BAB 3
PEMBAHASAN

3. 1. Analisis Sistem

Analisis sistem adalah penguraian dari satu sistem yang utuh ke dalam bagian-bagian komponen dengan maksud untuk mengidentifikasi dan mengevaluasi masalah, sehingga diperoleh solusi. Analisis merupakan tahapan paling penting karena kesalahan pada tahap ini menyebabkan kesalahan ditahap selanjutnya.

3. 1. 1. Analisis Masalah

Bagian *accounting* adalah salah satu bagian terpenting yang ada pada CV. Global Agency and Entertainment karena menyangkut masalah keuangan perusahaan. Salah satu tugas dari bagian *accounting* ini adalah membuat laporan keuangan bulanan.

Pada saat ini pengolahan laporan keuangan tersebut dilakukan dengan menggunakan *Microsoft Office Excel*. Tetapi penggunaan *Microsoft Office Excel* dalam pengolahan laporan keuangan bulanan sering menyebabkan data yang ada menjadi tidak terstruktur dan sulit dalam proses pencarian data. Selain itu pencarian dan pencetakan laporan dalam jumlah yang banyak akan memakan waktu yang tidak sedikit mengingat banyaknya data keuangan yang harus diolah.

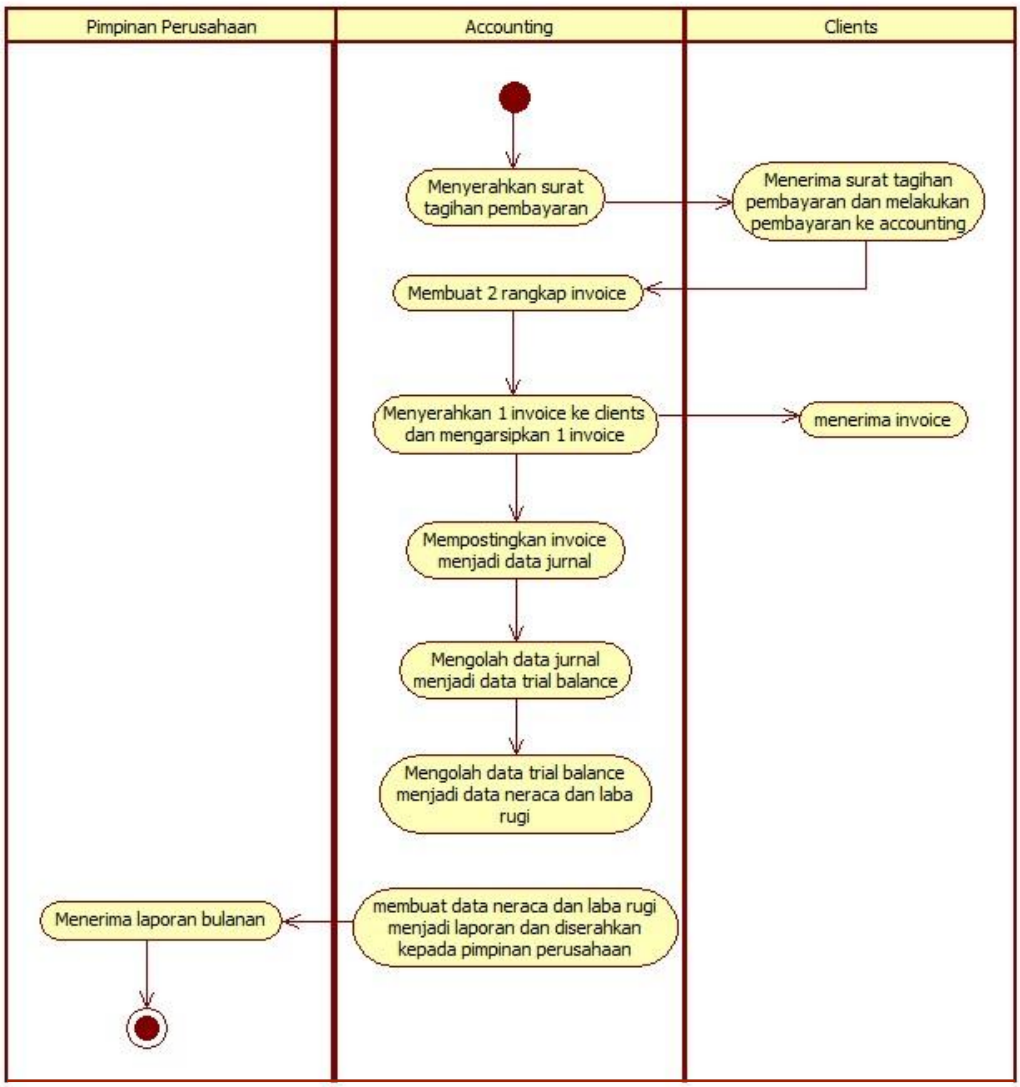
Melihat permasalah tersebut bagian *accounting* CV. Global Agency and Entertainment membutuhkan aplikasi yang dapat melakukan pengolahan data keuangan dan pembuatan laporan keuangan setiap bulannya.

3. 1. 2. Analisis Sistem yang Sedang Berjalan

Analisis sistem yang berjalan berisi tentang pemaparan sistem pengolahan data keuangan yang ada saat ini. Analisis ini dimaksudkan agar perangkat lunak yang dibangun tidak keluar dari cakupan sistem pengolahan data keuangan yang ada. Adapun prosedur pengolahan data keuangan yang ada di CV. Global Agency and Entertainment adalah :

1. *Accounting* membuat surat tagihan pembayaran yang ditujukan kepada *clients.*
2. Setelah *clients* melakukan pembayaran, *accounting* membuat 2 rangkap *invoice*. Satu *invoice* diserahkan kepada *clients* sebagai tanda bukti telah melakukan pembayaran, dan *invoice* yang ke dua di arsipkan oleh *accounting*.
3. *Invoice* yang telah diarsipkan akan di posting ke dalam data *journal* oleh *accounting*.
4. Kemudian, data *journal* diolah menjadi data *trial balance*. Lalu data *trial balance* diproses untuk membuat data neraca serta data laba-rugi.
5. Kedua data laba rugi dan neraca diarsipkan menjadi sebuah laporan bulanan, lalu dilaporkan kepada pimpinan perusahaan.

Prosedur pengolahan data keuangan tersebut digambarkan pada *activity diagram* yang terlihat pada Gambar 3.1



Gambar 3.1 Pengolahan Data Keuangan yang Sedang Berjalan

3. 1. 3. Analisis Kebutuhan NonFungsional

Analisis kebutuhan nonfungsional menggambarkan kebutuhan luar sistem yang diperlukan untuk menjalankan aplikasi yang dibangun. Adapun kebutuhan nonfungsional untuk menjalankan aplikasi pengolahan data laporan keuangan meliputi kebutuhan perangkat keras, kebutuhan perangkat lunak, dan pengguna yang akan memakai aplikasi. Analisis kebutuhan nonfungsional bertujuan agar aplikasi yang dibangun dapat digunakan sesuai dengan kebutuhan bagian keuangan dalam pengolahan data laporan keuangan.

3. 1. 3. 1. Analisis Perangkat Keras

Untuk menjalankan suatu aplikasi maka diperlukan perangkat keras yang dapat mendukung proses kerja dari sistem itu sendiri. Spesifikasi perangkat keras yang digunakan oleh bagian *accounting* dalam melakukan pengolahan data laporan keuangan tertera dalam Tabel 3.4.

Tabel 3.1 Spesifikasi Perangkat Keras Bagian Keuangan

No	Perangkat Keras	Spesifikasi
1	Processor	Intel Pentium 4
2	Monitor	Montor LCD 17 inch
3	VGA	VGA On-Board 32MB
4	Memori	1 GB
5	Keyboard	Standar
6	Mouse	Standar
7	Printer	Printer hitam-putih

Spesifikasi perangkat keras yang ada sudah mencukupi untuk menjalankan aplikasi *general ledger* bulanan di CV. Global Agency and Entertainment.

3. 1. 3. 2. Analisis Perangkat Lunak

Perangkat lunak yang digunakan di CV. Global Agency and Entertainment untuk membantu pengolahan data laporan keuangannya adalah sebagi berikut :

Sistem Operasi : Windows XP

Software Lainnya : Microsoft Office 2007

3. 1. 3. 3. Analisis Pengguna

Suatu aplikasi akan berjalan optimal apabila ditunjang oleh perangkat pikir yang memiliki kemampuan dalam menjalankan aplikas yang bersangkutan. Pengguna sistem, yaitu *accounting* yang membawahi bidang pengolahan data laporan keuangan dijelaskan sebagai berikut :

Tabel 3.2 Karakteristik Pengguna

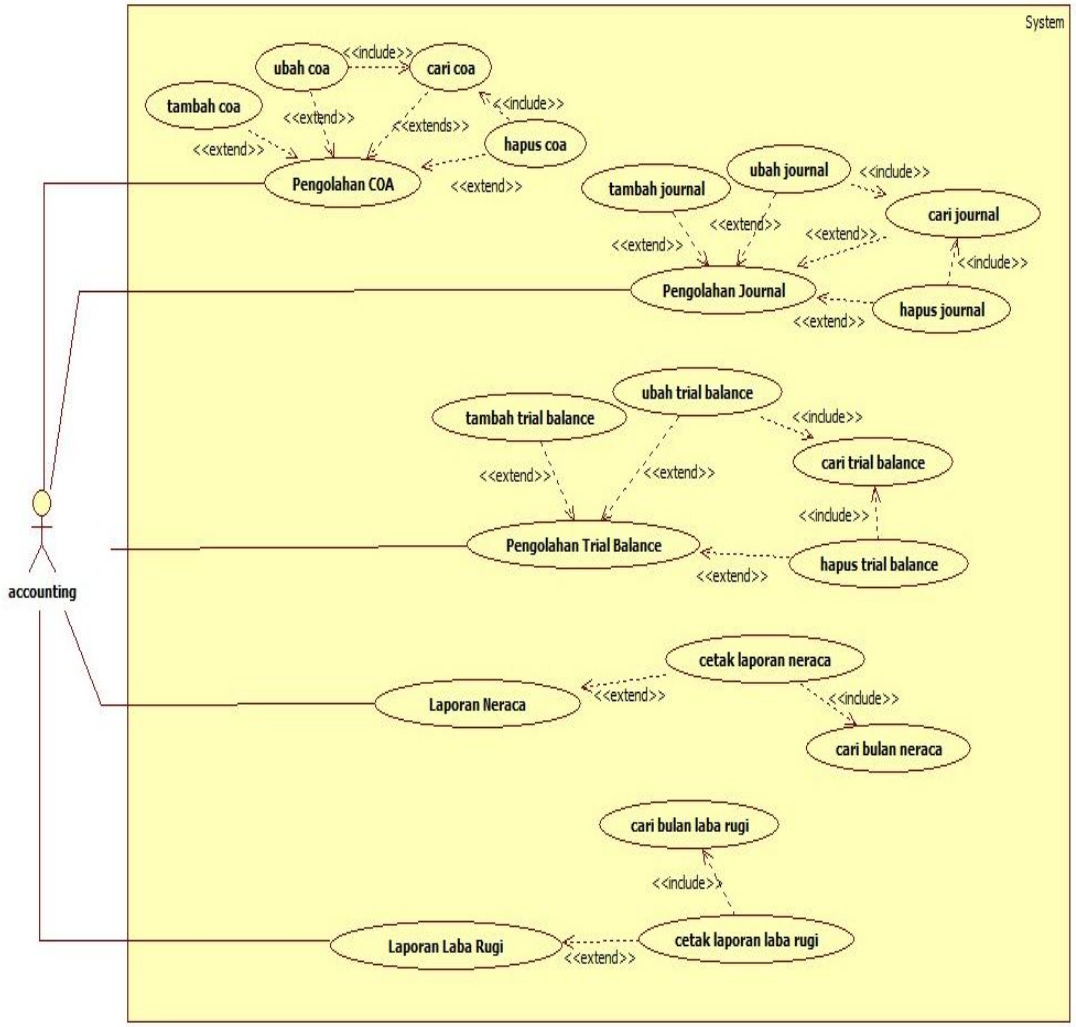
Pengguna	Tanggung jawab	Hak akses	Tingkat pendidikan	Tingkat keterampilan
Accounting	Mengelola data keuangan	Hak Akses Penuh	Min D3 Jurusan akuntansi	Bisa mengikuti petunjuk dan megoperasikan komputer

3. 1. 4. Analisis Kebutuhan Fungsional

Analisis sistem yang dilakukan menggunakan tools UML, adapun tahapan analisis sistem menggunakan UML adalah *use case diagram, use case scenario, activity diagram, state diagram, squence diagram,* dan *class diagram* .

3. 1. 4. 1. *Use Case Diagram*

Use case diagram merupakan konstruksi untuk mendeskripsikan hububungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada sistem. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem dengan mendefinisikan skenario penggunaan yang disepakati antara pemakai dan pengembang. Dari analisis pengguna aplikasi yang ada maka *use case* diagram untuk aplikasi *general ledger* bulanan di CV. Global Agency and Entertainment dapat dilihat dalam Gambar 3.2.



Gambar 3.2 *Use Case Diagram* Aplikasi *General Ledger* Bulanan

3. 1. 4. 2. *Use Case Scenario*

Use Case Scenario mendeskripsikan urutan langkah-langkah dalam proses bisnis baik yang dilakukan aktor terhadap sistem maupun yang dilikakukan oleh sistem terhadap aktor. Berdasarkan *use case diagram* pada gambar 3.1 maka *use case scenario* untuk aplikasi yang dibangun akan dijelaskan sebagai berikut :

3. 1. 4. 2. 1 *Use Case Scenario* Pengolahan COA

Tabel 3.3 *Use Case Scenario* Pengolahan COA

Identifikasi	
Nomor	1
Nama	Pengolahan COA
Tujuan	Mengolah data COA
Deskripsi	Proses ini untuk mengelola data COA mulai dari menambah, mengubah, menghapus data COA yang digunakan sebagai kode acuan untuk membuat *Journal* dan *Trial Balance*.
Aktor	*Accounting*
Use case yang berkaitan	Tambah COA, ubah COA, hapus COA
Skenario Utama	
Kondisi Awal	*Form* data COA ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Memilih tombol pilihan Pengolahan COA (tambah/ubah/hapus)	
	2. Menampilkan form menu yang dipilih oleh aktor (tambah/ubah/hapus)

1. *Use Case Scenario* Tambah COA

Tabel 3.4 *Use Case Scenario* Tambah Coa

Identifikasi	
Nomor	2
Nama	Tambah COA
Tujuan	Menambah data COA
Deskripsi	Proses penambah data COA
Aktor	*Accounting*
Use case yang berkaitan	-
Skenario Utama	
Kondisi Awal	*Form* tambah COA ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Mengisi *form* data tambah COA	
	2. Melakukan proses penambahan data COA
	3. Menyimpan data hasil proses penambahan data COA
	4. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data gagal ditambahkan

2. Mengisi kembali *form* data tambah COA	
	3. Menyimpan data hasil penambahan oleh aktor
	4. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat menambah data COA sesuai dengan kebutuhan

2. *Use Case Scenario* Ubah COA

Tabel 3.5 *Use Case Scenario* Ubah Coa

Identifikasi	
Nomor	3
Nama	Ubah COA
Tujuan	Mengubah data COA
Deskripsi	Proses pengubahan data COA
Aktor	*Accounting*
Use case yang berkaitan	Cari COA
Skenario Utama	
Kondisi Awal	*Form* ubah COA ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. mencari/memilih data COA yang akan diubah	
	2. Menampilkan data hasil pencarian
3. Mengisi *form* data COA yang akan diubah	
	4. Melakukan proses ubah data COA yang diisi oleh aktor
	5. Menyimpan data hasil proses ubah data COA yang diisi oleh aktor
	6. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data gagal diubah
2. Mengisi kembali yang *form* ubah data COA yang akan diubah	3. Melakukan proses ubah data COA yang diisi oleh aktor
	4. Menyimpan data hasil proses ubah data COA yang diisi oleh aktor
	5. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat mengubah data COA sesuai dengan kebutuhan

3. *Use Case Scenario* Hapus COA

Tabel 3.6 *Use Case Scenario* Hapus Coa

Identifikasi	
Nomor	4
Nama	Hapus COA
Tujuan	Menghapus data COA
Deskripsi	Proses penghapusan data COA
Aktor	*Accounting*
Use case yang berkaitan	Cari COA
Skenario Utama	
Kondisi Awal	*Form* hapus COA ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. mencari/memilih data COA yang akan dihapus	
	2. Menampilkan data hasil pencarian
3. Menghapus data COA	
	4. Menampilkan pesan persetujuan
5. Menghapus data COA	
	6. Melakukan proses hapus data COA
	7. Menyimpan data hasil proses hapus data COA
	8. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data gagal dihapus
2. Menghapus kembali data COA	
	3. Menampilkan pesan persetujuan
4. Menghapus kembali data COA	
	5. Melakukan proses hapus data COA
	6. Menyimpan data hasil proses hapus data COA
	7. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat menghapus data COA sesuai dengan kebutuhan

3. 1. 4. 2. 2 *Use Case Scenario* Pengolahan *Journal*

Tabel 3.7 *Use Case Scenario* Pengolahan *Journal*

Identifikasi	
Nomor	5
Nama	Pengolahan *Journal*
Tujuan	Mengolah data *Journal*
Deskripsi	Proses ini untuk mengelola data *Journal* mulai dari menambah, mengubah, menghapus data *Journal*
Aktor	*Accounting*
Use case yang berkaitan	Tambah *Journal*, ubah *Journal*, hapus *Journal*
Skenario Utama	
Kondisi Awal	*Form* data *Journal* ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Memilih tombol pilihan Pengolahan *Journal* (tambah/ubah/hapus)	
	2. Menampilkan *form* menu yang dipilih oleh aktor (tambah/ubah/hapus)

1. *Use Case Scenario* Tambah *Journal*

Tabel 3.8 *Use Case Scenario* Tambah *Journal*

Identifikasi	
Nomor	6
Nama	Tambah *Journal*
Tujuan	Menambah data *Journal*
Deskripsi	Proses penambah data *Journal*
Aktor	*Accounting*
Use case yang berkaitan	-
Skenario Utama	
Kondisi Awal	*Form* tambah *Journal* ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Mengisi *form* data tambah *Journal*	
	2. Melakukan proses penambahan data *Journal*
	3. Menyimpan data hasil proses penambahan data *Journal*
	4. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data gagal ditambahkan
2. Mengisi kembali yang	

form data tambah *Journal*	
	3. Menyimpan data hasil penambahan oleh aktor
	4. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat menambah data *Journal* sesuai dengan kebutuhan

2. *Use Case Scenario* Ubah *Journal*

Tabel 3.9 *Use Case Scenario* Ubah *Journal*

dentifikasi	
Nomor	7
Nama	Ubah *Journal*
Tujuan	Mengubah data *Journal*
Deskripsi	Proses pengubahan data *Journal*
Aktor	*Accounting*
Use case yang berkaitan	-
Skenario Utama	
Kondisi Awal	*Form* ubah *Journal* ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. mencari/memilih data *Journal* yang akan diubah	
	2. Menampilkan data hasil pencarian
3. Mengisi *form* data *Journal* yang akan diubah	
	4. Melakukan proses ubah data *Journal* yang diisi oleh aktor
	5. Menyimpan data hasil proses ubah data *Journal* yang diisi oleh aktor
	6. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data gagal diubah
2. Mengisi kembali yang *form* ubah data *Journal* yang akan diubah	3. Melakukan proses ubah data *Journal* yang diisi oleh aktor
	4. Menyimpan data hasil proses ubah data *Journal* yang diisi oleh aktor
	5. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat mengubah data *Journal* sesuai dengan kebutuhan

3. *Use Case Scenario* Hapus *Journal*

Tabel 3.10 *Use Case Scenario* Hapus *Journal*

Identifikasi	
Nomor	8
Nama	Hapus *Journal*
Tujuan	Menghapus data *Journal*
Deskripsi	Proses penghapusan data *Journal*
Aktor	*Accounting*
Use case yang berkaitan	-
Skenario Utama	
Kondisi Awal	*Form* hapus *Journal* ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. mencari/memilih data *Journal* yang akan dihapus	
	2. Menampilkan data hasil pencarian
3. Menghapus data *Journal*	
	4. Menampilkan pesan persetujuan
5. Menghapus data *Journal*	
	6. Melakukan proses hapus data *Journal*
	7. Menyimpan data hasil proses hapus data *Journal*
	8. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data gagal dihapus
2. Menghapus kembali data *Journal*	
	3. Menampilkan pesan persetujuan
4. Menghapus kembali data *Journal*	
	5. Melakukan proses hapus data *Journal*
	6. Menyimpan data hasil proses hapus data *Journal*
	7. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat menghapus data *Journal* sesuai dengan kebutuhan

3. 1. 4. 2. 3 *Use Case Scenario* Pengolahan *Trial Balance*

Tabel 3.11 *Use Case Scenario* Pengolahan *Trial Balance*

Identifikasi	
Nomor	9
Nama	Pengolahan *Trial Balance*
Tujuan	Mengolah data *Trial Balance*
Deskripsi	Proses ini untuk mengelola data *Trial Balance* mulai dari menambah, mengubah, menghapus data *Trial Balance*
Aktor	*Accounting*
Use case yang berkaitan	Tambah *Trial Balance*, ubah *Trial Balance*, hapus *Trial Balance*
Skenario Utama	
Kondisi Awal	*Form* data *Trial Balance* ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Memilih tombol pilihan Pengolahan *Trial Balance* (tambah/ubah/hapus)	
	2. Menampilkan *form* menu yang dipilih oleh aktor (tambah/ubah/hapus)

1. *Use Case Scenario* Tambah *Trial Balance*

Tabel 3.12 *Use Case Scenario* Tambah *Trial Balance*

Identifikasi	
Nomor	10
Nama	Tambah *Trial Balance*
Tujuan	Menambah data *Trial Balance*
Deskripsi	Proses penambah data *Trial Balance*
Aktor	*Accounting*
Use case yang berkaitan	-
Skenario Utama	
Kondisi Awal	*Form* tambah *Trial Balance* ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Mengisi *form* data tambah *Trial Balance*	
	2. Melakukan proses penambahan data *Trial Balance*
	3. Menyimpan data hasil proses penambahan data *Trial Balance*
	4. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data

	gagal ditambahkan
2. Mengisi kembali yang *form* data tambah *Trial Balance*	
	3. Menyimpan data hasil penambahan oleh aktor
	4. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat menambah data *Trial Balance* sesuai dengan kebutuhan

2. *Use Case Scenario* Ubah *Trial Balance*

Tabel 3.13 *Use Case Scenario* Ubah *Trial Balance*

dentifikasi	
Nomor	11
Nama	Ubah *Trial Balance*
Tujuan	Mengubah data *Trial Balance*
Deskripsi	Proses pengubahan data *Trial Balance*
Aktor	*Accounting*
Use case yang berkaitan	Cari *Trial Balance*
Skenario Utama	
Kondisi Awal	*Form* ubah *Trial Balance* ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. mencari/memilih data *Trial Balance* yang akan diubah	
	2. Menampilkan data hasil pencarian
3. Mengisi *form* data *Trial Balance* yang akan diubah	
	4. Melakukan proses ubah data *Trial Balance* yang diisi oleh aktor
	5. Menyimpan data hasil proses ubah data *Trial Balance* yang diisi oleh aktor
	6. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data gagal diubah
2. Mengisi kembali yang *form* ubah data *Trial Balance* yang akan diubah	3. Melakukan proses ubah data *Trial Balance* yang diisi oleh aktor
	4. Menyimpan data hasil proses ubah data *Trial Balance* yang diisi oleh aktor
	5. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat mengubah data *Trial Balance* sesuai dengan kebutuhan

3. *Use Case Scenario* Hapus *Trial Balance*

Tabel 3.14 *Use Case Scenario* Hapus *Trial Balance*

Identifikasi	
Nomor	12
Nama	Hapus *Trial Balance*
Tujuan	Menghapus data *Trial Balance*
Deskripsi	Proses penghapusan data *Trial Balance*
Aktor	*Accounting*
Use case yang berkaitan	Cari *Trial Balance*
Skenario Utama	
Kondisi Awal	*Form* hapus *Trial Balance* ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. mencari/memilih data *Trial Balance* yang akan dihapus	
	2. Menampilkan data hasil pencarian
3. Menghapus data *Trial Balance*	
	4. Menampilkan pesan persetujuan
5. Menghapus data *Trial Balance*	
	6. Melakukan proses hapus data *Trial Balance*
	7. Menyimpan data hasil proses hapus data *Trial Balance*
	8. Menampilkan pesan data berhasil disimpan
Skenario Alternatif (Proses Gagal)	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa pemrosesan data gagal dihapus
2. Menghapus kembali data *Trial Balance*	
	3. Menampilkan pesan persetujuan
4. Menghapus kembali data *Trial Balance*	
	5. Melakukan proses hapus data *Trial Balance*
	6. Menyimpan data hasil proses hapus data *Trial Balance*
	7. Menampilkan pesan data berhasil disimpan
Kondisi Akhir	*Accounting* dapat menghapus data *Trial Balance* sesuai dengan kebutuhan

3. 1. 4. 2. 4 *Use Case Scenario* Laporan Neraca

Tabel 3.15 *Use Case Scenario* Laporan Neraca

Identifikasi	
Nomor	13
Nama	Laporan Neraca
Tujuan	Mencetak laporan hasil pengolahan data neraca
Deskripsi	Proses ini untuk mencetak laporan dari data neraca
Aktor	*Accounting*
Use case yang berkaitan	Cari tahun neraca
Skenario Utama	
Kondisi Awal	*Form* pengolahan laporan neraca ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Mencari data neraca berdasarkan bulan dan tahun	
	2. Menampilkan data neraca sesuai dengan bulan dan tahun yang dipilih
3. Memilih tombol cetak	
	4. Menampilkan laporan data neraca
	5. Memproses pencetakan laporan yang dipilih
Kondisi Akhir	Laporan data neraca telah dicetak

3. 1. 4. 2. 5 *Use Case Scenario* Laporan Laba Rugi

Tabel 3.16 *Use Case Scenario* Laporan Laba Rugi

Identifikasi	
Nomor	14
Nama	Laporan Laba Rugi
Tujuan	Mencetak laporan hasil pengolahan data laba rugi
Deskripsi	Proses ini untuk mencetak laporan dari data laba rugi
Aktor	*Accounting*
Use case yang berkaitan	Cari tahun laba rugi
Skenario Utama	
Kondisi Awal	*Form* pengolahan laporan laba rugi ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Mencari data laba rugi berdasarkan bulan dan tahun	
	2. Menampilkan data laba rugi sesuai dengan bulan dan tahun yang dipilih
3. Memilih tombol cetak	

	4. Menampilkan laporan data laba rugi
	5. Memproses pencetakan laporan yang dipilih
Kondisi Akhir	Laporan data laba rugi telah dicetak

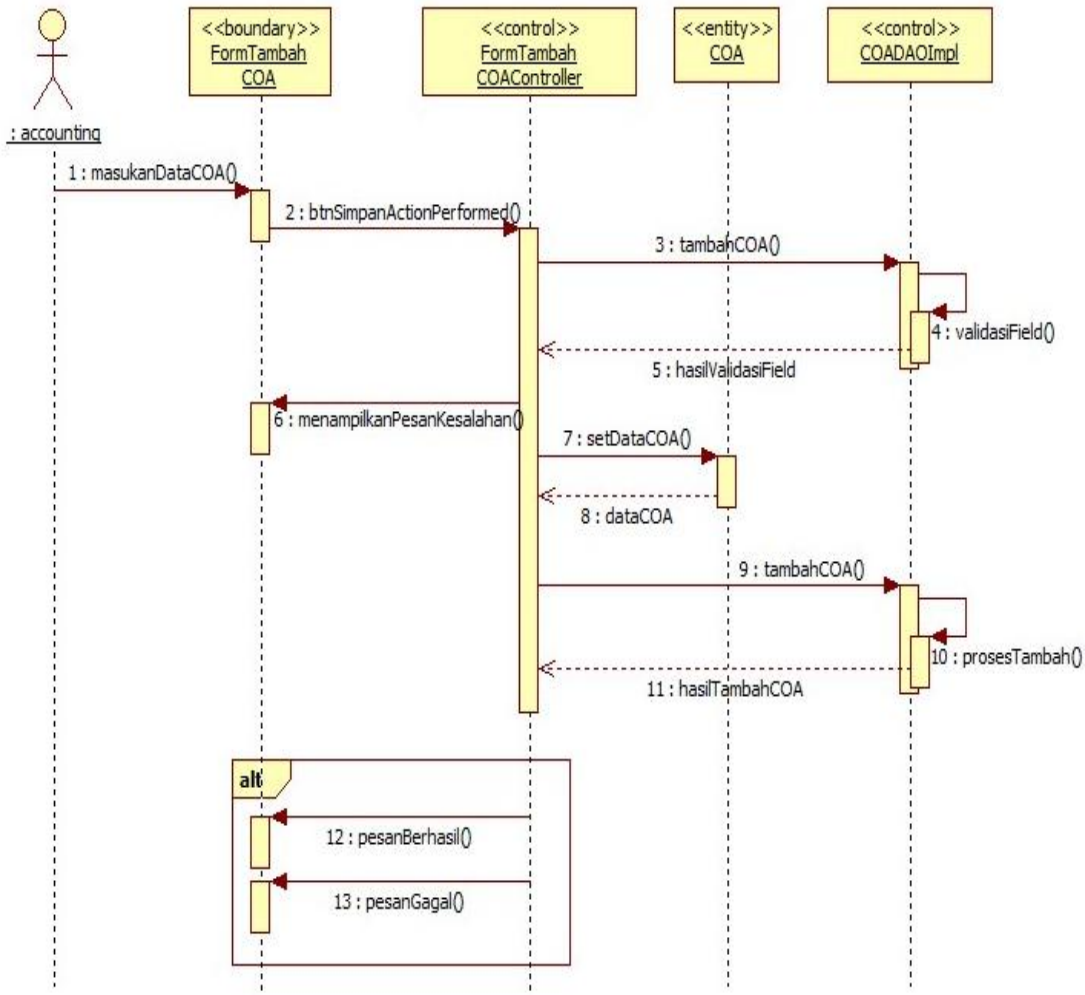
3. 1. 4. 3. *Sequence Diagram*

Sequence Diagram menggambarkan interaksi antar masing-masing objek pada setiap *use case* dalam urutan waktu. Interaksi ini berupa pengiriman serangkaian data antar objek-objek yang saling berinteraksi. *Sequence Diagram* pada aplikasi *general ledger* bulanan di CV. Global Agency and Entertainment terdiri dari *sequence diagram* pengolahan data coa, *sequence diagram* pengolahan data *journal*, *sequence diagram* pengolahan data *trial balance*, *sequence diagram* cetak laporan neraca dan e*squence diagram* cetak laporan laba rugi. Berikut ini penjelasan dari masing-masing *sequence diagram*:

3. 1. 4. 3. 1 *Sequence Diagram* Pengolahan Data Coa

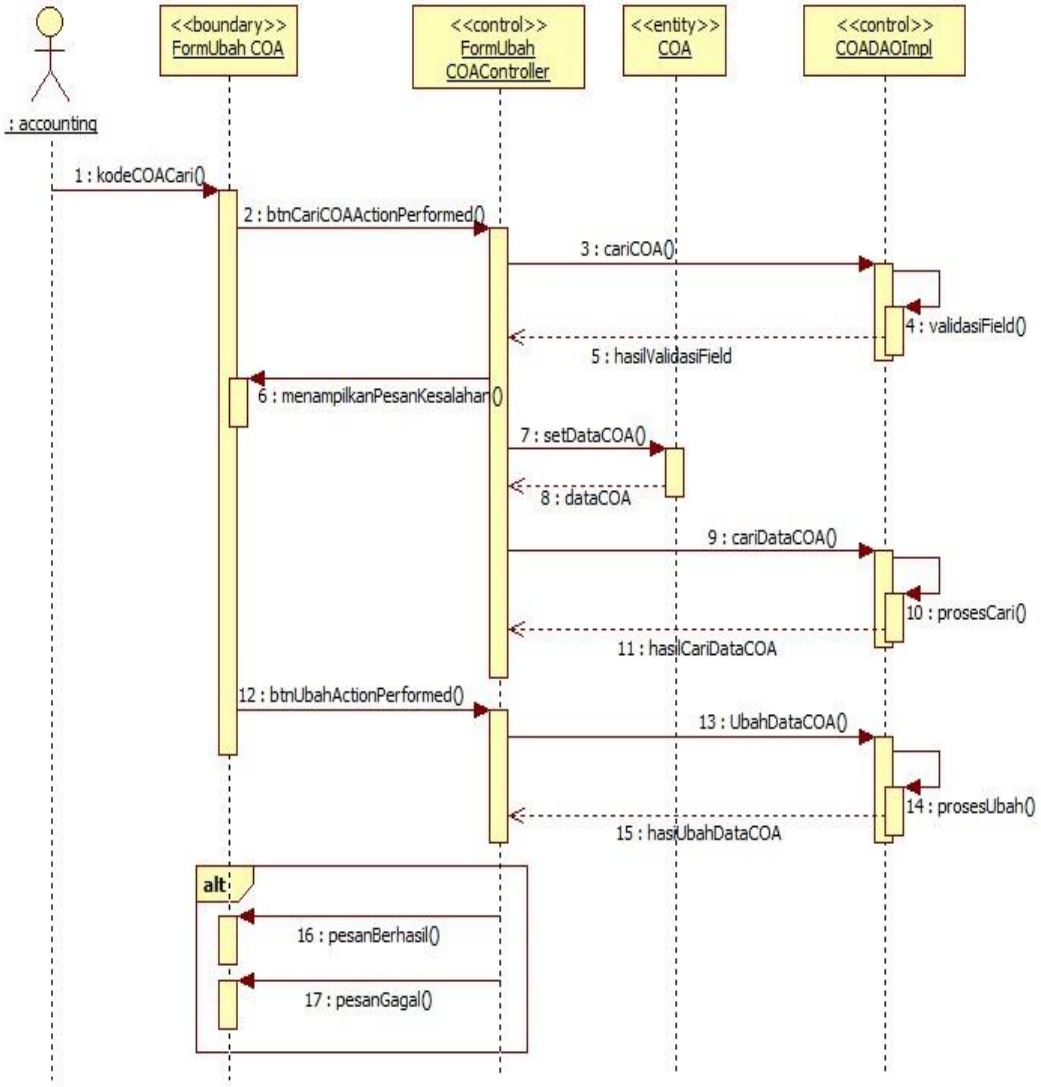
Sequence Diagram pengolahan data coa menunjukkan rangkaian pesan yang dikirim antara objek *accounting* dengan objek lainnya dalam proses pengolahan data coa. *Sequence diagram* pengolahan data coa terdiri dari se*quence diagram* tambah coa (Gambar 3.3), *sequence diagram* ubah coa (Gambar 3.4), *sequence diagram* hapus coa (Gambar 3.5).

1. *Sequence Diagram* Tambah COA



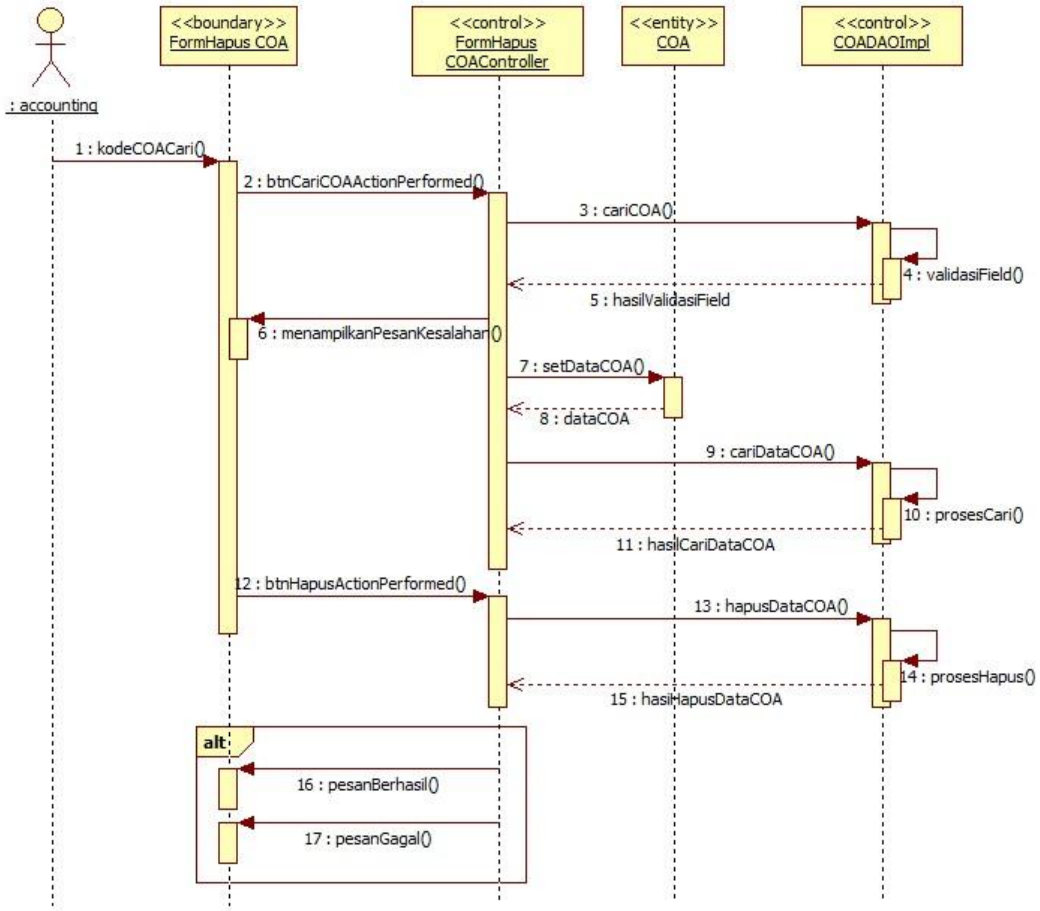
Gambar 3.3 *Sequence Diagram* Tambah COA

2. *Sequence Diagram* Ubah COA



Gambar 3.4 *Sequence Diagram* Ubah COA

3. *Sequence Diagram* Hapus COA

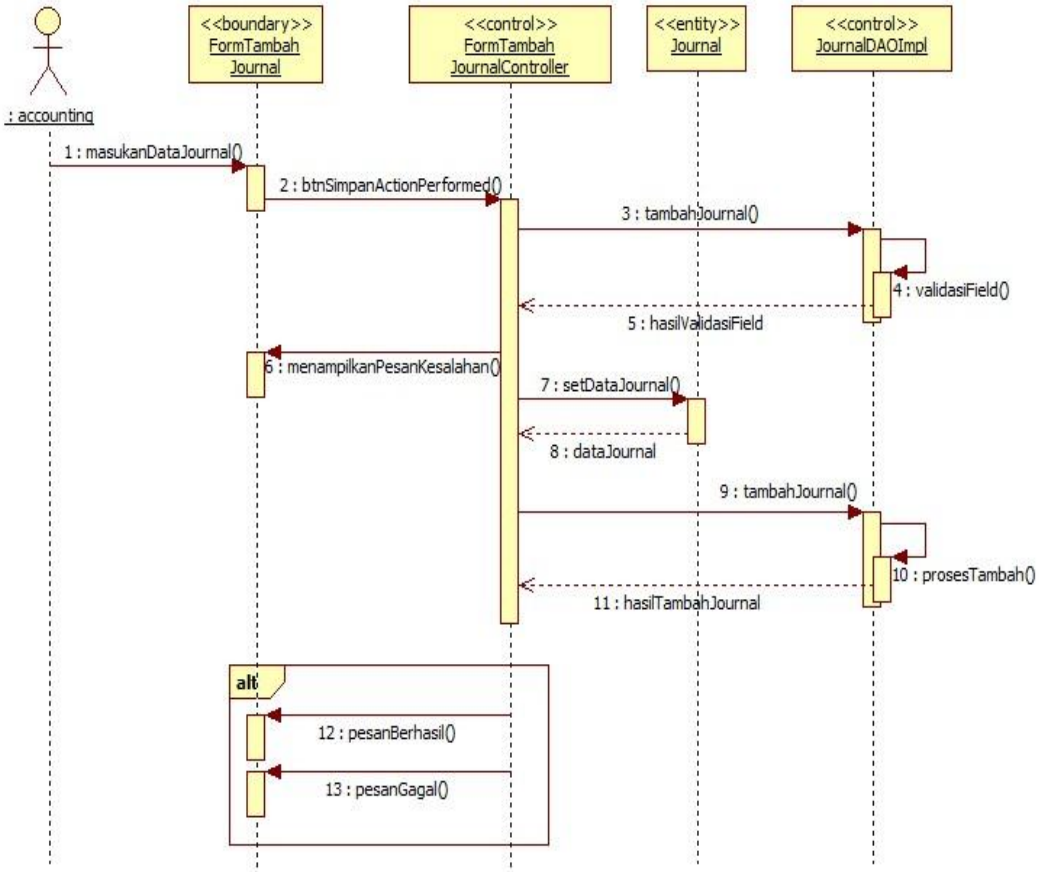


Gambar 3.5 *Sequence Diagram* Hapus COA

3. 1. 4. 3. 2 *Sequence Diagram* Pengolahan Data *Journal*

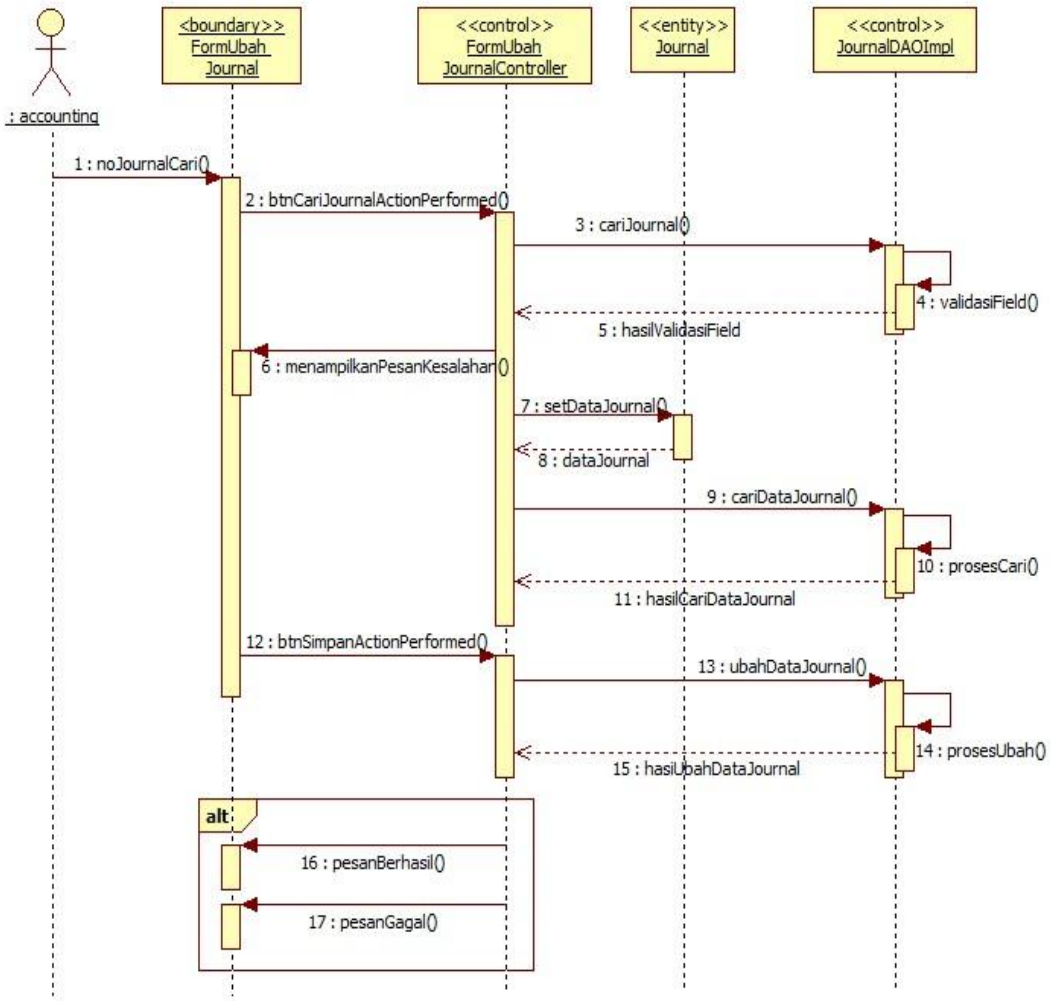
Sequence Diagram pengolahan data *journal* menunjukkan rangkaian pesan yang dikirim antara objek *accounting* dengan objek lainnya dalam proses pengolahan data *journal*. *Sequence diagram* pengolahan data *journal* terdiri dari *sequence diagram* tambah *journal* (Gambar 3.6), *sequence diagram* ubah *journal* (Gambar 3.7), dan *sequence diagram* hapus *journal* (Gambar 3.8).

1. *Sequence Diagram* Tambah *Journal*



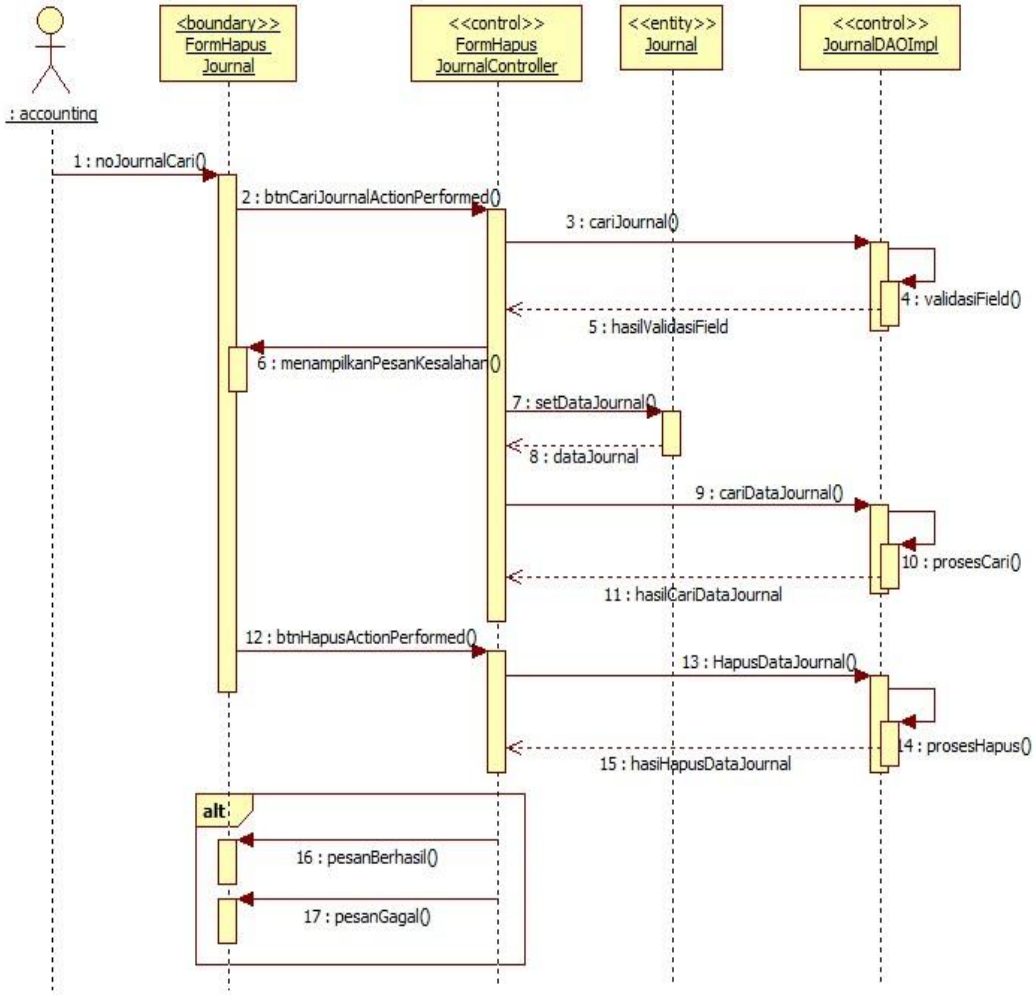
Gambar 3.6 *Sequence Diagram* Tambah *Journal*

2. *Sequence Diagram* Ubah *Journal*



Gambar 3.7 *Sequence Diagram* Ubah *Journal*

3. *Sequence Diagram* Hapus *Journal*

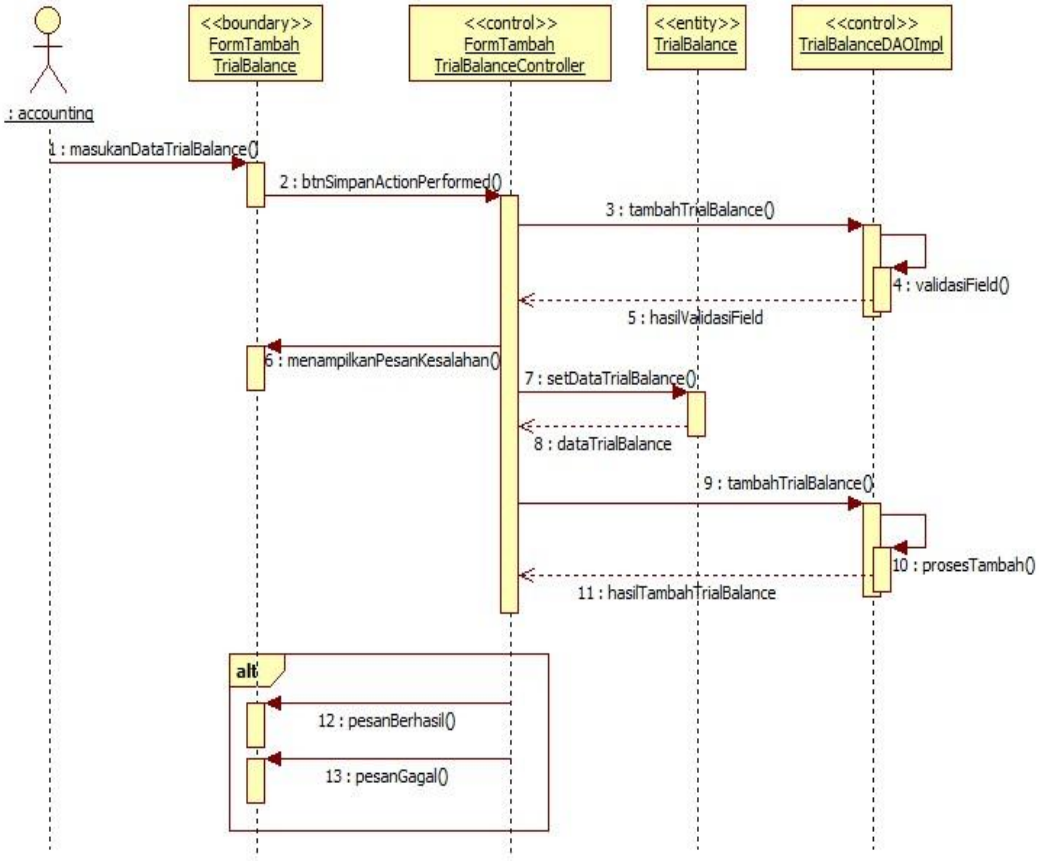


Gambar 3.8 *Squence Diagram* Hapus *Journal*

3. 1. 4. 3. 3 *Sequence Diagram* Pengolahan Data *Trial Balance*

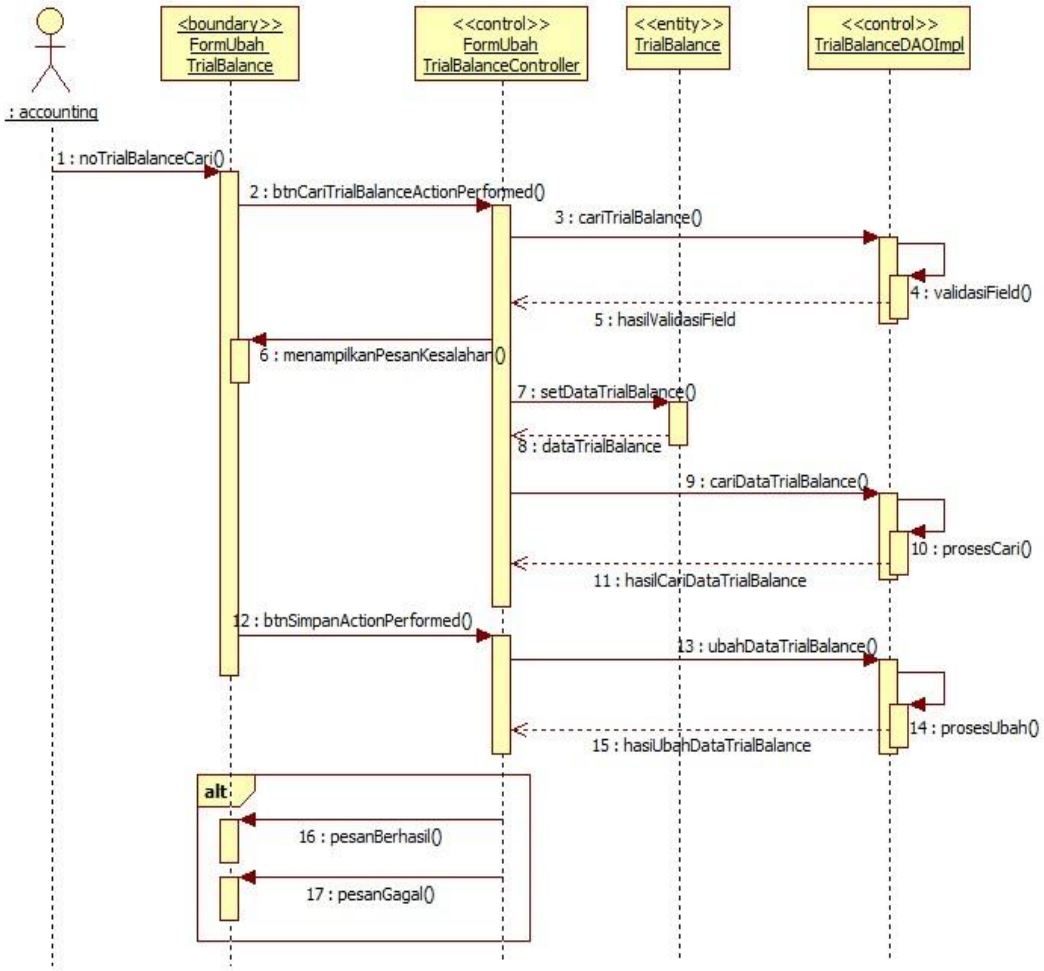
Sequence Diagram pengolahan data *trial balance* menunjukkan rangkaian pesan yang dikirim antara objek *accounting* dengan objek lainnya dalam proses pengolahan data *trial balance*. *Sequence Diagram* pengolahan data *trial balance* terdiri dari *sequence diagram* tambah *trial balance* (Gambar 3.9) , *sequence diagram* ubah *trial balance* (Gambar 3.10), dan *sequence diagram* hapus *trial balance* (Gambar 3.11).

1. *Sequence Diagram* Tambah *Trial Balance*



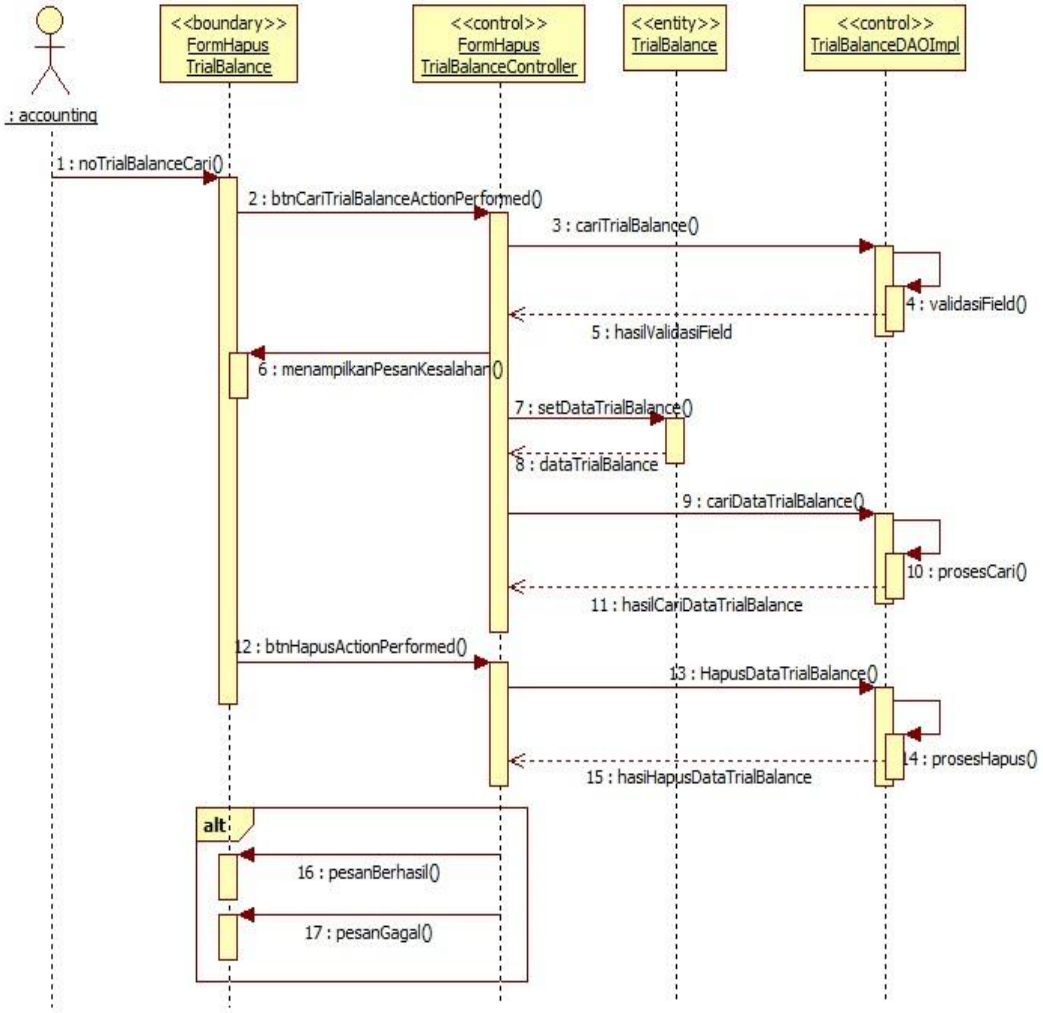
Gambar 3.9 *Sequence Diagram* Tambah *Trial Balance*

2. *Sequence Diagram* Ubah *Trial Balance*



Gambar 3.10 *Sequence Diagram* Ubah *Trial Balance*

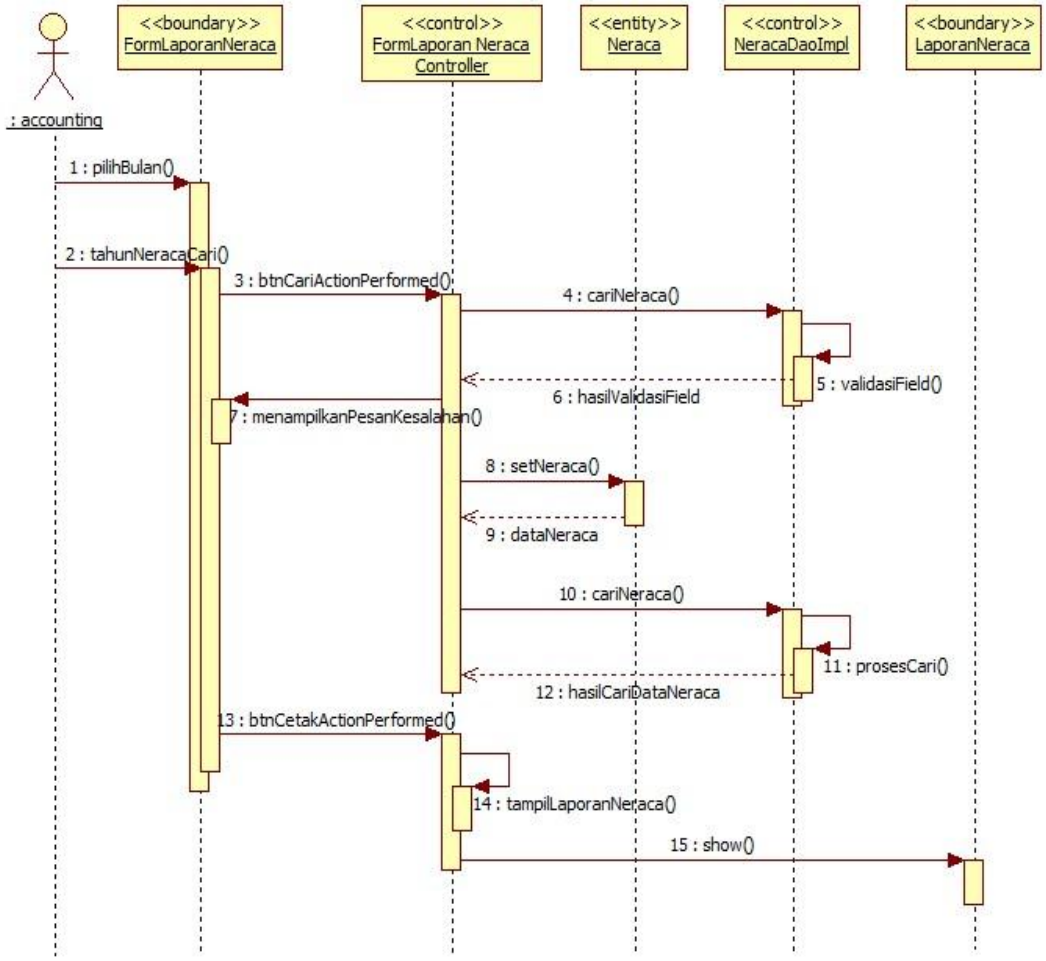
3. *Sequence Diagram* Hapus *Trial Balance*



Gambar 3.11 *Sequence Diagram* Hapus *Trial Balance*

3. 1. 4. 3. 4 *Sequence Diagram* Laporan Neraca

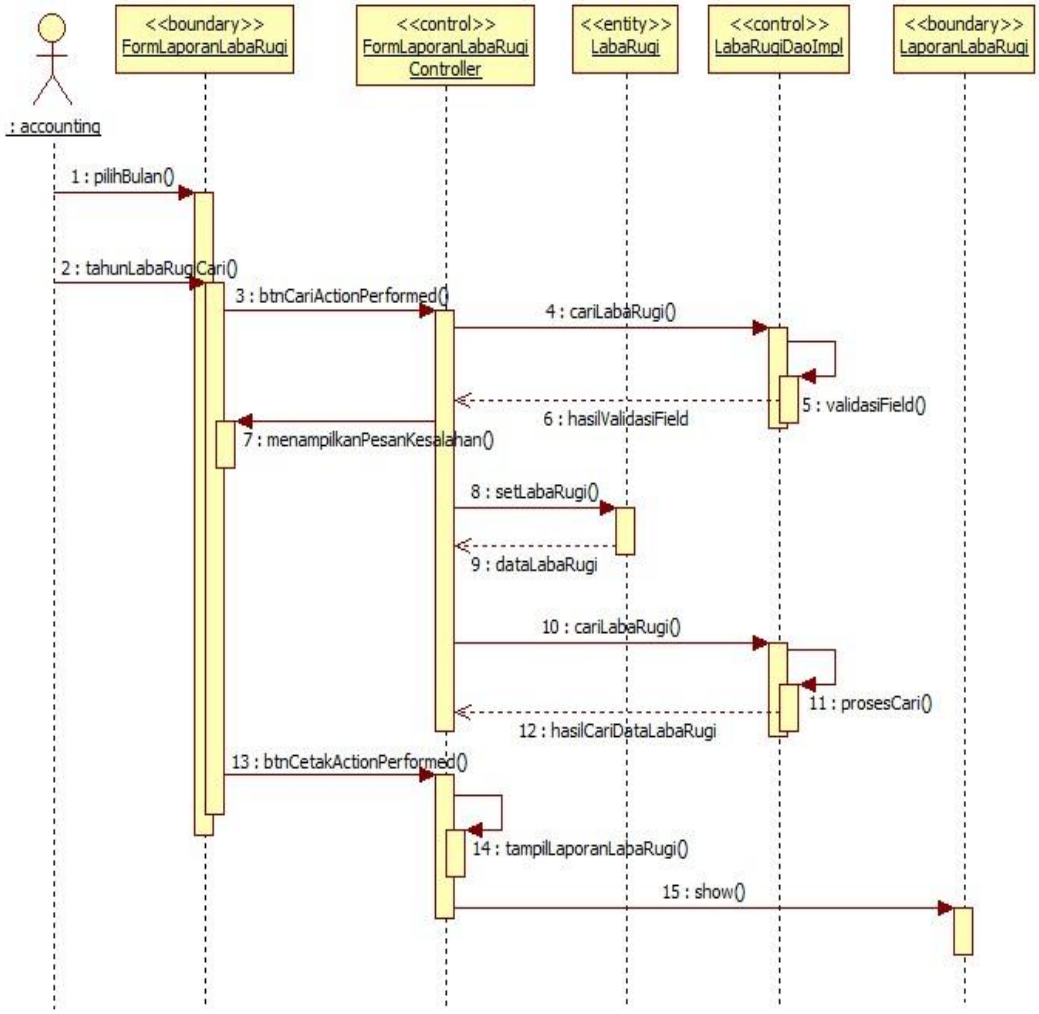
Sequence Diagram cetak laporan neraca menunjukkan rangkaian pesan yang dikirim antara objek *accounting* dengan objek lainnya dalam proses pencetakan laporan neraca. *Sequence Diagram* cetak laporan neraca dapat dilihat pada Gambar 3.12.



Gambar 3.12 *Sequence Diagram* Laporan Neraca

3. 1. 4. 3. 5 *Sequence Diagram* Laporan Laba Rugi

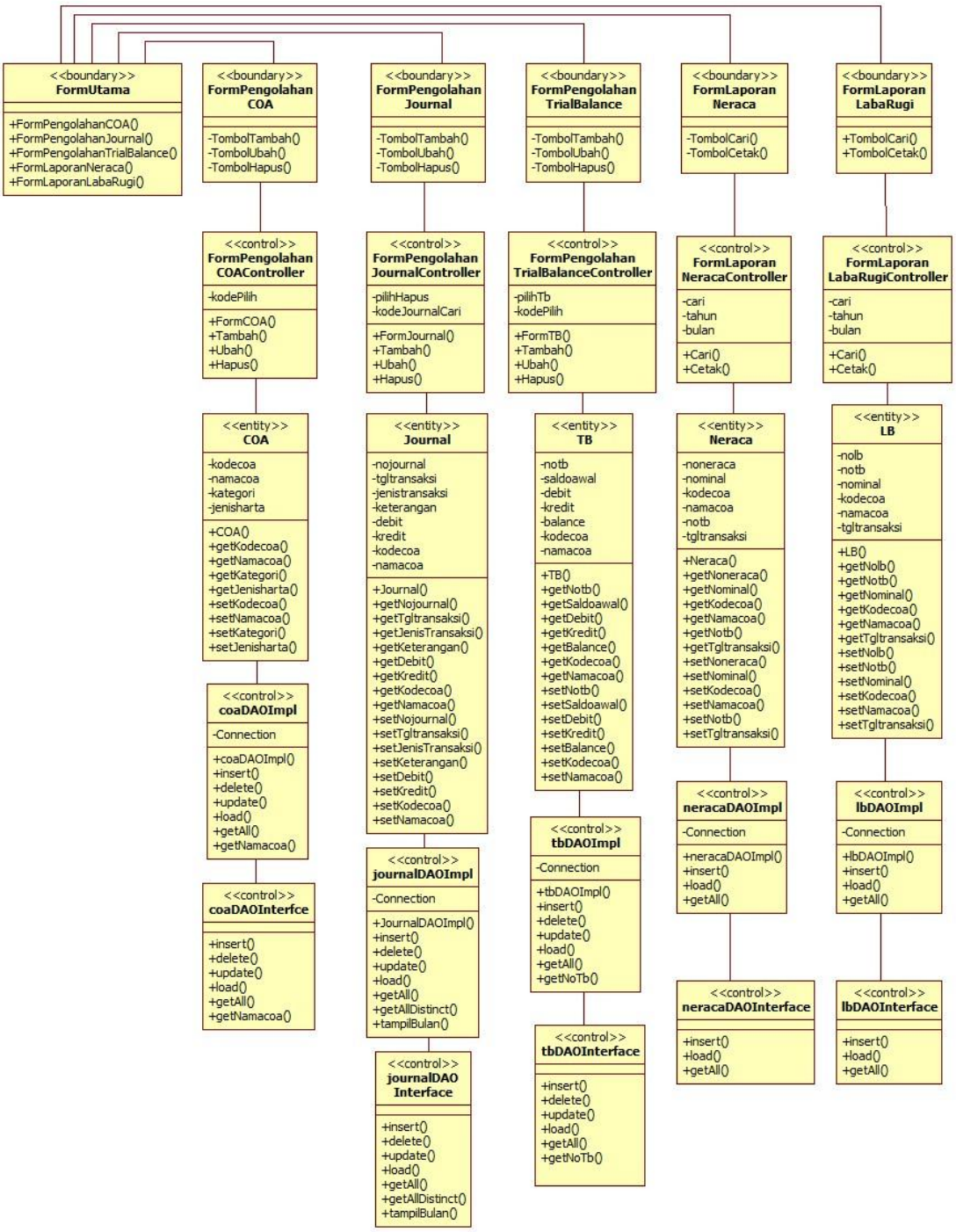
Sequence Diagram cetak laporan laba rugi menunjukkan rangkaian pesan yang dikirim antara objek *accounting* dengan objek lainnya dalam proses pencetakan laporan laba rugi. *Sequence Diagram* cetak laporan laba rugi dapat dilihat pada Gambar 3.13.



Gambar 3.13 *Sequence Diagram* Laporan Laba Rugi

3. 1. 4. 4. *Class Diagram*

Class Diagram menggambarkan struktur dan hubungan antar objek-objek yang ada pada sistem. Struktur itu meliputi atribut dan *method* yang ada pada masing-masing *class*. Hubungnnya dari masing-masing *class* yang ada tersebut digambarkan dengan menggunakan pewarisan dan generalisasi seperti terlihat pada Gambar 3.25



Gambar 3.14 ***Class Diagram***

3. 2. Perancangan Sistem

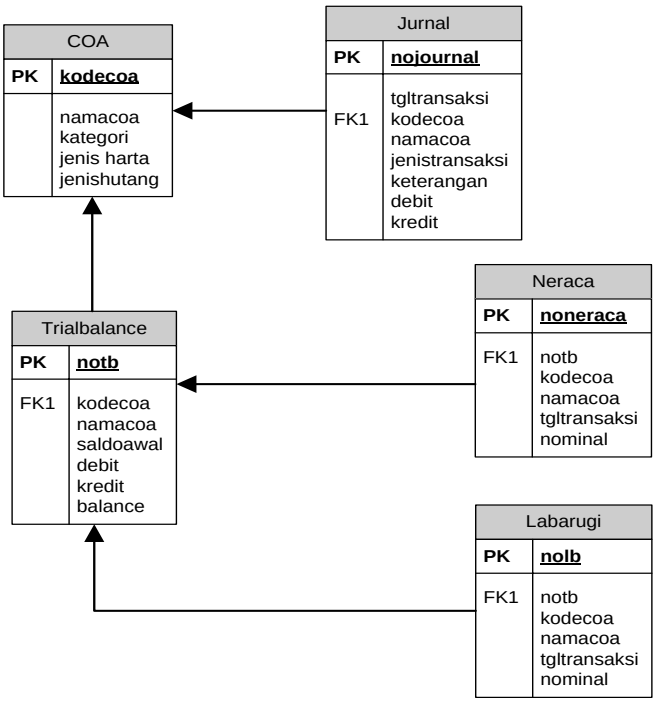
Perancangan merupakan penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam suatu kesatuan yang utuh dan berfungsi.

3. 2. 1. Perancangan Data

Perancangan data merupakan tahapan untuk memetakan model konseptual ke model basis data yang akan dipakai. Perancangan basis data terbagi menjadi dua yaitu skema relasi dan perancangan struktur tabel.

3. 2. 1. 1. Skema Relasi

Skema relasi merupakan rangkaian hubungan antara dua tabel atau lebih pada sistem basis data. Gambar 3.31 merupakan penjelasan rangkaian basis data pada Aplikasi *General Ledger* Bulanan pada CV. Global Agency and Entertainment.



Gambar 3.15 Skema Relasi

3. 2. 1. 2. Struktur Tabel

Struktur tabel menggambarkan detail tabel yang berisi *field*, tipe data, panjang data, dan keterangan lainnya. Tabel yang terdapat pada basis data aplikasi *General Ledger* bulanan di CV. Global Agency and Entertainment adalah tabel coa, tabel *Journal*, tabel *Trial Balance*, tabel neraca, dan tabel laba rugi. Berikut ini deskripsi dari masing-masing tabel tersebut.

1. **Tabel COA**

Tabel 3.17 Struktur Tabel COA

Nama Field	Tipe Data	Panjang	Null	Kunci
kodecoa	integer	-	No	*Primary Key*
namacoa	varchar	80	No	-
kategori	varchar	19	No	-
jenisharta	varchar	20	Yes	-

2. **Tabel *Journal***

Tabel 3.18 Struktur Tabel *Journal*

Nama Field	Tipe Data	Panjang	Null	Kunci
no*Journal*	integer	-	No	*Primary Key*
tgltransaksi	date	-	No	-
kodecoa	integer	-	No	*Foreign Key*
namacoa	varchar	80	No	-
jenistransaksi	varchar	9	No	-
keterangan	varchar	100	No	-
debit	double	-	No	-
kredit	double	-	No	-

3. Tabel *Trial Balance*

Tabel 3.19 Struktur Tabel *Trial Balance*

Nama Field	Tipe Data	Panjang	Null	Kunci
notb	integer	-	No	*Primary Key*
kodecoa	integer	-	No	*Foreign Key*
namacoa	varchar	80	No	-
saldoawal	double	-	No	-
debit	double	-	No	-
kredit	double	-	No	-
balance	double	-	No	-

4. Tabel Neraca

Tabel 3.20 Struktur Tabel Neraca

Nama Field	Tipe Data	Panjang	Null	Kunci
noneraca	integer	-	No	*Primary Key*
notb	integer	-	No	*Foreign Key*
kodecoa	integer	-	No	-
namacoa	varchar	80	No	-
tgltransaksi	date	-	No	-
nominal	double	-	No	-

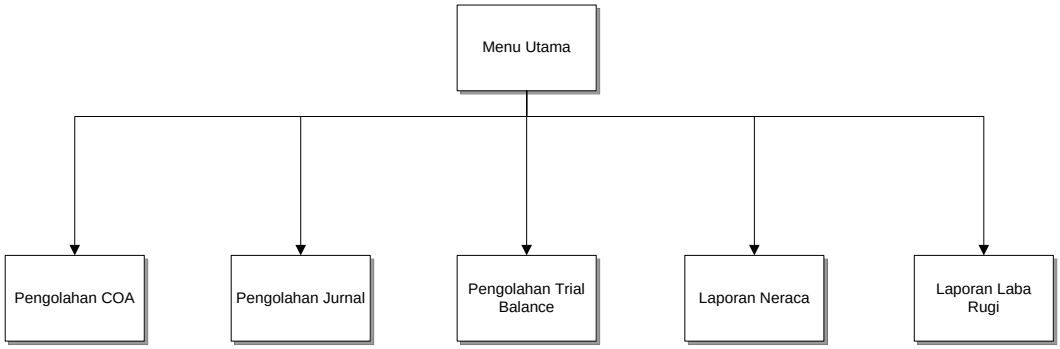
5. Tabel Laba Rugi

Tabel 3.21 Struktur Tabel Laba Rugi

Nama Field	Tipe Data	Panjang	Null	Kunci
nolb	integer	-	No	*Primary Key*
notb	integer	-	No	*Foreign Key*
kodecoa	integer	-	No	-
namacoa	varchar	80	No	-
tgltransaksi	date	-	No	-
nominal	double	-	No	-

3. 2. 2. Struktur Menu

Perancangan struktur menu merupakan gambaran jalur pemakaian aplikasi sehingga aplikasi yang dibangun mudah dipahami dan mudah digunakan. Perancangan struktur menu dari aplikasi *General Ledger* bulanan di CV. Global Agency and Entertainment dapat dilihat pada Gambar 3.27.



Gambar 3.16 Struktu Menu Aplikasi *General Ledger* Bulanan

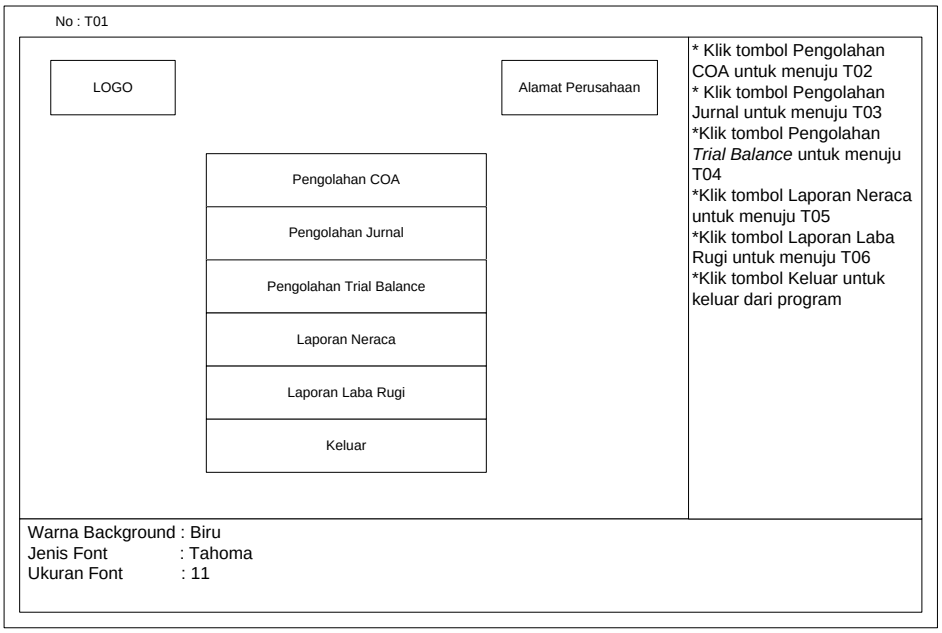
3. 2. 3. Perancangan Antar Muka

Perancangan antarmuka mendeskripsikan rencana tampilan dari setiap tampilan yang akan digunakan pada aplikasi *General Ledger* bulanan di CV. Global Agency and Entertainment. Perancangan antarmuka pada aplikasi *General Ledger* bulanan di CV. Global Agency and Entertainment terdiri dari perancangan *form* dan pesan.

3. 2. 3. 1. Perancangan *Form*

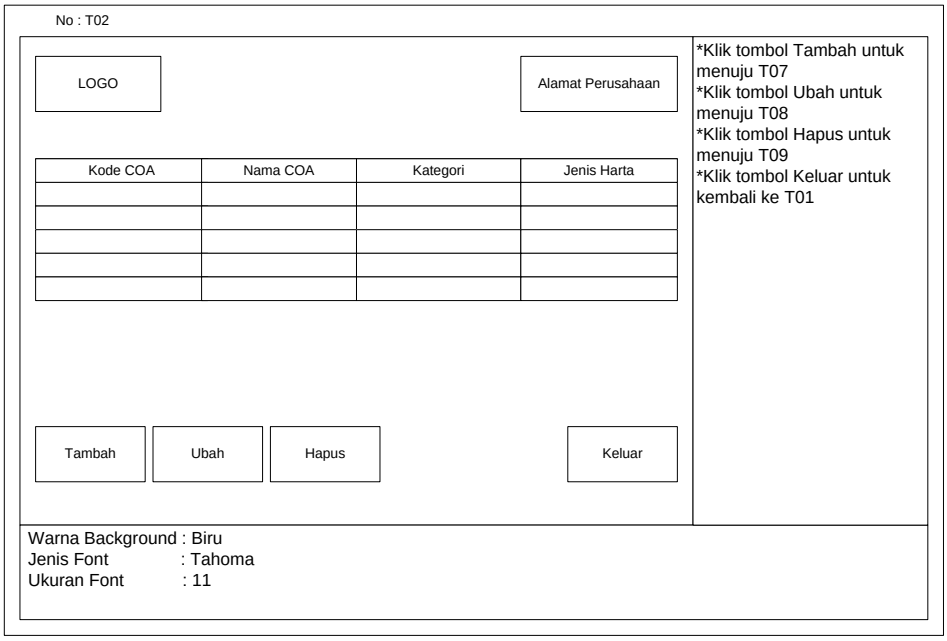
Perancangan antarmuka *form* pada aplikasi *General Ledger* bulanan di CV. Global Agency and Entertainment dapat dilihat pada Gambar 3.28 sampai dengan Gambar 3.42.

1. *Form* Menu Utama



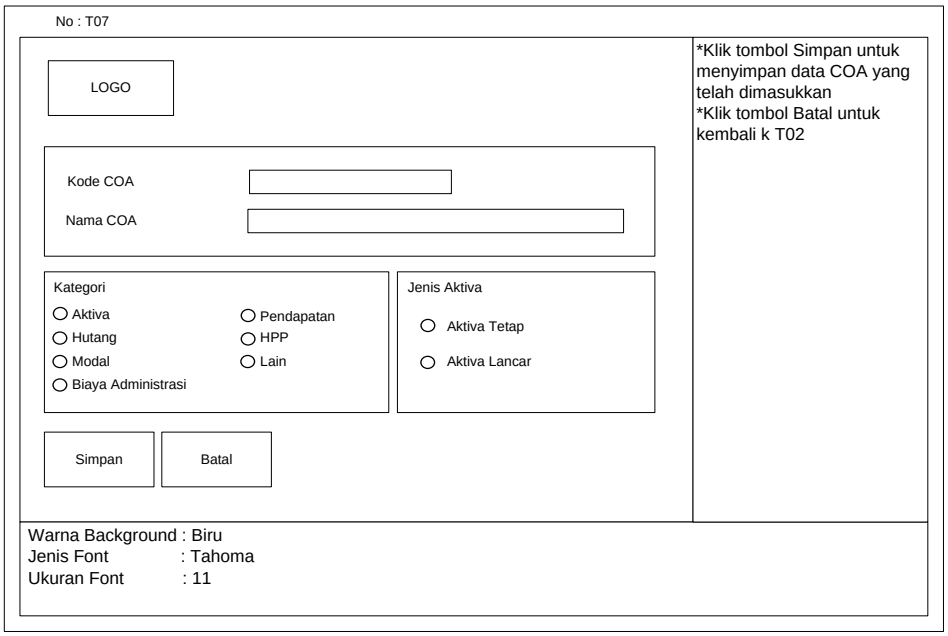
Gambar 3.17 *Form* Menu Utama

2. *Form* Pengolahan Data COA



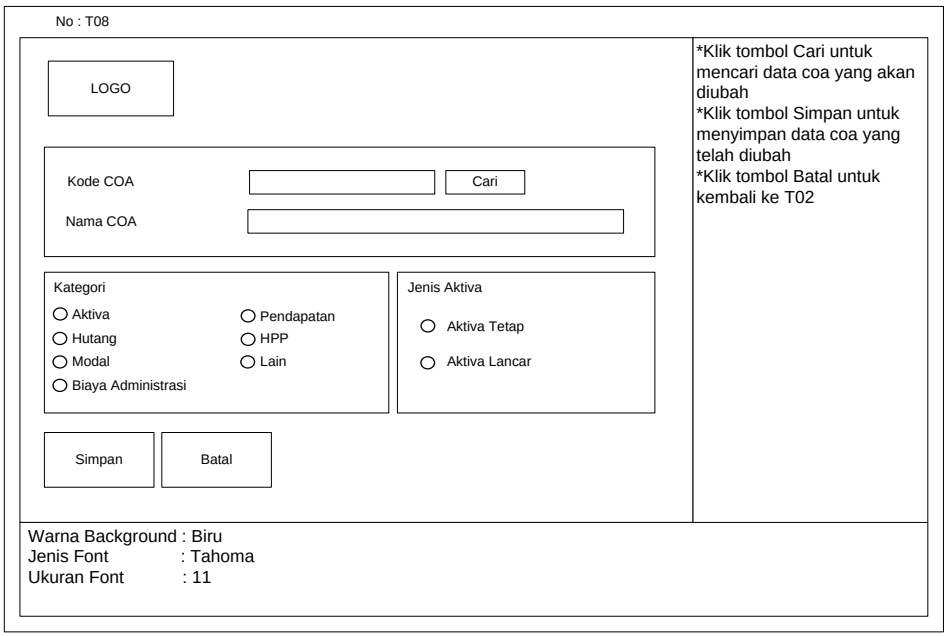
Gambar 3.18 *Form* Pengolahan Data COA

3. *Form* Tambah Data COA



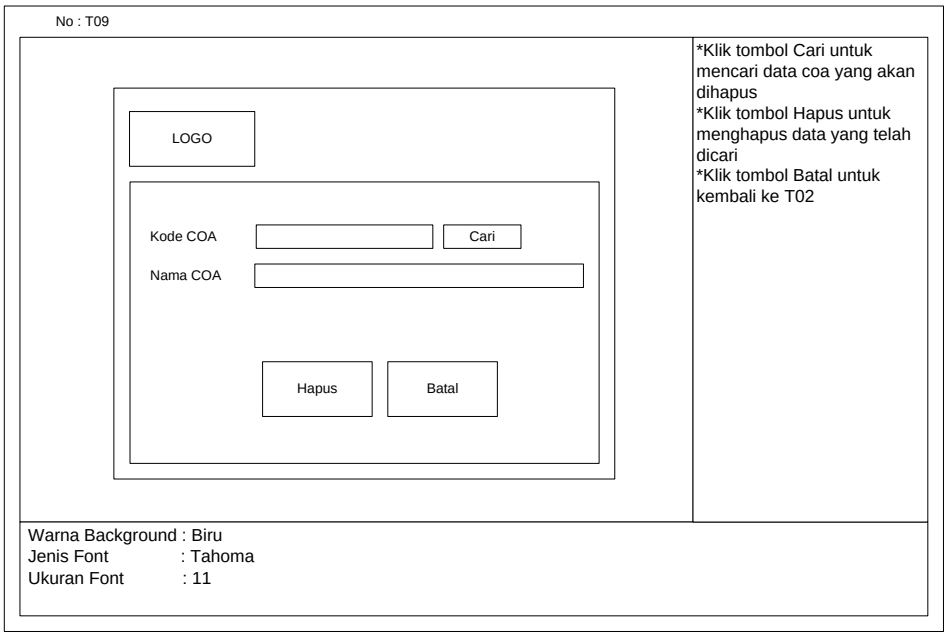
Gambar 3.19 *Form* Tambah Data COA

4. *Form* Ubah Data COA



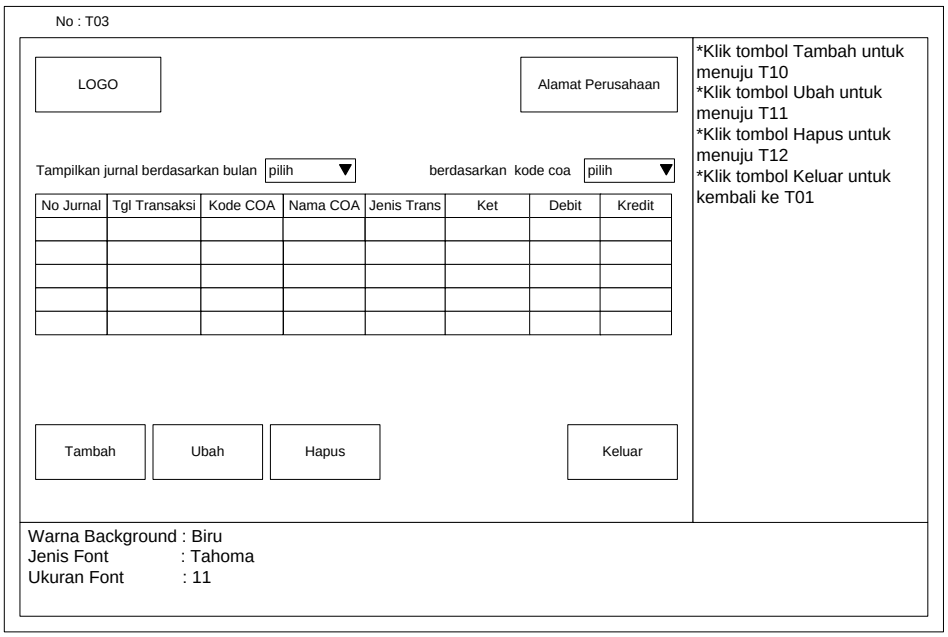
Gambar 3.20 *Form* Ubah Data COA

5. *Form* Hapus Data COA



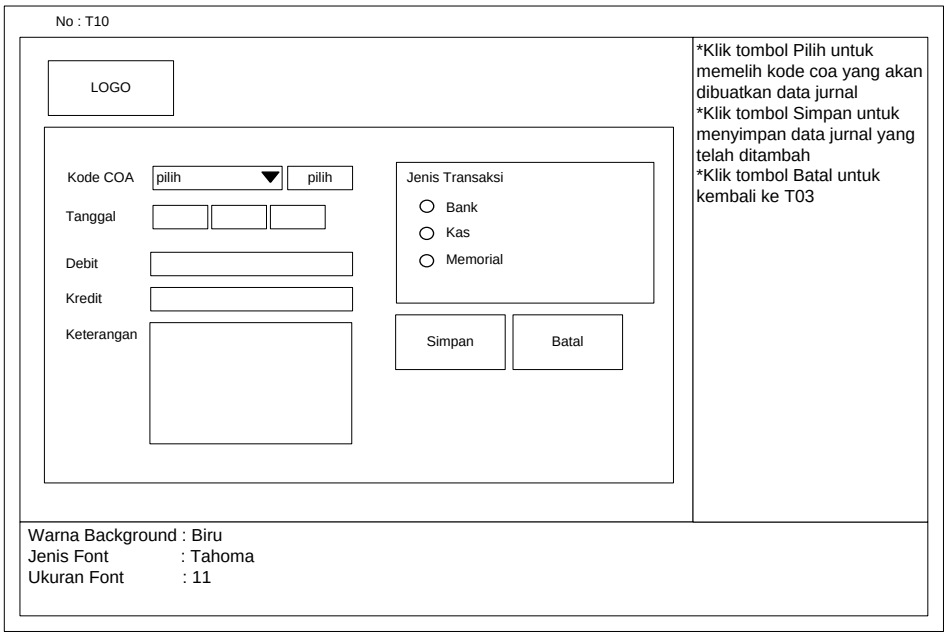
Gambar 3.21 *Form* Hapus Data COA

6. *Form* Pengolahan Data *Journal*



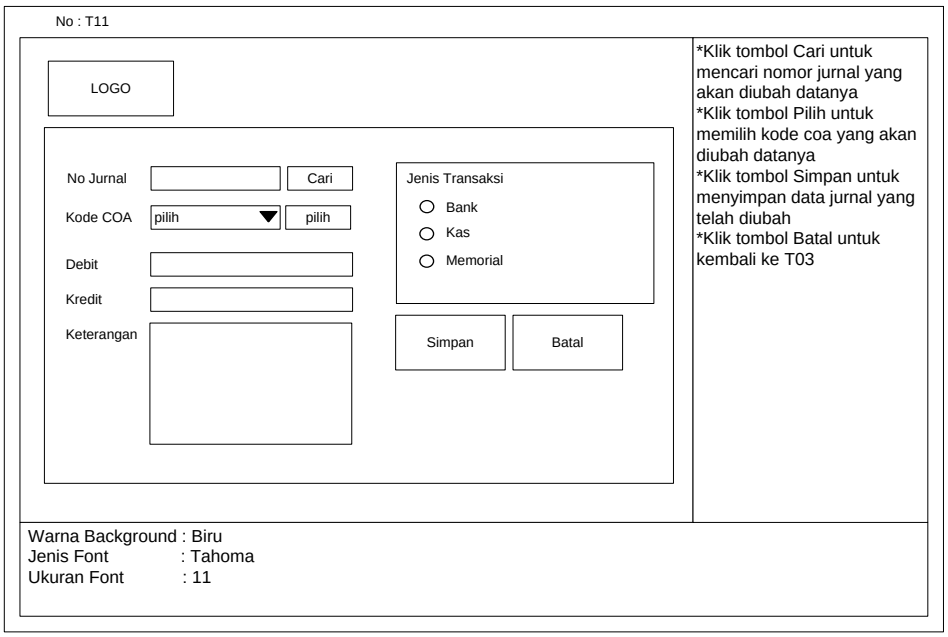
Gambar 3.22 *Form* Pengolahan Data *Journal*

7. *Form* Tambah Data *Journal*



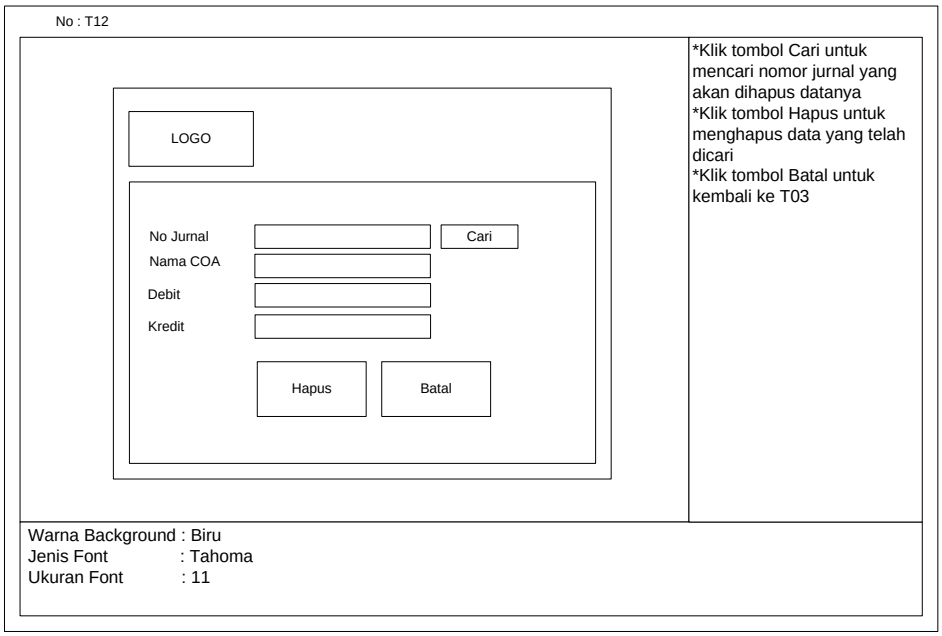
Gambar 3.23 *Form* Tambah Data *Journal*

8. *Form* Ubah Data *Journal*



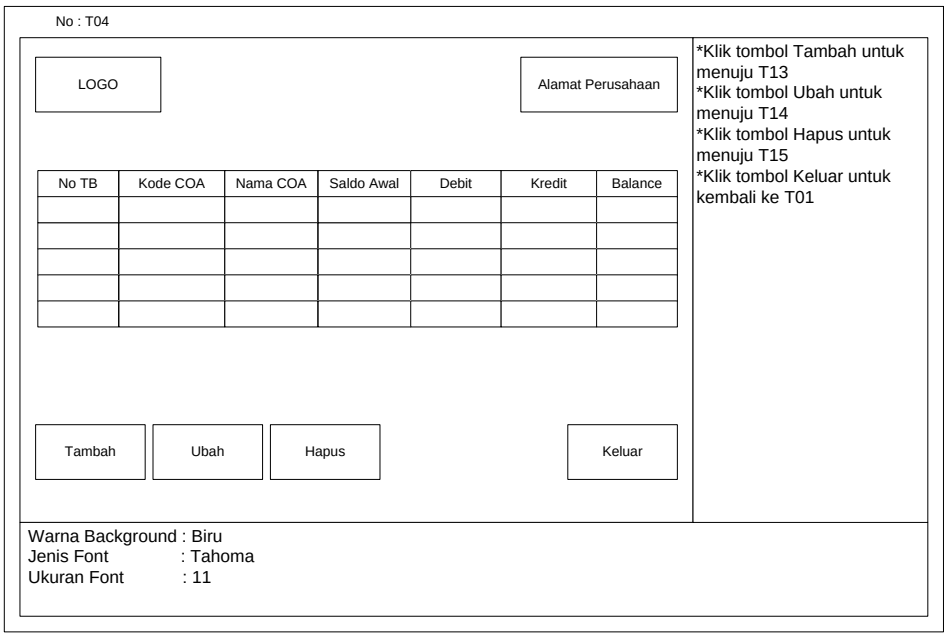
Gambar 3.24 *Form* Ubah Data *Journal*

9. *Form* Hapus Data *Journal*



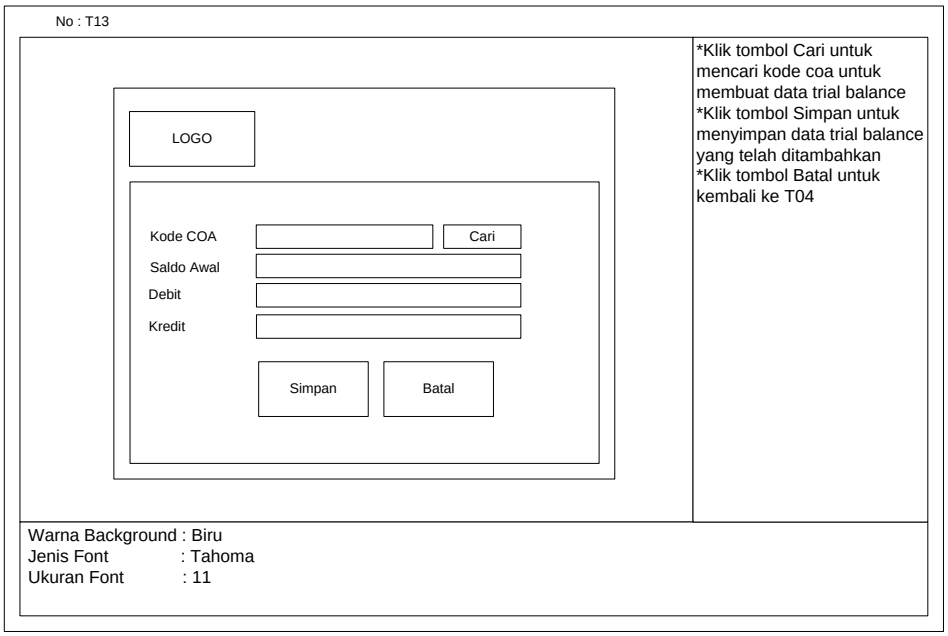
Gambar 3.25 *Form* Hapus Data *Journal*

10. *Form* Pengolahan Data *Trial Balance*



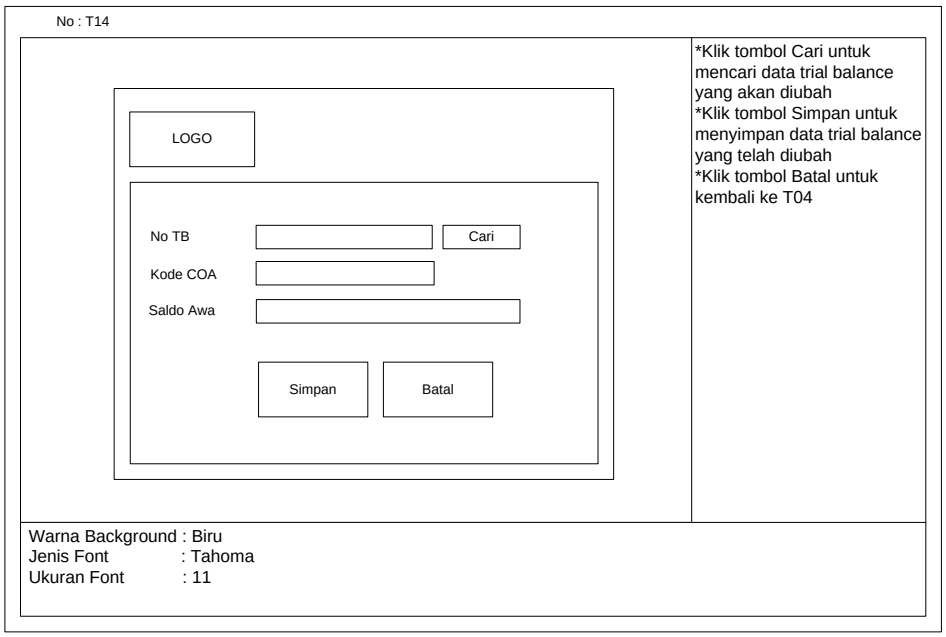
Gambar 3.26 *Form* Pengolahan Data *Trial Balance*

11. *Form* Tambah Data *Trial Balance*



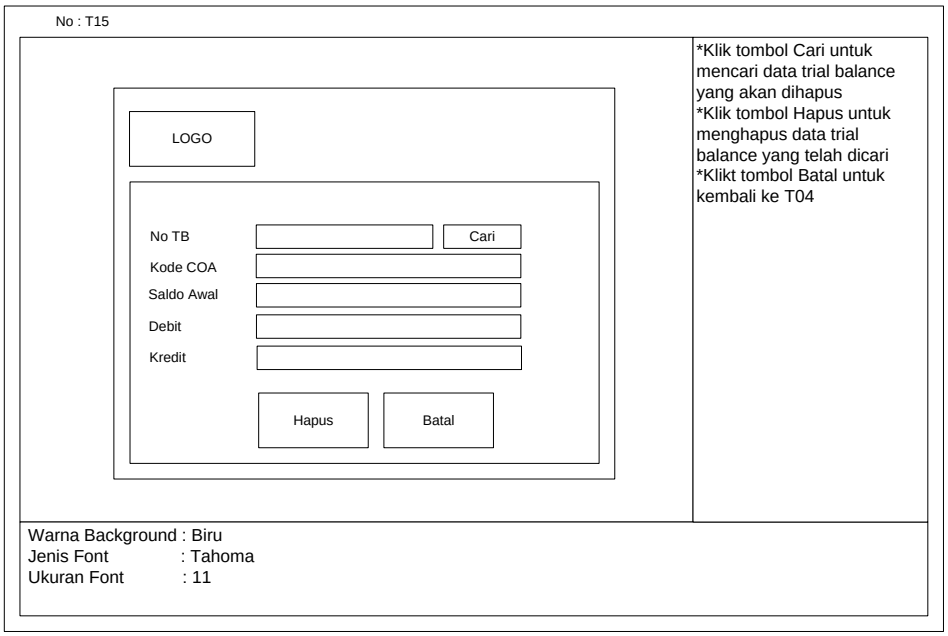
Gambar 3.27 *Form* Tambah Data *Trial Balance*

12. *Form* Ubah Data *Trial Balance*



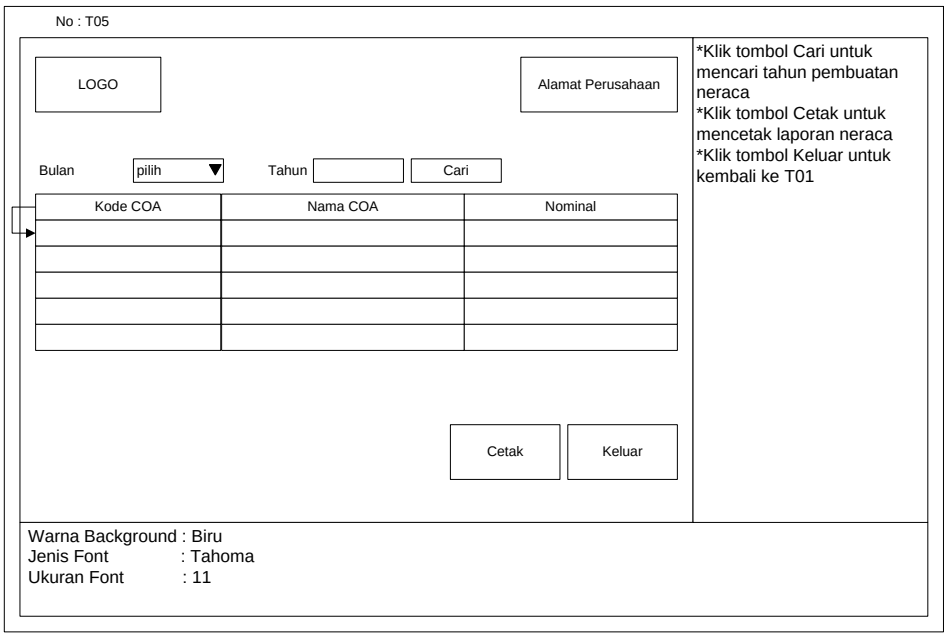
Gambar 3.28 *Form* Ubah Data *Trial Balance*

13. *Form* Hapus Data *Trial Balance*



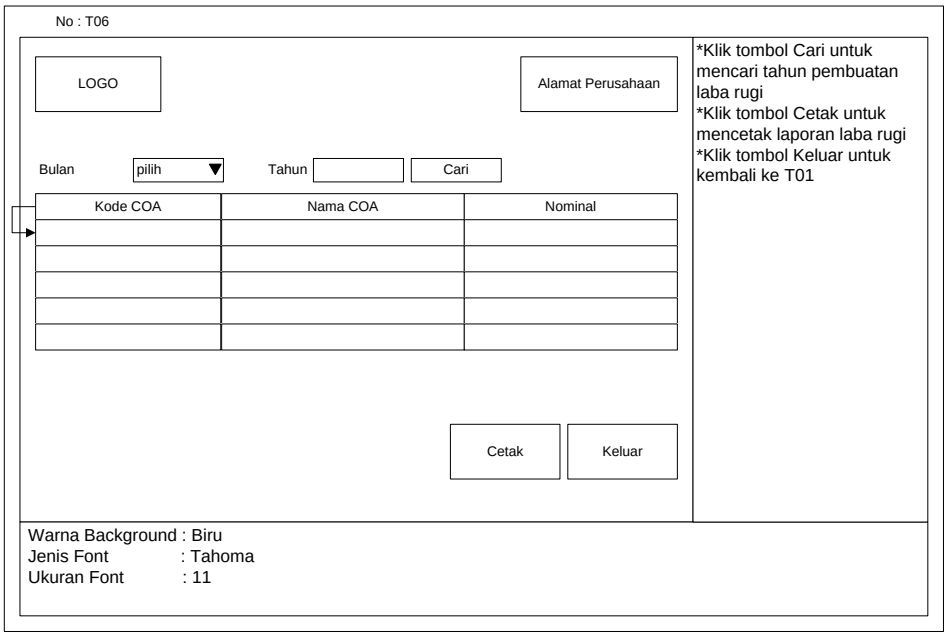
Gambar 3.29 *Form* Hapus Data *Trial Balance*

14. *Form* Pengolahan Laporan Neraca



Gambar 3.30 *Form* Pengolahan Laporan Neraca

15. *Form* Pengolahan Laporan Laba Rugi

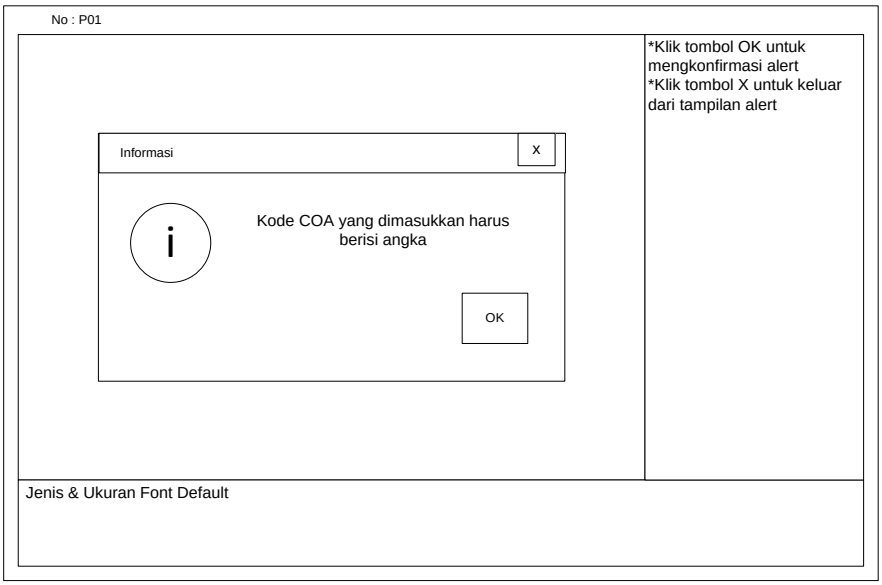


Gambar 3.31 *Form* Pengolahan Laporan Laba Rugi

3. 2. 3. 2. Perancangan Pesan

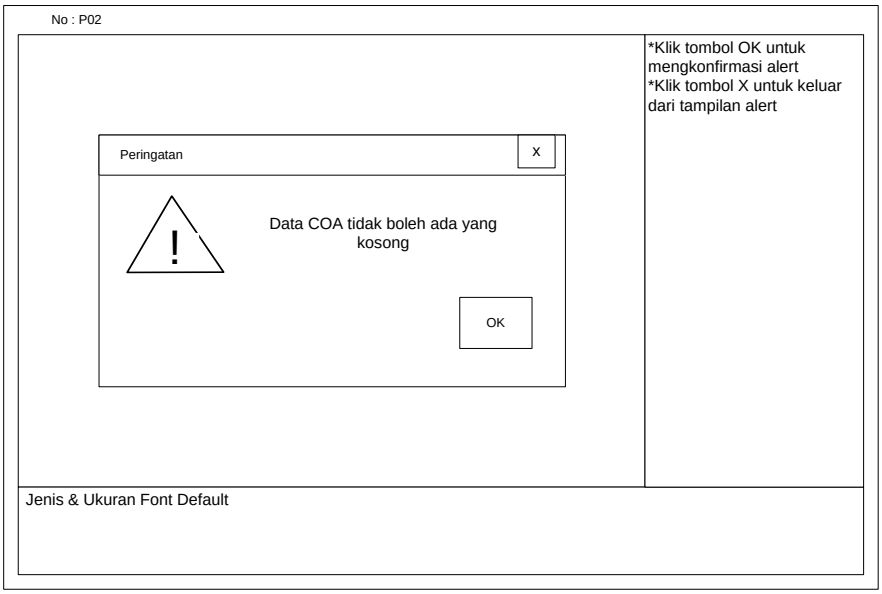
Perancangan antarmuka *form* pada aplikasi *General Ledger* bulanan di CV. Global Agency and Entertainment dapat dilihat pada Gambar 3.43 sampai dengan Gambar 3.73.

1. Perancangan Pesan P01



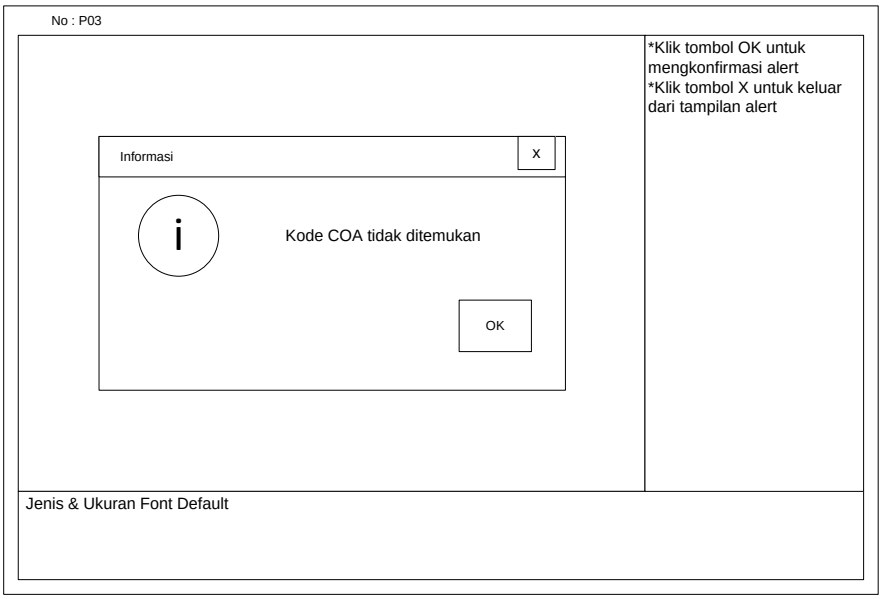
Gambar 3.32 Perancangan Pesan P01

2. Perancangan Pesan P02



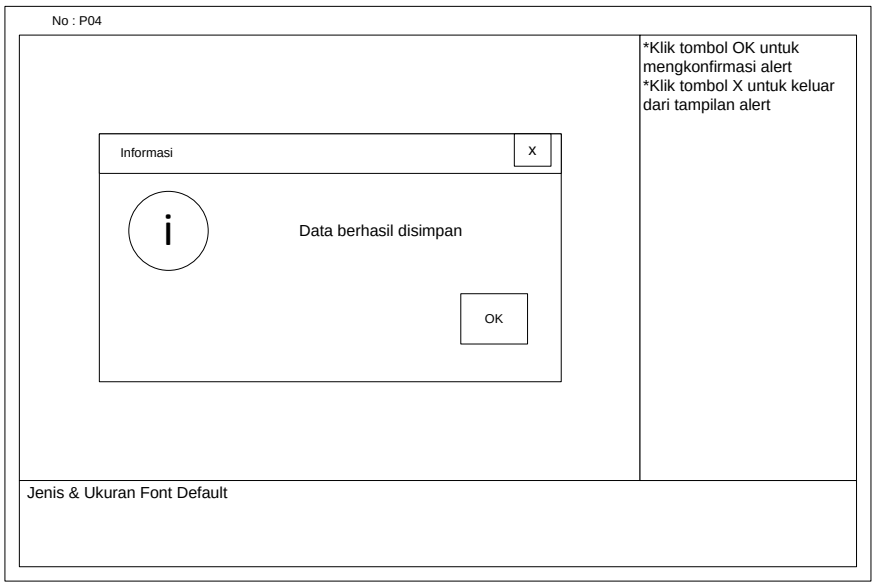
Gambar 3.33 Perancangan Pesan P02

3. Perancangan Pesan P03



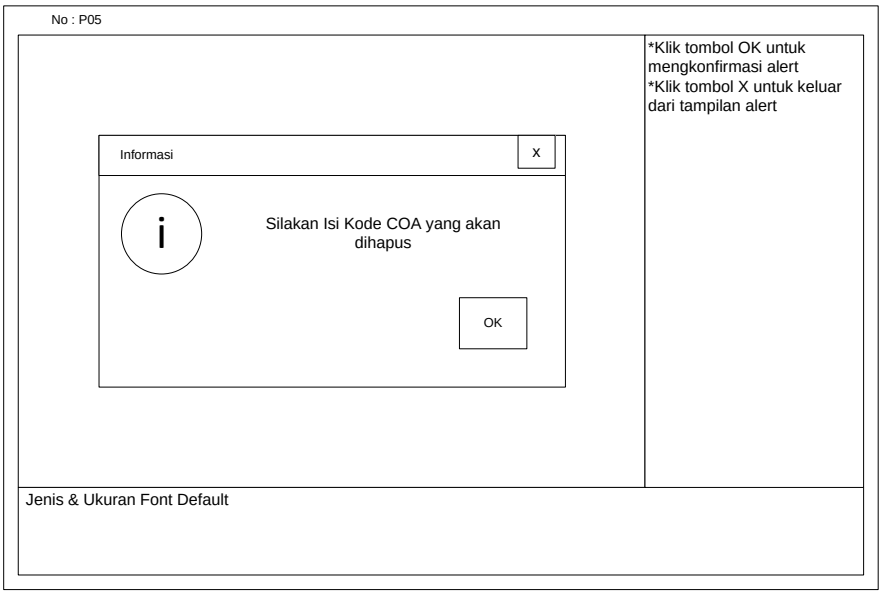
Gambar 3.34 Perancangan Pesan P03

4. Perancangan Pesan P04



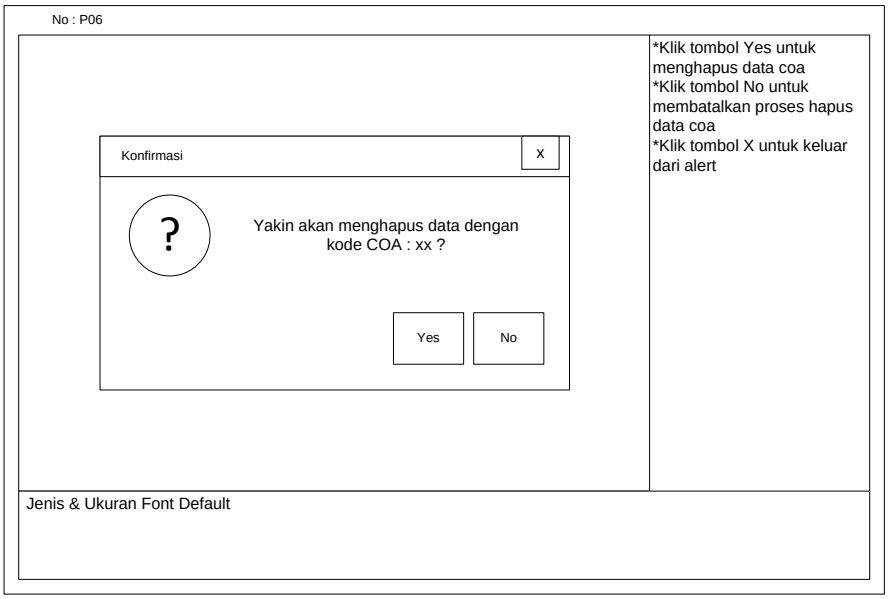
Gambar 3.35 Perancangan Pesan P04

5. Perancangan Pesan P05



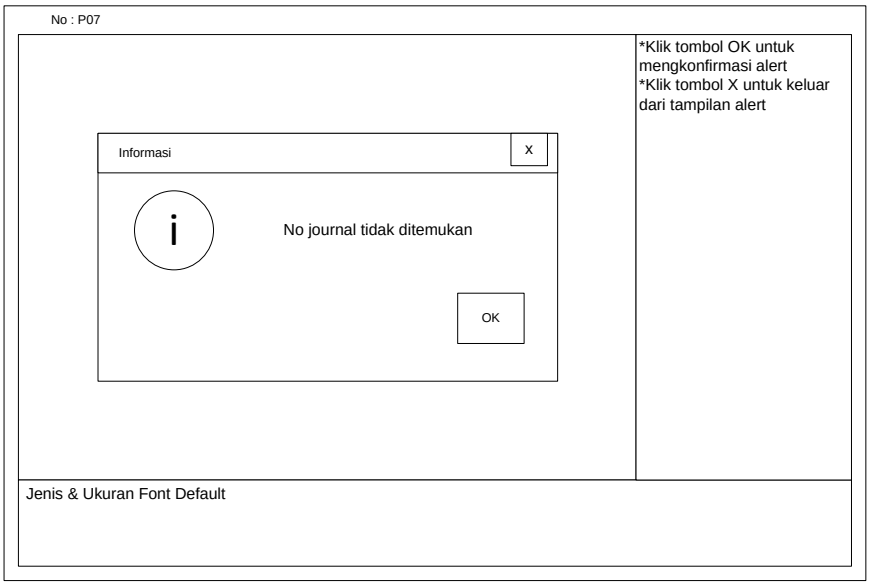
Gambar 3.36 Perancangan Pesan P05

6. Perancangan Pesan P06



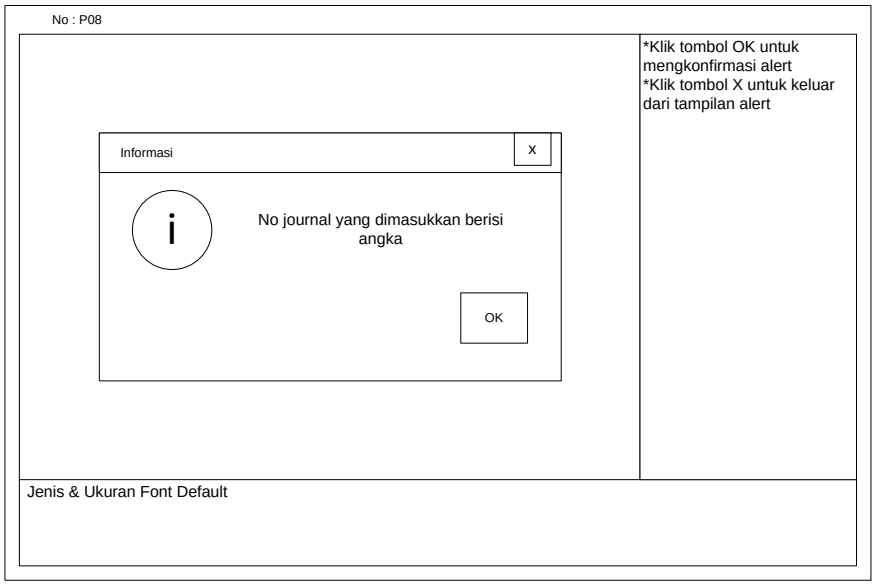
Gambar 3.37 Perancangan Pesan P06

7. Perancangan Pesan P07



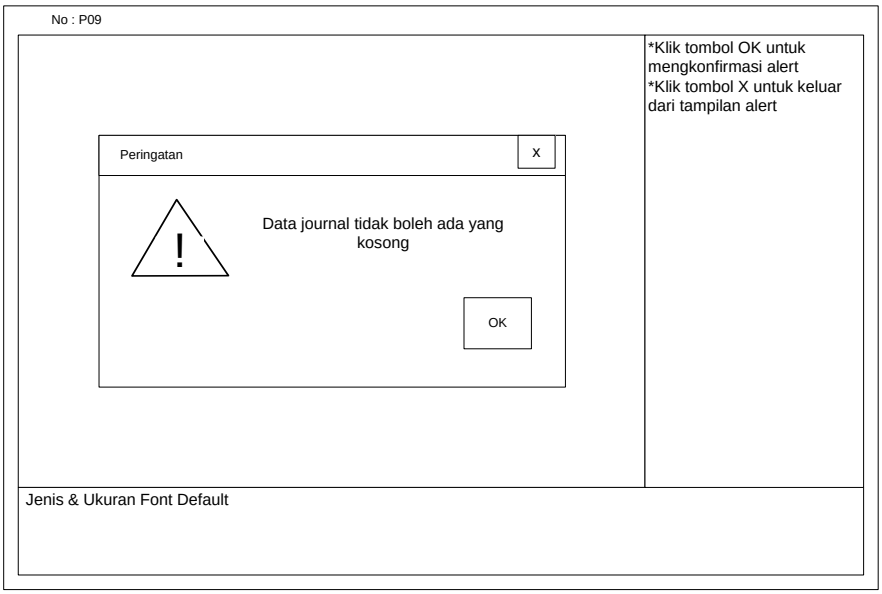
Gambar 3.38 Perancangan Pesan P07

8. Perancangan Pesan P08



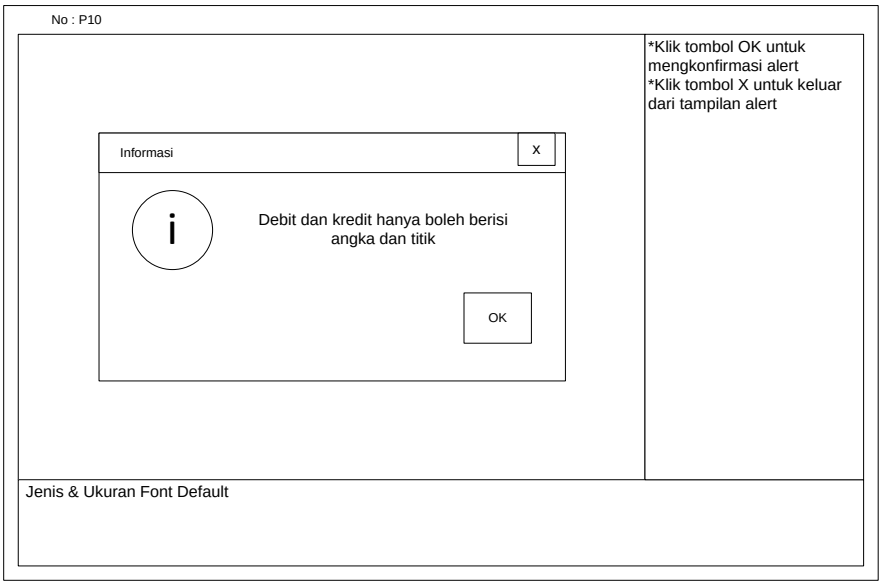
Gambar 3.39 Perancangan Pesan P08

9. Perancangan Pesan P09



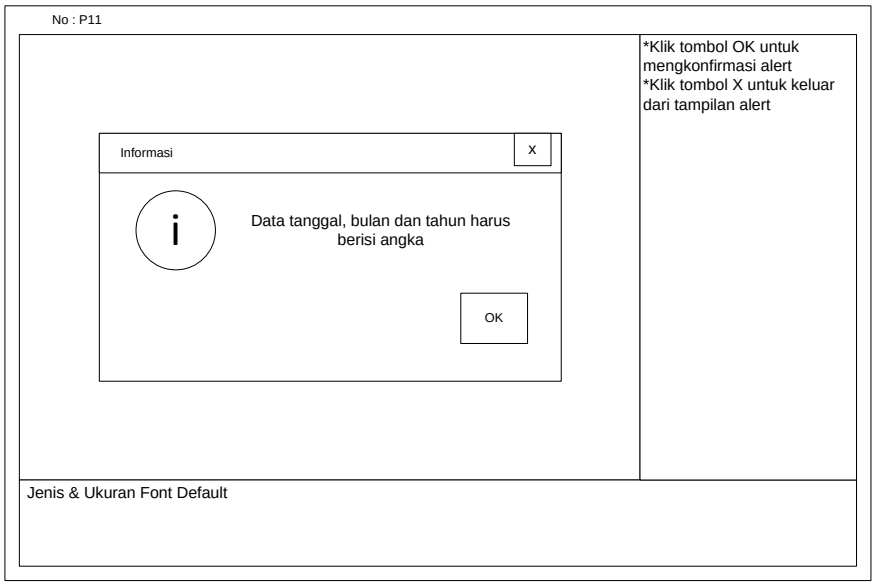
Gambar 3.40 Perancangan Pesan P09

10. Perancangan Pesan P10



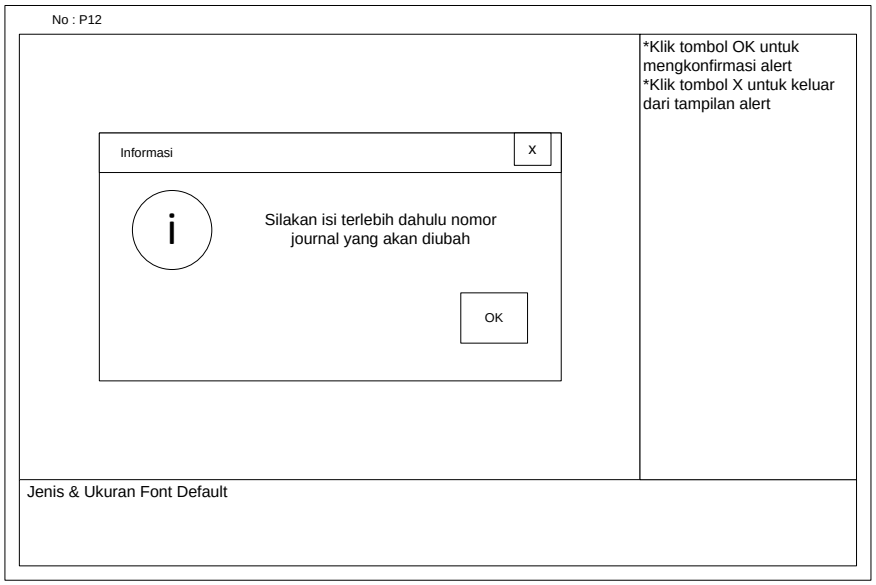
Gambar 3.41 Perancangan Pesan P10

11. Perancangan Pesan P11



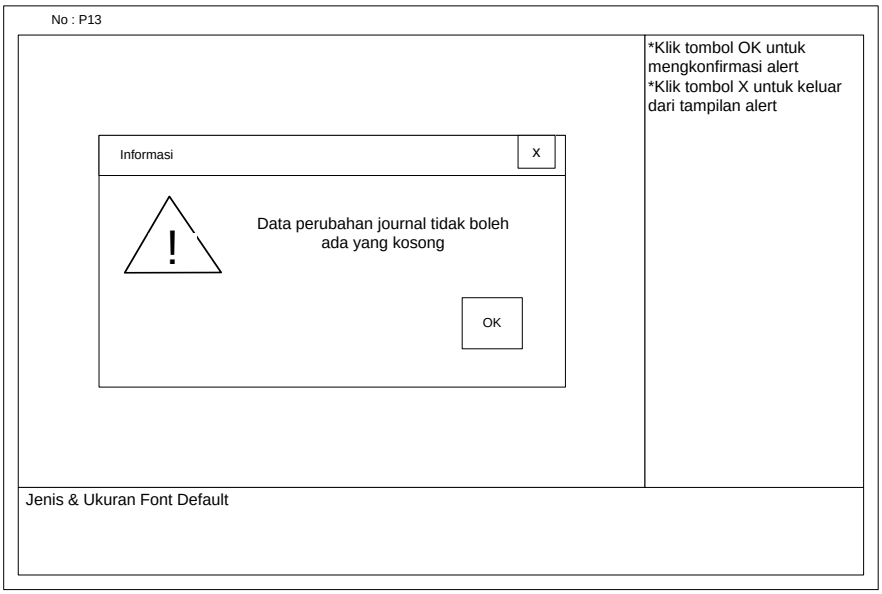
Gambar 3.42 Perancangan Pesan P11

12. Perancangan Pesan P12



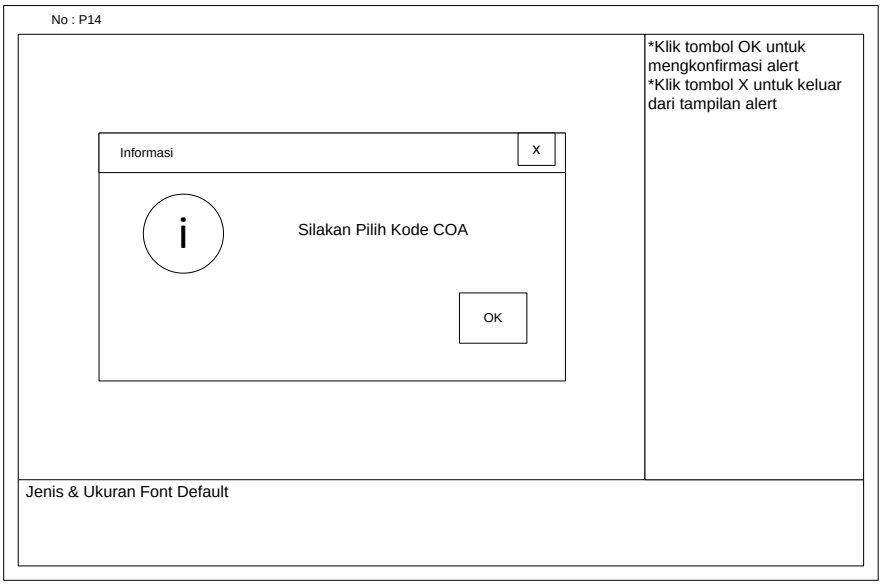
Gambar 3.43 Perancangan Pesan P12

13. Perancangan Pesan P13



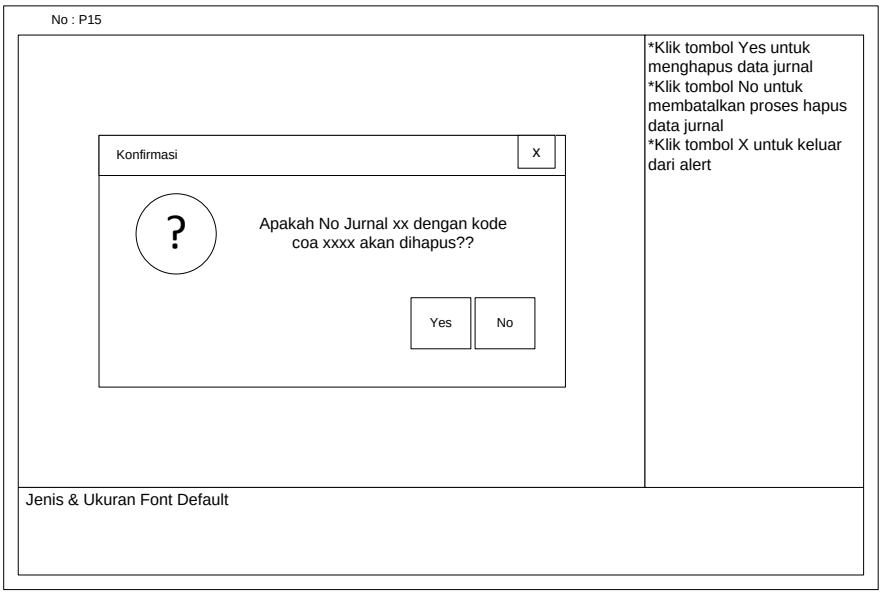
Gambar 3.44 Perancangan Pesan P13

14. Perancangan Pesan P14



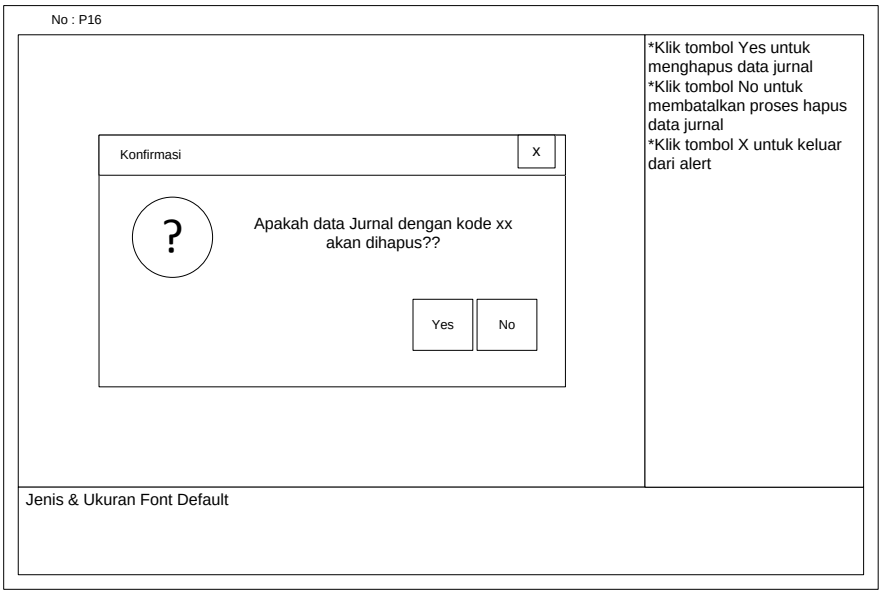
Gambar 3.45 Perancangan Pesan P14

15. Perancangan Pesan P15



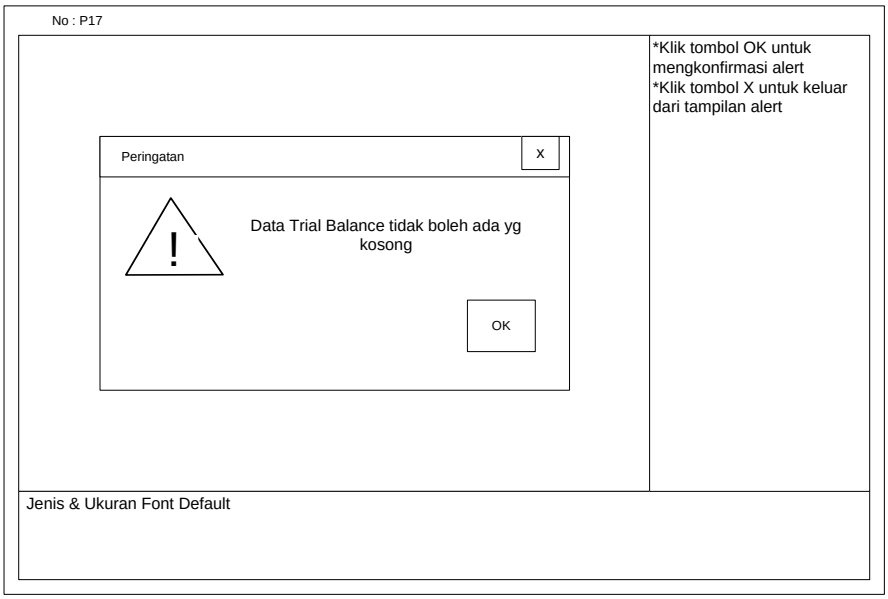
Gambar 3.46 Perancangan Pesan P15

16. Perancangan Pesan P16



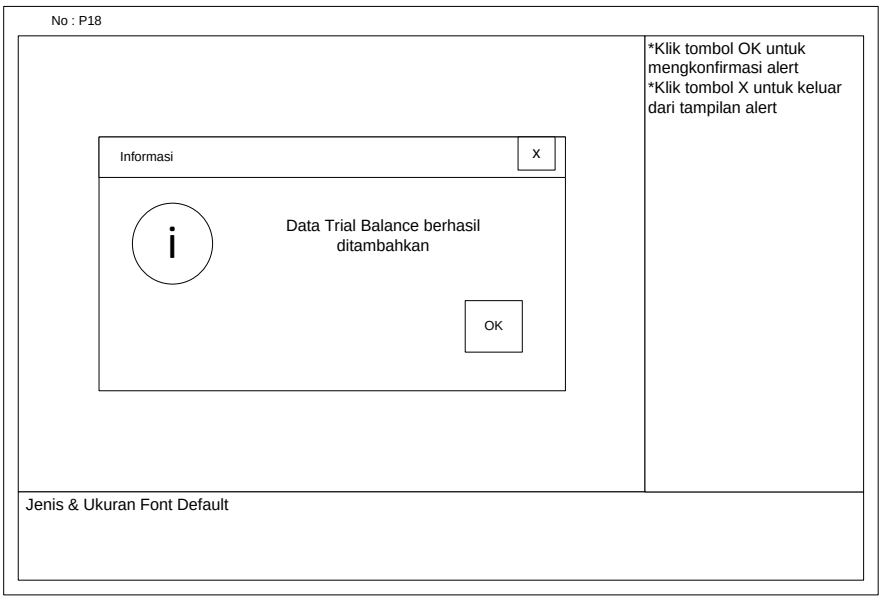
Gambar 3.47 Perancangan Pesan P16

17. Perancangan Pesan P17



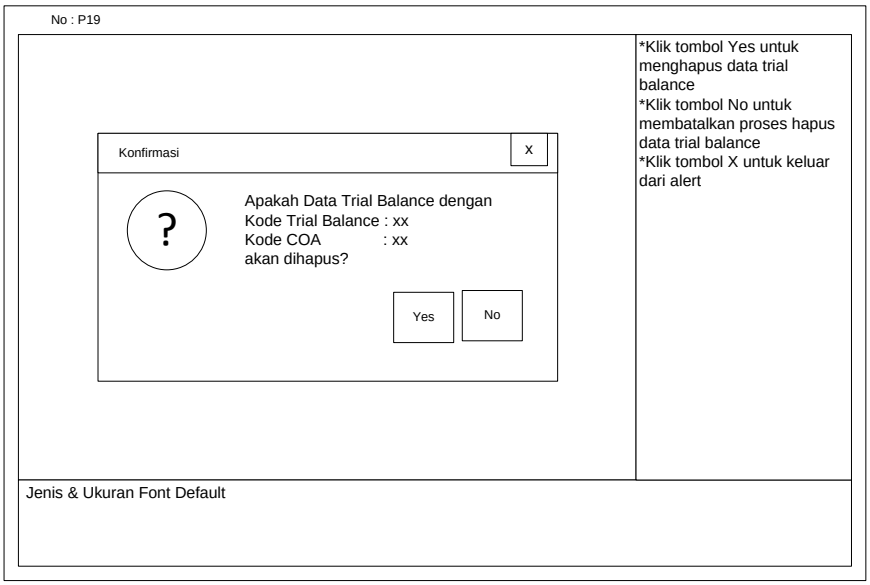
Gambar 3.48 Perancangan Pesan P17

18. Perancangan Pesan P18



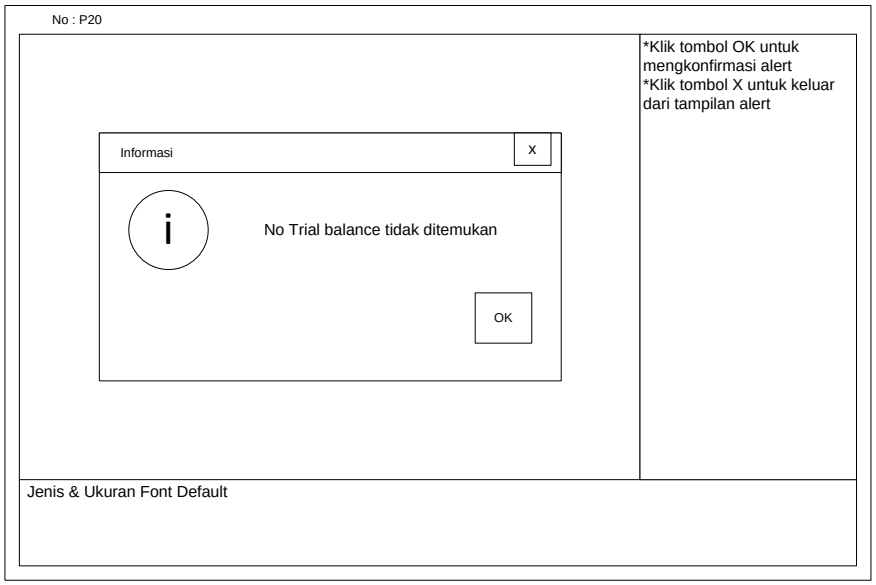
Gambar 3.49 Perancangan Pesan P18

19. Perancangan Pesan P19



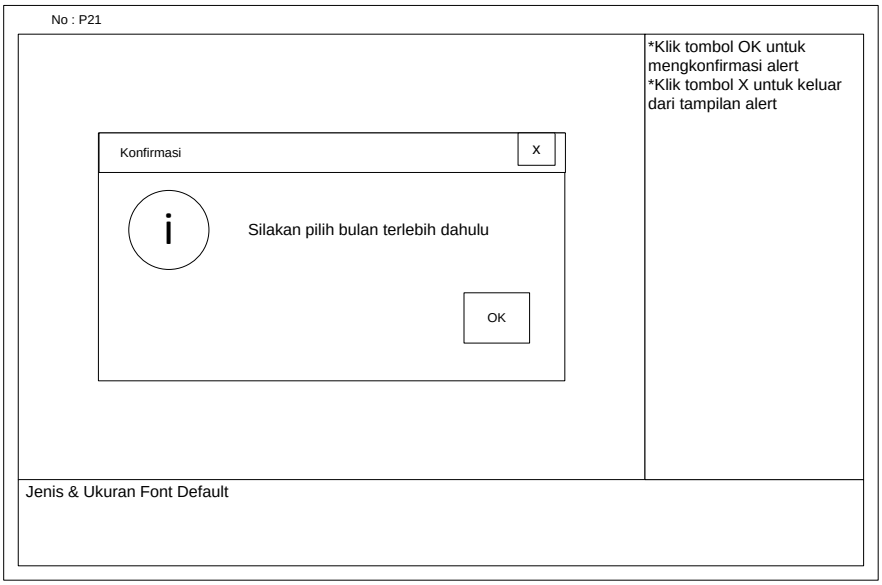
Gambar 3.50 Perancangan Pesan P19

20. Perancangan Pesan P20



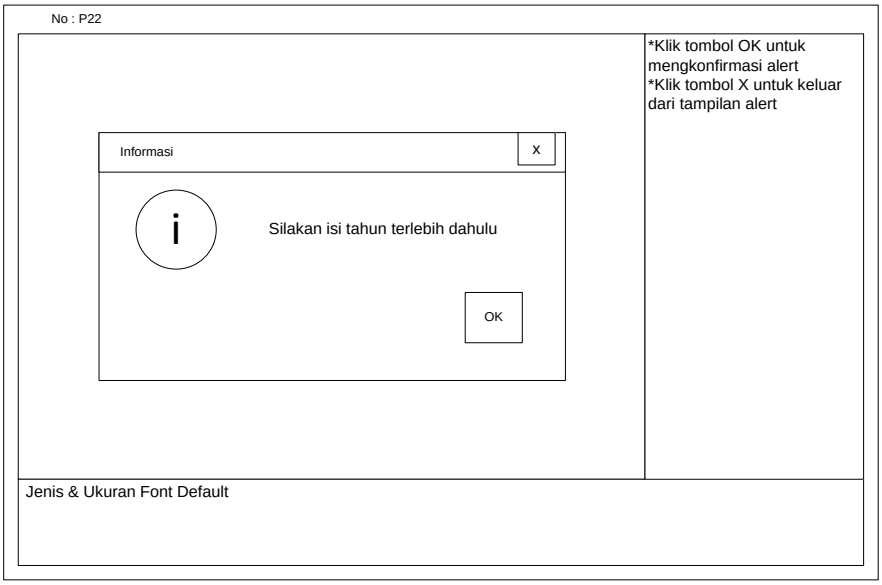
Gambar 3.51 Perancangan Pesan 20

21. Perancangan Pesan P21



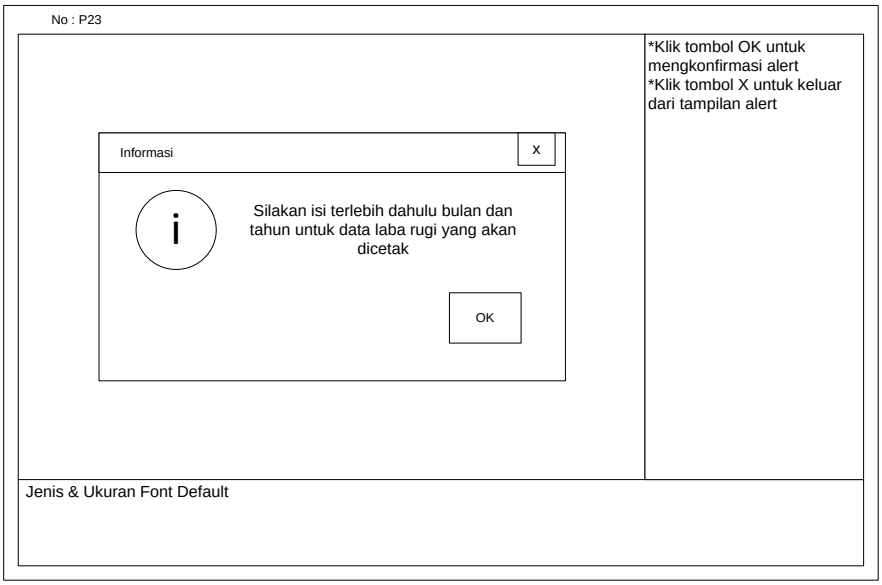
Gambar 3.52 Perancangan Pesan 21

22. Perancangan Pesan P22



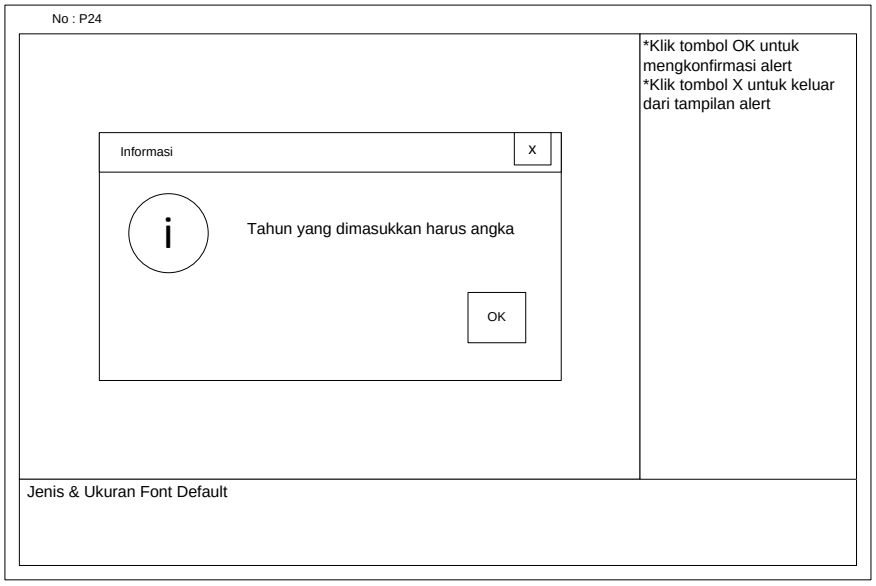
Gambar 3.53 Perancangan Pesan P22

23. Perancangan Pesan P23



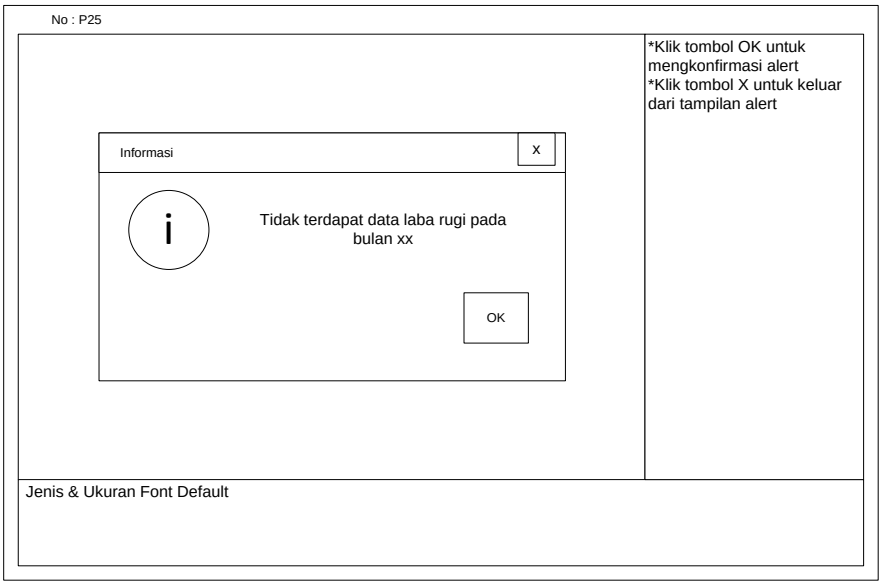
Gambar 3.54 Perancangan Pesan P23

24. Perancangan Pesan P24



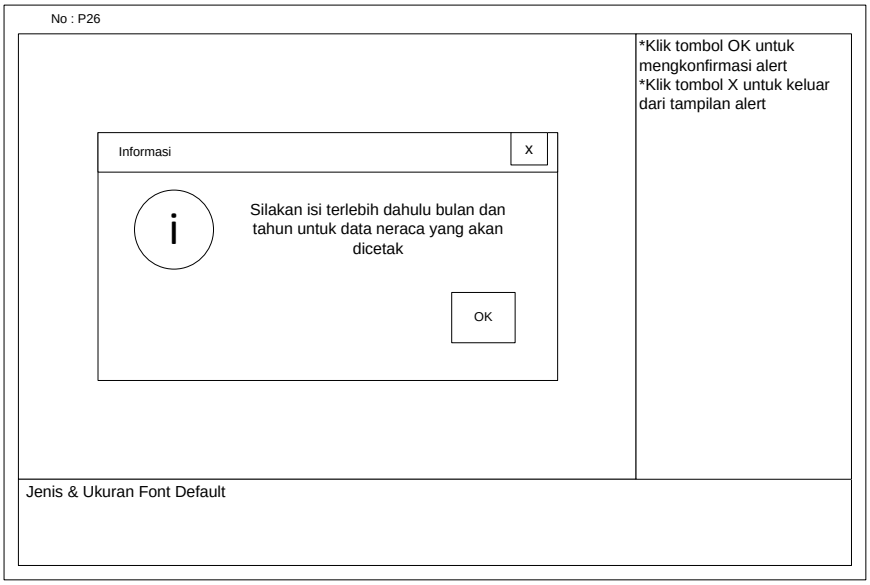
Gambar 3.55 Perancangan Pesan P24

25. Perancangan Pesan P25



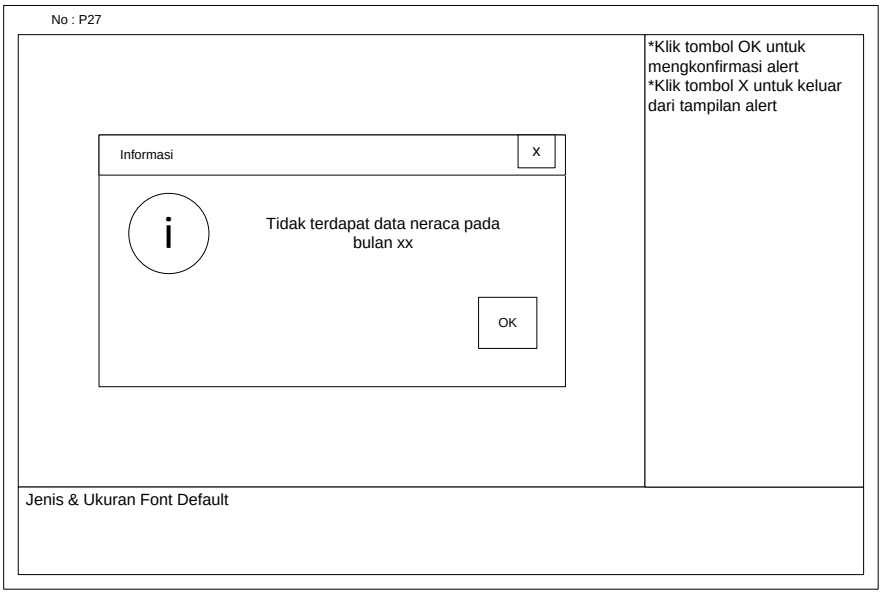
Gambar 3.56 Perancangan Pesan P25

26. Perancangan Pesan P26



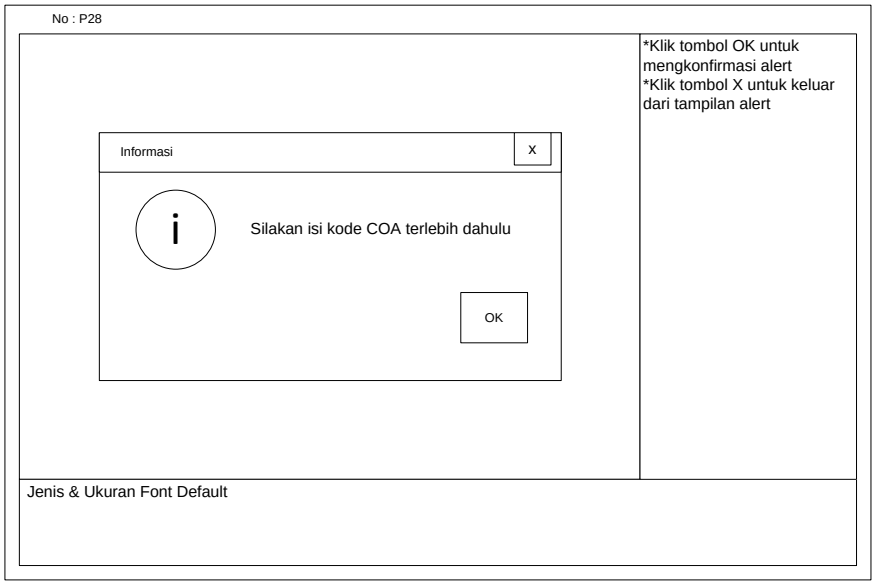
Gambar 3.57 Perancangan Pesan P26

27. Perancangan Pesan P27



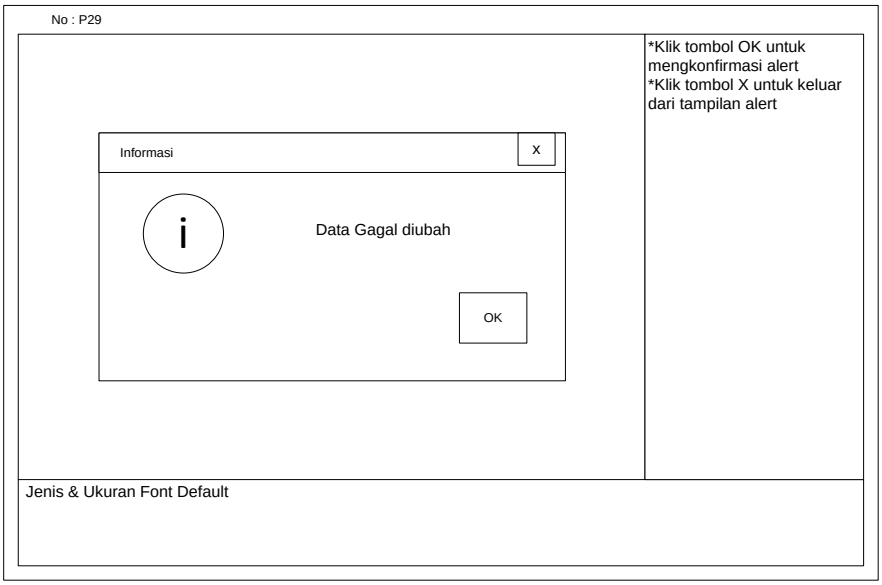
Gambar 3.58 Perancangan Pesan 27

28. Perancangan Pesan P28



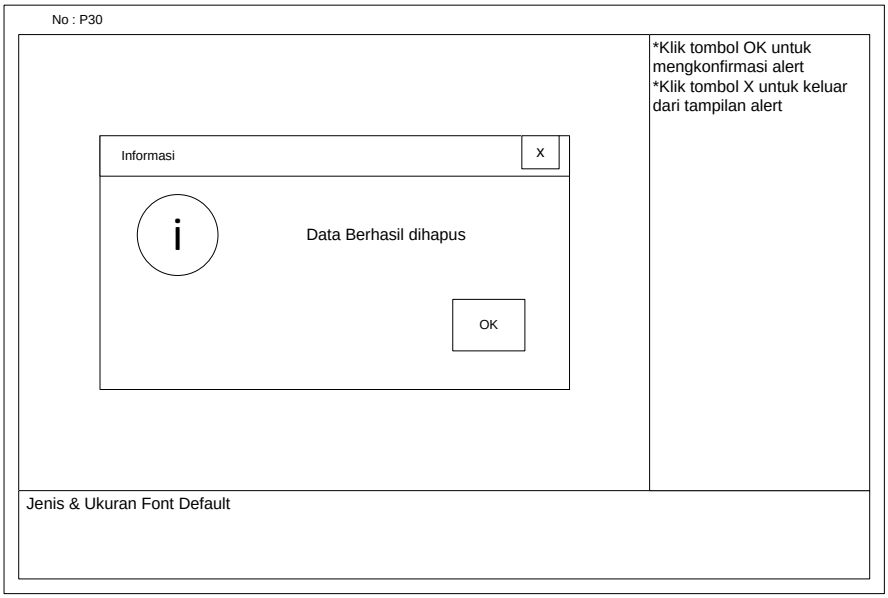
Gambar 3.59 Perancangan Pesan P28

29. Perancangan Pesan P29



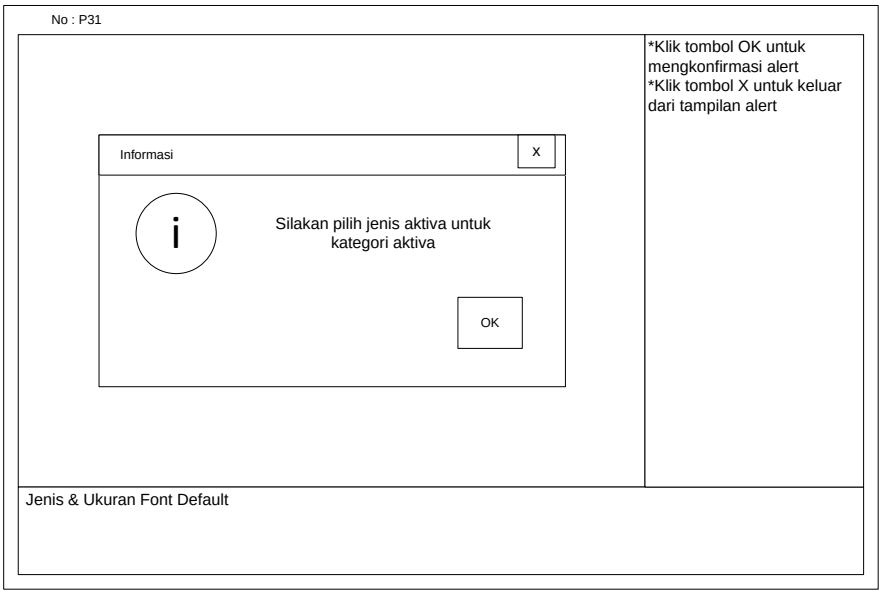
Gambar 3.60 Perancangan Pesan P29

30. Perancangan Pesan P30



Gambar 3.61 Perancangan Pesan P30

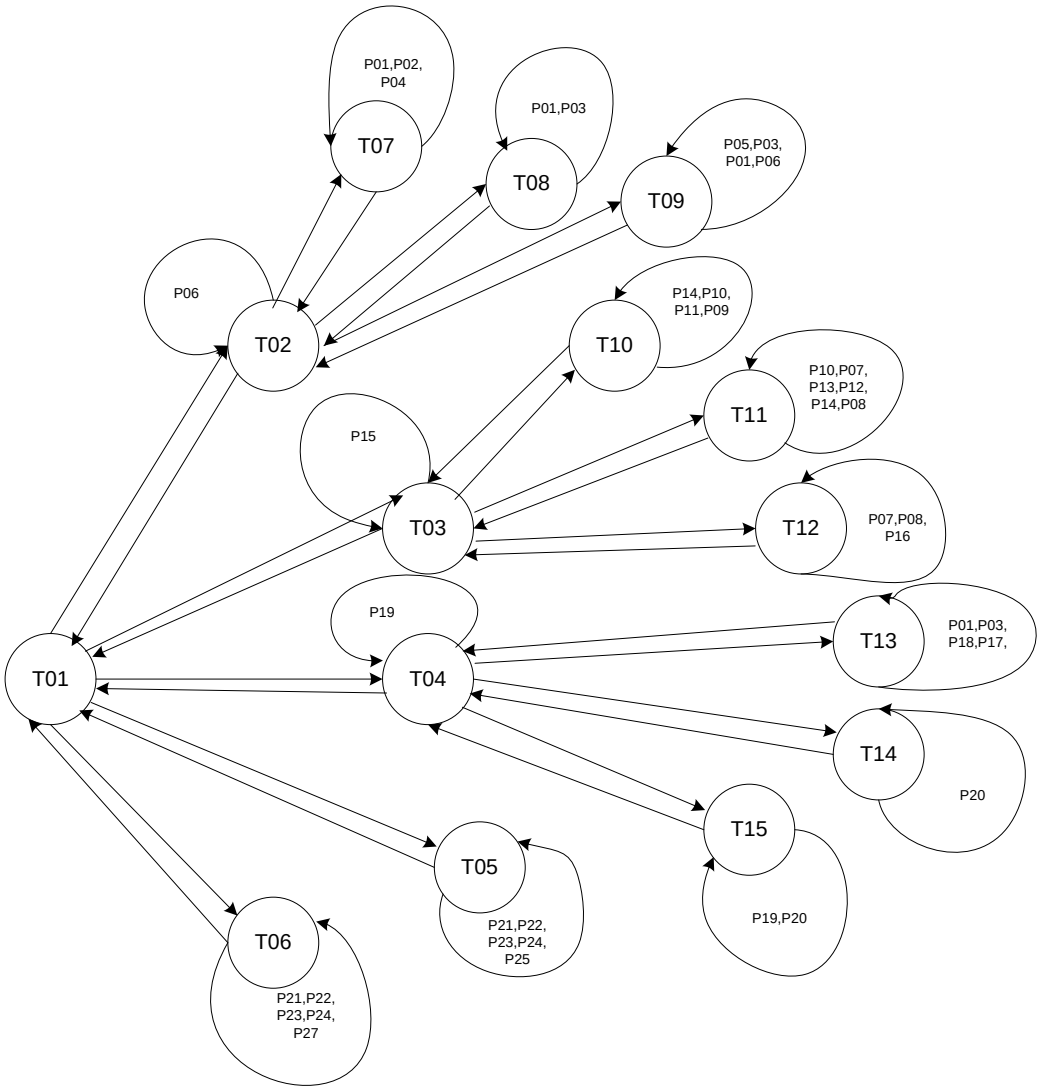
31. Perancangan Pesan P31



Gambar 3.62 Perancangan Pesan P31

3. 2. 4. Jaringan Semantik

Jaringan Semantik merupakan gambaran pengetahuan grafis yang menunjukkan hubungan antar berbagai objek. Jaringan semantik terdiri dari lingkaran-lingkaran yang menunjukkan objek dan informasi tentang objek-objek tertentu. Jaringan semantik untuk aplikasi *General Ledger* bulanan di CV.Global Agency and Entertainment dapat dilihat pada Gambar 3.74.



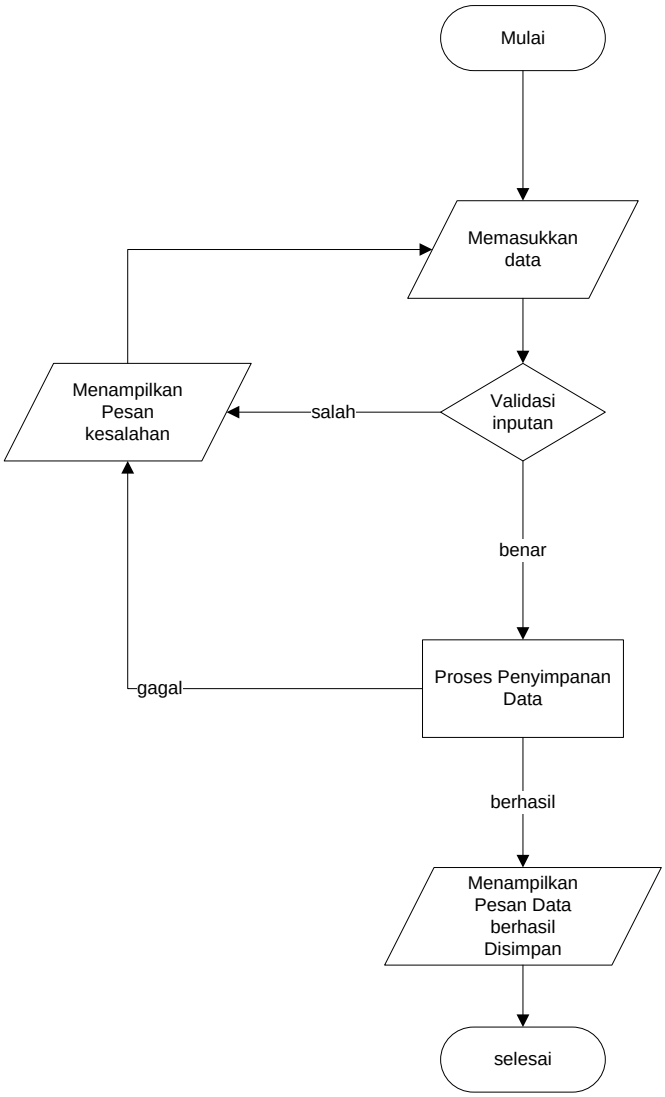
Gambar 3.63 Jaringan Semantik

3. 2. 5. Perancangan Prosedural

Perancangan prosedural merupakan representasi algoritma dari perangkat lunak yang dibangun. Perancangan prosedural pada aplikasi *General Ledger* bulanan di CV.Global Agency and Entertainment digambarkan dengan menggunakan *flowchart*. *Flowchart* perancangan prosedural aplikasi *General Ledger* bulanan terdiri dari *flowchart* penambahan data, *flowchart* perubahan data, *flowchart* pencarian data, dan *flowchart* penghapusan data. Adapun penjelasan beserta gambaran dari masing-masing *flowchart* yang ada adalah sebagai berikut :

1. Prosedural Penambahan Data

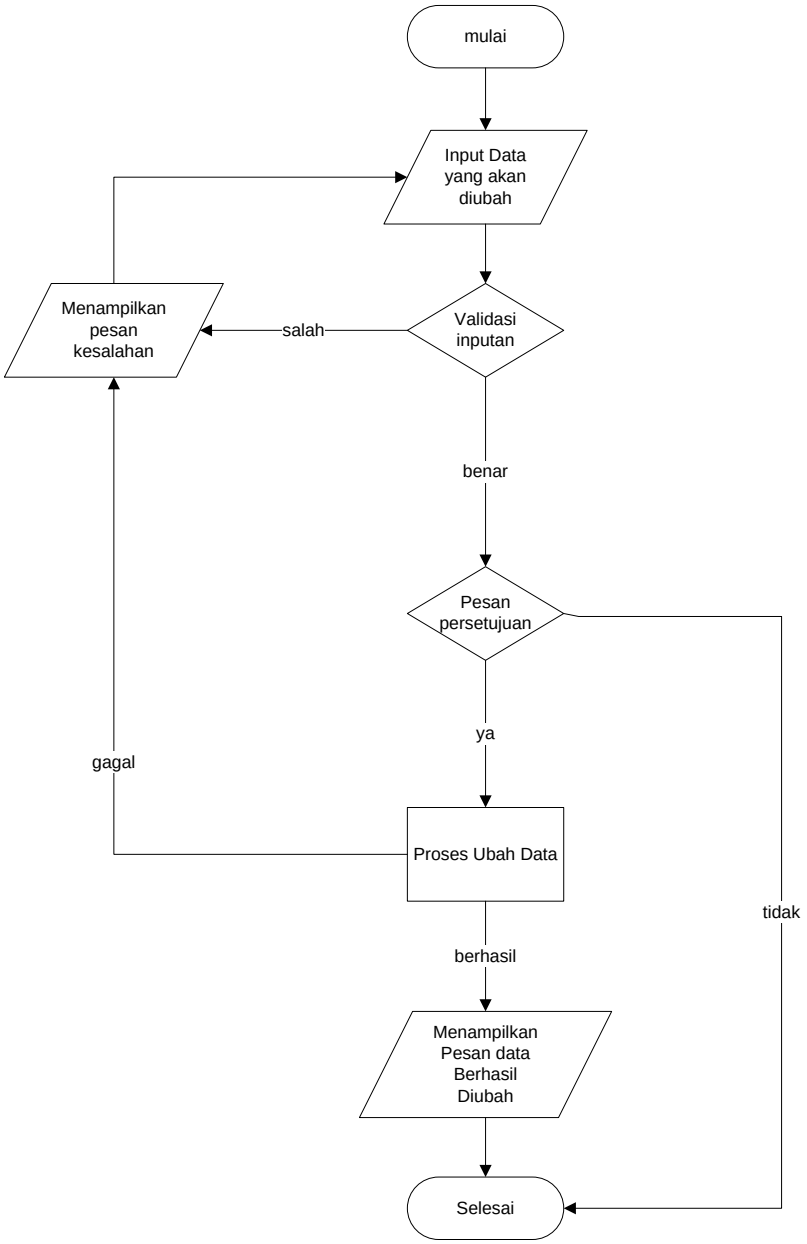
Prosedural penambahan data menggambarkan langkah-langkah yang dilakukan oleh aktor *accounting* untuk melakukan penambahan data pada aplikasi yang dibangun. Adapun gambaran prosedural dari proses penambahan data dapat dilihat pada gambar 3.75.



Gambar 3.64 *Flowchart* Penambahan Data

2. Prosedural Perubahan Data

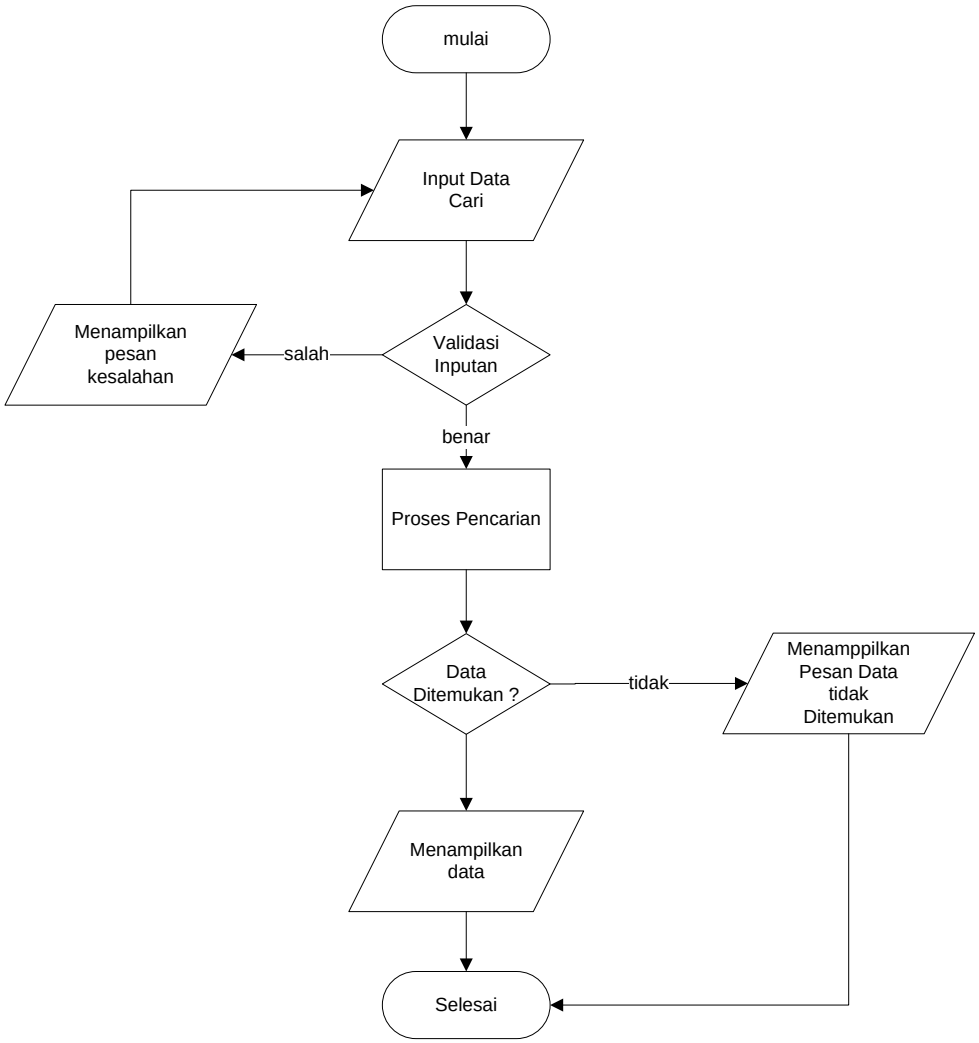
Prosedural perubahan data menggambarkan langkah-langkah yang dilakukan oleh aktor *accounting* untuk melakukan perubahan data pada aplikasi yang dibangun. Adapun gambaran prosedural dari proses perubahan data dapat dilihat pada gambar 3.76.



Gambar 3.65 *Flowchart* Perubahan Data

3. Prosedural Pencarian Data

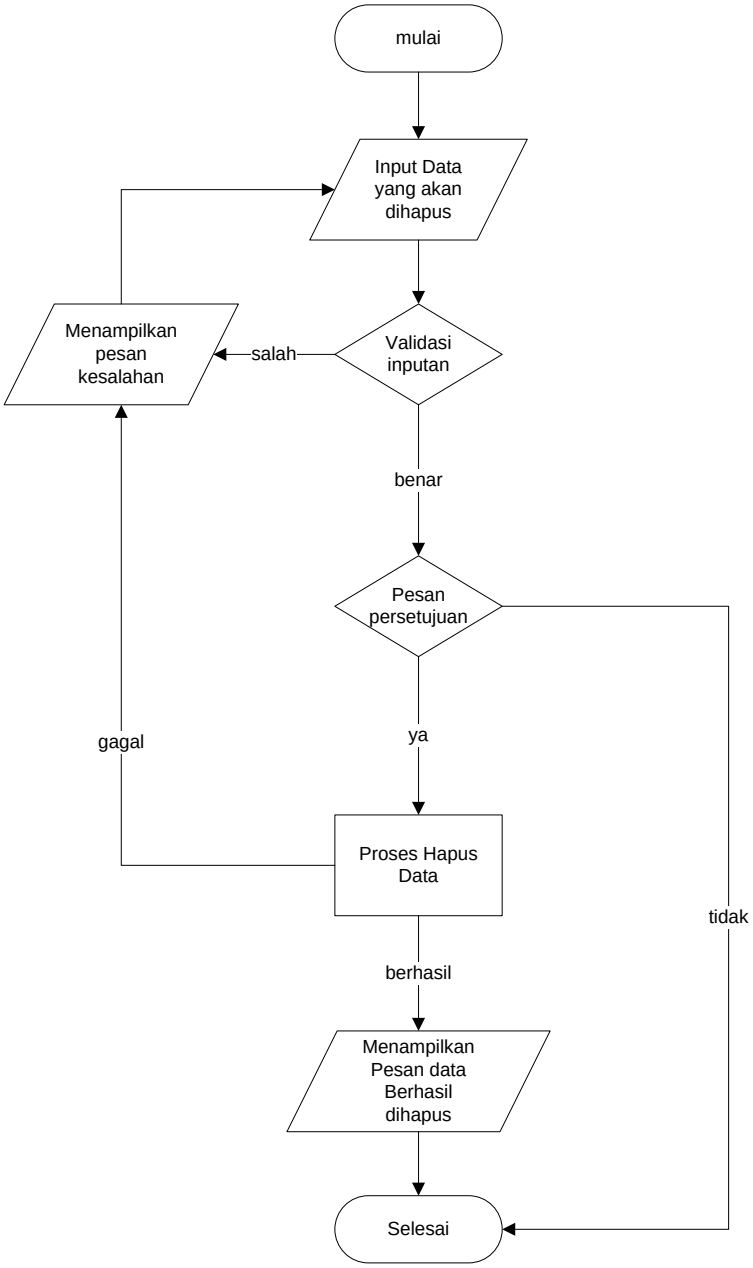
Prosedural pencarian data menggambarkan langkah-langkah yang dilakukan oleh aktor *accounting* untuk melakukan pencarian data pada aplikasi yang dibangun. Adapun gambaran prosedural dari proses pencarian data dapat dilihat pada gambar 3.77.



Gambar 3.66 *Flowchart* Pencarian Data

4. Prosedural Penghapusan Data

Prosedural pencarian data menggambarkan langkah-langkah yang dilakukan oleh aktor *accounting* untuk melakukan penghapusan data pada aplikasi yang dibangun. Adapun gambaran prosedural dari proses penghapusan data dapat dilihat pada gambar 3.78.



Gambar 3.67 *Flowchart* Penghapusan Data

3. 3. Implementasi

Setelah sistem dianalisis dan didesain secara rinci maka selanjutnya akan menuju tahap implementasi. Tujuan implementasi adalah untuk menjelaskan tentang manual modul kepada semua user yang akan menggunakan sistem. Sehingga user tersebut dapat merespon apa yang ditampilkan di sistem dan memberikan masukan kepada pembuat sistem untuk dilakukan perbaikan agar sistem lebih baik lagi.

3. 3. 1. Kebutuhan Perangkat Keras Pembangun

Spesifikasi perangkat keras yang digunakan untuk membangun aplikasi *General Ledger* Bulanan di CV. Global Agency and Entertainment :

1. Monitor LCD 14 inchi
2. Harddisk 320 GB
3. Memory 2 GB
4. Processor Core 2 Duo 2,2 GHz
5. Mouse
6. Keyboard

3. 3. 2. Kebutuhan Perangkat Lunak Pembangun

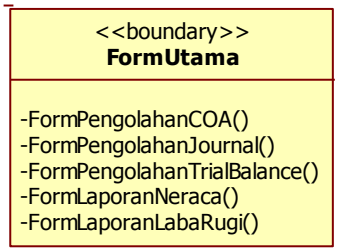
Beberapa perangkat lunak pendukung yang dapat mengoptimalkan program yang akan dibangun, diantaranya adalah sebagai berikut.

1. NetBeans IDE 6.8
2. JDK 1.6 update 17
3. MySQL Versi 5.1
4. SQLyog
5. iReport

3. 3. 3. Implementasi Kelas

3. 3. 3. 1. Kelas *Form* Utama

Kelas *form* utama merupakan kelas yang digunakan untuk menghubungkan semua kelas *form* yang ada di aplikasi *general ledger* bulanan di CV. Global Agency and Entertainment.



Gambar 3.68 Kelas *Form* Utama

A. Deskripsi Atribut

Tidak memiliki atribut

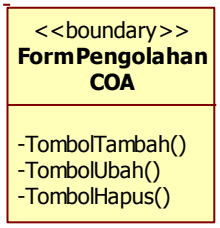
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. FormPengolahanCOA(), *method* untuk menghubungkan *form* pengolahan coa dengan *form* utama.
2. FormPengolahanJournal(), *method* untuk menghubungkan *form* pengolahan *journal* dengan *form* utama.
3. FormPengolahanTrialBalance(), *method* untuk menghubungkan *form* pengolahan *trial balance* dengan *form* utama.
4. FormPengolahanNeraca(), *method* untuk menghubungkan *form* pengolahan neraca dengan *form* utama.
5. FormPengolahanLabaRugi(), *method* untuk menghubungkan *form* pengolahan laba rugi dengan *form* utama.

3. 3. 3. 2. Kelas *Form* Pengolahan COA

Kelas *form* pengolahan coa merupakan kelas *interface* untuk pengolahan data coa.



Gambar 3.69 Kelas *Form* Pengolahan COA

A. Deskripsi Atribut

Tidak memilik atribut

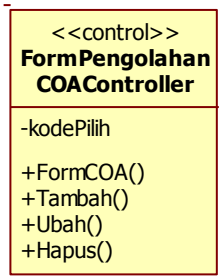
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. TombolTambah(), *interface* untuk tombol tambah
2. TombolUbah(), *interface* untuk tombol ubah
3. TombolHapus(), *interface* untuk tombol hapus

3. 3. 3. 3. Kelas *Form* Pengolahan COA *Controller*

Kelas *form* pengolahan coa *controller* merupakan kelas yang digunakan untuk melakukan proses manipulasi data.



Gambar 3.70 Kelas *Form* Pengolahan COA *Controller*

A. Deskripsi Atribut

Tabel 3.22 Deskripsi Atribut Kelas *Form* Pengolahan COA *Controller*

Nama	Tipe	Panjang	Range	Nilai Tetap
kodePilih	int	-	-	-

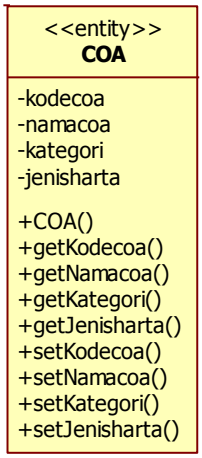
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. FormCOA(), konstruktor dari *form* coa yang bisa menampilkan data ke tabel
2. Tambah(), *method* operasi untuk memanggil *form* tambah coa
3. Ubah(), *method* operasi untuk memanggil *form* ubah coa
4. Hapus(), *method* operasi untuk memanggil *form* hapus coa

3. 3. 3. 4. Kelas COA

Kelas coa merupakan kelas yang merepresentasikan coa.



Gambar 3.71 Kelas COA

A. Deskripsi Atribut

Tabel 3.23 Deskripsi Atrinut Kelas COA

Nama	Tipe	Panjang	Range	Nilai Tetap
kodecoa	Int	10	-	Not Null
namacoa	Varchar	80	a-z, A-Z	Not Null
kategori	Varchar	19	a-z, A-Z	Not Null
jenisharta	Varchar	20	a-z, A-Z	Null

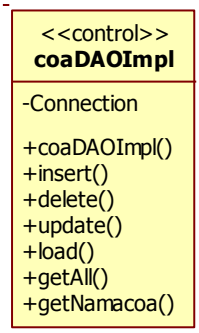
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. COA(), konstruktor coa
2. getKodeCOA(), *method* untuk menampilkan kode coa
3. getNamaCOA(), *method* untuk menampilkan nama coa
4. getKategori(), *method* untuk menampilkan kategori
5. getJenisHarta(), *method* untuk menampilkan jenis harta
6. setKodeCOA(), *method* untuk memasukan kode coa
7. setNamaCOA(), *method* untuk memasukan nama coa
8. setKategori(), *method* untuk memasukan kategori
9. setJenisHarta(), *method* untuk memasukan jenis harta

3. 3. 3. 5. Kelas COA DAO *Implement*

Kelas coa dao *implement* merupakan kelas yang digunakan untuk melakukan proses manipulasi data coa di *database*.



Gambar 3.72 Kelas COA DAO *Implement*

A. Deskripsi Atribut

Tabel 3.24 Deskripsi Atribut Kelas COA DAO *Implement*

Nama	Tipe	Panjang	Range	Nilai Tetap
connection	-	-	-	-

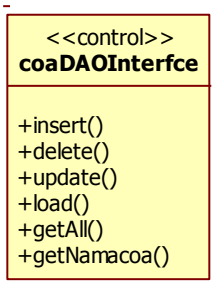
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. coaDAOImpl(), *method* untuk menghubungkan dengan *database*
2. insert(), *method* untuk memasukan data ke *database*
3. delete(), *method* untuk menghapus data dari *database*
4. update(), *method* untuk mengubah data di *database*
5. load(), *method* untuk mengambil data berdasarkan kode coa tertentu dari *database*
6. getAll (), *method* untuk mengambil semua data dari *database*
7. getNamaCOA(), *method* untuk menampilkan nama coa

3. 3. 3. 6. Kelas COA DAO *Interface*

Kelas coa dao *interface* merupakan kelas yang terdiri dari *method-method* yang akan diimplementasikan ke dalam kelas coa dao *implement*.



Gambar 3.73 Kelas COA DAO *Interface*

A. Deskripsi Atribut

Tidak memiliki atribut

B. Deskripsi Layanan

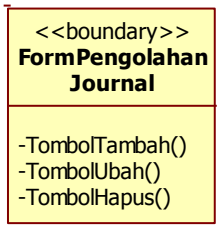
Operasi – operasi yang ada pada kelas ini adalah :

1. insert(), *method* untuk memasukan data ke *database*
2. delete(), *method* untuk menghapus data dari *database*
3. update(), *method* untuk mengubah data di *database*

4. load(), *method* untuk mengambil data berdasarkan kode coa tertentu dari *database*
5. getAll (), *method* untuk mengambil semua data dari *database*
6. getNamaCOA(), *method* untuk menampilkan nama coa dari *database*

3. 3. 3. 7. Kelas *Form* Pengolahan *Journal*

Kelas *form* pengolahan *journal* merupakan kelas *interface* untuk pengolahan data *journal*.



Gambar 3.74 Kelas *Form* Pengolahan *Journal*

A. Deskripsi Atribut

Tidak memiliki atribut

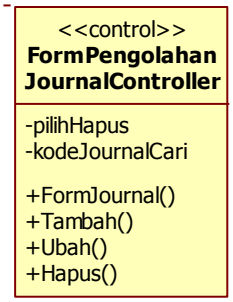
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. TombolTambah(), *interface* untuk tombol tambah
2. TombolUbah(), *interface* untuk tombol ubah
3. TombolHapus(), *interface* untuk tombol hapus

3. 3. 3. 8. Kelas *Form* Pengolahan *Journal Controller*

Kelas *form* pengolahan *journal controller* merupakan kelas yang digunakan untuk melakukan proses manipulasi data.



Gambar 3.75 Kelas *Form* Pengolahan *Journal Controller*

A. Deskripsi Atribut

Tabel 3.25 Deskripsi Atribut Kelas *Form* Pengolahan *Journal Controller*

Nama	Tipe	Panjang	Range	Nilai Tetap
pilihHapus	Boolean	-	-	-
kodeJournalCari	int	-	-	-

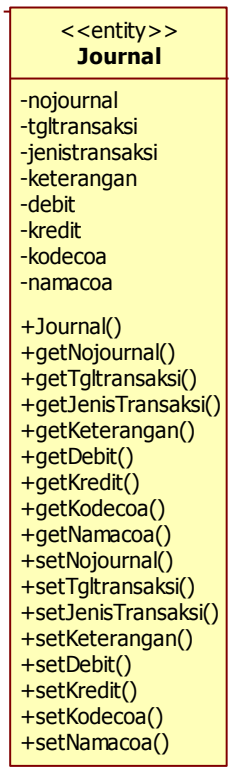
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. FormJournal(), konstruktor dari *form journal* yang bisa menampilkan data ke tabel
2. Tambah(), *method* operasi untuk memanggil *form* tambah *journal*
3. Ubah(), *method* operasi untuk memanggil *form* ubah *journal*
4. Hapus(), *method* operasi untuk memanggil *form* hapus *journal*

3. 3. 3. 9. Kelas *Journal*

Kelas *journal* merupakan kelas yang merepresentasikan *journal*.



Gambar 3.76 Kelas *Journal*

A. Deskripsi Atribut

Tabel 3.26 Deskripsi Atribut Kelas *Journal*

Nama	Tipe	Panjang	Range	Nilai Tetap
nojournal	Int	11	-	Not Null
tgltransaksi	Date	-	-	Not Null
jenistransaksi	Varchar	9	a-z, A-Z	Not Null
keterangan	Varchar	100	a-z, A-Z	Not Null
Debit	Double	-	-	Not Null
Kredit	Double	-	-	Not Null
Kodecoa	Int	10	-	Not Null
Namacoa	Varchar	80	a-z, A-Z	Not Null

B. Deskripsi Layanan

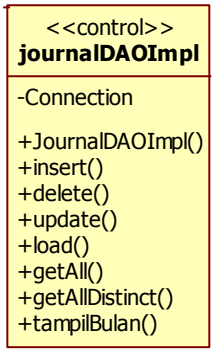
Operasi – operasi yang ada pada kelas ini adalah :

1. *Journal*(), konstruktor *journal*
2. getNoJournal(), *method* untuk menampilkan nomor *journal*
3. getTglTransaksi(), *method* untuk menampilkan tanggal transaksi
4. getJenisTransaksi(), *method* untuk menampilkan jenis transaksi

5. getKeterangan(), *method* untuk menampilkan keterangan
6. getDebit(), *method* untuk menampilkan debit
7. getKreditCOA(), *method* untuk menampilkan kredit
8. getKodeCoa (), *method* untuk menampilkan kode coa
9. getNamaCoa(),*method* untuk menampilkan nama coa
10. setNoJournal(), *method* untuk memasukan nomor *journal*
11. setTglTransaksi(), *method* untuk memasukan tanggal transaksi
12. setJenisTransaksi(), *method* untuk memasukan jenis transaksi
13. setKeterangan(), *method* untuk memasukan keterangan
14. setDebit(), *method* untuk memasukan debit
15. setKreditCOA(), *method* untuk memasukan kredit
16. setKodeCoa (), *method* untuk memasukan kode coa
17. setNamaCoa(),*method* untuk memasukan nama coa

3. 3. 3. 10. Kelas *Journal* DAO *Implement*

Kelas *journal* dao *implement* merupakan kelas yang digunakan untuk melakukan proses manipulasi data *journal* di *database*.



Gambar 3.77 Kelas *Journal* DAO *Implement*

A. Deskripsi Atribut

Tabel 3.27 Deskripsi Atribut *Journal* DAO *Implement*

Nama	Tipe	Panjang	Range	Nilai Tetap
connection	-	-	-	-

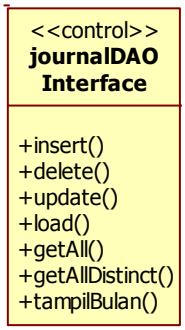
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. JournalDAOImpl(), *method* untuk menghubungkan dengan *database*
2. insert(), *method* untuk memasukan data ke *database*
3. delete(), *method* untuk menghapus data dari *database*
4. update(), *method* untuk mengubah data di *database*
5. load(), *method* untuk mengambil data berdasarkan kode coa, bulan dan tahun tertentu dari *database*
6. getAll (), *method* untuk mengambil semua data dari *database*
7. getAllDistinct (), *method* untuk menampilkan satu data yang telah dipilih
8. tampilBulan(), *method* untuk menampilkan bulan dari *database*

3. 3. 3. 11. Kelas *Journal* DAO *Interface*

Kelas *journal* dao *interface* merupakan kelas yang terdiri dari *method-method* yang akan diimplementasikan ke dalam kelas *journal* dao *implement*.



Gambar 3.78 Kelas *Journal* DAO *Interface*

A. Deskripsi Atribut

Tidak memiliki atribut.

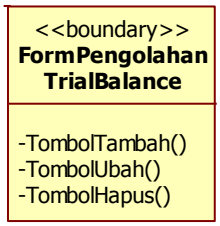
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. insert(), *method* untuk memasukan data ke *database*
2. delete(), *method* untuk menghapus data dari *database*
3. update(), *method* untuk mengubah data di *database*
4. load(), *method* untuk mengambil data berdasarkan kode coa tertentu dari *database*
5. getAll (), *method* untuk mengambil semua data dari *database*
9. getAllDistinct (), *method* untuk menampilkan satu data yang telah dipilih
6. tampilBulan(), *method* untuk menampilkan bulan dari *database*

3. 3. 3. 12. Kelas *Form* Pengolahan *Trial Balance*

Kelas *form* pengolahan *trial balance* merupakan kelas *interface* untuk pengolahan data *trial balance*.



Gambar 3.79 Kelas *Form* Pengolahan *Trial Balance*

A. Deskripsi Atribut

Tidak memiliki atribut.

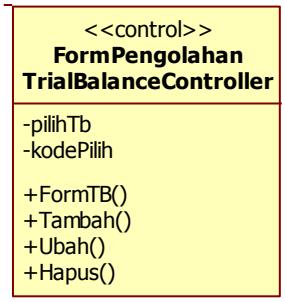
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. TombolTambah(), *interface* untuk tombol tambah
2. TombolUbah(), *interface* untuk tombol ubah
3. TombolHapus(), *interface* untuk tombol hapus

3. 3. 3. 13. Kelas *Form* Pengolahan *Trial Balance Controller*

Kelas *form* pengolahan *trial balance controller* merupakan kelas yang digunakan untuk melakukan proses manipulasi data.



Gambar 3.80 Kelas *Form* Pengolahan *Trial Balance Controller*

A. Deskripsi Atribut

Tabel 3.28 Deskripsi Atribut Kelas *Form* Pengolahan *Trial Balance Controller*

Nama	Tipe	Panjang	Range	Nilai Tetap
pilihTb	Boolean	-	-	-
kodePilih	int	-	-	-

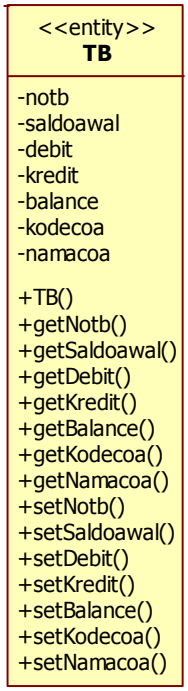
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. FormTB(), konstruktor dari *form trial balance* yang bisa menampilkan data ke tabel
2. Tambah(), fungsi operasi untuk memanggil *form* tambah *trial balance*
3. Ubah(), fungsi operasi untuk memanggil *form* ubah *trial balance*
4. Hapus(), fungsi operasi untuk memanggil *form* hapus *trial balance*

3. 3. 3. 14. Kelas *Trial Balance*

Kelas *trial balance* merupakan kelas yang merepresentasikan *trial balance*.



Gambar 3.81 Kelas *Trial Balance*

A. Deskripsi Atribut

Tabel 3.29 Deskripsi Atribut Kelas *Trial Balance*

Nama	Tipe	Panjang	Range	Nilai Tetap
notb	Int	11	-	Not Null
saldoawal	Date	-	-	Not Null
debit	Double	-	-	Not Null
kredit	Double	-	-	Not Null
kodecoa	Int	10	-	Not Null
namacoa	Varchar	80	a-z, A-Z	Not Null

B. Deskripsi Layanan

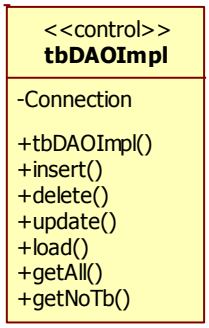
Operasi – operasi yang ada pada kelas ini adalah :

1. TB(), konstruktor *trial balance*
2. getNoTb(), *method* untuk menampilkan nomor *trial balance*
3. getSaldoawal (), *method* untuk menampilkan saldo awal
4. getDebit(), *method* untuk menampilkan debit
5. getKreditCOA(), *method* untuk menampilkan kredit
6. getKodeCoa (), *method* untuk menampilkan kode coa

7. getNamaCoa(),*method* untuk menampilkan nama coa
8. setNoTb(), *method* untuk memasukan nomor *trial balance*
9. setSaldoawal(), *method* untuk memasukan saldo awal
10. setDebit(), *method* untuk memasukan debit
11. setKreditCOA(), *method* untuk memasukan kredit
12. setKodeCoa (), *method* untuk memasukan kode coa
13. setNamaCoa(),*method* untuk memasukan nama coa

3. 3. 3. 15. Kelas *Trial Balance* DAO *Implement*

Kelas *trial balance* dao *implement* merupakan kelas yang digunakan untuk melakukan proses manipulasi data *trial balance* di *database*.



Gambar 3.82 Kelas *Trial Balance* DAO *Implement*

A. Deskripsi Atribut

Tabel 3.30 Deskripsi Atribut Kelas *Trial Balance* DAO *Implement*

Nama	Tipe	Panjang	Range	Nilai Tetap
connection	-	-	-	-

B. Deskripsi Layanan

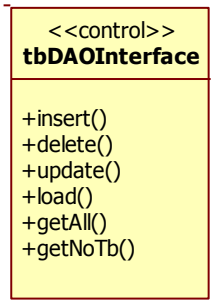
Operasi – operasi yang ada pada kelas ini adalah :

1. tbDAOImpl(), *method* untuk menghubungkan dengan *database*
2. insert(), *method* untuk memasukan data ke *database*
3. delete(), *method* untuk menghapus data dari *database*
4. update(), *method* untuk mengubah data di *database*

5. load(), *method* untuk mengambil data berdasarkan kode coa tertentu dari *database*
6. getAll (), *method* untuk mengambil semua data dari *database*
7. getNoTb(), *method* untuk menampilkan data *trial balance* berdasarkan urutan nomor *trial balance*

3. 3. 3. 16. Kelas *Trial Balance* DAO *Interface*

Kelas *trial balance* dao *interface* merupakan kelas yang terdiri dari *method-method* yang akan diimplementasikan ke dalam kelas *trial balance* dao *implement*.



Gambar 3.83 Kelas *Trial Balance* DAO *Interface*

A. Deskripsi Atribut

Tidak memiliki atribut.

B. Deskripsi Layanan

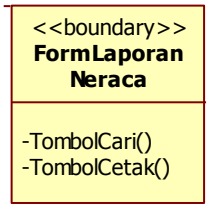
Operasi – operasi yang ada pada kelas ini adalah :

1. insert(), *method* untuk memasukan data ke *database*
2. delete(), *method* untuk menghapus data dari *database*
3. update(), *method* untuk mengubah data di *database*
4. load(), *method* untuk mengambil data berdasarkan nomor *trial balance* tertentu dari *database*
5. getAll (), *method* untuk mengambil semua data dari *database*

6. getNoTb(), *method* untuk menampilkan data *trial balance* berdasarkan urutan nomor *trial balance*

3. 3. 3. 17. Kelas *Form* Laporan Neraca

Kelas *form* laporan neraca merupakan kelas *interface* untuk pengolahan laporan neraca.



Gambar 3.84 Kelas *Form* Laporan Neraca

A. Deskripsi Atribut

Tidak memiliki atribut.

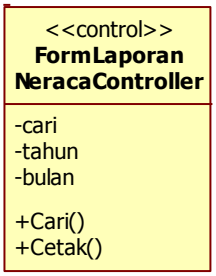
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. TombolCari(), *interface* untuk tombol cari
2. TombolCetak(), *interface* untuk tombol cetak

3. 3. 3. 18. Kelas *Form* Laporan Neraca *Controller*

Kelas *form* laporan neraca *controller* merupakan kelas yang digunakan untuk memproses laporan neraca yang akan dicetak.



Gambar 3.85 Kelas *Form* Laporan Neraca *Controller*

A. Deskripsi Atribut

Tabel 3.31 Deskripsi Atribut Kelas *Form* Laporan Neraca *Controller*

Nama	Tipe	Panjang	Range	Nilai Tetap
cari	Boolean	-	-	-
tahun	Int	-	-	-
bulan	Int	-	-	-

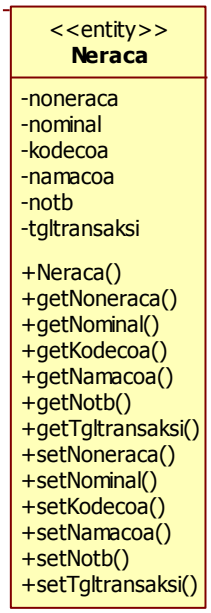
B. Deksripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. Cari(), *method* untuk mencari data neraca yang akan dicetak
2. Cetak(), *method* untuk mencetak data neraca yang telah dicari

3. 3. 3. 19. Kelas Neraca

Kelas neraca merupakan kelas yang merepresentasikan neraca.



Gambar 3.86 Kelas Neraca

A. Deskripsi Atribut

Tabel 3.32 Deskripsi Atribut Kelas Neraca

Nama	Tipe	Panjang	Range	Nilai Tetap
noneraca	Int	11	-	Not Null
nominal	Double	-	-	Not Null
kodecoa	Int	10	-	Not Null
namacoa	Varchar	80	a-z, A-Z	Not Null
Notb	Int	11	-	Not Null
tgltransaksi	Date	-	-	Not Null

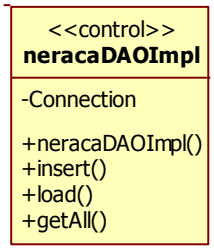
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. Neraca(), konstruktor Neraca
2. getNoneraca(), *method* untuk menampilkan nomor neraca
3. getNominal(), *method* untuk menampilkan nominal
4. getKodeCoa (), *method* untuk menampilkan kode coa
5. getNamaCoa(),*method* untuk menampilkan nama coa
6. getNoTb(), *method* untuk menampilkan nomor *trial balance*
7. getTgltransaksi(), *method* untuk menampilkan tanggal transaksi
8. setNoneraca(), *method* untuk memasukan nomor neraca
9. setNominal(), *method* untuk memasukan nominal
10. setKodeCoa (), *method* untuk memasukan kode coa
11. setNamaCoa(),*method* untuk memasukan nama coa
12. setNoTb(), *method* untuk memasukan nomor *trial balance*
13. setTgltransaksi(), *method* untuk memasukan tanggal transaksi

3. 3. 3. 20. Kelas Neraca DAO *Implement*

Kelas neraca dao *implement* merupakan kelas yang digunakan untuk melakukan proses manipulasi data neraca di *database*.



Gambar 3.87 Kelas Neraca DAO *Implement*

A. Deskripsi Atribut

Tabel 3.33 Deskripsi Atribut Kelas Neraca DAO *Implement*

Nama	Tipe	Panjang	Range	Nilai Tetap
connection	-	-	-	-

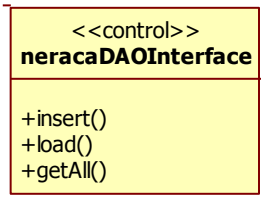
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. neracaDAOImpl(), *method* untuk menghubungkan dengan *database*
2. insert(), *method* untuk memasukan data ke *database*
3. load(), *method* untuk mengambil data berdasarkan nomor neraca tertentu dari *database*
4. getAll(),*method* untuk menampilkan semua data dari *database*

3. 3. 3. 21. Kelas Neraca DAO *Interface*

Kelas neraca dao *interface* merupakan kelas yang terdiri dari *method-method* yang akan diimplementasikan ke dalam kelas neraca dao *implement*.



Gambar 3.88 Kelas Neraca DAO *Interface*

A. Deskripsi Atribut

Tidak memiliki atribut.

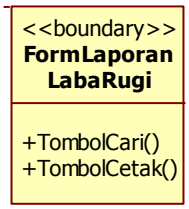
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. insert(), *method* untuk memasukan data ke *database*
2. load(), *method* untuk mengambil data berdasarkan nomor neraca tertentu dari *database*
3. getAll(), *method* untuk memanggil semua data di *database*

3. 3. 3. 22. Kelas *Form* Laporan Laba Rugi

Kelas *form* laporan laba rugi merupakan kelas *interface* untuk pengolahan laporan laba rugi.



Gambar 3.89 Kelas *Form* Laporan Laba Rugi

A. Deskripsi Atribut

Tidak memiliki atribut.

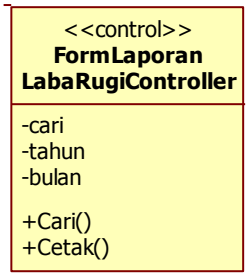
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. TombolCari(), *interface* untuk tombol cari
2. TombolCetak(), *interface* untuk tombol cetak

3. 3. 3. 23. Kelas *Form* Laporan Laba Rugi *Controller*

Kelas *form* laporan laba rugi *controller* merupakan kelas yang digunakan untuk memproses laporan laba rugi yang akan dicetak.



Gambar 3.90 Kelas *Form* Laporan Laba Rugi *Controller*

A. Deskripsi Atribut

Tabel 3.34 Deskripsi Atribut Kelas *Form* Laporan Laba Rugi *Controller*

Nama	Tipe	Panjang	Range	Nilai Tetap
cari	Boolean	-	-	-
tahun	Int	-	-	-
bulan	Int	-	-	-

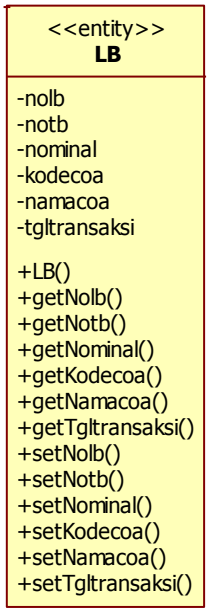
B. Deksripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. Cari(), *method* untuk mencari data laba rugi yang akan dicetak
2. Cetak(), *method* untuk mencetak data laba rugi yang telah dicari

3. 3. 3. 24. Kelas Laba Rugi

Kelas laba rugi merupakan kelas yang merepresentasikan laba rugi.



Gambar 3.91 Kelas Laba Rugi

A. Deskripsi Atribut

Tabel 3.35 Deskripsi Atribut Kelas Laba Rugi

Nama	Tipe	Panjang	Range	Nilai Tetap
nolb	Int	11	-	Not Null
notb	Int	11	-	Not Null
nominal	Double	-	-	Not Null
kodecoa	Int	10	-	Not Null
namacoa	Varchar	80	a-z, A-Z	Not Null
tgltransaksi	Date	-	-	Not Null

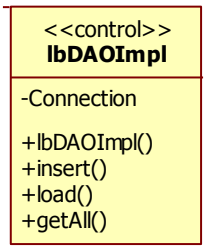
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. LB(), konstruktor laba rugi
2. getNolb(), *method* untuk menampilkan nomor laba rugi
3. getNotb(), *method* untuk menampilkan nomor *trial balance*
4. getNominal(), *method* untuk menampilkan nominal
5. getKodeCoa (), *method* untuk menampilkan kode coa
6. getNamaCoa(),*method* untuk menampilkan nama coa
7. getTgltransaksi(), *method* untuk menampilkan tanggal transaksi
8. setNolb(), *method* untuk memasukan nomor laba rugi
9. setNotb(), *method* untuk memasukan nomor *trial balance*
10. setNominal(), *method* untuk memasukan nominal
11. setKodeCoa (), *method* untuk memasukan kode coa
12. setNamaCoa(),*method* untuk memasukan nama coa
13. setTgltransaksi(), *method* untuk memasukan tanggal transaksi

3. 3. 3. 25. Kelas Laba Rugi DAO *Implement*

Kelas laba rugi dao *implement* merupakan kelas yang digunakan untuk melakukan proses manipulasi data laba rugi di *database*.



Gambar 3.92 Kelas Laba Rugi DAO *Implement*

A. Deskripsi Atribut

Tabel 3.36 Deskripsi Atribut Kelas Laba Rugi DAO *Implement*

Nama	Tipe	Panjang	Range	Nilai Tetap
connection	-	-	-	-

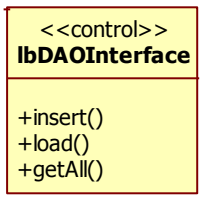
B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. lbDAOImpl(), *method* untuk menghubungkan dengan *database*
2. insert(), *method* untuk memasukan data ke *database*
3. load(), *method* untuk mengambil data berdasarkan nomor laba rugi tertentu dari *database*
4. getAll(),*method* untuk menampilkan semua data dari *database*

3. 3. 3. 26. Kelas Laba Rugi DAO *Interface*

Kelas laba rugi dao *interface* merupakan kelas yang terdiri dari *method-method* yang akan diimplementasikan ke dalam kelas laba rugi dao *implement*.



Gambar 3.93 Kelas Laba Rugi DAO *Interface*

A. Deskripsi Atribut

Tidak memiliki atribut.

B. Deskripsi Layanan

Operasi – operasi yang ada pada kelas ini adalah :

1. insert(), *method* untuk memasukan data ke *database*
2. load(), *method* untuk mengambil data berdasarkan nomor laba rugi tertentu dari *database*
3. getAll(), *method* untuk memanggil semua data di *database*

3. 3. 4. Implementasi Basis Data

Database Management Sistem (DBMS) yang digunakan untuk mengimplementasikan data aplikasi *General Ledger* Bulanan di CV. Global Agency and Entertainment ini adalah MySQL Versi 5.1. Implementasi *database* dalam aplikasi yang telah dirancang adalah sebagai berikut :

1. Tabel COA

Berikut *syntax* hasil *generate* tabel coa ke dalam *database* MySQL :

Tabel 3.37 Tabel COA

Tabel	*Syntax* MySQL

COA
PK: **kodecoa**
namacoa
kategori
jenis harta
jenishutang
	`CREATE TABLE `coa` (`
	`  `kodecoa` int(10) unsigned NOT NULL,`
	`  `namacoa` varchar(80) NOT NULL,`
	`  `kategori` varchar(19) NOT NULL,`
	`  `jenisharta` varchar(20) DEFAULT NULL,`
	`  PRIMARY KEY (`kodecoa`)`
	`) ENGINE=InnoDB`

2. Tabel *Journal*

Berikut *syntax* hasil *generate* tabel *Journal* ke dalam *database* MySQL :

Tabel 3.38 Tabel *Journal*

Tabel	*Syntax* MySQL

Jurnal
PK: **nojournal**
FK1: tgltransaksi, kodecoa, namacoa, jenistransaksi, keterangan, debit, kredit
	`CREATE TABLE `Journal` (`
	`  `noJournal` int(11) NOT NULL AUTO_INCREMENT,`
	`  `tgltransaksi` date NOT NULL,`
	`  `kodecoa` int(10) unsigned NOT NULL,`
	`  `namacoa` varchar(80) NOT NULL,`
	`  `jenistransaksi` varchar(9) NOT NULL,`
	`  `keterangan` varchar(100) NOT NULL,`
	`  `debit` double NOT NULL,`
	`  `kredit` double NOT NULL,`
	`  PRIMARY KEY (`noJournal`),`
	`  KEY `kodecoa` (`kodecoa`),`
	`  CONSTRAINT `Journal_ibfk_1` FOREIGN KEY (`kodecoa`) REFERENCES `coa` (`kodecoa`) ON DELETE CASCADE ON UPDATE CASCADE`
	`) ENGINE=InnoDB`

3. Tabel *Trial Balance*

Berikut *syntax* hasil *generate* tabel *Trial Balance* ke dalam *database* MySQL:

Tabel 3.39 Tabel *Trial Balance*

Tabel	*Syntax* MySQL

Trialbalance
PK: **notb**
FK1: kodecoa, namacoa, saldoawal, debit, kredit, balance
	`CREATE TABLE `trialbalance` (`
	`  `notb` int(11) NOT NULL AUTO_INCREMENT,`
	`  `kodecoa` int(10) unsigned NOT NULL,`
	`  `namacoa` varchar(80) NOT NULL,`
	`  `saldoawal` double NOT NULL,`
	`  `debit` double NOT NULL,`
	`  `kredit` double NOT NULL,`
	`  `balance` double NOT NULL,`
	`  PRIMARY KEY (`notb`),`
	`  KEY `kodecoa` (`kodecoa`),`
	`  CONSTRAINT `trialbalance_ibfk_1` FOREIGN KEY (`kodecoa`) REFERENCES `coa` (`kodecoa`) ON DELETE CASCADE ON UPDATE CASCADE`
	`) ENGINE=InnoDB`

4. Tabel Neraca

Berikut *syntax* hasil *generate* tabel neraca ke dalam *database* MySQL :

Tabel 3.40 Tabel Neraca

Tabel	*Syntax* MySQL

Neraca
PK noneraca
FK1 notb
kodecoa
namacoa
tgltransaksi
nominal
	`CREATE TABLE `neraca` (`
	`  `noneraca` int(11) NOT NULL AUTO_INCREMENT,`
	`  `notb` int(11) NOT NULL,`
	`  `kodecoa` int(10) unsigned NOT NULL,`
	`  `namacoa` varchar(80) NOT NULL,`
	`  `tgltransaksi` date NOT NULL,`
	`  `nominal` double NOT NULL,`
	`  PRIMARY KEY (`noneraca`),`
	`  KEY `notb` (`notb`),`
	`  CONSTRAINT `neraca_ibfk_1` FOREIGN KEY (`notb`) REFERENCES `trialbalance` (`notb`) ON DELETE CASCADE ON UPDATE CASCADE`
	`) ENGINE=InnoDB`

5. Tabel Laba Rugi

Berikut *syntax* hasil *generate* tabel laba rugi ke dalam *database* MySQL :

Tabel 3.41 Tabel Laba Rugi

Tabel	*Syntax* MySQL

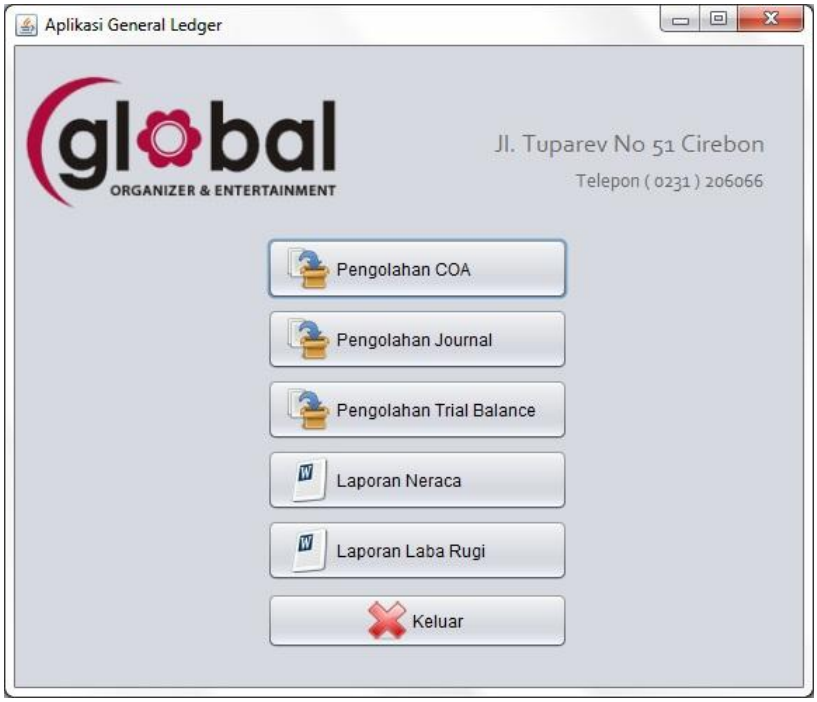
Labarugi
PK nolb
FK1 notb
kodecoa
namacoa
tgltransaksi
nominal
	`CREATE TABLE `labarugi` (`
	`  `nolb` int(11) NOT NULL AUTO_INCREMENT,`
	`  `notb` int(11) NOT NULL,`
	`  `kodecoa` int(10) unsigned NOT NULL,`
	`  `namacoa` varchar(80) NOT NULL,`
	`  `tgltransaksi` date NOT NULL,`
	`  `nominal` double NOT NULL,`
	`  PRIMARY KEY (`nolb`),`
	`  KEY `notb` (`notb`),`
	`  CONSTRAINT `labarugi_ibfk_1` FOREIGN KEY (`notb`) REFERENCES `trialbalance` (`notb`) ON DELETE CASCADE ON UPDATE CASCADE`
	`) ENGINE=InnoDB`

3. 3. 5. Tampilan Aplikasi

Setelah melakukan implementasi data, dilakukan implementasi antar muka yang berguna sebagai tampilan untuk pengguna sistem. Implementasi antar muka

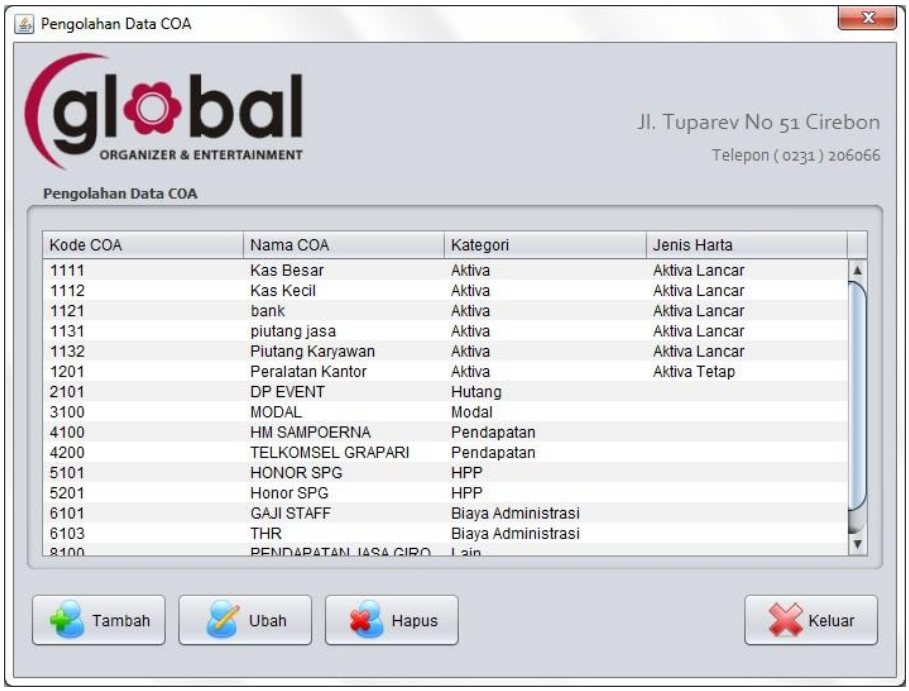
dilakukan pada setiap hasil perancangan antar muka yang telah dibuat kedalam aplikasi yang dibangun. Implementasi antarmuka pada aplikasi *General Ledger* Bulanan di CV. Global Agency and Entertainment dapat dilihat pada Gambar 3.79 sampai dengan Gambar 3.128.

1. Implementasi Antarmuka Tampilan Utama



Gambar 3.94 Implementasi Antarmuka Tampilan Utama

2. Implementasi Antarmuka *Form* Pengolahan COA



Gambar 3.95 Implementasi Antarmuka *Form* Pengolahan COA



Gambar 3.96 Implementasi Antarmuka *Form* Tambah COA

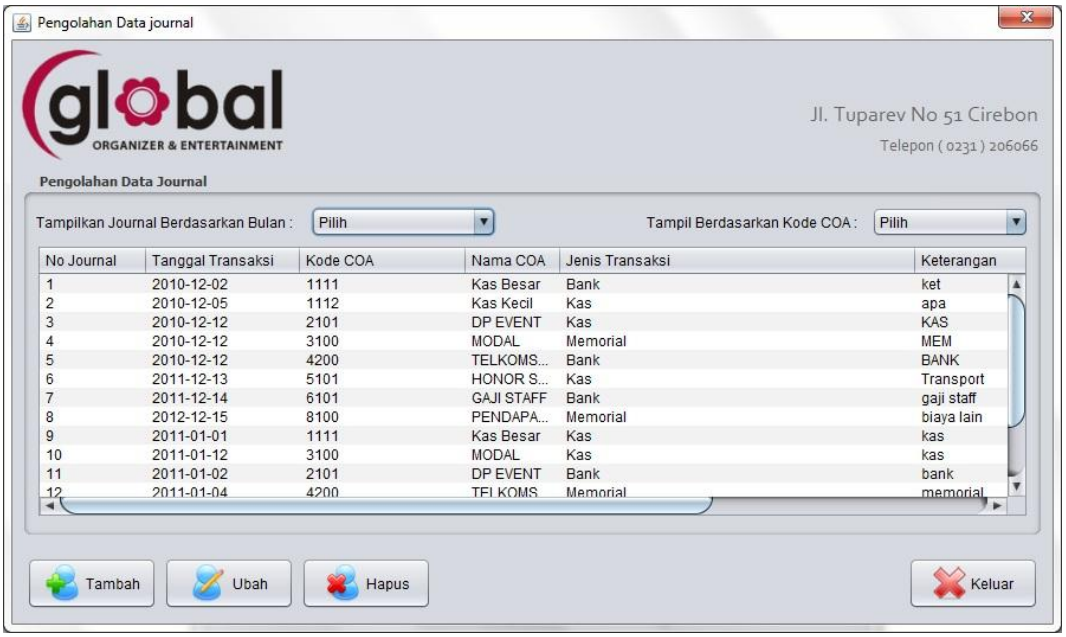


Gambar 3.97 Implementasi Antarmuka *Form* Ubah COA

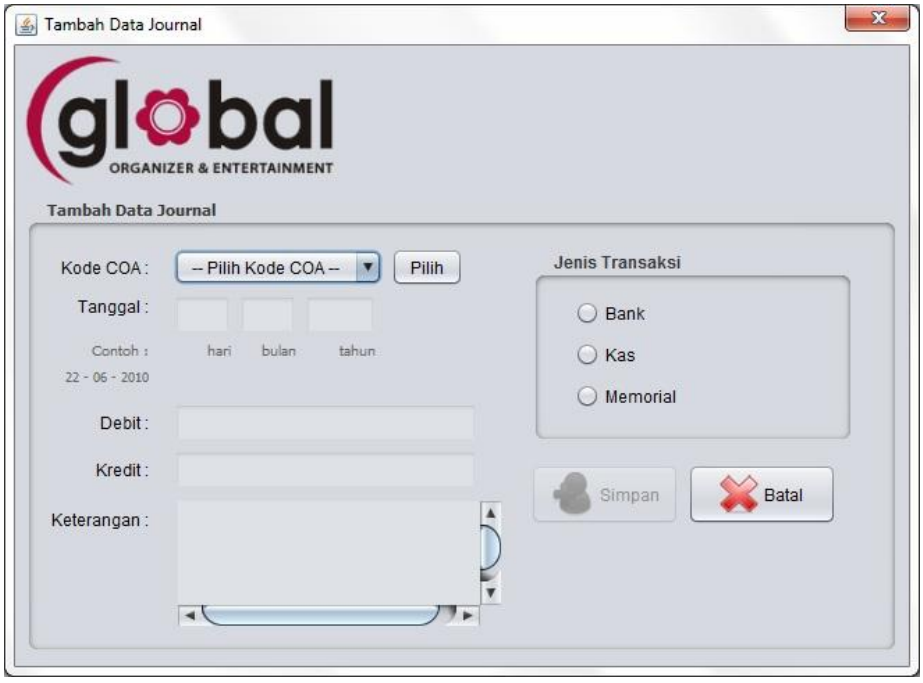


Gambar 3.98 Implementasi Antarmuka Form Hapus COA

3. Implementasi Antarmuka *Form* Pengolahan *Journal*



Gambar 3.99 Implementasi Antarmuka *Form* Pengolahan *Journal*



Gambar 3.100 Implementasi Antarmuka *Form* Tambah *Journal*

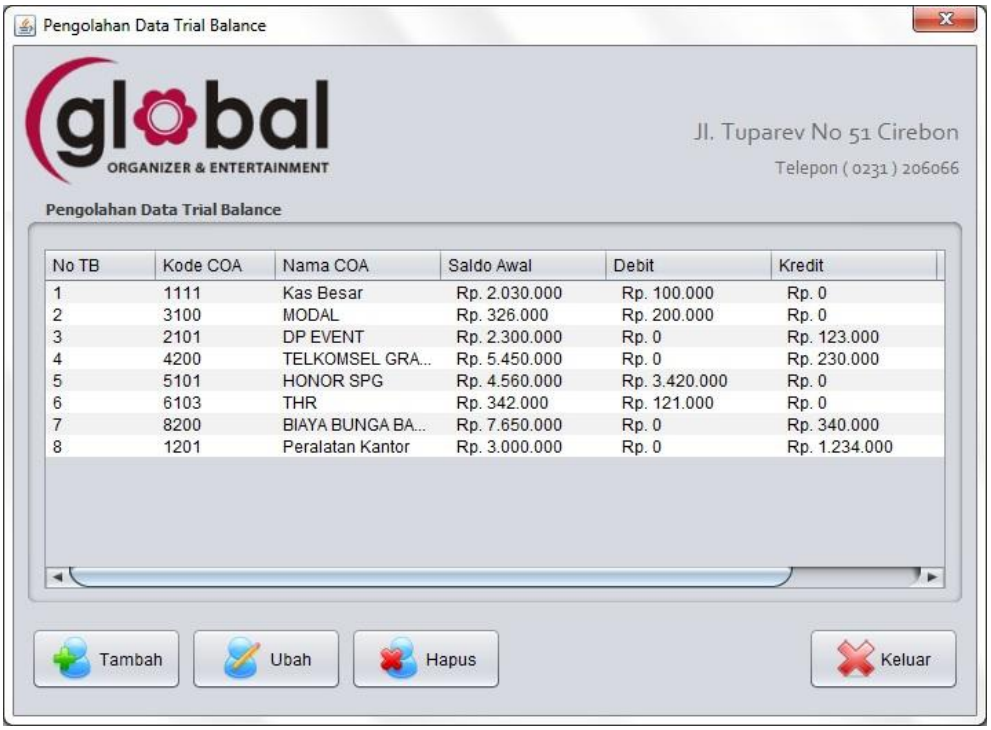


Gambar 3.101 Implementasi Antarmuka *Form* Ubah *Journal*



Gambar 3.102 Implementasi Antarmuka *Form* Hapus *Journal*

4. Implementasi Antarmuka *Form* Pengolahan *Trial Balance*



Gambar 3.103 Implementasi Antarmuka *Form* Pengolahan *Trial Balance*



Gambar 3.104 Implementasi Antarmuka *Form* Tambah *Trial Balance*



Gambar 3.105 Implementasi Antarmuka *Form* Ubah *Trial Balance*



Gambar 3.106 Implementasi Antarmuka *Form* Hapus *Trial Balance*

5. Implementasi Antarmuka *Form* Pengolahan Laporan Neraca



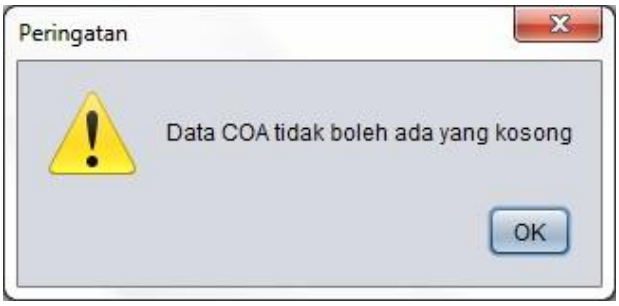
Gambar 3.107 Implementasi Antarmuka *Form* Pengolahan Laporan Neraca

6. Implementasi Antarmuka *Form* Pengolahan Laporan Laba Rugi

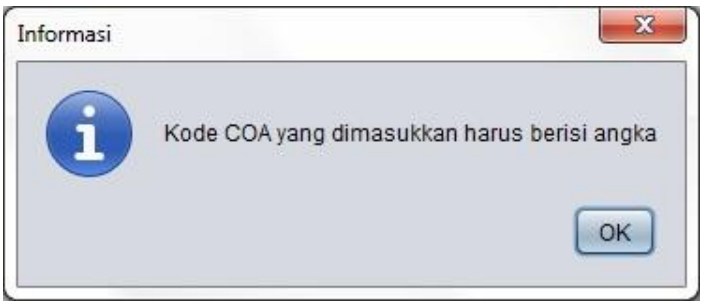


Gambar 3.108 Implementasi Antarmuka *Form* Pengolahan Laporan Laba Rugi

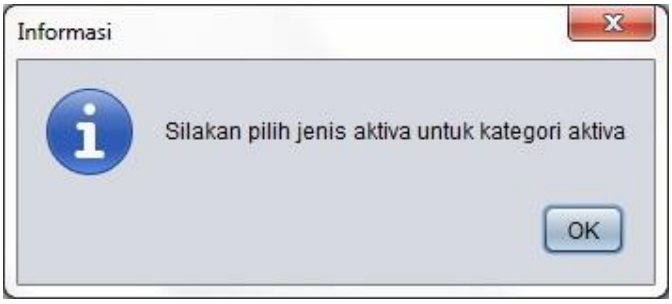
7. Implementasi Antarmuka Pesan Pengolahan COA



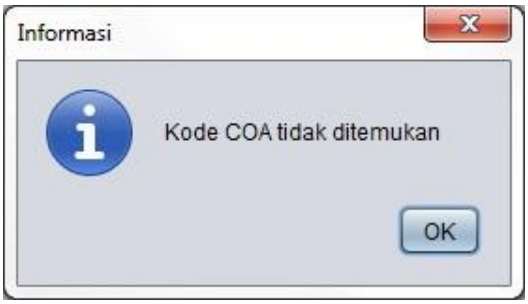
Gambar 3.109 Implementasi Antarmuka Pesan Peringatan Tambah COA



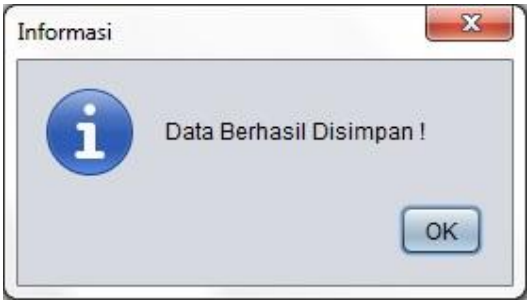
Gambar 3.110 Implementasi Antarmuka Pesan Informasi Kode COA



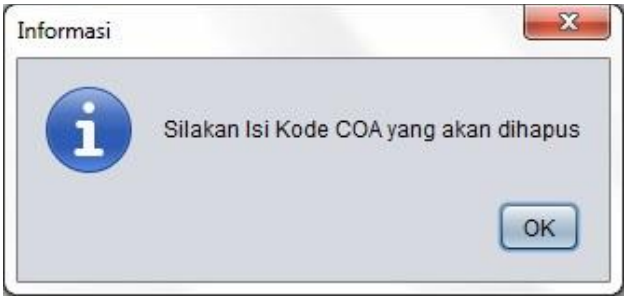
Gambar 3.111 Implementasi Antarmuka Pesan Informasi Pilih Kategori



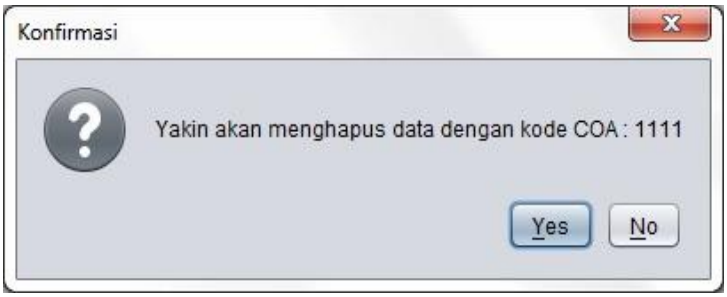
Gambar 3.112 Implementasi Antarmuka Pesan Informasi Kode COA



Gambar 3.113 Implementasi Antarmuka Pesan Informasi Simpan Data



Gambar 3.114 Implementasi Antarmuka Pesan Informasi Kode COA

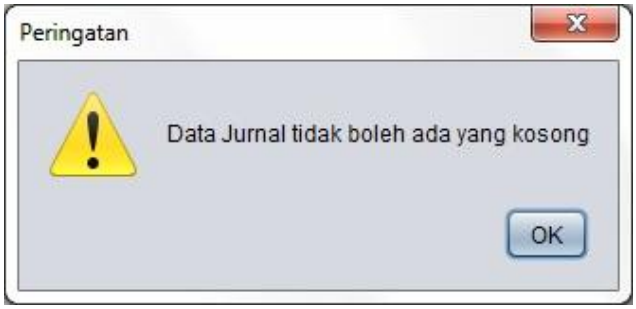


Gambar 3.115 Implementasi Antarmuka Pesan Konfirmasi Hapus COA

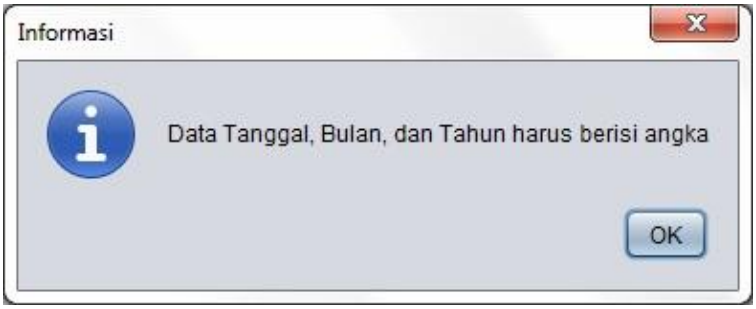
8. Implementasi Antarmuka Pesan Pengolahan *Journal*



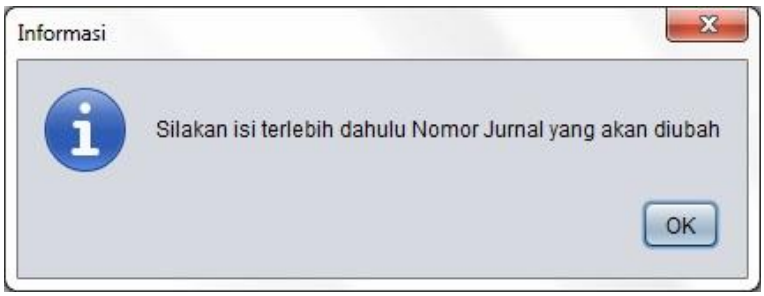
Gambar 3.116 Implementasi Antarmuka Pesan Informasi Kode COA



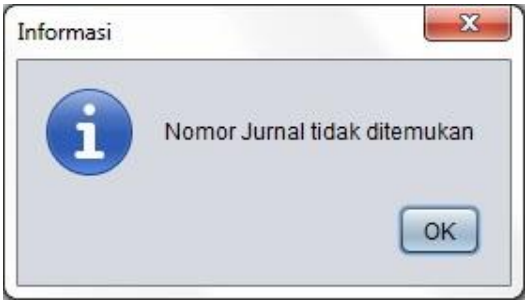
Gambar 3.117 Implementasi Antarmuka Pesan Peringatan Tambah *Journal*



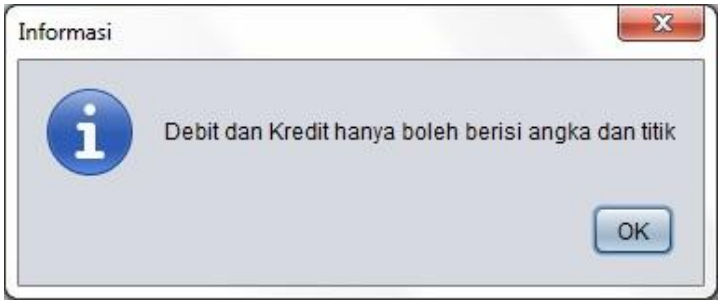
Gambar 3.118 Implementasi Antarmuka Pesan Informasi Validasi Angka



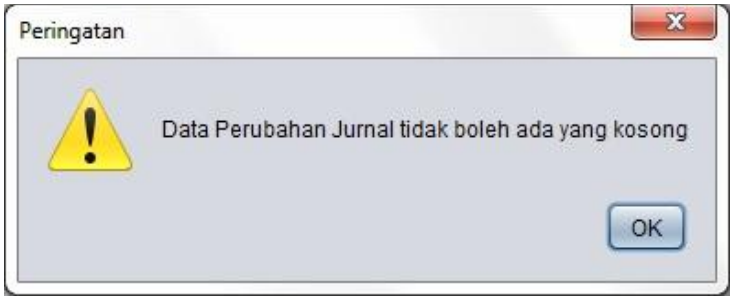
Gambar 3.119 Implementasi Antarmuka Pesan Informasi No *Journal*



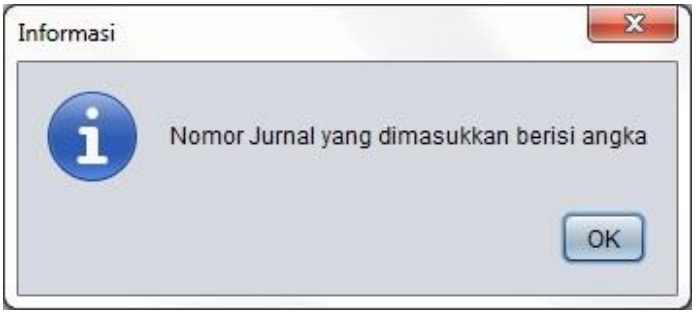
Gambar 3.120 Implementasi Antarmuka Pesan Informasi No *Journal*



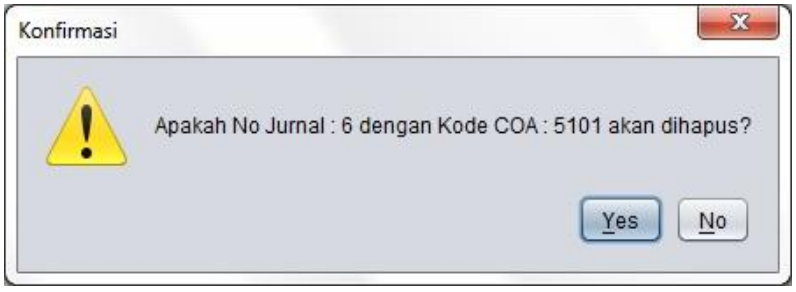
Gambar 3.121 Implementasi Antarmuka Pesan Informasi Validasi Angka



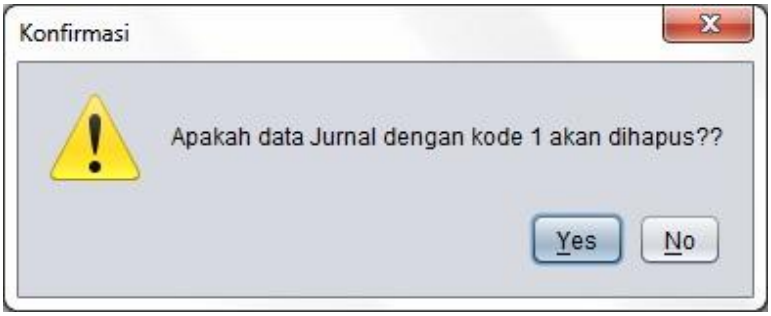
Gambar 3.122 Implementasi Antarmuka Pesan Peringatan Ubah *Journal*



Gambar 3.123 Implementasi Antarmuka Pesan Informasi Validasi Angka

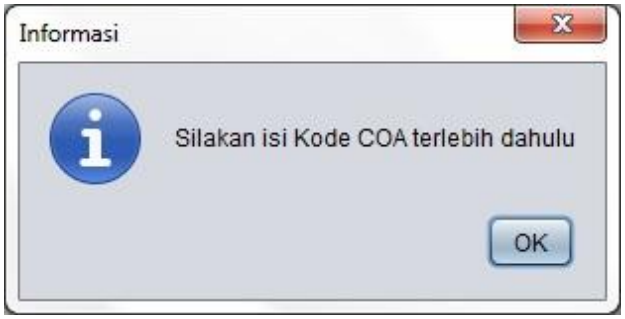


Gambar 3.124 Implementasi Antarmuka Pesan Konfirmasi Hapus *Journal*

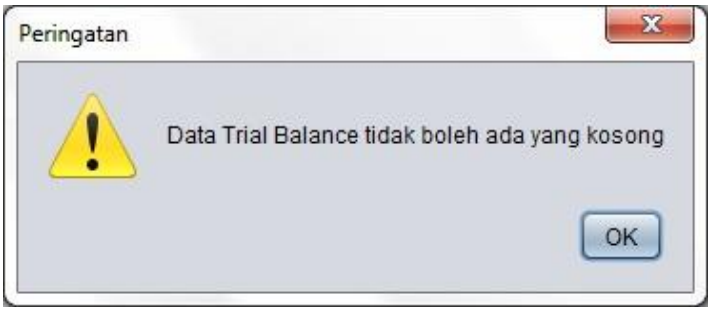


Gambar 3.125 Implementasi Antarmuka Pesan Konfirmasi Hapus *Journal*

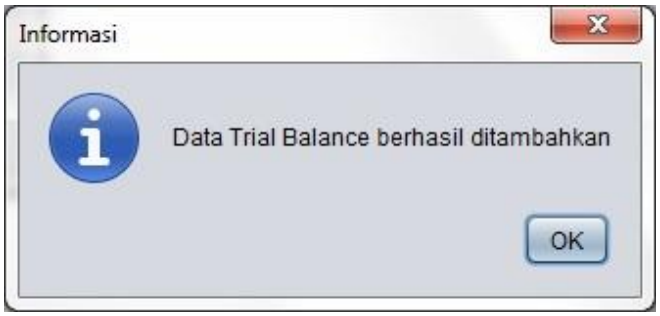
9. Implementasi Antarmuka Pesan Pengolahan *Trial Balance*



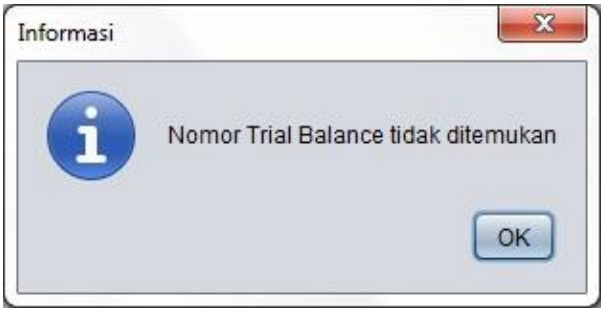
Gambar 3.126 Implementasi Antarmuka Pesan Informasi Tambah *TB*



Gambar 3.127 Implementasi Antarmuka Pesan Peringatan Tambah *TB*



Gambar 3.128 Implementasi Antarmuka Pesan Informasi Simpan Berhasil



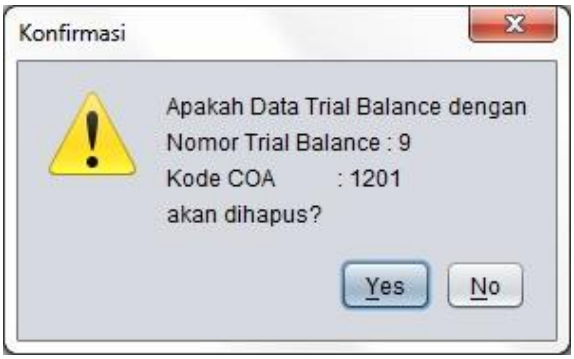
Gambar 3.129 Implementasi Antarmuka Pesan Informasi Kesalahan No *TB*



Gambar 3.130 Implementasi Antarmuka Pesan Informasi Simpan Gagal



Gambar 3.131 Implementasi Antarmuka Pesan Informasi Ubah Berhasil



Gambar 3.132 Implementasi Antarmuka Pesan Konfirmasi Hapus *TB*

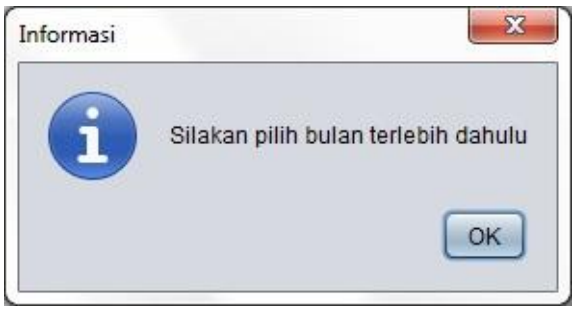


Gambar 3.133 Implementasi Antarmuka Pesan Informasi Hapus Berhasil

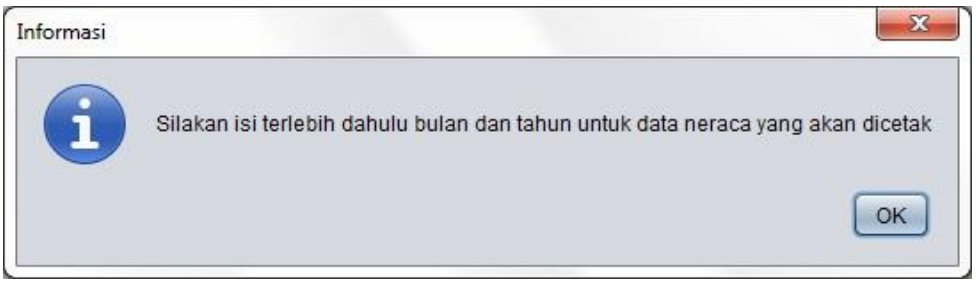
10. Implementasi Antarmuka Pesan Pengolahan Laporan Neraca



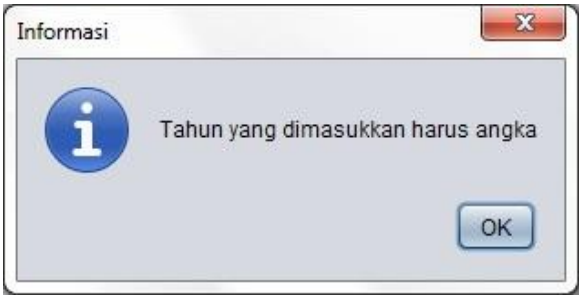
Gambar 3.134 Implementasi Antarmuka Pesan Informasi Data Tahun



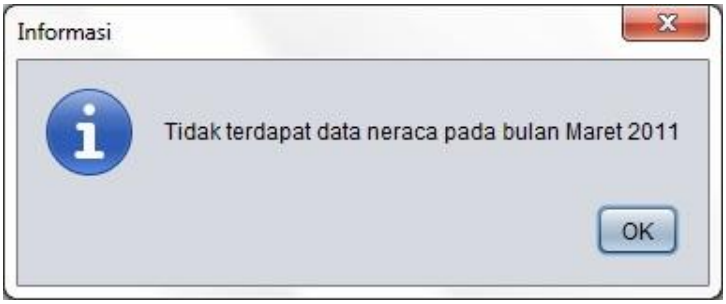
Gambar 3.135 Implementasi Antarmuka Pesan Informasi Data Bulan



Gambar 3.136 Implementasi Antarmuka Pesan Informasi Tahun dan Bulan



Gambar 3.137 Implementasi Antarmuka Pesan Informasi Validasi Angka

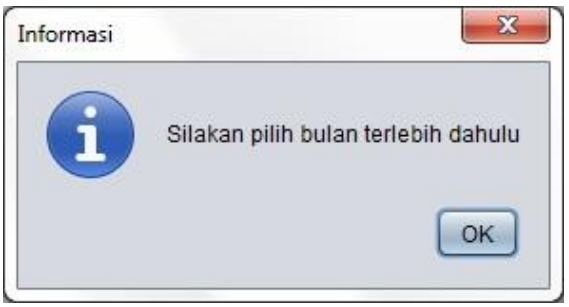


Gambar 3.138 Implementasi Antarmuka Pesan Informasi Pencarian

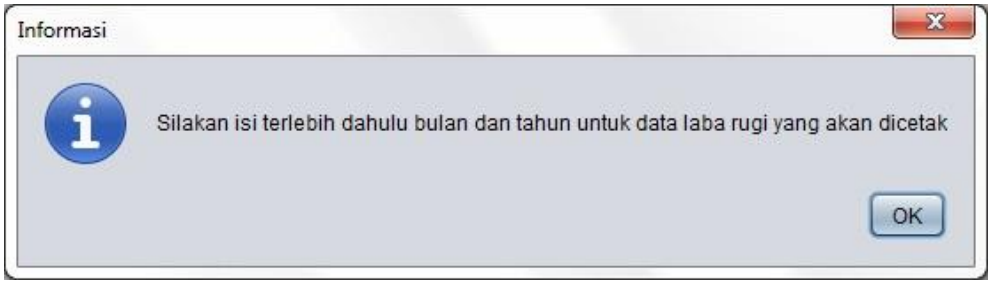
11. Implementasi Antarmuka Pesan Pengolahan Laporan Laba Rugi



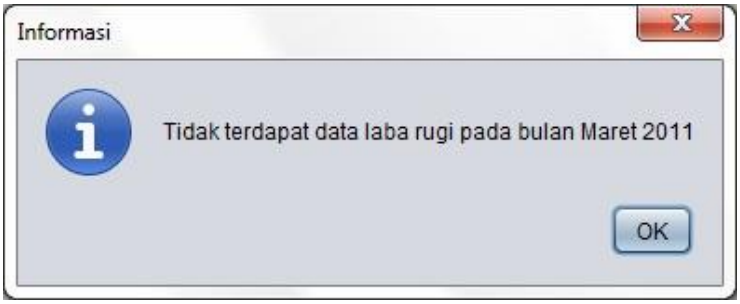
Gambar 3.139 Implementasi Antarmuka Pesan Informasi Data Tahun



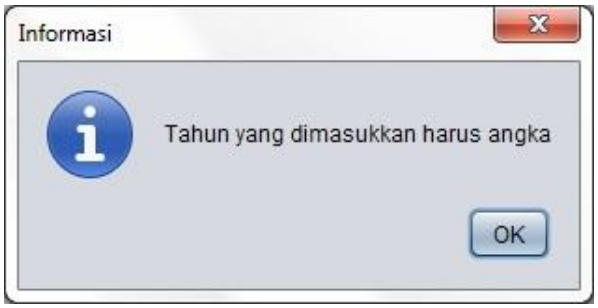
Gambar 3.140 Implementasi Antarmuka Pesan Informasi Data Bulan



Gambar 3.141 Implementasi Antarmuka Pesan Informasi Tahun dan Bulan



Gambar 3.142 Implementasi Antarmuka Pesan Informasi Pencarian



Gambar 3.143 Implementasi Antarmuka Pesan Informasi Validasi Angka