArray($params = null)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

*// Filter*

$whereExp = "";

if (isset($params["filters"])) {

if ($params["filters"] instanceof Zend\_Db\_Expr) {

$whereExp = $params["filter"]->\_\_toString();

} else {

foreach ($params["filters"] as $field => $filter) {

$whereExp += " AND " . ($filter == null)?

"node.$field is null":

$db->quoteInto("node.$field = ?", $filter);

}

}

}

$sql = "SELECT node.id, node.kodeAccount, node.account, "

. "node.normalPos, CASE node.normalPos WHEN 1 "

. "THEN 'Debit' ELSE 'Kredit' END AS normalPosKet, "

. "node.kelompok, CASE node.kelompok WHEN 'N' "

. "THEN 'Neraca' ELSE 'Laba/Rugi' END AS kelompokKet, "

. "(COUNT(parent.id) - 1) AS depth, "

. "CASE node.rgt - node.lft WHEN 1 THEN true ELSE false "

. " END AS hasChild "

. "FROM account AS node, account AS parent "

. "WHERE node.lft BETWEEN parent.lft AND parent.rgt $whereExp"

. "GROUP BY node.id "

. "ORDER BY node.lft";

return $db->fetchAll($sql);

}

/\*\*

\* Mendapatkan semua data dalam bentuk Zend\_Paginator\_Adapter\_Array.

\* **@return** Zend\_Paginator\_Adapter\_Array

\*/

public function fetchPaginatorAdapter($params = null) {

$data = $this->fetchArray($params);

$adapter = new Zend\_Paginator\_Adapter\_Array($data);

return $adapter;

}

/\*\*

\* Mendapatkan semua anak sesuai dengan id parent yang dilewatkan

\* (tidak termasuk anak dari anaknnya),

\* disini menghasilkan tambahan field yaitu

\* <code>depth</code>: kedalaman dari node, dan <code>hasChild</code>:

\* ada atau tidaknya child.

\*

\* **@param** int $parent OPTIONAL - Jika tidak dimasukkan, akan

\* mengembalikan semua data yang tidak mempunyai parent (root).

\* **@return** array

\*/

public function fetchChilds($parent = null)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

*// Generate sql*

if ($parent == null) {

*// $parent tidak ditentukan, select semua root*

$sql = "SELECT node.id, node.kodeAccount, node.account, "

. "node.normalPos, node.kelompok, null as parent, "

. "(COUNT(parent.id) - 1) AS depth, CASE "

. "node.rgt - node.lft WHEN 1 THEN true ELSE "

. "false END AS hasChild "

. "FROM account AS node, account AS parent "

. "WHERE node.lft BETWEEN parent.lft AND parent.rgt "

. "GROUP BY node.id "

. "HAVING depth = 0 "

. "ORDER BY node.lft";

} else {

*// Select anak dari $parent ini*

$parent = $db->quote($parent);

$sql = "SELECT node.id, node.kodeAccount, node.account, "

. "node.normalPos, node.kelompok, $parent as parent, "

. "CASE node.rgt - node.lft WHEN 1 THEN true ELSE false "

. "END AS hasChild, (COUNT(parent.id) - "

. "(subTree.depth + 1)) AS depth "

. "FROM account AS node, account AS parent, account "

. "AS subParent, "

. "( "

. "SELECT node.id, (COUNT(parent.id) - 1) AS depth "

. "FROM account AS node, account AS parent "

. "WHERE node.lft BETWEEN parent.lft AND parent.rgt "

. "AND node.id = $parent "

. "GROUP BY node.id "

. "ORDER BY node.lft "

. ") AS subTree "

. "WHERE node.lft BETWEEN parent.lft AND parent.rgt "

. "AND node.lft BETWEEN subParent.lft AND subParent.rgt "

. "AND subParent.id = subTree.id "

. "GROUP BY node.id "

. "HAVING depth = 1 "

. "ORDER BY node.lft";

}

return $db->fetchAll($sql);

}

/\*\*

\* Mendapatkan data pasangan key dan value ke dalam array.

\*

\* **@param** string $fieldKey

\* **@param** string $fieldValue

\* **@param** string $fieldKode - OPTIONAL

\* **@return** array

\*/

public function fetchKeyValue($fieldKey, $fieldValue, $fieldKode = "")

{

$data = array();

$rows = $this->fetchArray();

foreach ($rows as $row) {

$data[$row[$fieldKey]] = str\_repeat(". . . . ", $row["depth"])

. (($fieldKode)? " [" . $row[$fieldKode] . "] - ": "") . " "

. $row[$fieldValue];

}

return $data;

}

/\*\*

\* Mendapatkan data account berdasarkan id

\* **@param** int idAccount

\* **@return** array

\*/

public function getAccountById($idAccount)

{

$select = $this->select();

$select->where("id = ?", $idAccount);

$db = $this->getAdapter();

return $db->fetchAll($select);

}

/\*\*

\* Mendapatkan data account berdasarkan kode

\* **@param** string kodeAccount

\* **@return** array

\*/

public function getAccountByKode($kodeAccount)

{

$select = $this->select();

$select->where("kodeAccount = ?", $kodeAccount);

$db = $this->getAdapter();

return $db->fetchAll($select);

}

public function createNew($data)

{

*// Menentukan lft/rgt dari node*

if ($data["after"] == 0) { *// Insert paling awal*

$rowParent = $this->find($data["parent"])->current();

$no = ($rowParent)? $rowParent->lft: 0;

} else {

$rowParent = $this->find($data["after"])->current();

$no = ($rowParent)? $rowParent->rgt: 0;

}

*// Update lft dan rgt terlebih dahulu*

$adapter = $this->getAdapter();

$adapter->beginTransaction();

$this->update(array("lft" => new Zend\_Db\_Expr("lft + 2")),

$adapter->quoteInto("lft > ?", $no));

$this->update(array("rgt" => new Zend\_Db\_Expr("rgt + 2")),

$adapter->quoteInto("rgt > ?", $no));

*// Buat baris baru pada tabel account*

$row = $this->createRow();

if (!$row) {

$adapter->rollBack();

throw new Zend\_Exception("Tidak dapat membuat account baru!");

}

*// Set data pada baris baru*

$row->parent = ($data["parent"])? $data["parent"]: null;

$row->lft = $no + 1;

$row->rgt = $no + 2;

$row->kodeAccount = $data["kodeAccount"];

$row->account = $data["account"];

$row->normalPos = $data["normalPos"];

$row->kelompok = $data["kelompok"];

*// Simpan baris baru*

$result = $row->save();

$adapter->commit();

return $result;

}

public function updateData($id, $data)

{

*// Cari baris yang cocok dengan id*

$row = $this->find($id)->current();

if (!$row) throw new Zend\_Exception("Update gagal, data tidak

ditemukan!");

*// Update data*

$row->kodeAccount = $data["kodeAccount"];

$row->account = $data["account"];

$row->normalPos = $data["normalPos"];

$row->kelompok = $data["kelompok"];

*// Simpan baris yang di-update*

$row->save();

*// Update child*

$db = $this->getAdapter();

$db->update("account", array("kodeAccount" => new Zend\_Db\_Expr(

"concat('" . $data["kodeAccount"] . "', SubString(kodeAccount, "

. ($lenOldKode + 1) . ")) ")),

"Left(kodeAccount, $lenOldKode) = '$oldKodeAccount'");

return true;

}

public function deleteData($id)

{

$row = $this->find($id)->current();

if (!$row) throw new Zend\_Exception("Delete gagal, data tidak

ditemukan.");

$adapter = $this->getAdapter();

$adapter->beginTransaction();

$left = $row->lft;

$right = $row->rgt;

$width = $row->rgt - $row->lft + 1;

switch ($this->\_deleteMethod) {

case self::*DELETE\_ALL\_CHILD*:

*// Hapus beserta anak-anaknya*

$this->delete("lft BETWEEN $left AND $right");

*// Update lft dan rgt*

$this->update(array("lft" => new Zend\_Db\_Expr("lft - $width")),

"lft > $right");

$this->update(array("rgt" => new Zend\_Db\_Expr("rgt - $width")),

"rgt > $right");

break;

case self::*ALL\_CHILD\_TO\_PARENT*:

*// Hapus node ini*

$this->delete("lft = $left");

*// Update lft dan rgt*

$this->update(array("lft" => new Zend\_Db\_Expr("lft - 1"),

"rgt" => new Zend\_Db\_Expr("rgt - 1")),

"lft BETWEEN $left AND $right");

$this->update(array("lft" => new Zend\_Db\_Expr("lft - 2")),

"lft > $right");

$this->update(array("rgt" => new Zend\_Db\_Expr("rgt - 2")),

"rgt > $right");

break;

case self::*FIRSTCHILD\_TO\_PARENT*:

*//* **TODO** *Menghapus node account kemudian anak pertamanya dijadikan*

*// parent*

throw new Zend\_Exception("TODO Menghapus node account kemudian

anak pertamanya dijadikan parent");

break;

}

$adapter->commit();

return true;

}

/\*\*

\* Update field lft dan rgt pada tabel berdasarkan field parent.

\*/

public function rebuildTree()

{

*// mendapatkan top level tree (tidak mempunyai parent)*

$result = $this->fetchData(array("filters" => array(

"parent" => null)));

*// Rebuild child*

$left = 1; *// Nilai awal left = 1*

foreach ($result as $row) {

$left = $this->\_rebuildChild($row["id"], $left);

}

}

/\*\*

\* Fungsi rekursi untuk mengupdate semua anak-anaknya.

\*

\* **@param** int $parent

\* **@param** int $left

\* **@return** int

\*/

private function \_rebuildChild($parent, $left) {

*// Nilai right dari node ini adalah nilai left + 1*

$right = $left + 1;

*// Mendapatkan semua anak dari node ini*

$child = $this->fetchData(array("filters" => array(

"parent" => $parent)));

foreach ($child as $row) {

*// Eksekusi rekursi untuk setiap anak dari node ini.*

*// $right adalah nilai right saat ini, yang dinaikkan nilainya oleh*

*// fungsi ini.*

$right = $this->\_rebuildChild($row["id"], $right);

}

*// Kita telah mendapatkan nilai left, dan sekarang kita memproses*

*// anak dari node ini, kita juga tahu nilai right*

$$this->update(array("lft" => $left, "rgt" => $right),

"id = $parent");

*// Kembalikan nilai right dari node ini + 1*

return $right + 1;

}

}

## File: /application/models/Crud.php

<?php

/\*\*

\* Interface untuk Create, Read, Update, Delete data.

\* **@author** Adi Sayoga

\*/

interface Model\_Crud

{

/\*\*

\* Menambah data baru.

\* **@param** array $data

\* **@return** Primary key dari tabel yang baru saja ditambahkan.

\* **@throws** Zend\_Exception

\*/

public function createNew($data);

/\*\*

\* Mendapatkan semua data.

\* **@param** array $params OPTIONAL - Parameter yang diterima:

\* <code>array filters</code> - filter pada where clause yang berupa

\* <code>key => value</code>; <code>string sortFields </code> -

\* field untuk pengurutan.

\* **@return** Zend\_Db\_Table\_Rowset\_Abstract

\*/

public function fetchData($params = null);

/\*\*

\* Mendapatkan semua data dalam bentuk Zend\_Paginator\_Adapter\_DbSelect.

\* **@param** array $params OPTIONAL - Parameter yang diterima:

\* <code>array filters</code> - filter pada where clause yang

\* berupa <code>key => value</code>; <code>string sortFields

\* </code> - field untuk pengurutan.

\* **@return** Zend\_Paginator\_Adapter\_DbSelect

\*/

public function fetchPaginatorAdapter($params = null);

/\*\*

\* Mendapatkan semua data dalam bentuk array.

\* **@param** array $params OPTIONAL - Parameter yang diterima:

\* <code>array filters</code> - filter pada where clause yang berupa

\* <code>key => value</code>; <code>string sortFields </code> -

\* field untuk pengurutan.

\* **@return** array

\*/

public function fetchArray($params = null);

/\*\*

\* Mengupdate data berdasarkan id tabel.

\* **@param** int $id

\* **@param** array $data

\* **@return** bool - True jika update berhasil.

\* **@throws** Zend\_Exception

\*/

public function updateData($id, $data);

/\*\*

\* Menghapus data berdasarkan id tabel.

\* **@param** int $id

\* **@return** bool - True jika delete berhasil.

\* **@throws** Zend\_Exception

\*/

public function deleteData($id);

}

## File: /application/models/Journal.php

<?php

/\*\*

\* Class untuk manage data journal.

\* **@author** Adi Sayoga

\*/

class Model\_Journal extends Zend\_Db\_Table\_Abstract implements Model\_Crud

{

protected $\_name = "bukti\_transaksi"; *// Nama tabel yang digunakan*

protected $\_tipeJournal = ""; *// Tipe journal yang digunakan*

protected $\_tahun; *// Periode tahun*

protected $\_bulan; *// Periode bulan*

const TIPE\_JOURNAL\_UMUM = 1;

const TIPE\_JOURNAL\_PENYESUAIAN = 2;

/\*\*

\* Mendapatkan instansi dari objek Zend\_Db\_Table\_Select.

\* **@param** array $params

\* **@return** Zend\_Db\_Table\_Select

\*/

private function \_getSelect($params)

{

$select = new Zend\_Db\_Select($this->getAdapter());

$select->from(

array("b" => "bukti\_transaksi"),

array("id", "noBukti", "tipeJournal",

"tipeJournalKet" => new Zend\_Db\_Expr("CASE b.tipeJournal "

. "WHEN " . self::*TIPE\_JOURNAL\_UMUM* . " THEN 'Jurnal Umum' "

. "WHEN " . self::*TIPE\_JOURNAL\_PENYESUAIAN* . " THEN 'Jurnal "

. "Penyesuaian' "

. "ELSE 'Jurnal' END "),

"tanggal" => new Zend\_Db\_Expr("date\_format(from\_unixtime("

. "b.tanggal), '%d/%m/%Y')"),

"keterangan", "total" => new Zend\_Db\_Expr("format(total, "

. "'#.##0')"), "validatedBy",

"validatedDate" => new Zend\_Db\_Expr("date\_format(from\_unixtime("

. "b.validatedDate), "'%d/%m/%Y')")))

->join(

array("j" => "journal"), "b.id = j.idBukti",

array("idAccount",

"debit" => new Zend\_Db\_Expr("format(debit, '#.##0')"),

"kredit" => new Zend\_Db\_Expr("format(kredit, '#.##0')"),

"pos" => new Zend\_Db\_Expr("CASE j.debit WHEN 0 THEN -1 "

. "ELSE 1 END"),

"posKet" => new Zend\_Db\_Expr("CASE j.debit WHEN 0 THEN

. "'Kredit' ELSE 'Debit' END")))

->join(array("a" => "account"), "a.id = j.idAccount",

array("kodeAccount", "account"));

*// Tipe journal*

$this->\_tipeJournal

&& $select->where("tipeJournal = ?", $this->\_tipeJournal);

*// Periode*

$this->\_tahun

&& $select->where("year(from\_unixtime(tanggal)) = ?",

$this->\_tahun);

$this->\_bulan

&& $select->where("month(from\_unixtime(tanggal)) = ?",

$this->\_bulan);

*// Filter*

if (isset($params["filters"])) {

foreach ($params["filters"] as $field => $filter) {

if ($filter == null) {

$select->where("$field is null");

} else {

$select->where("$field = ?", $filter);

}

}

}

*// Sort*

isset($params["sortFields"])

&& $select->order($params["sortFields"]);

return $select;

}

/\*\*

\* Mengeset tipe jurnal (jurnal umum/jurnal penyesuaian)

\* **@param** int $tipeJournal

\*/

public function setTipeJournal($tipeJournal)

{

$this->\_tipeJournal = $tipeJournal;

}

/\*\*

\* Mengeset filter periode (tahun dan bulan)

\* **@param** $tahun int

\* **@param** $bulan int

\*/

public function setPeriode($tahun, $bulan)

{

$this->\_tahun = $tahun;

$this->\_bulan = $bulan;

}

public function fetchData($params = null)

{

$select = $this->\_getSelect($params);

return $this->fetchAll($select);

}

public function fetchPaginatorAdapter($params = null)

{

$select = $this->\_getSelect($params);

*// Ciptakan instansi bari dari paginator adapter*

$adapter = new Zend\_Paginator\_Adapter\_DbSelect($select);

return $adapter;

}

public function fetchArray($params = null)

{

$select = $this->\_getSelect($params);

$db = $this->getAdapter();

return $db->fetchAll($select);

}

/\*\*

\* Mendapatkan data hanya dari tabel bukti transaksi

\* **@param** int $id

\* **@return** array

\*/

public function fetchBukti($id)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

$select = $db->select();

$select->from("bukti\_transaksi",

array("id", "noBukti", "tipeJournal",

"tipeJournalKet" => new Zend\_Db\_Expr("CASE tipeJournal "

. "WHEN " . self::*TIPE\_JOURNAL\_UMUM* . " THEN 'Jurnal Umum' "

. "WHEN " . self::*TIPE\_JOURNAL\_PENYESUAIAN* . " THEN "

. "'Jurnal Penyesuaian' ELSE 'Jurnal' END "),

"tanggal" => new Zend\_Db\_Expr("date\_format(from\_unixtime("

. "tanggal), '%d/%m/%Y')"),

"keterangan", "total", "validatedBy",

"validatedDate" => new Zend\_Db\_Expr("date\_format(from\_unixtime("

. "validatedDate), '%d/%m/%Y')")))

->where("id = ?", $id);

return $db->fetchRow($select);

}

/\*\*

\* Mendapatkan semua data journal per bukti transaksi

\* **@param** int $id

\* **@return** array

\*/

public function fetchDetails($id)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

$select = $db->select();

$select->from(array("j" => "journal"))

->join(array("a" => "account"), "a.id = j.idAccount",

array("kodeAccount"))

->where("j.idBukti = ?", $id);

return $db->fetchAll($select);

}

public function createNew($data)

{

return $this->updateData(0, $data);

}

/\*\*

\* **@return** Primary key dari tabel

\*/

public function updateData($id, $data)

{

*//return $data;*

$db = $this->getAdapter();

$db->beginTransaction();

*// Cari baris yang cocok dengan id, jika tidak ditemukan maka tambah*

*// baru*

$row = $this->find($id)->current();

$row || $row = $this->createRow();

if (!$row) {

$db->rollBack();

throw new Zend\_Exception("Tidak dapat menambah atau mengubah

data.");

}

$now = new Zend\_Date();

$tanggal = new Zend\_Date($data["tanggal"]);

$auth = Zend\_Auth::*getInstance*();

$username = ($auth->hasIdentity())? $auth->getIdentity()->username:

"";

*// Set data pada baris baru*

$row->tipeJournal = $data["tipeJournal"];

$row->noBukti = $data["noBukti"];

$row->tanggal = $tanggal->get(Zend\_Date::*TIMESTAMP*);

$row->keterangan = $data["keterangan"];

$row->total = $data["total"];

$row->validatedBy = $username;

$row->validatedDate = $now->get(Zend\_Date::*TIMESTAMP*);

*// Simpan baris yang di-update*

$id = $row->save();

*// Hapus data terlebih dahulu*

$db->delete("journal", "idBukti = $id");

*// Kemudian insert detail baru*

foreach ($data["details"] as $detail) {

$db->insert("journal", array(

"idBukti" => $id,

"idAccount" => $detail["idAccount"],

"debit" => $detail["debit"],

"kredit" => $detail["kredit"]

));

}

$db->commit();

return $id;

}

public function deleteData($id)

{

*// Cari baris yang cocok dengan id*

$row = $this->find($id)->current();

if (!$row) throw new Zend\_Exception("$id Delete gagal, data tidak

ditemukan!");

$row->delete();

return true;

}

/\*\*

\* <p>Membuat nomor bukti otomatis sesuai dengan tipe journal dan

\* tanggal dan dengan format yang ditentukan.</p>

\* **@param** Zend\_Date $date - Tanggal dengan format timestamp

\* **@param** array $pattern - Pola yang digunakan. pattern[prefix],

\* pattern[dateFormat], pattern[digitNum]

\* **@return** string

\*/

public function autoNum($date, $pattern)

{

$defaultPattern = array("prefix" => "BJU", "dateFormat" => "yyMM",

"digitNum" => 4);

*// prefix*

$prefix = "";

if (isset($pattern["dateFormat"])) {

$pattern["dateFormat"]

&& $prefix .= $date->toString($pattern["dateFormat"]) . "/";

} else {

$prefix .= $date->toString($defaultPattern["dateFormat"]) . "/";

}

if (isset($pattern["prefix"])) {

$pattern["prefix"] && $prefix .= $pattern["prefix"] . "/";

} else {

$prefix .= $defaultPattern["prefix"] . "/";

}

$lenPrefix = strlen($prefix);

*// jumlah digit*

if (isset($pattern["digitNum"])) {

$pattern["digitNum"] && $digitNum = $pattern["digitNum"];

} else {

$digitNum = $defaultPattern["digitNum"];

}

*// sql: SELECT MAX(noBukti) FROM bukti\_transaksi*

*// WHERE LEFT(noBukti, [lenPrefix]) = [prefix]*

$select = $this->select();

$select->from("bukti\_transaksi", array(

"noBukti" => new Zend\_Db\_Expr("MAX(noBukti)")));

$select->where("LEFT(noBukti, $lenPrefix) = ?", $prefix);

$db = $this->getAdapter();

$noBukti = $db->fetchOne($select);

*// Nomor otomatis*

$no = 1;

$noBukti && $no = intval(substr($noBukti, $lenPrefix + 1)) + 1;

return $prefix . substr(str\_repeat("0", $digitNum)

. $no, -$digitNum);

}

}

## File: /application/models/JournalPenyesuaian.php

<?php

/\*\*

\* Class untuk manage data journal penyesuaian.

\* **@author** Adi Sayoga

\*/

class Model\_JournalPenyesuaian extends Model\_Journal

{

protected $\_tipeJournal = self::*TIPE\_JOURNAL\_PENYESUAIAN*;

}

## File: /application/models/JournalUmum.php

<?php

/\*\*

\* Class untuk manage data journal umum.

\* **@author** Adi Sayoga

\*/

class Model\_JournalUmum extends Model\_Journal

{

protected $\_tipeJournal = self::*TIPE\_JOURNAL\_UMUM*;

}

## File: /application/models/JsonRpc.php

<?php

class Model\_JsonRpc

{

/\* ------------------------------------------------------------------

\* Account

\* ------------------------------------------------------------------\*/

/\*\*

\* Validasi form account.

\* **@param** array $data

\* **@param** string $exclude

\* **@return** array

\*/

public function validateAccountForm($data, $exclude = null)

{

$accountForm = new Form\_Account($exclude);

foreach ($accountForm->getElements() as $element) {

$elementId = $element->getId();

if ($element->getType() != "Zend\_Form\_Element\_Hidden"

&& !key\_exists($elementId, $data)) {

*// Hapus element yang tidak diikutsertakan dalam validasi*

$accountForm->removeElement($elementId);

}

}

$accountForm->isValid($data);

$messages = $accountForm->getMessages();

return $messages;

}

/\*\*

\* Mendapatkan semua data account.

\* **@return** array

\*/

public function fetchAccount()

{

$accountModel = new Model\_Account();

return $accountModel->fetchArray();

}

/\*\*

\* Mendapatkan data account per halaman.

\* **@param** int $page

\* **@return** returns the items for a given page.

\*/

public function fetchAccountPaginator($page)

{

$accountModel = new Model\_Account();

$adapter = $accountModel->fetchPaginatorAdapter();

$paginator = new Zend\_Paginator($adapter);

$paginator->setPageRange(5);

$page || $page = 1;

$paginator->setCurrentPageNumber($page);

$result = array(

"pagination" => $paginator->getPages("sliding"),

"data" => $paginator->getCurrentItems()

);

return $result;

}

/\*\*

\* Mendapatkan data pasangan key dan value ke dalam array.

\* **@return** array

\*/

public function fetchAccountKeyValue()

{

$accountModel = new Model\_Account();

return $accountModel->fetchKeyValue("id", "account", "kodeAccount");

}

/\*\*

\* Mendapatkan semua anak sesuai dengan id parent yang dilewatkan

\* (tidak termasuk anak dari anaknnya),

\* disini menghasilkan tambahan field yaitu

\* <code>depth</code>: kedalaman dari node, dan <code>hasChild</code>:

\* ada atau tidaknya child.

\*

\* **@param** int $parent OPTIONAL - Jika tidak dimasukkan, akan

\* mengembalikan semua data yang tidak mempunyai parent (root).

\* **@return** array

\*/

public function fetchAccountChilds($parent = null)

{

$accountModel = new Model\_Account();

return $accountModel->fetchChilds($parent);

}

/\*\*

\* Mendapatkan data account berdasarkan id account

\* **@param** int $idAccount

\* **@return** array

\*/

public function getAccountById($idAccount)

{

$accountModel = new Model\_Account();

return $accountModel->getAccountById($idAccount);

}

/\*\*

\* Mendapatkan data account berdasarkan kode account

\* **@param** string $kodeAccount

\* **@return** array

\*/

public function getAccountByKode($kodeAccount)

{

$accountModel = new Model\_Account();

return $accountModel->getAccountByKode($kodeAccount);

}

/\*\*

\* Menambah data account

\* **@param** array $data

\* **@return** Primary key dari tabel yang baru saja ditambahkan.

\*/

public function addAccount($data)

{

$accountModel = new Model\_Account();

return $accountModel->createNew($data);

}

/\*\*

\* Mengubah data account

\* **@param** int $id

\* **@param** array $data

\* **@return** bool - true jika update berhasil

\*/

public function updateAccount($id, $data)

{

$accountModel = new Model\_Account();

return $accountModel->updateData($id, $data);

}

/\*\*

\* Menghapus data account

\* **@param** int $id

\* **@return** bool - True jika delete berhasil

\*/

public function deleteAccount($id)

{

$accountModel = new Model\_Account();

return $accountModel->deleteData($id);

}

/\* ------------------------------------------------------------------

\* Journal

\* ------------------------------------------------------------------\*/

/\*\*

\* Validasi form journal.

\* **@param** array $data

\* **@param** string $exclude

\* **@return** array

\*/

public function validateJournalForm($data, $exclude = null)

{

$journalForm = new Form\_Journal($exclude);

foreach ($journalForm->getElements() as $element) {

$elementId = $element->getId();

if ($element->getType() != "Zend\_Form\_Element\_Hidden"

&& !key\_exists($elementId, $data)) {

*// Hapus element yang tidak diikutsertakan dalam validasi*

$journalForm->removeElement($elementId);

}

}

$journalForm->isValid($data);

$messages = $journalForm->getMessages();

return $messages;

}

/\*\*

\* Mendapatkan data journal per halaman.

\* **@param** int $page

\* **@param** int $tipeJournal

\* **@param** int $tahun

\* **@param** int $bulan

\* **@return** returns the items for a given page.

\*/

public function fetchJournalPaginator($page, $tipeJournal, $tahun,

$bulan)

{

$journalModel = new Model\_Journal();

$journalModel->setTipeJournal($tipeJournal);

$journalModel->setPeriode($tahun, $bulan);

$adapter = $journalModel->fetchPaginatorAdapter();

$paginator = new Zend\_Paginator($adapter);

$paginator->setPageRange(5);

$page || $page = 1;

$paginator->setCurrentPageNumber($page);

$result = array(

"pagination" => $paginator->getPages("sliding"),

"data" => $paginator->getCurrentItems()

);

return $result;

}

/\*\*

\* <p>Membuat nomor bukti otomatis sesuai dengan tipe journal dan

\* tanggal dan dengan

\* format yang ditentukan.</p>

\* **@param** Zend\_Date $date - Tanggal dengan format timestamp

\* **@param** array $pattern - Pola yang digunakan. pattern[prefix],

\* pattern[dateFormat], pattern[digitNum]

\* **@return** string

\*/

public function journalAutoNum($date, $pattern)

{

$journalModel = new Model\_Journal();

return $journalModel->autoNum(new Zend\_Date($date), $pattern);

}

/\*\*

\* Menambah data journal

\* **@param** array $data

\* **@return** Primary key dari tabel yang baru saja ditambahkan.

\*/

public function addJournal($data)

{

$journalModel = new Model\_Journal();

return $journalModel->createNew($data);

}

/\*\*

\* Mengupdate data journal berdasarkan id bukti transaksi

\* **@param** int $id

\* **@param** array $data

\*/

public function updateJournal($id, $data)

{

$journalModel = new Model\_Journal();

return $journalModel->updateData($id, $data);

}

/\*\*

\* Menghapus data journal

\* **@param** int $id

\* **@return** bool - True jika delete berhasil

\*/

public function deleteJournal($id)

{

$journalModel = new Model\_Journal();

return $journalModel->deleteData($id);

}

/\* ------------------------------------------------------------------

\* User

\* ------------------------------------------------------------------\*/

/\*\*

\* Validasi form user.

\* **@param** array $data

\* **@param** string $eclude

\* **@return** array

\*/

public function validateUserForm($data, $exclude = null)

{

$userForm = new Form\_User($exclude);

foreach ($userForm->getElements() as $element) {

$elementId = $element->getId();

*// Hapus element yang tidak diikutsertakan dalam validasi kecuali*

*// hidden element*

if ($element->getType() != "Zend\_Form\_Element\_Hidden"

&& !key\_exists($elementId, $data)) {

$userForm->removeElement($elementId);

}

}

$userForm->isValid($data);

$messages = $userForm->getMessages();

return $messages;

}

/\*\*

\* Mendapatkan data user per halaman.

\* **@param** int $page

\* **@return** returns the items for a given page.

\*/

public function fetchUserPaginator($page)

{

$userModel = new Model\_User();

$adapter = $userModel->fetchPaginatorAdapter();

$paginator = new Zend\_Paginator($adapter);

$paginator->setPageRange(5);

$page || $page = 1;

$paginator->setCurrentPageNumber($page);

$result = array(

"pagination" => $paginator->getPages("sliding"),

"data" => $paginator->getCurrentItems()

);

return $result;

}

/\*\*

\* Menambah data user

\* **@param** array $data

\* **@return** Primary key dari tabel yang baru saja ditambahkan.

\*/

public function addUser($data)

{

$userModel = new Model\_User();

return $userModel->createNew($data);

}

/\*\*

\* Mengupdate data user

\* **@param** int $id

\* **@param** array $data

\* **@return** bool - true jika berhasil diupdate, false lainnya

\*/

public function updateUser($id, $data)

{

$userModel = new Model\_User();

return $userModel->updateData($id, $data);

}

/\*\*

\* Mengubah password

\* **@param** int $id

\* **@param** string $password

\* **@return** bool - true jika update berhasil

\*/

public function updatePassword($id, $password)

{

$userModel = new Model\_User();

return $userModel->updatePassword($id, $password);

}

/\*\*

\* Menghapus data user

\* **@param** int $id

\* **@return** bool - True jika delete berhasil

\*/

public function deleteUser($id)

{

$userModel = new Model\_User();

return $userModel->deleteData($id);

}

/\*\*

\* Login

\* **@param** string $username

\* **@param** string $password

\*/

public function login($username, $password)

{

$userModel = new Model\_User();

return $userModel->login($username, $password);

}

}

## File: /application/models/Laporan.php

<?php

/\*\*

\* Class untuk manage laporan.

\* **@author** Adi Sayoga

\*/

class Model\_Laporan

{

const ROOT\_AKTIVA = "1";

const ROOT\_PASIVA = "2";

const ROOT\_MODAL = "3";

const ROOT\_PENDAPATAN = "4";

const ROOT\_BIAYA = "5";

const ROOT\_PENDAPATAN\_NON\_OPERASIONAL = "6.01.01";

const ROOT\_BIAYA\_NON\_OPRASIONAL = "6.02.01";

const LABA\_DITAHAN = "3.02";

const LABARUGI\_BERJALAN = "3.03";

const DEFAULT\_DISPLAY\_LEVEL = 2;

/\*\*

\* Implode array

\* **@param** string $glue

\* **@param** array $pieces

\* **@return** string - data yang sudah di-escape

\*/

private function \_implode($glue, $pieces) {

$db = Zend\_Db\_Table::*getDefaultAdapter*();

$result = "";

foreach ($pieces as $piece) {

if ($result) $result .= $glue;

$result .= $db->quote($piece);

}

return $result;

}

/\*\*

\* Mendapatkan data account (id, lft, rgt) berdasarkan kode account

\* **@param** string $kodeAccount

\* **@return** Zend\_Db\_Table\_Row\_Abstract

\*/

private function \_fetchRowAccount($kodeAccount)

{

$accountModel = new Model\_Account();

$db = Zend\_Db\_Table::*getDefaultAdapter*();

$result = $accountModel->fetchRow($db->quoteInto("kodeAccount = ?",

$kodeAccount));

if (!$result) {

return $result;

} else {

throw new Zend\_Exception("Kode akun tidak ditemukan.");

}

}

/\*\*

\* Mendapatkan nilai saldo journal (debit - kredit) berdasarkan account

\* (beserta anak-anaknya).

\* Jika tanggal awal = 0, maka akan dihitung dari paling awal sampai

\* tanggal akhir.

\* **@param** string|array $root - Kode root account

\* **@param** int $tglAwal - Periode tanggal awal

\* **@param** int $tglAkhir - Periode tanggal akhir

\* **@return** money

\*/

private function \_getTotalTrans($root, $tglAwal, $tglAkhir)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

*//* **TODO** *Kode tidak selevel apakah menghasilkan data yang benar?*

$rootList = (is\_array($root)) ? $this->\_implode(", ", $root): $root;

$root = $db->quote($root);

$tglAwal = $db->quote($tglAwal);

$tglAkhir = $db->quote($tglAkhir);

$periode = ($tglAwal == "0") ? "b.tanggal <= $tglAkhir":

"b.tanggal between $tglAwal and $tglAkhir";

$sql = "select ifnull(sum(j.saldo), 0) as jumlah "

. "from account as node "

. "join ( "

. "select id, lft, rgt from account where kodeAccount "

. "in($rootList) "

. ") as parent "

. "join ( "

. "select j.idAccount, sum(j.debit - j.kredit) as saldo "

. "from bukti\_transaksi as b "

. "inner join journal as j on b.id = j.idBukti "

. "inner join account as a on a.id = j.idAccount "

. "where $periode "

. "group by j.idAccount "

. ") as j "

. "where node.lft between parent.lft and parent.rgt and "

. "node.id = j.idAccount ";

$result = $db->fetchOne($sql);

if (!$result) $result = 0;

return $result;

}

/\*\*

\* Mendapatkan nilai laba/rugi.

\* Jika tanggal awal = 0, maka akan dihitung dari paling awal sampai

\* tanggal akhir.

\* **@param** int $tglAwal - Periode tanggal awal

\* **@param** int $tglAkhir - Periode tanggal akhir

\* **@return** money

\*/

private function \_getLabaRugi($tglAwal, $tglAkhir)

{

$result = $this->\_getTotalTrans(array(

self::*ROOT\_PENDAPATAN*,

self::*ROOT\_BIAYA*,

self::*ROOT\_PENDAPATAN\_NON\_OPERASIONAL*,

self::*ROOT\_BIAYA\_NON\_OPRASIONAL*),

$tglAwal, $tglAkhir);

return $result;

}

/\*\*

\* Mendapatkan jumlah journal (debit, kredit) berdasarkan account per

\* periode awal s/d akhir.

\* Jika tanggal awal = 0, maka akan dihitung dari paling awal sampai

\* tanggal akhir.

\* **@param** int $root - Kode root account

\* **@param** int $tglAwal - Periode tanggal awal

\* **@param** int $tglAkhir - Periode tanggal akhir

\* **@param** int $maxLevel - Maksimum level yang ditampilkan

\* **@return** array - Berupa data account (tree) dan nilai debit/kreditnya

\*/

private function \_fetchTransPeriode($root, $tglAwal, $tglAkhir,

$maxLevel)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

$root = $db->quote($root);

$tglAwal = $db->quote($tglAwal);

$tglAkhir = $db->quote($tglAkhir);

$maxLevel = $db->quote($maxLevel);

$periode = ($tglAwal == "'0'") ? "b.tanggal <= $tglAkhir":

"b.tanggal between $tglAwal and $tglAkhir";

$sql = "select node.id, node.kodeAccount, node.account, "

. "count(parent.id) - (subTree.lvl + 1) as lvl, "

. "ifnull(j.debit, 0) as debit, ifnull(j.kredit, 0) "

. "as kredit, ifnull(j.saldo, 0) as saldo, "

. "case when (node.lft = node.rgt - 1) or "

. "(count(parent.id) - (subTree.lvl + 1) = "

. "$maxLevel) then true else false end as isLeaf "

*// Node*

. "from account as node "

*// Parent*

. "join account as parent "

*// Sub Parent*

. "join account as subParent "

*// Sub Tree*

. "join ( "

. "select node.id, (count(parent.id) - 1) as lvl "

. "from account node join account parent "

. "where node.lft between parent.lft and parent.rgt and "

. "node.kodeAccount = $root "

. "group by node.id "

. "order by node.lft "

. ") as subTree "

*// Total Transaksi*

. "left join ( "

. "select parent.id as idAccount, sum(j.debit) as debit, "

. "sum(j.kredit) as kredit, sum(j.debit - j.kredit) \* "

. "parent.normalPos as saldo "

. "from account as node "

. "join account as parent "

. "join ( "

. "select j.idAccount, j.debit, j.kredit "

. "from bukti\_transaksi b "

. "inner join journal j on b.id = j.idBukti "

. "where $periode "

. ") as j "

. "where (node.lft between parent.lft and parent.rgt) and "

. "(node.id = j.idAccount) "

. "group by parent.id "

. "order by parent.lft "

. ") as j on node.id = j.idAccount "

. "where node.lft between parent.lft and parent.rgt "

. "and node.lft between subParent.lft and subParent.rgt "

. "and subParent.id = subTree.id "

. "group by node.id "

. "having lvl <= $maxLevel "

. "order by node.lft";

$result = $db->fetchAll($sql);

return $this->\_insertTotal($result);

}

/\*\*

\* Menyisipkan total pada baris data

\* **@param** array $data

\* **@return** array - $data yang sudah di-insertkan total

\*/

private function \_insertTotal($data)

{

$oldLevel = 0;

$oldHasChild = false;

$oldKode = array("1" => "", "2" => "", "3" => "");

$oldAccount = array("1" => "", "2" => "", "3" => "");

$oldDebit = array("1" => 0, "2" => 0, "3" => 0);

$oldKredit = array("1" => 0, "2" => 0, "3" => 0);

$oldTotal = array("1" => 0, "2" => 0, "3" => 0);

$result = array();

foreach ($data as $row) {

*// Mengecek apakah mempunyai anak/tidak. Jika ya, tandai induk*

*// has\_child = 1*

*// Level saat ini lebih besar dari level sebelumnya, ini artinya*

*// loop memasuki level anaknya*

if ($row["lvl"] > $oldLevel) { $oldHasChild = true; }

*// Mengecek subtotal*

if ($row["lvl"] < $oldLevel) {

*// Akhir dari level anak. Dengan kata lain, loop kembali ke*

*// induknya*

*// Level 2*

if ($row["lvl"] <= 2 && $oldLevel > 2) {

$result[] = array(

"id" => 0,

"kodeAccount" => $oldKode["2"] . "-TOTAL",

"account" => "Total " . $oldAccount["2"],

"lvl" => 2,

"debit" => $oldDebit["2"],

"kredit" => $oldKredit["2"],

"saldo" => $oldTotal["2"],

"isLeaf" => 0,

);

}

*// Level 1*

if ($row["lvl"] <= 1 && $oldLevel > 1) {

$result[] = array(

"id" => 0,

"kodeAccount" => $oldKode["1"] . "-TOTAL",

"account" => "Total " . $oldAccount["1"],

"lvl" => 1,

"debit" => $oldDebit["1"],

"kredit" => $oldKredit["1"],

"saldo" => $oldTotal["1"],

"isLeaf" => 0,

);

}

*// Level 0*

if ($row["lvl"] <= 0 && $oldLevel > 0) {

$result[] = array(

"id" => 0,

"kodeAccount" => $oldKode["0"] . "-TOTAL",

"account" => "Total " . $oldAccount["0"],

"lvl" => 0,

"debit" => $oldDebit["0"],

"kredit" => $oldKredit["0"],

"saldo" => $oldTotal["0"],

"isLeaf" => 0,

);

}

}

if ($row["lvl"] == 2) {

$oldKode["2"] = $row["kodeAccount"];

$oldAccount["2"] = $row["account"];

$oldDebit["2"] = $row["debit"];

$oldKredit["2"] = $row["kredit"];

$oldTotal["2"] = $row["saldo"];

}

if ($row["lvl"] == 1) {

$oldKode["1"] = $row["kodeAccount"];

$oldAccount["1"] = $row["account"];

$oldDebit["1"] = $row["debit"];

$oldKredit["1"] = $row["kredit"];

$oldTotal["1"] = $row["saldo"];

}

if ($row["lvl"] == 0) {

$oldKode["0"] = $row["kodeAccount"];

$oldAccount["0"] = $row["account"];

$oldDebit["0"] = $row["debit"];

$oldKredit["0"] = $row["kredit"];

$oldTotal["0"] = $row["saldo"];

}

$oldLevel = $row["lvl"];

$oldHasChild = false; *// reset hasChild*

$result[] = $row;

}

*// Tambahkan total untuk diakhir*

*// Level 2*

if ($oldLevel > 2) {

$result[] = array(

"id" => 0,

"kodeAccount" => $oldKode["2"] . "-TOTAL",

"account" => "Total " . $oldAccount["2"],

"lvl" => 2,

"debit" => $oldDebit["2"],

"kredit" => $oldKredit["2"],

"saldo" => $oldTotal["2"],

"isLeaf" => 0,

);

}

*// Level 1*

if ($oldLevel > 1) {

$result[] = array(

"id" => 0,

"kodeAccount" => $oldKode["1"] . "-TOTAL",

"account" => "Total " . $oldAccount["1"],

"lvl" => 1,

"debit" => $oldDebit["1"],

"kredit" => $oldKredit["1"],

"saldo" => $oldTotal["1"],

"isLeaf" => 0,

);

}

*// Level 0*

if ($oldLevel > 0) {

$result[] = array(

"id" => 0,

"kodeAccount" => $oldKode["0"] . "-TOTAL",

"account" => "Total " . $oldAccount["0"],

"lvl" => 0,

"debit" => $oldDebit["0"],

"kredit" => $oldKredit["0"],

"saldo" => $oldTotal["0"],

"isLeaf" => 0,

);

}

return $result;

}

/\*\*

\* Mendapatkan laporan journal umum

\* **@param** int $tglAwal

\* **@param** int $tglAkhir

\* **@return** array

\*/

public function fetchJournalUmum($tglAwal, $tglAkhir)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

$select = new Zend\_Db\_Select($db);

$select->from(array("b" => "bukti\_transaksi"),

array("tipeJournal", "noBukti", "keterangan", "tanggal"

=> new Zend\_Db\_Expr("date\_format(from\_unixtime(b.tanggal),

. "'%d/%m/%Y')")))

->join(array("j" => "journal"), "b.id = j.idBukti",

array("debit", "kredit", "idAccount", "pos"

=> new Zend\_Db\_Expr("CASE j.debit WHEN 0 THEN -1 ELSE 1 END")))

->join(array("a" => "account"), "j.idAccount = a.id",

array("kodeAccount", "account"))

->where("b.tanggal >= ?", $tglAwal)

->where("b.tanggal <= ?", $tglAkhir)

->order(array("b.tanggal", "b.noBukti", "pos DESC"));

return $db->fetchAll($select);

}

/\*\*

\* Mendapatkan laporan buku besar.

\* **@param** int $tglAwal

\* **@param** int $tglAkhir

\* **@param** int $idAccount

\* **@return** array

\*/

public function fetchBukubesar($tglAwal, $tglAkhir, $idAccount)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

*// Saldo awal*

$selectSaldo = new Zend\_Db\_Select($db);

$selectSaldo->from(array("b" => "bukti\_transaksi"),

new Zend\_Db\_Expr("SUM((j.debit - j.kredit) \* a.normalPos)"))

->join(array("j" => "journal"), "b.id = j.idBukti", null)

->join(array("a" => "account"), "a.id = j.idAccount", null)

->where("j.idAccount = ?", $idAccount)

->where("b.tanggal < ?", $tglAwal);

$saldoAwal = ($db->fetchOne($selectSaldo))?

$db->fetchOne($selectSaldo): 0;

*// Data journal*

$selectJournal = new Zend\_Db\_Select($db);

$selectJournal->from(array("b" => "bukti\_transaksi"),

array("noBukti", "keterangan", "tanggal" => new Zend\_Db\_Expr(

"date\_format(from\_unixtime(b.tanggal), '%d/%m/%Y')")))

->join(array("j" => "journal"), "b.id = j.idBukti",

array("idAccount", "debit", "kredit",

new Zend\_Db\_Expr("(j.debit - j.kredit) \* a.normalPos AS saldo"),

new Zend\_Db\_Expr("CASE j.debit WHEN 0 THEN -1 ELSE 1 END AS "

. "pos")))

->join(array("a" => "account"), "j.idAccount = a.id",

array("kodeAccount", "account", "normalPos"))

->where("j.idAccount = ?", $idAccount)

->where("b.tanggal >= ?", $tglAwal)

->where("b.tanggal <= ?", $tglAkhir)

->order(array("b.tanggal", "b.noBukti"));

$dataJournal = $db->fetchAll($selectJournal);

*// Generate Laporan*

$result = array();

*// Tambahkan saldo awal*

$tglSaldo = new Zend\_Date($tglAwal);

$tglSaldo->sub(1, Zend\_Date::*DAY*);

$result[] = array(

"tanggal" => $tglSaldo->toString("dd/MM/yyyy"),

"noBukti" => "--",

"keterangan" => "Saldo Awal",

"debit" => 0,

"kredit" => 0,

"saldo" => $saldoAwal,

"pos" => "1"

);

*// Tambahkan dari journal*

$saldo = $saldoAwal;

foreach ($dataJournal as $row) {

$saldo += $row["saldo"];

$result[] = array(

"tanggal" => $row["tanggal"],

"noBukti" => $row["noBukti"],

"keterangan" => $row["keterangan"],

"debit" => $row["debit"],

"kredit" => $row["kredit"],

"saldo" => $saldo,

"pos" => $row["pos"]

);

}

return $result;

}

/\*\*

\* Mendapatkan laporan laba/rugi.

\* **@param** int $tglAwal

\* **@param** int $tglAkhir

\* **@return** array

\*/

public function fetchLabarugi($tglAwal, $tglAkhir)

{

$db = Zend\_Db\_Table::*getDefaultAdapter*();

$result = array();

*// Pendapatan*

$result[self::*ROOT\_PENDAPATAN*] = $this->\_fetchTransPeriode(

self::*ROOT\_PENDAPATAN*, $tglAwal, $tglAkhir,

self::*DEFAULT\_DISPLAY\_LEVEL*);

*// Biaya*

$result[self::*ROOT\_BIAYA*] = $this->\_fetchTransPeriode(

self::*ROOT\_BIAYA*, $tglAwal, $tglAkhir, self::*DEFAULT\_DISPLAY\_LEVEL*);

*// Pendapatan Non Operasional*

$result[self::*ROOT\_PENDAPATAN\_NON\_OPERASIONAL*] = $this

->\_fetchTransPeriode(

self::*ROOT\_PENDAPATAN\_NON\_OPERASIONAL*, $tglAwal, $tglAkhir,

self::*DEFAULT\_DISPLAY\_LEVEL*);

*// Biaya Non Operasional*

$result[self::*ROOT\_BIAYA\_NON\_OPRASIONAL*] = $this->\_fetchTransPeriode(

self::*ROOT\_BIAYA\_NON\_OPRASIONAL*, $tglAwal, $tglAkhir,

self::*DEFAULT\_DISPLAY\_LEVEL*);

return $result;

}

/\*\*

\* Mendapatkan laporan neraca

\* **@param** int $tglAwal

\* **@param** int $tglAkhir

\*/

public function fetchNeraca($tglAwal, $tglAkhir)

{

*// Laba ditahan: total dari awal sampai tanggal awal - 1*

$tglAkhirDitahan = new Zend\_Date($tglAwal, Zend\_Date::*TIMESTAMP*);

$tglAkhirDitahan->sub("1", Zend\_Date::*DAY*); *// Kurangi 1 hari*

$labaDitahan = $this->\_getLabaRugi(0, $tglAkhirDitahan

->get(Zend\_Date::*TIMESTAMP*));

*// Laba/rugi periode berjalan: total dari tanggal awal sampai tanggal*

*// akhir*

$labarugiBerjalan = $this->\_getLabaRugi($tglAwal, $tglAkhir);

*// Fetch data*

$result = array();

*// Aktiva*

$result[self::*ROOT\_AKTIVA*] = $this

->\_fetchTransPeriode(self::*ROOT\_AKTIVA*, 0, $tglAkhir,

self::*DEFAULT\_DISPLAY\_LEVEL*);

*// Pasiva*

$result[self::*ROOT\_PASIVA*] = $this

->\_fetchTransPeriode(self::*ROOT\_PASIVA*, 0, $tglAkhir,

self::*DEFAULT\_DISPLAY\_LEVEL*);

*// Modal*

$modal = $this->\_fetchTransPeriode(self::*ROOT\_MODAL*, 0, $tglAkhir,

self::*DEFAULT\_DISPLAY\_LEVEL*);

*// Update laba ditahan dan laba/rugi periode berjalan*

*// Laba ditahan dan laba/rugi periode berjalan berada pada akun modal*

*// (tepat anaknya modal)*

*// jadi cuma update laba ditahan dan laba/rugi periode berjalan saja*

*// + update modal.*

*//* **TODO** *Ada yang lebih baik?*

foreach ($modal as $key => $rowModal) {

switch ($rowModal["kodeAccount"]) {

case self::*ROOT\_MODAL*:

case self::*ROOT\_MODAL* . "-TOTAL":

$labarugi = $labaDitahan + $labarugiBerjalan;

if ($labarugi >= 0) {

$modal[$key]["debit"] += $labarugi;

} else {

$modal[$key]["kredit"] -= $labarugi;

}

$modal[$key]["saldo"] -= $labarugi;

break;

case self::*ROOT\_MODAL* . ".02": *// laba ditahan*

if ($labaDitahan >= 0) {

$modal[$key]["debit"] += $labaDitahan;

} else {

$modal[$key]["kredit"] -= $labaDitahan;

}

$modal[$key]["saldo"] -= $labaDitahan;

break;

case self::*ROOT\_MODAL* . ".03": *// laba ditahan*

if ($labarugiBerjalan >= 0) {

$modal[$key]["debit"] += $labarugiBerjalan;

} else {

$modal[$key]["kredit"] -= $labarugiBerjalan;

}

$modal[$key]["saldo"] -= $labarugiBerjalan;

break;

}

}

$result[self::*ROOT\_MODAL*] = $modal;

return $result;

}

}

## File: /application/models/User.php

<?php

/\*\*

\* Class untuk manage data user.

\* **@author** Adi Sayoga

\*/

class Model\_User extends Zend\_Db\_Table\_Abstract implements Model\_Crud

{

protected $\_name = "users";

/\*\*

\* Mendapatkan instansi dari objek Zend\_Db\_Table\_Select.

\* **@param** array $params

\* **@return** Zend\_Db\_Table\_Select

\*/

private function \_getSelect($params)

{

$select = $this->select();

*// Filter*

if (isset($params["filters"])) {

foreach ($params["filters"] as $field => $filter) {

if ($filter == null) {

$select->where("$field is null");

} else {

$select->where("$field = ?", $filter);

}

}

}

*// Sort*

isset($params["sortFields"])

&& $select->order($params["sortFields"]);

return $select;

}

/\*\*

\* Login

\* **@param** string $username

\* **@param** string $password

\* **@return** bool

\*/

public function login($username, $password)

{

*// Dapatkan default db adapter*

$db = Zend\_Db\_Table::*getDefaultAdapter*();

*// Ciptakan auth adapter*

$authAdapter = new Zend\_Auth\_Adapter\_DbTable($db,

"users", "username", "password");

*// Set username dan password*

$authAdapter->setIdentity($username);

$authAdapter->setCredential(md5($password));

*// Authenticate*

$result = $authAdapter->authenticate();

if (!$result->isValid()) return false;

*// Taruh username dan displayName dari user*

$auth = Zend\_Auth::*getInstance*();

$storage = $auth->getStorage();

$storage->write($authAdapter->getResultRowObject(array(

"username", "displayName", "role")));

return true;

}

/\*\*

\* Logout

\*/

public function logout()

{

$authAdapter = Zend\_Auth::*getInstance*();

$authAdapter->clearIdentity();

}

public function fetchData($params = null)

{

$select = $this->\_getSelect($params);

return $this->fetchAll($select);

}

public function fetchPaginatorAdapter($params = null)

{

$select = $this->\_getSelect($params);

*// Ciptakan instansi bari dari paginator adapter*

$adapter = new Zend\_Paginator\_Adapter\_DbSelect($select);

return $adapter;

}

public function fetchArray($params = null)

{

$select = $this->\_getSelect($params);

$db = $this->getAdapter();

return $db->fetchAssoc($select);

}

public function createNew($data)

{

*// Buat baris baru*

$row = $this->createRow();

if (!$row) throw new Zend\_Exception("Tidak dapat membuat user

baru!");

*// Update nilai row*

$row->username = $data["username"];

$row->password = md5($data["password"]);

$row->displayName = $data["displayName"];

$row->role = $data["role"];

return $row->save();

}

public function updateData($id, $data)

{

*// Cari baris yang cocok dengan id*

$row = $this->find($id)->current();

if (!$row) throw new Zend\_Exception("Update gagal, data tidak

ditemukan!");

$row->username = $data["username"];

$row->displayName = $data["displayName"];

$row->role = $data["role"];

*// Simpan baris yang di-update*

$row->save();

return true;

}

/\*\*

\* Mengupdate hanya password.

\*

\* **@param** int $id

\* **@param** string $password

\*

\* **@return** bool - True jika update berhasil.

\* **@throws** Zend\_Exception

\*/

public function updatePassword($id, $password)

{

*// Cari baris yang cocok dengan id*

$row = $this->find($id)->current();

if (!$row) throw new Zend\_Exception("Update gagal, data tidak

ditemukan!");

$row->password = md5($password);

$row->save();

return true;

}

public function deleteData($id)

{

*// Cari baris yang cocok dengan id*

$row = $this->find($id)->current();

if (!$row) throw new Zend\_Exception("Delete gagal, data tidak

ditemukan!");

$row->delete();

return true;

}

}

## File: /application/views/scripts/\_partials/pagination-control.phtml

<?php if ($this->pageCount):

$params = Zend\_Controller\_Front::*getInstance*()->getRequest()

->getParams();

*// hapus parameter sistem*

unset($params["module"]);

unset($params["controller"]);

unset($params["action"]); ?>

<div class="pagination-control">

*<!-- Previous -->*

<?php if (isset($this->previous)): ?>

<a class="button" href="<?php echo $this->url(array\_merge(

$params, array('page' => $this->previous))); ?>">

*&lt; Sebelumnya*</a>

<?php else: ?>

<span class="button button-disabled">*&lt; Sebelumnya*</span>

<?php endif; ?>

*<!-- Penomoran -->*

<?php foreach ($this->pagesInRange as $page): ?>

<?php if ($page != $this->current): ?>

<a class="button" href="<?php echo $this->url(array\_merge(

$params, array('page' => $page))); ?>">

<?php echo $page; ?></a>

<?php else: ?>

<span class="button button-current"><?php echo $page; ?></span>

<?php endif; ?>

<?php endforeach; ?>

*<!-- Next -->*

<?php if (isset($this->next)): ?>

<a class="button" href="<?php echo $this->url(array\_merge(

$params, array('page' => $this->next))); ?>">

*Berikutnya &gt;*</a>

<?php else: ?>

<span class="button button-disabled">*Berikutnya &gt;*</span>

<?php endif; ?>

</div>

<?php endif; ?>

## File: /application/views/scripts/account/\_table-row.phtml

<tr>

<td class="action"><a class="delete" href="/account/delete/id/

<?php echo $this->id; ?>"></a>

<a class="update" href="/account/update/id/

<?php echo $this->id; ?>"></a></td>

<td><?php echo $this->kodeAccount; ?></td>

<td style="padding-left:

<?php echo ($this->depth \* 20 + 3); ?>px">

<?php echo $this->account; ?></td>

<td style="text-align: *center*; ">

<?php echo $this->normalPosKet; ?></td>

<td style="text-align: *center*; "><?php echo $this->kelompokKet; ?></td>

</tr>

## File: /application/views/scripts/account/create.phtml

<script type="text/javascript">*<!--*

var api;

$(document).ready(function() {

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

$(".form").validate({

smd: api,

functionName: "validateAccountForm",

toValidate: "#kodeAccount, #account, #kelompok, #normalPos",

onSubmit: addNew

});

displayChilds(null);

$("#parent").change(function() { displayChilds($(this).val()); });

});

/\*\*

\* Menampilkan list account pada parent

\* @param string lastParentValue

\*/

function displayListParent(lastParentValue) {

var $parentElement = $("#parent");

$parentElement.empty();

$parentElement.append('<option value="0" label="(Tidak Ada)">

(Tidak Ada)</option>');

*// Update comboBox*

api.fetchAccount({ success: function(rows, id, metod) {

for (var idx in rows) {

$parentElement.append('<option value="' + rows[idx].id

+ '" label="' + ". . . . ".repeat(rows[idx].depth)

+ '[' + rows[idx].kodeAccount + '] - ' + rows[idx].account

+ '">' + ". . . . ".repeat(rows[idx].depth)

+ '[' + rows[idx].kodeAccount + '] - ' + rows[idx].account

+ '</option>');

}

$parentElement.val(lastParentValue);

}});

}

/\*\*

\* Menampilkan list anak berdasarkan parent

\* @param int parent

\*/

function displayChilds(parent) {

if (parent == 0) parent = null; *// paling awal*

var $afterElement = $("#after");

$afterElement.empty();

$afterElement.append('<option value="0" label="(Paling Awal)">

(Paling Awal)</option>');

api.fetchAccountChilds(parent, { success: function(childs) {

for (var idx in childs) {

$afterElement.append('<option value="' + childs[idx].id

+ '" label="' + '[' + childs[idx].kodeAccount + '] - '

+ childs[idx].account + '">' + '[' + childs[idx].kodeAccount

+ '] - ' + childs[idx].account + '</option>');

}

$afterElement.val($afterElement.find(":last").val());

}});

}

/\*\*

\* Menambah data baru

\*/

var addNew = function() {

var data = {

parent: $("#parent").val(),

after: $("#after").val(),

kodeAccount: $("#kodeAccount").val(),

account: $("#account").val(),

kelompok: $("#kelompok").val(),

normalPos: $("#normalPos").val()

};

api.addAccount(data, { success: function(result) {

if (!result) {

displayMessage("Tidak dapat menambah data baru!", "error");

return;

}

var lastParentValue = $("#parent").val();

displayListParent(lastParentValue);

displayChilds(lastParentValue);

*// Kosongkan field*

$("#kodeAccount").val("");

$("#account").val("");

displayMessage("Data berhasil ditambahkan.");

}});

};

*//-->*</script>

<h1>*Tambah Akun*</h1>

<p>*Isi form dibawah ini kemudian tekan tombol simpan.*</p>

<?php if ($this->message): ?>

<div class="ui-state-highlight ui-corner-all message">

<div class="ui-icon ui-icon-info"></div>

<div><?php echo $this->message; ?></div>

</div>

<?php endif; ?>

<?php echo $this->form; ?>

## File: /application/views/scripts/account/index.phtml

<script type="text/javascript">*<!--*

$(document).ready(function() {

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

initializeHandle();

});

var cache = [];

/\*\*

\* Pergi ke halaman tertentu

\* @param int page

\*/

function gotoPage(page) {

if (cache[page]) {

displayData(cache[page]);

return; *// Data hanya di-request sekali saja, setelah itu*

*// digunakan cache*

}

api.fetchAccountPaginator(page, { success: function(result) {

cache[page] = result;

displayData(cache[page]);

}});

}

/\*\*

\* Menampilkan data

\* @param json result

\*/

function displayData(result) {

$(".spreadsheet").displayPagedData({

data: result.data,

url: "/account",

fields: {

0: { name: "kodeAccount" },

1: { name: "account", depth: "depth" },

2: { name: "normalPosKet", align: "center" },

3: { name: "kelompokKet", align: "center" }

}

});

$(".pagination").paginationControl({

pagination: result.pagination,

url: "/account/index"

});

initializeHandle();

}

/\*\*

\* Setelah pagination control dibuat ulang, click handle awal tidak

\* berlaku lagi maka disini dibuat lagi.

\*/

function initializeHandle() {

$(".pagination a").click(function(e) {

e.preventDefault();

var href = $(this).attr("href");

var page = href.substr(href.indexOf("/page/") + 6);

gotoPage(page);

});

$(".action .delete").click(function(e) {

e.preventDefault();

if (!confirm("Anda yakin mau menghapus data ini?")) return

var href = $(this).attr("href");

var id = href.substr(href.indexOf("/id/") + 4);

api.deleteAccount(id, { success: function(result) {

if (!result) {

displayMessage("Data tidak dapat dihapus!", "error");

return;

}

var currentPage = $(".pagination .current").text();

cache[currentPage] = null;

gotoPage(currentPage); *// Refresh data*

displayMessage("Data berhasil dihapus");

}});

});

}

*//-->*</script>

<h1>*Data Akun*</h1>

<table class='spreadsheet' cellspacing='0' style="width: *100%*;">

<thead>

<tr>

<th class="action">*&nbsp;*</th>

<th style="width: *100px*;">*Kode Akun*</th>

<th>*Akun*</th>

<th style="width: *100px*;">*Normal Pos*</th>

<th style="width: *100px*;">*Kelompok*</th>

</tr>

</thead>

<tbody>

<?php echo $this->partialLoop('account/\_table-row.phtml',

$this->paginator); ?>

</tbody>

</table>

<div class="pagination-control" style="width: *100%*;">

<div class="action"><a class="button" href="/account/create">

*Tambah Akun Baru*</a></div>

<div class="pagination">

<?php echo $this->paginationControl($this->paginator, "Sliding",

"\_partials/pagination-control.phtml"); ?>

</div>

<div style="clear: *both*; "></div>

</div>

## File: /application/views/scripts/account/update.phtml

<script type="text/javascript">

*<!--*

var api;

var lastKodeAccount = <?php echo(($this->lastKodeAccount)? '"'

. $this->lastKodeAccount . '"': "null"); ?>;

$(document).ready(function() {

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

$(".form").validate({

smd: api,

functionName: "validateAccountForm",

toValidate: "#kodeAccount, #account, #kelompok, #normalPos",

exclude: lastKodeAccount,

onSubmit: updateAccount

});

});

/\*\* Mengubah data account \*/

var updateAccount = function() {

var data = {

kodeAccount: $("#kodeAccount").val(),

account: $("#account").val(),

kelompok: $("#kelompok").val(),

normalPos: $("#normalPos").val()

};

api.updateAccount($("#id").val(), data, { success: function(result) {

if (!result) {

displayMessage("Data tidak dapat diubah!", "error");

return;

}

window.location = "/account";

}});

};

*//-->*

</script>

<h1>*Update Akun*</h1>

<p>*Untuk meng-update akun ini, ubah yang anda perlukan, dan kemudian klik tombol update.*</p>

<?php echo $this->form; ?>

## File: /application/views/scripts/journal/\_table-row.phtml

<tr>

<td class="action"><a class="delete" href="/journal/delete/id/

<?php echo $this->id; ?>"></a>

<a class="update" href="/journal/update/id/

<?php echo $this->id; ?>"></a></td>

<td><?php echo $this->tanggal; ?></td>

<td><?php echo $this->noBukti; ?></td>

<td><?php echo $this->kodeAccount; ?></td>

<td<?php echo ($this->pos == -1)? ' style="padding-left: 20px;"': '';

?>><?php echo $this->account; ?></td>

<td style="text-align: *right*;"><?php echo $this->debit; ?></td>

<td style="text-align: *right*;"><?php echo $this->kredit; ?></td>

<td><?php echo $this->keterangan; ?></td>

</tr>

## File: /application/views/scripts/journal/create.phtml

<h1>*Input Data Jurnal*</h1>

<p>*Isi form dibawah ini kemudian tekan tombol simpan.*</p>

<?php if ($this->message): ?>

<div class="<?php echo $this->uiState; ?> ui-corner-all message">

<div class="ui-icon <?php echo $this->uiIcon; ?>"></div>

<div><?php echo $this->message; ?></div>

</div>

<?php endif; ?>

<form id="journal" enctype="application/x-www-form-urlencoded" class="form" action="/journal/create" method="post">

<dl class="zend\_form"><?php

*// karena form ini berisi detail, maka harus di-print per element*

echo $this->form->getElement("id");

echo $this->form->getElement("tipeJournal");

echo $this->form->getElement("noBukti");

echo $this->form->getElement("tanggal");

echo $this->form->getElement("keterangan");

echo $this->form->getElement("total");

?></dl>

<table class='spreadsheet journal' cellspacing='0'

style="width: *100%*;">

<thead>

<tr class="ui-widget-header">

<th style="width: *20px*;">*&nbsp;*</th>

<th style="width: *100px*;">*Kode Akun*</th>

<th>*Akun*</th>

<th style="width: *100px*;">*Debit*</th>

<th style="width: *100px*;">*Kredit*</th>

</tr>

</thead>

<tbody>

<?php $i = 0; foreach ($this->details as $row): ?>

<tr>

<td class="action"><input type="checkbox" name="include

<?php echo $i++; ?>"

checked="checked" class="include" /></td>

<td><input type="text" name="kodeAccount[]" value="

<?php echo $row["kodeAccount"]; ?>"

class="kodeAccount" disabled="disabled" style="width:

*100px*;"/></td>

<td><select name="account[]" class="account" style="width: *99%*; ">

<?php

foreach($this->accountList as $key => $value) {

echo "<option value='$key' label='$value' "

. (($row["idAccount"] == $key)? "selected='selected'": "")

. ">$value</option>";

}

?>

</select></td>

<td><input type="text" name="debit[]" value="<?php echo

$row["debit"]; ?>" class="debit" style="width: *100px*; " /></td>

<td><input type="text" name="kredit[]" value="<?php echo

$row["kredit"]; ?>" class="kredit" style="width: *100px*; "

/></td>

</tr>

<?php endforeach; ?>

*<!-- Blank input -->*

<tr>

<td class="action"><input type="checkbox" name="include

<?php echo $i++; ?>"

checked="checked" class="include" /></td>

<td><input type="text" name="kodeAccount[]" value=""

class="kodeAccount" disabled="disabled" style="width:

*100px*;"/></td>

<td><select name="account[]" class="account" style="width: *99%*; ">

<?php

foreach($this->accountList as $key => $value) {

echo "<option value='$key' label='$value'>$value</option>";

}

?>

</select></td>

<td><input type="text" name="debit[]" value="0" class="debit"

style="width: *100px*; " /></td>

<td><input type="text" name="kredit[]" value="0" class="kredit"

style="width: *100px*; " /></td>

</tr>

</tbody>

<tfoot>

<tr>

<td>*&nbsp;*</td>

<td colspan="2"><div class="addRow">*&nbsp;*</div></td>

<td><span class="totalDebit" style="padding: *8px*; ">

<?php echo $this->totalDebit; ?></span></td>

<td><span class="totalKredit" style="padding: *8px*; ">

<?php echo $this->totalKredit; ?></span></td>

</tr>

</tfoot>

</table>

<dl><dt>*&nbsp;*</dt><dd><input type="submit" value="Simpan" /></dd></dl>

</form>

*<!-- ini akan dihapus jika javascript enable -->*

<a href="/journal/create" class="noJavascript">*Kosongkan data untuk menambah baru*</a>

## File: /application/views/scripts/journal/index.phtml

<script type="text/javascript">*<!--*

$(document).ready(function() {

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

initializeHandle();

*// Tombol filter tidak diperlukan*

$(".filter input:submit").hide();

$(".filter #tahun, .filter #bulan").change(function(e) {

cache = [];

gotoPage(1);

});

});

var tipeJournal = 1; *// Journal umum*

var cache = [];

/\*\*

\* Pergi ke halaman tertentu

\* @param int page

\*/

function gotoPage(page) {

if (cache[page]) {

displayData(cache[page]);

return; *// Data hanya di-request sekali saja, setelah itu*

*// digunakan cache*

}

var tahun = $(".filter #tahun").val();

var bulan = $(".filter #bulan").val();

api.fetchJournalPaginator(page, tipeJournal, tahun, bulan, {

success: function(result, id, method) {

cache[page] = result;

displayData(cache[page]);

}});

}

/\*\*

\* Menampilkan data

\* @param json result

\*/

function displayData(result) {

$(".spreadsheet").displayPagedData({

data: result.data,

url: "/journal",

fields: {

0: { name: "tanggal" },

1: { name: "noBukti" },

2: { name: "kodeAccount" },

3: { name: "account", pos: "pos" },

4: { name: "debit", align: "right" },

5: { name: "kredit", align: "right" },

6: { name: "keterangan" }

}

});

$(".pagination").paginationControl({

pagination: result.pagination,

url: "/journal/index"

});

initializeHandle();

}

/\*\*

\* Setelah pagination control dibuat ulang, click handle awal tidak

\* berlaku lagi maka disini dibuat lagi.

\*/

function initializeHandle() {

$(".pagination a").click(function(e) {

e.preventDefault();

var href = $(this).attr("href");

var page = href.substr(href.indexOf("/page/") + 6);

gotoPage(page);

});

$(".action .delete").click(function(e) {

e.preventDefault();

if (!confirm("Menghapus jurnal ini juga akan menghapus jurnal

lain dengan no bukti yang sama. Untuk menghapus hanya jurnal

ini, gunakan edit.\n\nYakin mau menghapus data ini?"))

return

var href = $(this).attr("href");

var id = href.substr(href.indexOf("/id/") + 4);

api.deleteJournal(id, { success: function(result) {

if (!result) {

displayMessage("Data tidak dapat dihapus!", "error");

return;

}

var currentPage = $(".pagination .current").text();

cache[currentPage] = null;

gotoPage(currentPage); *// Refresh data*

displayMessage("Data berhasil dihapus");

}});

});

}

*//-->*

</script>

<h1>*Jurnal*</h1>

<div class="filter">

<form id="periode" enctype="application/x-www-form-urlencoded" action="/journal" method="post" class="form form-filter">

<table border="0">

<tr>

<td>*Periode: &nbsp;&nbsp;*</td>

<td><select name="bulan" id="bulan">

<?php

$bulan = array(

"1" => "Januari",

"2" => "Februari",

"3" => "Maret",

"4" => "April",

"5" => "Mei",

"6" => "Juni",

"7" => "Juli",

"8" => "Agustus",

"9" => "September",

"10" => "Oktober",

"11" => "Nopember",

"12" => "Desember"

);

?>

<?php foreach($bulan as $key => $value): ?>

<option value="<?php echo $key; ?>" label="<?php echo $value; ?>"

<?php if ($this->bulan == $key) echo 'selected="selected"'; ?>>

<?php echo $value; ?>

</option>

<?php endforeach; ?>

</select></td>

<td><select name="tahun" id="tahun">

<?php for ($i = $this->tahun-30; $i <= $this->tahun + 10; $i++): ?>

<option value="<?php echo $i; ?>" label="<?php echo $i; ?>"

<?php if ($this->tahun == $i) echo 'selected="selected"'; ?>>

<?php echo $i; ?></option>

<?php endfor; ?>

</select>

<td><input type="submit" value="OK" /></td>

</tr>

</table>

</form>

</div>

<table class='spreadsheet' cellspacing='0' style="width: *100%*;

min-height: *100%*; overflow: *auto*; ">

<thead>

<tr class>

<th class="action">*&nbsp;*</th>

<th style="*/\*width: 60px;\*/*">*Tanggal*</th>

<th style="*/\*width: 70px;\*/*">*No. Bukti*</th>

<th style="*/\*width: 70px;\*/*">*Kode Akun*</th>

<th>*Akun*</th>

<th style="*/\*width: 80px;\*/*">*Debit*</th>

<th style="*/\*width: 80px;\*/*">*Kredit*</th>

<th>*Keterangan*</th>

</tr>

</thead>

<tbody>

<?php echo $this->partialLoop('journal/\_table-row.phtml',

$this->paginator); ?>

</tbody>

</table>

<div class="pagination-control" style="width: *100%*;">

<div class="action"><a class="button" href="/journal/create">

*Tambah Jurnal Baru*</a></div>

<div class="pagination">

<?php echo $this->paginationControl($this->paginator, "Sliding",

"\_partials/pagination-control.phtml"); ?>

</div>

<div style="clear: *both*; "></div>

</div>

**Catatan:** File /application/views/scripts/journal/update.phtml hampir sama dengan file /application/views/scripts/journal/create.phtml.

Jurnal penyesuaian juga hampir sama dengan jurnal ini.

## File: /application/views/scripts/user/\_table-row.phtml

<tr>

<td class="action"><a class="delete" href="/user/delete/id/

<?php echo $this->id; ?>"></a>

<a class="update" href="/user/update/id/<?php echo $this->id;

?>"></a></td>

<td><?php echo $this->username; ?></td>

<td><?php echo $this->displayName; ?></td>

<td><?php echo $this->role; ?></td>

</tr>

## File: /application/views/scripts/user/create.phtml

<script type="text/javascript">*<!--*

var api;

$(document).ready(function() {

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

$(".form").validate({

smd: api,

functionName: "validateUserForm",

toValidate: "#username, #displayName, #password, #role",

onSubmit: addNew

});

});

/\*\* Menambah data baru \*/

var addNew = function() {

var data = {

username: $("#username").val(),

displayName: $("#displayName").val(),

password: $("#password").val(),

role: $("#role").val()

};

api.addUser(data, { success: function(result) {

if (!result) {

displayMessage("Tidak dapat menambah data baru!", "error");

return;

}

*// Kosongkan field*

$("#username").val("");

$("#displayName").val("");

$("#password").val("");

displayMessage("Data berhasil ditambahkan.");

}});

};

*//-->*</script>

<h1>*Tambah Data User*</h1>

<p>*Untuk menambah data user, lengkapi form ini dan klik tombol simpan...*</p>

<?php if ($this->message): ?>

<div class="ui-state-highlight ui-corner-all message">

<div class="ui-icon ui-icon-info"></div>

<div><?php echo $this->message; ?></div>

</div>

<?php endif; ?>

<?php echo $this->form; ?>

## File: /application/views/scripts/user/index.phtml

<script type="text/javascript">*<!--*

$(document).ready(function() {

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

initializeHandle();

});

var cache = [];

/\*\*

\* Pergi ke halaman tertentu

\* @param int page

\*/

function gotoPage(page) {

if (cache[page]) {

displayData(cache[page]);

return; *// Data hanya di-request sekali saja, setelah itu digunakan*

*// cache*

}

api.fetchUserPaginator(page, { success: function(result) {

cache[page] = result;

displayData(cache[page]);

}});

}

/\*\*

\* Menampilkan data

\* @param json result

\*/

function displayData(result) {

$(".spreadsheet").displayPagedData({

data: result.data,

url: "/user",

fields: {

0: { name: "username" },

1: { name: "displayName" },

2: { name: "role" }

}

});

$(".pagination").paginationControl({

pagination: result.pagination,

url: "/user/index"

});

initializeHandle();

}

/\*\*

\* Setelah pagination control dibuat ulang, click handle awal tidak

\* berlaku lagi maka disini dibuat lagi.

\*/

function initializeHandle() {

$(".pagination a").click(function(e) {

e.preventDefault();

var href = $(this).attr("href");

var page = href.substr(href.indexOf("/page/") + 6);

gotoPage(page);

});

$(".action .delete").click(function(e) {

e.preventDefault();

if (!confirm("Anda yakin mau menghapus data ini?")) return;

var href = $(this).attr("href");

var id = href.substr(href.indexOf("/id/") + 4);

api.deleteUser(id, { success: function(result) {

if (!result) {

displayMessage("Data tidak dapat dihapus!", "error");

return;

}

var currentPage = $(".pagination .current").text();

cache[currentPage] = null;

gotoPage(currentPage); *// Refresh data*

displayMessage("Data berhasil dihapus");

}});

});

}

*//-->*</script>

<h1>*Data User*</h1>

<table class='spreadsheet' cellspacing='0' style="width: *100%*;">

<thead>

<tr>

<th class="action">*&nbsp;*</th>

<th style="width: *30%*; ">*Nama User*</th>

<th>*Nama Ditampilkan*</th>

<th style="width: *20%*; ">*Role*</th>

</tr>

</thead>

<tbody>

<?php echo $this->partialLoop('user/\_table-row.phtml', $this->paginator); ?>

</tbody>

</table>

<div class="pagination-control" style="width: *100%*;">

<div class="action"><a class="button" href="/user/create">

*Tambah User Baru*</a></div>

<div class="pagination">

<?php echo $this->paginationControl($this->paginator, "Sliding",

"\_partials/pagination-control.phtml"); ?>

</div>

<div style="clear: *both*; "></div>

</div>

## File: /application/views/scripts/user/login.phtml

<script type="text/javascript">*<!--*

var api;

$(document).ready(function() {

layout.hide("west"); *// Sembunyikan sideba*r

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

$("#submit").click(function(e) {

e.preventDefault();

var username = $("#username").val();

var password = $("#password").val();

api.login(username, password, { success: function(result) {

if (!result) {

displayMessage("Nama user dan atau password salah!");

return;

}

window.location = "<?php echo $this->ref; ?>";

}});

});

});

*//-->*</script>

<style>*<!--*

*.welcome* **img** {

border: *1px solid #cacaca*;

-webkit-border-radius: *20px*; -moz-border-radius: *20px*;

border-radius: *20px*;

-webkit-box-shadow: *3px 3px 5px rgba(0, 0, 0, .2)*;

-moz-box-shadow: *3px 3px 5px rgba(0, 0, 0, .2)*;

box-shadow: *3px 3px 5px rgba(0, 0, 0, .2)*;

}

*.login* {

width: *200px*; height: *350px*; padding: *20px 10px*; margin-right: *10px*;

border: *1px solid #cacaca*; -webkit-border-radius: *5px*;

-moz-border-radius: *5px*; border-radius: *5px*;

-webkit-box-shadow: *3px 3px 5px rgba(0, 0, 0, .2)*;

-moz-box-shadow: *3px 3px 5px rgba(0, 0, 0, .2)*;

box-shadow: *3px 3px 5px rgba(0, 0, 0, .2)*;

}

*.login* *.form* { font-size: *13px*; margin: *5px 0*; padding: *0*; border: *0* }

*.login* *.form* **dt** { float: *none*; text-align: *left*; margin-top: *10px*; }

*.login* *.form* **dt,** *.login* *.form* **dd** { width: *200px*; }

*.login* *.form* **input** { margin: *5px 0*; }

*.login* *.form* **dd** **input**[type="text"]**,** *.login* *.form* **dd**

**input**[type="password"] {

width: *190px*; font-size: *inherit*;

}

*.login* *.form* **dd** **input**[type="submit"] {

width: *100px*; padding: *8px*; font-size: *inherit*; font-weight: *bold*;

margin-left: *50px*;

}

*.login* **p** { font-style: *italic*; color: *#666*; }

*-->*</style>

*<!-- TODO layout menggunakan css (float) tidak bekerja -->*

<table>

<tr valign="top"><td>

<div class="welcome">

<h1>*Selamat Datang*</h1>

<p>*PT. Bumi Sentosa merupakan perusahaan yang berisikan personil-*

*personil yang dinamis, enerjik dan meiliki bakat enterpreneur yang*

*tinggi dengan pengalaman pengelolaan dan manajemen outsourcing yang*

*sukses.*</p>

<img alt="Bumi Sentosa" src="/images/hub.jpg" />

</div>

</td>

<td>

<div class="login">

<h1>*User Login*</h1>

<p>*Masukkan username dan password anda dibawah ini...*</p>

<?php if ($this->message): ?>

<div class="ui-state-error ui-corner-all message">

<div class="ui-icon ui-icon-alert"></div>

<div><?php echo $this->message; ?></div>

</div>

<?php endif; ?>

<?php echo $this->form; ?>

</div>

</td>

</tr>

</table>

## File: /application/views/scripts/user/update-password.phtml

<script type="text/javascript">*<!--*

var api;

$(document).ready(function() {

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

$(".form").validate({

smd: api,

functionName: "validateUserForm",

toValidate: "#password",

onSubmit: updatePassword

});

});

/\*\* Mengubah password user \*/

var updatePassword = function() {

api.updatePassword($("#id").val(), $("#password").val(), {

success: function(result, id, method) {

if (!result) {

displayMessage("Data tidak dapat diubah!", "error");

return;

}

window.location = "/user";

}});

};

*//-->*</script>

<h1>*Update Password* <?php echo ($this->user)? $this->user->displayName: "User"; ?></h1>

<p>*Masukkan password baru kemudian klik tombol simpan...*</p>

<?php echo $this->form; ?>

## File: /application/views/scripts/user/update.phtml

<script type="text/javascript">

*<!--*

var api;

var lastUsername = <?php echo(($this->lastUsername)? '"'

. $this->lastUsername . '"': "null"); ?>;

$(document).ready(function() {

api = jQuery.Zend.jsonrpc({url: '/api/v1.0/jsonrpc.php',

async: true });

$(".form").validate({

smd: api,

functionName: "validateUserForm",

toValidate: "#username, #displayName, #role",

exclude: lastUsername,

onSubmit: updateUser

});

});

/\*\* Mengubah data user \*/

var updateUser = function() {

var data = {

username: $("#username").val(),

displayName: $("#displayName").val(),

role: $("#role").val()

};

api.updateUser($("#id").val(), data, { success: function(result) {

if (!result) {

displayMessage("Data tidak dapat diubah!", "error");

return;

}

window.location = "/user";

}});

};

*//-->*

</script>

<h1>*Update User*</h1>

<p>*Untuk meng-update user ini, ubah yang anda perlukan, dan kemudian klik tombol update.*</p>

<?php echo $this->form; ?>

<div class="pagination-control">

<a href="/user/update-password/id/<?php echo $this->form

->getValue("id"); ?>">*Ubah Password*</a>

</div>

## File: /library/Acc/Controller/Plugin/Acl.php

<?php

/\*\*

\* ACL Controller Plug-in.

\* **@author** Adi Sayoga

\*/

class Acc\_Controller\_Plugin\_Acl extends Zend\_Controller\_Plugin\_Abstract

{

public function preDispatch(Zend\_Controller\_Request\_Abstract $request)

{

*//* **TODO** *hard code, seharusnya role disimpan di-database*

*// Set up acl*

$acl = new Zend\_Acl();

*// Tambahkan resources*

$resources = array("index", "error", "settings", "user", "account",

"journal", "journal-penyesuaian", "laporan");

foreach ($resources as $resourceId => $resource) {

$acl->addResource(new Zend\_Acl\_Resource($resource));

}

*// Tambahkan role*

$roles = array(

array("roleId" => "guest", "parent" => null),

array("roleId" => "user", "parent" => "guest"),

array("roleId" => "admin", "parent" => "user"));

foreach ($roles as $role) {

$acl->addRole($role["roleId"], $role["parent"]);

}

$acl->allow(null, "error");

*// Guest hanya bisa mengakses login*

$acl->allow("guest", "user", array("login", "logout"));

*// User hanya bisa mengakses account, journal, dan laporan*

$acl->allow("user", array("index", "account", "journal",

"journal-penyesuaian", "laporan"));

*// Administrator bisa melakukan apa saja*

$acl->allow("admin", null);

*// Fetch user saat ini*

$auth = Zend\_Auth::*getInstance*();

if ($auth->hasIdentity()) {

$identity = $auth->getIdentity();

$role = strtolower($identity->role);

} else {

$role = "guest";

}

$controller = $request->controller;

$action = $request->action;

if (!$acl->isAllowed($role, $controller, $action)) {

if ($role == "guest") {

$request->setControllerName("user");

$request->setActionName("login");

*// tambahkan referensi request pada login sehingga jika login*

*// berhasil maka akan kembali ke halaman sebelumnya yang*

*// direquest*

if ($controller != "user")

$request->setParam("ref", $controller . "/" . $action);

} else {

$request->setControllerName("error");

$request->setActionName("noauth");

}

}

}

}