BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem yang sedang berjalan merupakan kegiatan mempelajari interaksi sistem, prosedur dan data informasi yang terkait. Pada bagian analisis sistem ini akan dibahas mengenai analisis masalah, analisis penerapan teori antrian, analisis penggunaan multithreading, analisis kebutuhan fungsional dan analisis kebutuhan fungsional. Kegiatan ini memetakan suatu sistem pada model yang dapat menganalisis seluruh aspek yang terlibat dalam sistem tersebut.

3.1.1 Analisis Masalah

Proses pengiriman laporan data dengan menggunakan sistem *single thread* belum tentu bisa mengoptimalkan sistem kerja prosesor dan proses pengiriman data laporan itu sendiri. Hal ini disebabkan belum adanya sistem yang bisa mengoptimalkan kinerja prosesor sehingga bisa mempercepat proses pengiriman data.

Berdasarkan hal tersebut penyusun mencoba untuk membuat suatu sistem yang lebih baik untuk proses pengiriman data. Dengan menggunakan sistem *multithreading* yang mengizinkan beberapa *thread* terjadi dalam sebuah proses, tentu bisa mempercepat proses pengiriman laporan data serta mengoptimalkan proses kinerja prosesor. Untuk lebih mengenal sistem *multithreading* maka diambil sebuah contoh studi kasus pada perusahaan besar yang bergerak dalam bidang distributor yaitu PT Bina San Prima Bogor sebagai salah satu anak cabang dari PT Bina San Prima.

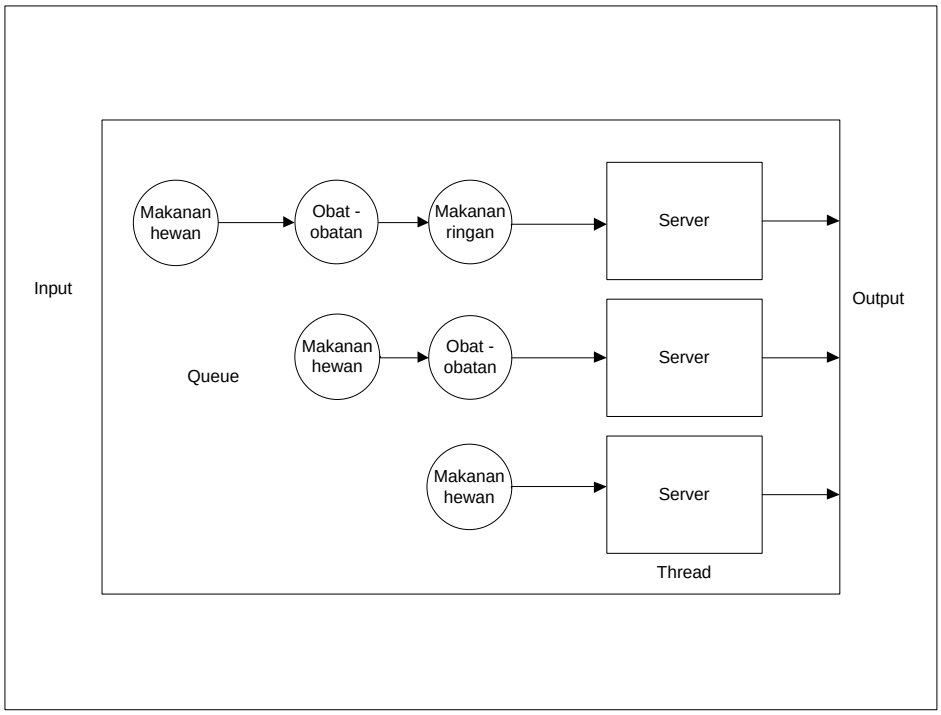
PT Bina San Primas sekarang memiliki lebih dari 200 macam produk yang terbagi dalam 35 divisi produk yang aktif dalam transaksi faktur. Apabila dalam satu proses penarikan data untuk membentuk laporan seluruh produk maka sistem yang sekarang berjalan adalah dengan menarik divisi produk secara *one by one*, dengan pemrosesan divisi produk pertama hingga selesai kemudian dilanjutkan

dengan divisi produk kedua hingga selesai dan seterusnya hingga divisi produk sejumlah ke-n.

3.1.2 Analisis Penerapan Teori Antrian

Analisis penerapan teori antrian ini menggunakan penjadwalan algoritma *First-Come First-Served Scheduling* (FCFS*)* ini, merupakan algoritma penjadwalan yang paling sederhana yang digunakan CPU. Dengan menggunakan algoritma ini setiap proses yang berada pada status *ready* dimasukkan ke dalam *queue* sesuai dengan waktu kedatangannya. Proses yang tiba terlebih dahulu yang akan dieksekusi. Untuk mengenal lebih jelas tentang penerapan teori antrian FCFS didalam pengiriman laporan data maka diberikan contoh kasus.

PT Bina San Prima Bogor sebagai anak cabang perusahaan distributor yang memiliki 3 divisi produk yang berdeda yaitu produk makanan ringan, obat – obatan dan makanan hewan. PT Bina San Prima Bogor ini selalu melakukan proses penutupan buku pada tiap akhir bulan. Permasalahan yang sering dihadapi adalah ketika PT Bina san Prima bogor ini akan mengirimkan laporan data pada perusahaan pusat dimana divisi produk dari cabang perusahaan makanan ringan yang masih aktif mengirimkan laporan data maka divisi produkobat –obatan dan makanan hewan yang akan mengirimkan laporan data harus menunggu divisi makanan ringan selesai dikirim. Deskripsi contoh permasalahan dapat dilihat pada gambar 3.1.



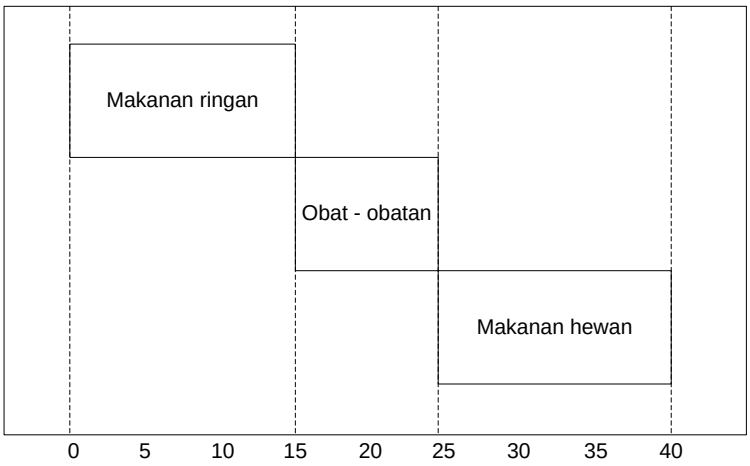
Gambar 3.1 Penerapan teori antrian PT Bina San Prima Bogor

3.1.3 Analisis Penggunaan *Multithreading*

Analisis penggunaan *multithreading* merupakan suatu proses yang mengimplementasikan sistem yang akan digunakan kedalam suatu aplikasi. Untuk memperjelas penggunaan *multithreading* ini maka diberikan contoh kasus berdasarkan proses pengiriman data antara *single thread* dengan *multithreading*.

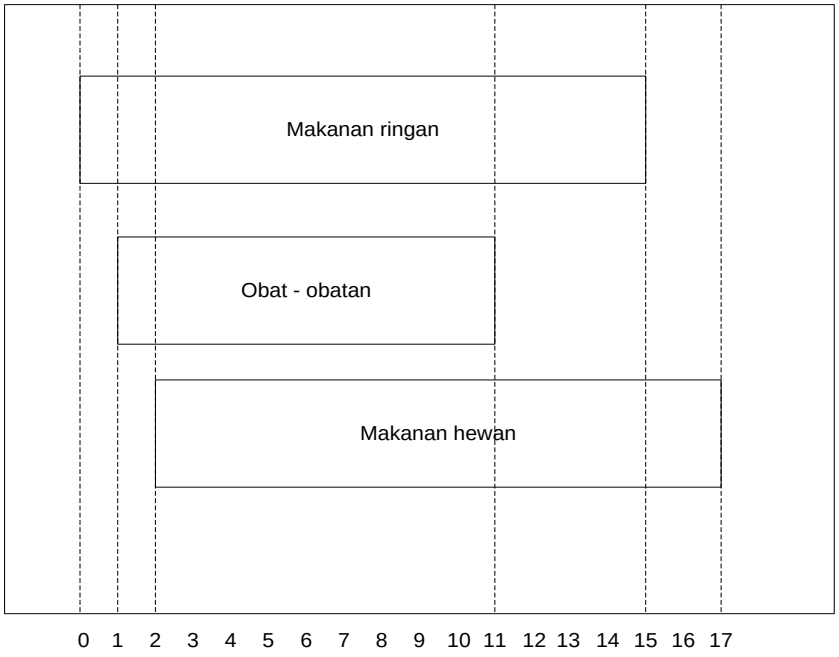
PT Bina San Prima Bogor akan melakukan proses evaluasi dimana proses evaluasi ini dilakukan rutin setiap bulannya. PT Bina San Prima Bogor ini memasarkan tiga jenis produk yang berbeda yaitu produk pertama adalah makanan ringan produk kedua adalah obat-obatan dan produk yang ketiga adalah makanan hewan.

Diberikan contoh perbedaan proses *single thread* dan *multithread* berdasarkan dari waktu penarikan dan proses yang berjalan dapat dilihat seperti pada gambar 3.2 dan gambar 3.3.



Gambar 3.2 Proses Penarikan Laporan *Single Thread*

Pada gambar 3.2 proses penarikan laporan data menggunakan *single thread* hanya mengijinkan satu proses terjadi sehingga pada gambar 3.3 proses penarikan laporan data makanan dimulai dari 0 sampai waktu penarikan selesai yaitu 15 kemudian proses penarikan laporan data Obat – obatan dimulai dari 15 sampai 25 dan proses laporan penarikan laporan produk makanan hewan dimulai dari 25 sampai 40. Dengan *single thread* jika ada tiga produk yang mempunyai waktu yang berbeda dan akan dilakukan proses penarikan laporan data maka akan terjadi antrian penarikan laporan data.



Gambar 3.3 Proses Penarikan Laporan *Multithreading*

Pada gambar 3.3 proses penarikan laporan data dengan menggunakan *multithreading* yang mengijinkan beberapa *thread* terjadi dalam sebuah proses penarikan maka laporan data penarikan dimulai secara bersamaan dengan jeda waktu tertentu dari proses pertama. Dari kedua proses diatas dapat ditarik kesimpulan seperti pada tabel 3.1.

Tabel 3.1 Kesimpulan Hasil Data Penarikan

No	Proses penarikan	Awal penarikan
1	single *thread*	Harus menunggu proses selesai baru bisa melakukan proses penarikan kembali, waktu yang dibutuhkan untuk penarikan *single thread* adalah 40 detik.
2	*multithreading*	Penarikan data dimulai secara bersamaan dengan menunggu jeda waktu proses pertama berjalan waktu yang dibutuhkan untuk melakukan proses multithread adalah 17 detik.

3.1.4 Analisis Kebutuhan *non*-Fungsional

Analisis kebutuhan non fungsional menggambarkan kebutuhan sistem yang menitik beratkan pada properti prilaku yang dimiliki oleh sistem, diantaranya kebutuhan perangkat keras, perangkat lunak, serta *user* sebagai bahan analisis kekurangan dan kebutuhan yang harus dipenuhi dalam perancangan sistem yang akan diterapkan.

3.1.4.1 Analisis Kebutuhan Perangkat Keras

Kebutuhan sistem perangkat keras untuk membangun aplikasi sistem *multithreading* ini terbagi atas *client* sebagai peminta layanan dan *server* sebagai penyedia layanan dengan kebutuhan minimun dari perangkat keras ini yaitu:

Tabel 3.2 Perbandingan Kebutuhan Perangkat Keras

Client	*Server*
a. Prosessor 2Ghz	a. Prosessor 2Ghz
b. Ram 2 GB	b. Ram 4 GB
c. Hardisk 500 GB	c. Hardisk 1TB
d. Keyboard dan Mouse	d. Keyboard dan Mouse
e. Monitor 17”	e. Monitor 17”
f. Operating sistem yang digunakan adalah Windows Xp,7	f. Operating sistem yang digunakan adalah Windows Xp,7

3.1.4.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak atau software adalah program yang berisi kumpulan instruksi untuk melakukan proses pengolahan data. Berikut perangkat lunak yang digunakan dalam pembangunan aplikasi *multithreading* adalah:

a. Mysql

b. Borland Delphi

c. Microsotf Office Excel

d. Microsof Office visio 2007

e. Rational Rose

3.1.4.3 Analisis Pengguna Sistem

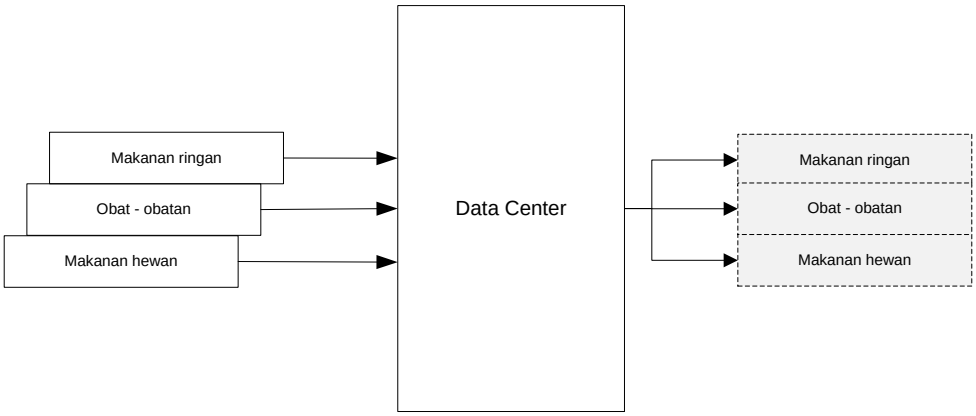
Analisis pengguna sistem dimaksudkan untuk mengetahui siapa saja aktor yang terlibat dalam menjalankan sistem. Pengguna atau *user* disini bertugas menganalisis dua sistem aplikasi yang akan digunakan dan membandingkan aplikasi mana yang bekerja lebih baik. Berikut ini karakteristik pengguna sistem yang ditunjukan oleh tabel 3.3.

Tabel 3.3 Karakteristik Pengguna Sistem Pada Perusahaan

Pengguna	Usia	Hak Akses	Tingkat Pendidikan	Pengalaman
User	Lebih dari 21 tahun	Mengakses data-data laporan Faktur.	Min. S1	Min. bisa menggunakan *Borland Delphi, Mysql dan Microsof Office Excel*

3.1.4.4 Analisis Metode *Multithreading*

Analisis metode *multithreading* dilakukan untuk memberikan gambaran terhadap pengertian dan model yang akan diambil berdasarkan kebutuhan sistem. *Multithreading* adalah cara pengeksekusian yang mengijinkan beberapa *thread* terjadi dalam sebuah proses, saling berbagi sumber daya tetapi dapat dijalankan secara independen. Model *multithreading* yang akan dipakai dalam sistem aplikasi ini adalah model *many to many* seperti contoh pada gambar 3.4.



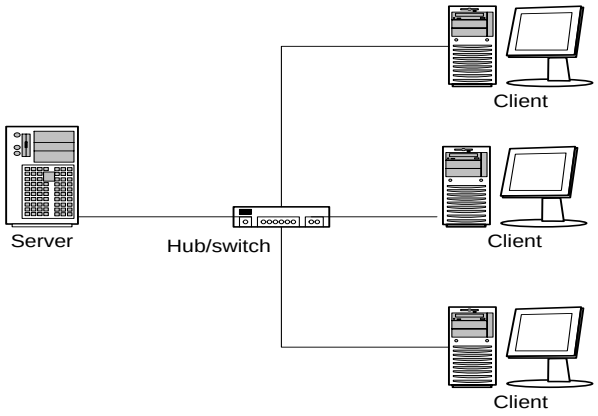
Gambar 3.4 Sistem Penarikan Laporan Data *Multithread*

Gambar 3.4 terlihat bahwa sebuah *processor* berpindah dari satu *thread* ke *thread* yang lain. proses berjalan secara bersamaan berpindah dari satu *thread* ke *thread* yang lain. Hal tersebut terjadi secara cepat sehingga *user* akan beranggapan bahwa suatu *thread* atau *task* berjalan secara bersamaan.

3.1.4.5 Analisis *Client Server*

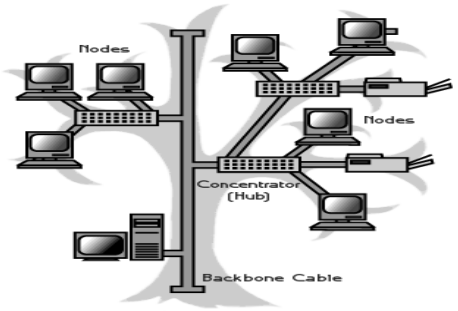
Konsep Jaringan *client-server* ini membedakan dengan jelas kedudukan suatu komputer, mana yang dapat memberikan layanan jaringan (*server*) dan mana yang hanya meminta layanan (*client*). Jika suatu komputer *diinstall* sebagai *server* yang memberikan segala sumber daya (*resource*), contohnya : *printer*, *modem* kepada komputer lain (*client)* yang terhubung ke jaringan. Untuk dapat saling berkomunikasi antara *server* dan *client*, maka digunakan suatu aplikasi jaringan yang dinamakan *client-server program*, dimana *server* menggunakan aplikasi jaringan yang disebut *server program*, sementara *client* menggunakan

client program untuk dapat berkomunikasi dengan *server program* pada *server*. Model *client-server* yang digunakan dapat terlihat pada gambar 3.5.



Gambar 3.5 Model Jaringan *Client-Server*

Topologi yang akan digunakan dalam Client Server ini berupa topologi pohon karena topologi model ini merupakan perpaduan antara topologi *Linear BUS* / Garis Lurus dan Bintang, yang terdiri dari kelompok-kelompok dari *workstation* konfigurasi Bintang yang terkoneksi ke kabel utama yang menggunakan topologi *Linear BUS* / Garis Lurus. Topologi ini memungkinkan untuk pengembangan jaringan yang telah ada, dan memungkinkan sebuah perusahaan mengkonfigurasi jaringan sesuai dengan kebutuhannya. Model topologi yang digunakan dapat terlihat pada gambar 3.6.



Gambar 3.6 Topologi Jaringan Pohon

3.1.5 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional dilakukan untuk memberikan gambaran aliran data yang ada pada program aplikasi yang akan dibangun. Tahapan-tahapan yang ada yaitu dengan mengkonfigurasi dari komponen-komponen perangkat

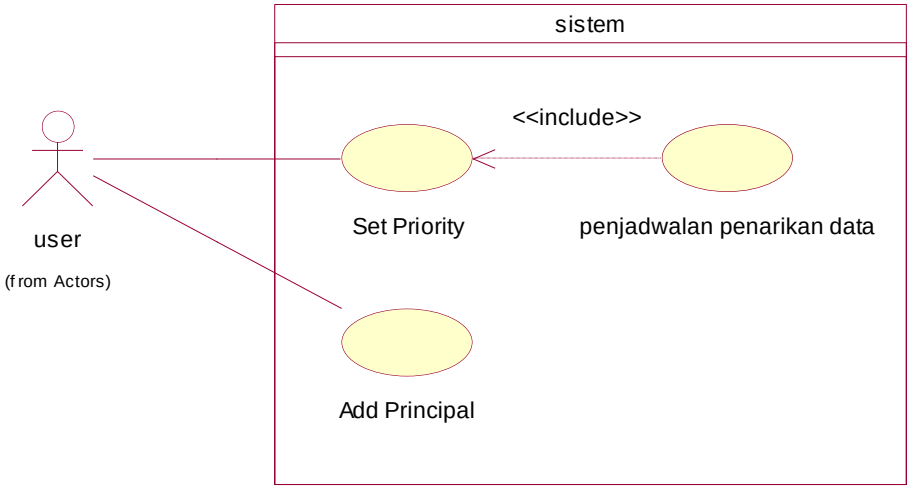
lunak dan perangkat keras dari suatu sistem. Adapun sebagai alat bantu yang digunakan untuk menggambarkan atau memodelkan sistem secara umum yang akan dibangun yaitu dengan menggunakan *Unified Modeling Language* (UML).

Pemodelan sistem dengan menggunakan *Unified Modeling Language* (UML) memiliki beberapa tahapan. Tahap-tahap pemodelan dalam analisis tersebut antara lain*Usecase diagram, Sequence Diagram, Activity Diagram, Class Diagram.*

A. Use case Diagram

Diagram *Use case* merupakan bagian tertinggi dari fungsionalitas yang dimiliki sistem yang akan menggambarkan bagaimana seseorang atau aktor akan menggunakan dan memanfaatkan sistem. Diagram ini juga mendeskripsikan apa yang akan dilakukan oleh sistem. *Use case* terdiri dari tiga bagian yaitu identifikasi aktor, identifikasi *Use case* dan skenario *Use case.*

Diagram ini menggambarkan transisis dari perubahan keadaan suatu objek, akibat dari stimulus/*input* yang diterimanya, statechart diagram menelusuri individu-individu obyek melalui keseluruhan daur hidupnya, menspesifikasikan semua urutan yang mungkin dari pesan-pesan yang akan diterima obyek tersebut, bersama-sama dengan tanggapan atas pesan-pesan tersebut.



Gambar 3.7 *Use case* penarikan laporan data

Gambar 3.8 menjelaskan bahwa aktor sebagai pengguna aplikasi bisa memilih apa saja yang akan diproses atau jika perlu menambah data sebelum proses penarikan laporan data dijalankan. Actor disini bisa memilih salah satu dari dua proses yang ada yaitu proses single *thread* atau proses *multithread* untuk dijalankan.

1. **Identifikasi Aktor**

   Identifikasi aktor adalah Aktor yang berperan dalam menjalankan sistem, aktor yang berperan dapat dilihat pada tabel 3.4.

Tabel 3.4 Aktor Sistem

Aktor	Deskripsi
Pengguna	Merupakan aktor yang menggunakan aplikasi *single thread* atau *multithreading* untuk penarikan laporan data
Admin	Merupakan aktor yang akan mengelola aplikasi *multithreading*.

Dari tabel 3.4 diperoleh keterangan pengguna aplikasi bisa melakukan penarikan data dengan menggunakan aplikasi *single thread* atau *multithread.* Sedangkan admin mengelola kedua proses yang dijalankan.

2. **Skenario *Use case***

   Skenario proses-proses yang terdapat didalam *Use case* dapat dilihat pada tabel-tabel berikut.

Tabel 3.5 Skenario *Use case Add Principal*

Identifikasi	
Nama *Use case*	*Add principal*
Deskripsi	Fungsional untuk menambah data principal
Aktor	Pengguna
Tujuan	Menambahkan divisi *principal*
Keadaan Awal	Sistem menampilkan informasi *principal* yang ada
Skenario Utama	
Aksi Aktor	**Reaksi sistem**
1. Menambahkan data *principal*	
2. Menekan tanda tambah	
	3. Melakukan pengecekan data apabila data sesuai maka akan kelangkah 5, jika data tidak sesuai maka akan kelangkah 4.
	4. Menampilkan data yang dimasukan tidak sesuai maka akan kembali kelangkah 1
	5. menyimpan data principal
Kondisi Akhir	Menampilkan data *principal* yang baru ditambahkan

Tabel 3.6 Skenario *Use case* Set *Priority*

Identifikasi	
Nama *Use case*	Set *Priority*
Deskripsi	Fungsionalitas untuk memilih data yang di*priority*
Aktor	Pengguna
Tujuan	Memilih pilihan data
Keadaan Awal	Sistem menampilkan informasi *principal* yang ada
Skenario Utama	
Aksi Aktor	**Reaksi sistem**
1. Memilih periode awal dan akhir	
2. Memilih jenis penjualan	
3. Memilih jenis *Principal*	
4. Mengklik Penjadwalan penarikan data	
	5. Melakukan pengecekan data apabila data sesuai maka akan kelangkah 7, jika data tidak sesuai maka akan kelangkah 6.
	6. Menampilkan data yang dipilih tidak sesuai maka akan kembali kelangkah 1.
	7. Menampilkan informasi penjadwalan
Kondisi Akhir	Menampilkan data *principal* sesuai yang dipilih

Tabel 3.7 Skenario *Use case* Penjadwalan penarikan data

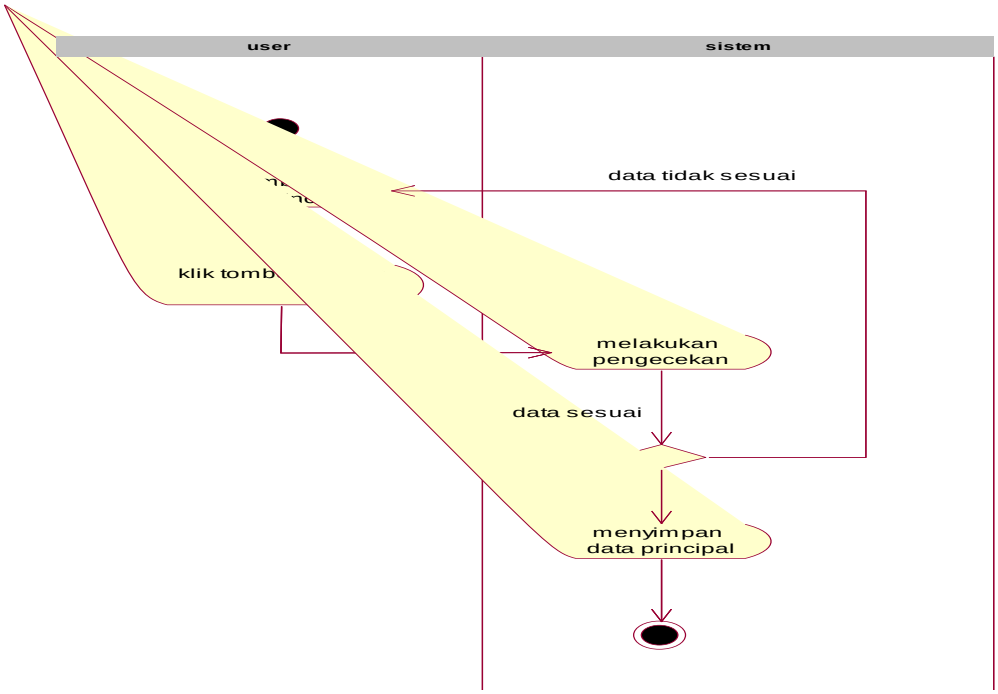
Identifikasi	
Nama *Use case*	Penjadwalan penarikan data
Deskripsi	Fungsionalitas untuk memilih penjadwalan penarikan data
Aktor	Pengguna
Tujuan	Memilih Antrian
Keadaan Awal	Sistem menampilkan informasi produk yang ada
Skenario Utama	
Aksi Aktor	**Reaksi sistem**
1. Memilih periode awal dan akhir	
2. Memilih jenis penjualan	
3. Memilih jenis *Principal*	
4. Mengklik Penjadwalan penarikan data	
	5. Melakukan pengecekan data apabila data sesuai maka akan kelangkah 7, jika data tidak sesuai maka akan kelangkah 6.
	6. Menampilkan data yang dipilih tidak sesuai maka akan kembali kelangkah 1.
	7. Menampilkan informasi penjadwalan
8. Memilih penjadwalan antian jika memilih *proses singlethread* maka kelangkah 9, jika memilih proses *multithread* maka kelangkah 12	
9. Memilih *proses singlethread*	
	10. Menjalankan proses penarikan secara *singlethread* Penarikan berjalan secara sendiri-sendiri sehingga didapatkan waktu dari hasil penarikan selama 40detik.
	11. Mengeksport database to pivot
12. Memilih *proses multithreading*	
	13. Menjalankan proses penarikan secara *singlethread* Penarikan berjalan secara sendiri-sendiri sehingga didapatkan waktu dari hasil penarikan selama 17detik.
	14. Mengeksport database to pivot
Kondisi Akhir	Setelah melakukan proses penarikan *singlethread* didapatkan waktu selama 40detik sedangkan proses penarikan *multithreading* didapatkan waktu selama 17 detik, sehingga dapat diambil kesimpulan bahwa proses penarikan dengan menggunakan *multitthread* lebih cepat dari s*inglethread*

B. *ActivityDiagram*

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis, aliran kerja dalam banyak kasus, menggambarkan rangkaian aliran dari aktivitas, dan digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *Use case* atau interaksi.

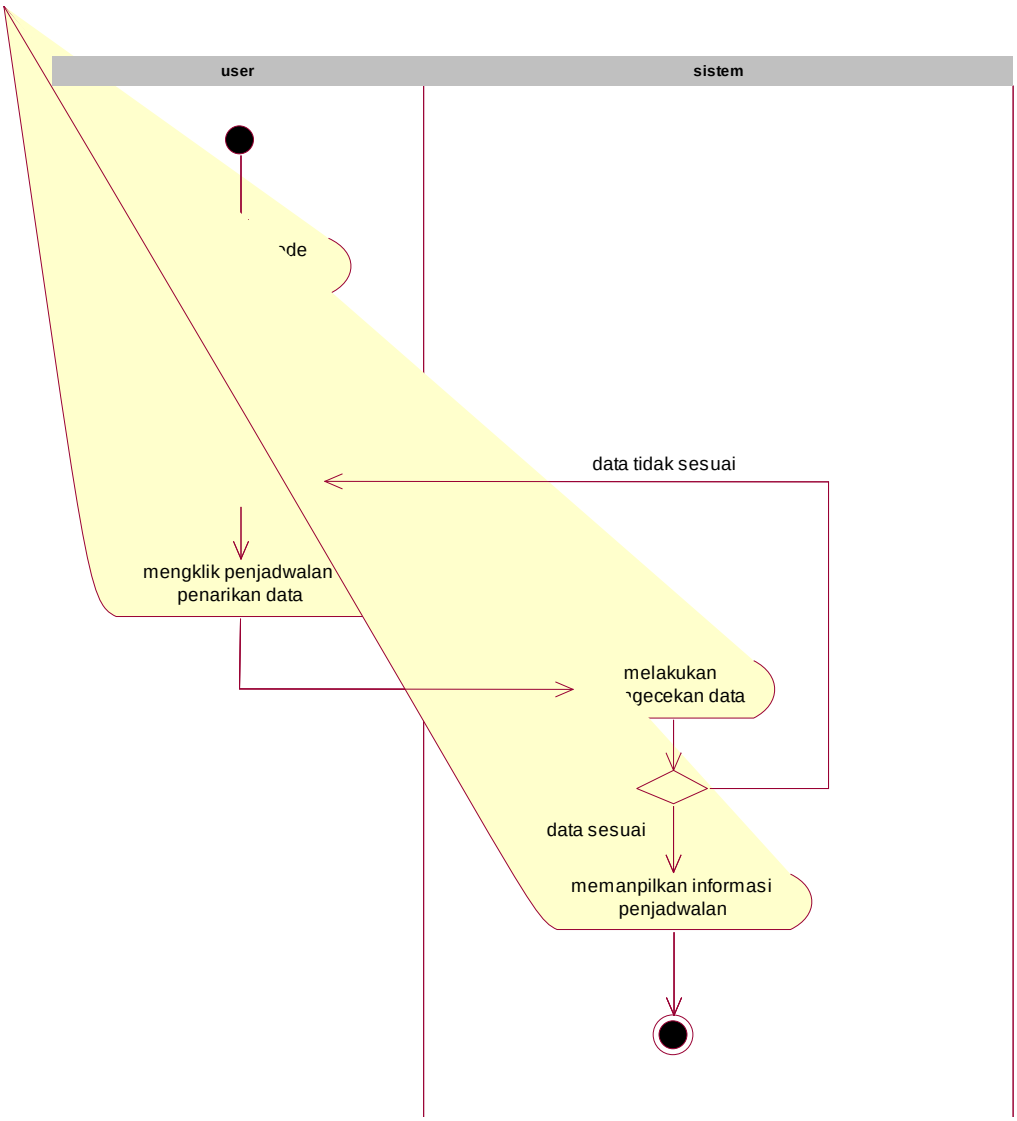
Activity diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku *parallel* sedangkan *flowchart* tidak bisa.

1. *Acivity Diagram* Add principal



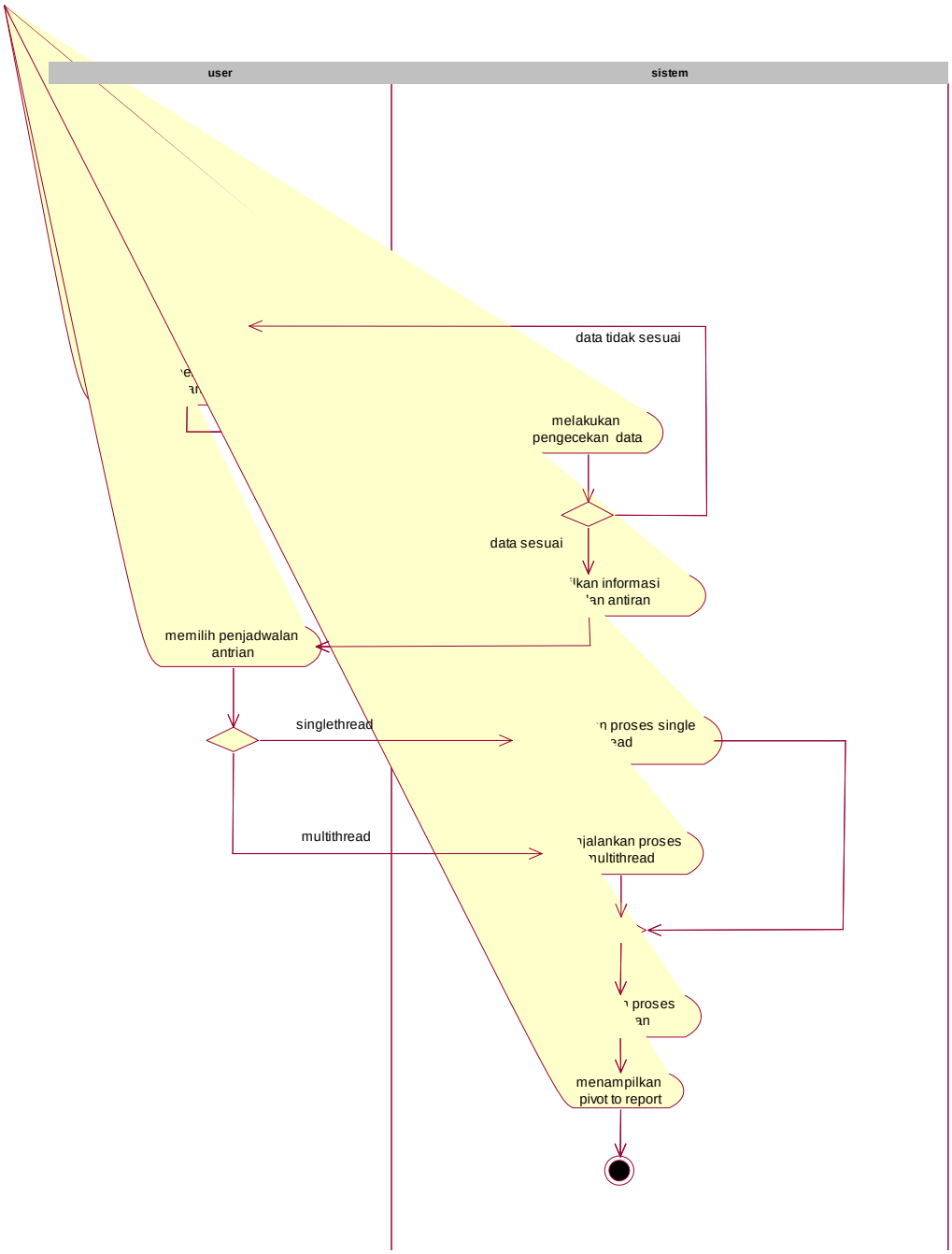
Gambar 3.8 ***Acivity Diagram Add principal***

2. *Acivity Diagram* Set *Priority*



Gambar 3.9 ***Acivity Diagram Set priority***

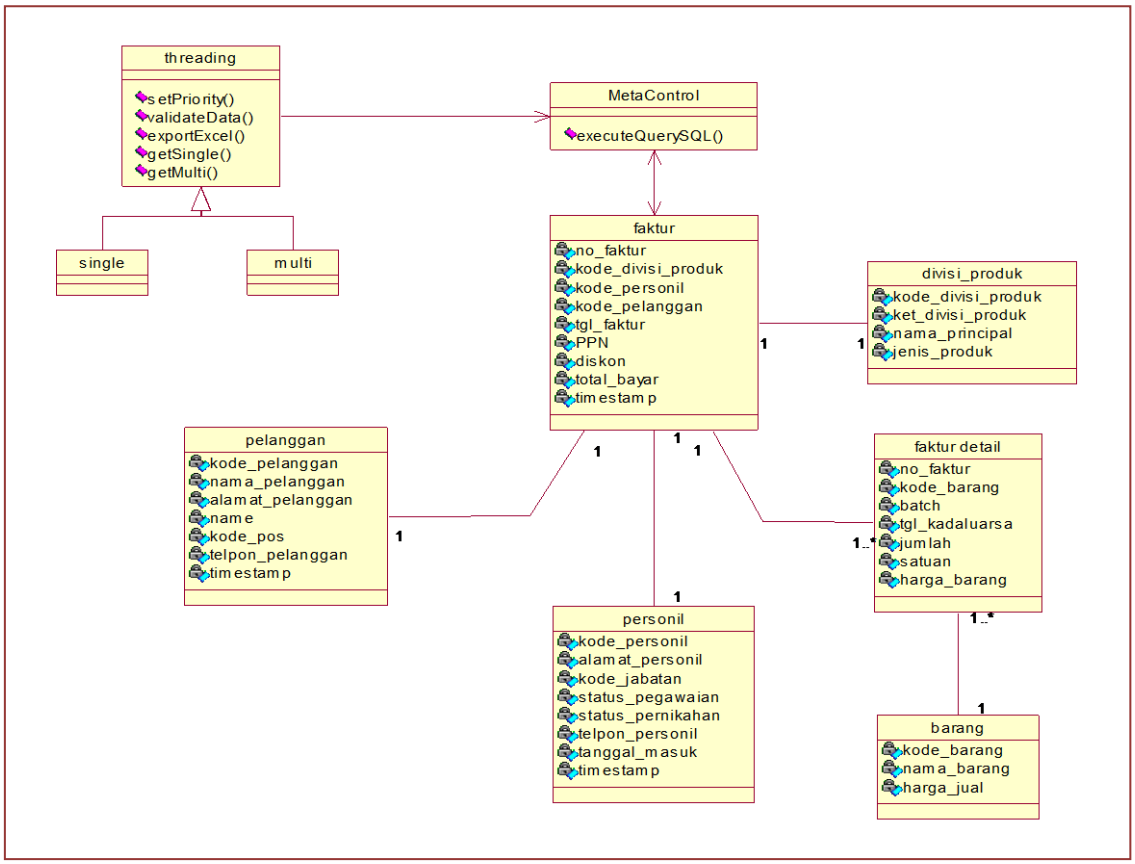
3. *Acivity Diagram* Penjadwalan penarikan data



Gambar 3.10 *Acivity Diagram* Penjadwalan penarikan data

C. *Class Diagram*

Tahap pertama yang dilakukan dalam melakukan analisis berorientasi objek menggunakan UML adalah menentukan aktor atau pengguna sistem. Kata aktor dalam konteks UML, menampilkan peran (*roles*) yang pengguna (atau sesuatu di luar sistem yang dikembangkan yang dapat berupa perangkat keras, *end user*, sistem yang lain, dan sebagainya).



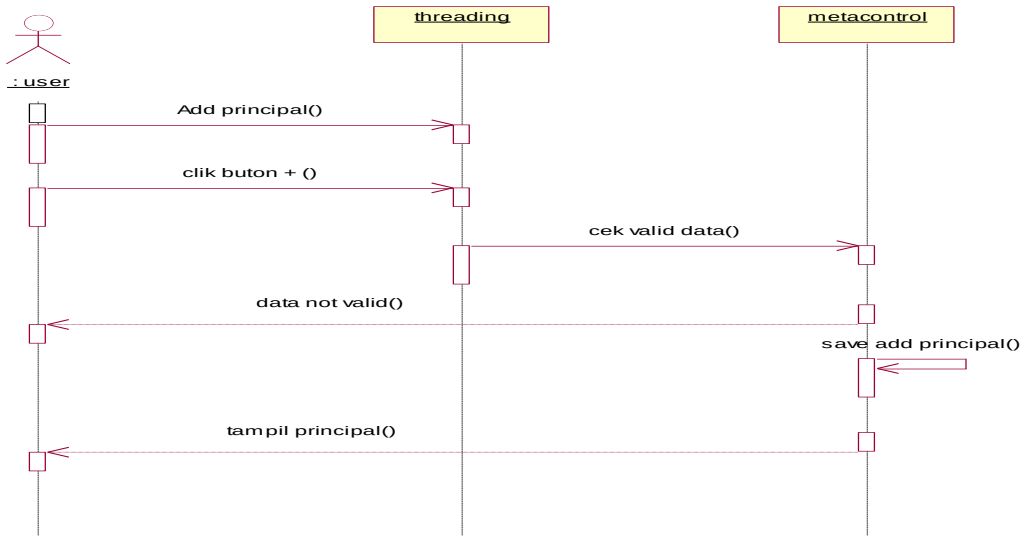
Gambar 3.11 *Class Diagram* penarikan laporan

D. *Sequence Diagram*

Pada tahap ini *Sequence Diagram* menjelaskan secara detil urutan proses yang dilakukan sistem untuk mencapai tujuan dari *Use case*. Interaksi yang

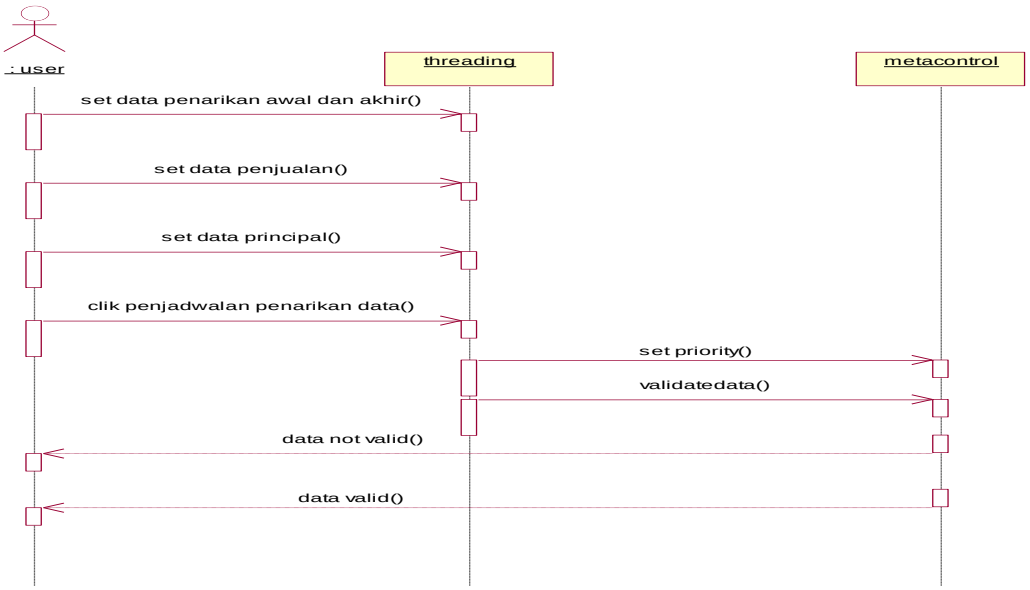
terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

1. *Sequence Add principal*



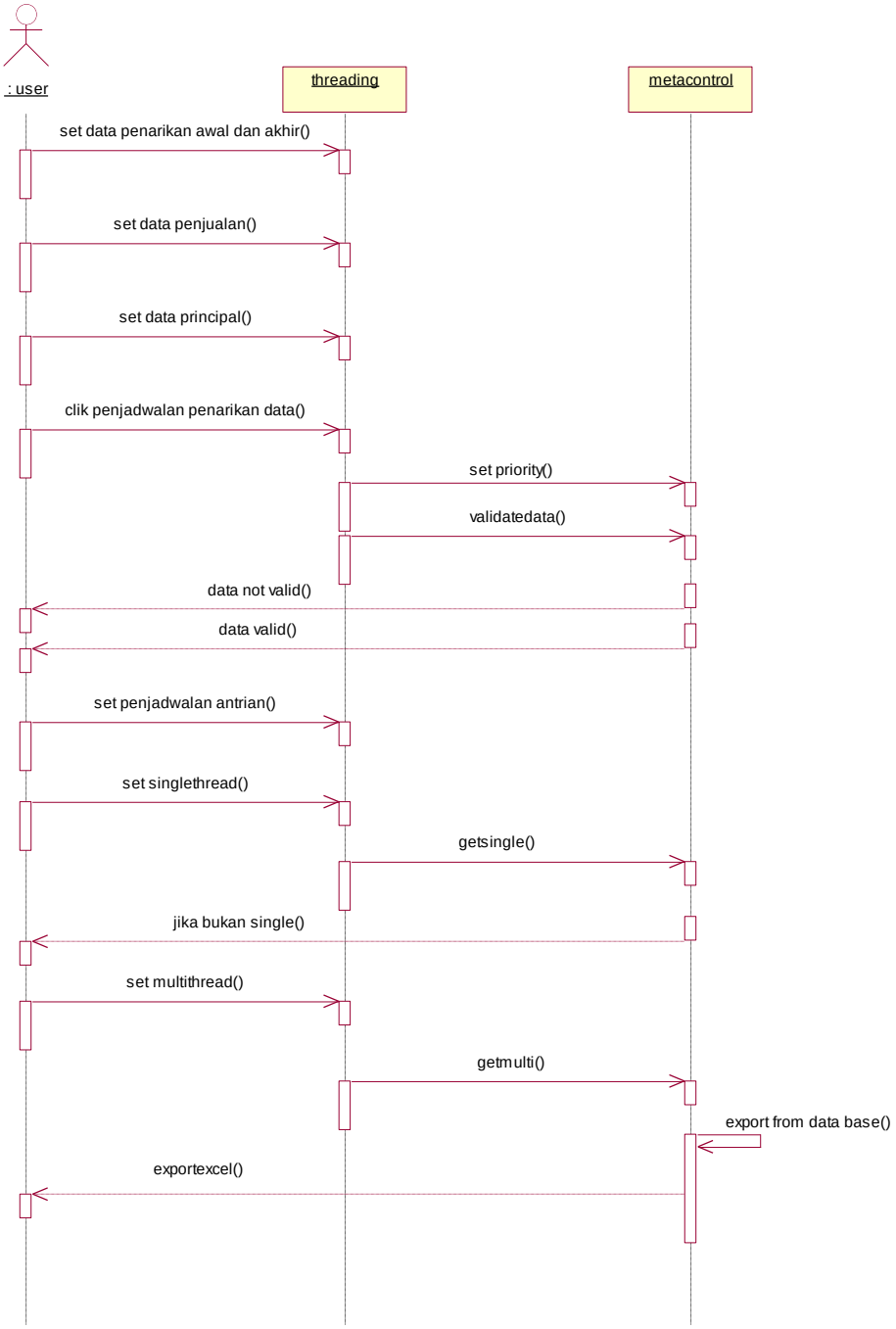
Gambar 3.12 *Sequence Add principal*

2. *Sequence Set priority*



Gambar 3.13 *Sequence Set priority*

3. *Sequence* Penjadwalan penarikan data

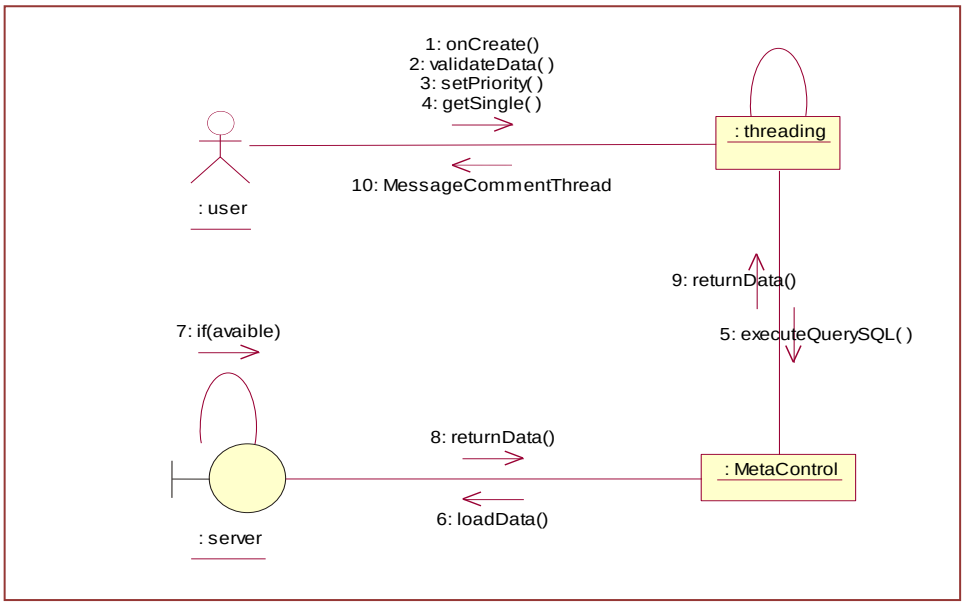


Gambar 3.14 *Sequence* Penjadwalan Penarikan Data

E. Collaboration Diagram

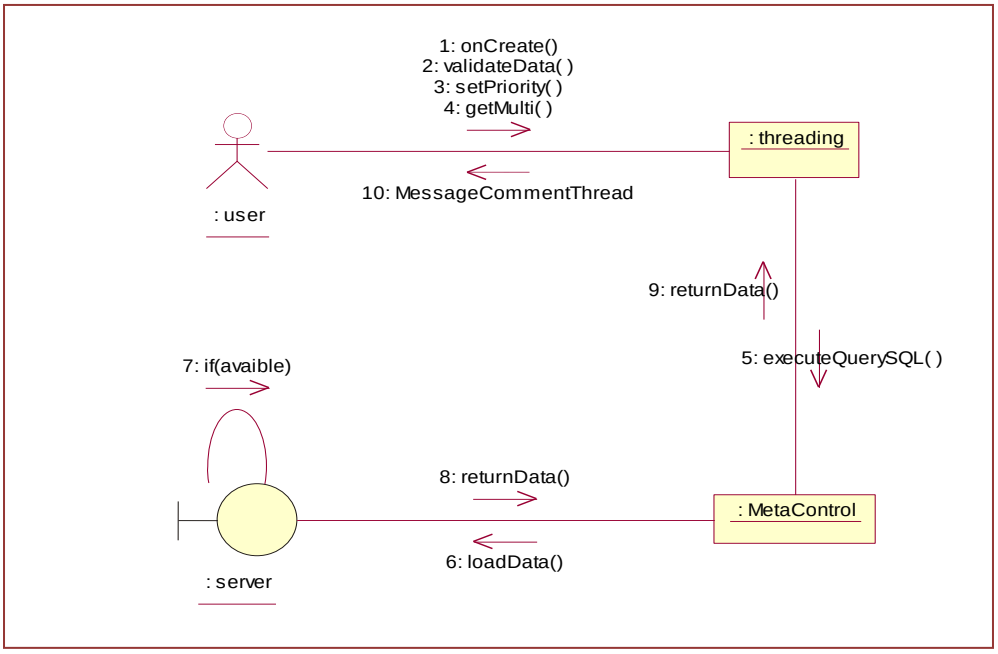
Pada tahap ini *Collaboration Diagram* dapat menggambarkan interaksi antar objek seperti *Sequence Diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap message memiliki *Sequence* number, di mana message dari level tertinggi memiliki nomor satu. Messages dari level yang sama memiliki prefiks yang sama.

1. *Collaboration Single Thread*



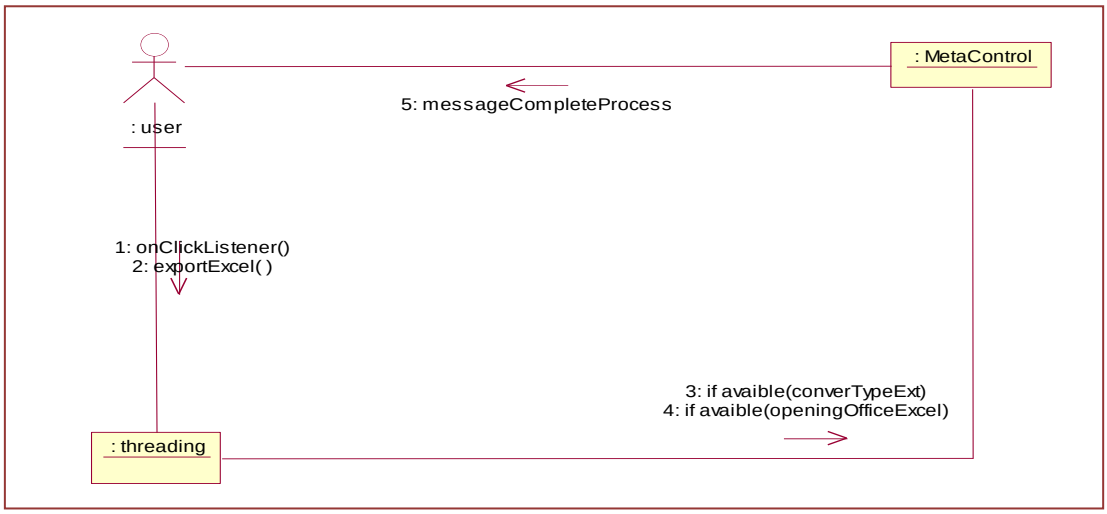
Gambar 3.15 *Collaboration Single*

2. *Collaboration Multithread*



Gambar 3.16 *Collaboration Multithread*

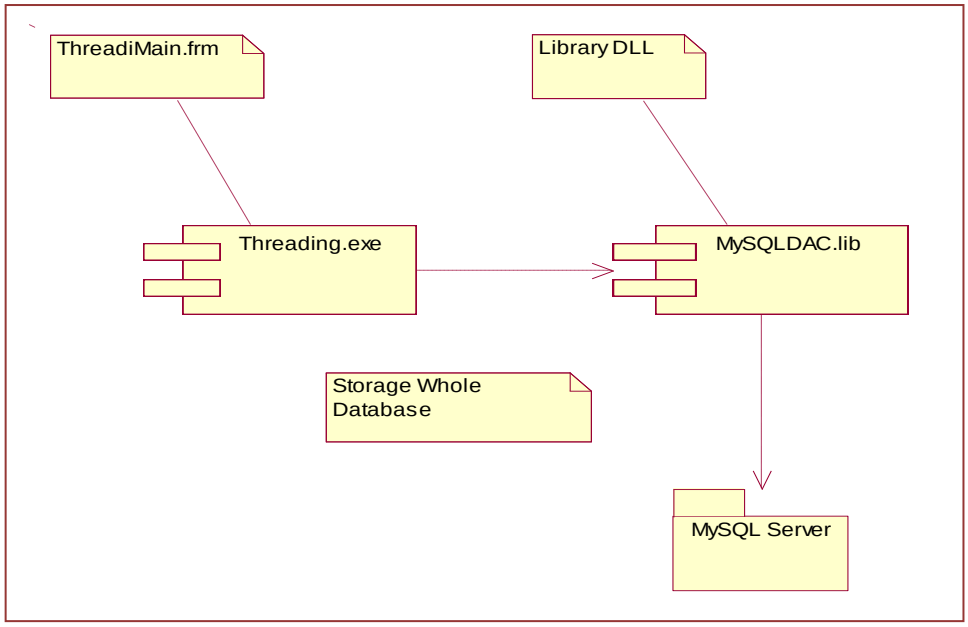
3. *Collaboration* Export Excel



Gambar 3.17 *Collaboration* Export

F. *Component* Diagram

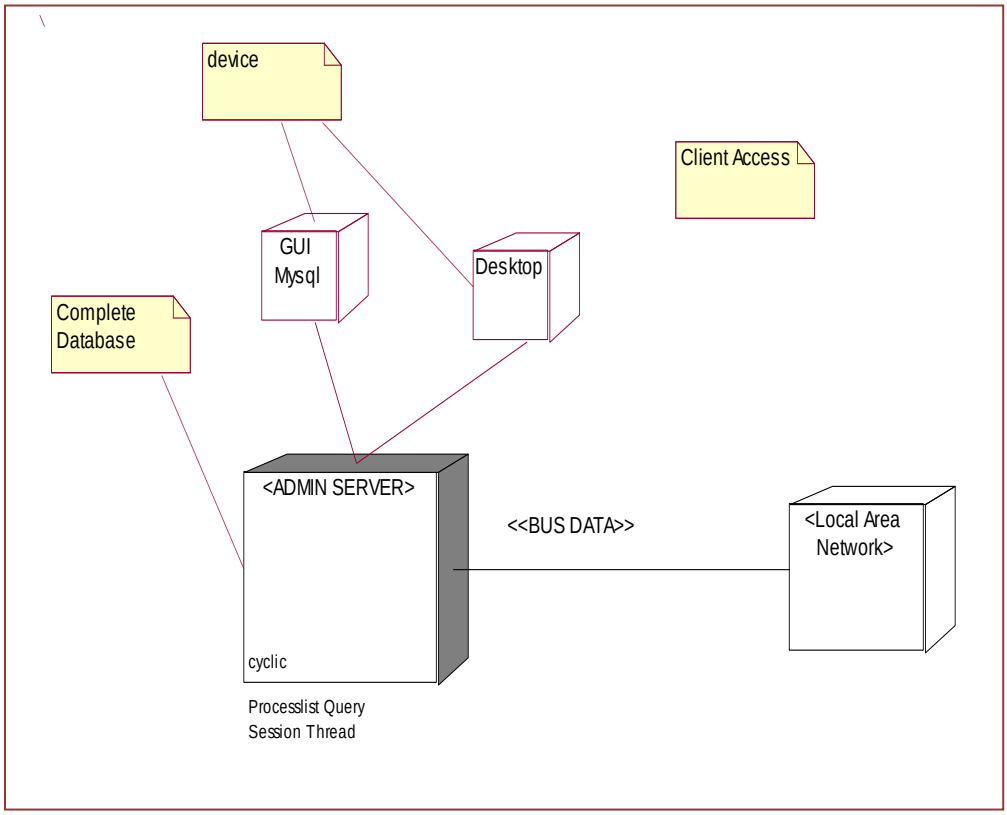
Pada tahap ini, *Component Diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak termasuk ketergantungan (*dependency*). Komponen piranti lunak yang yang dimaksud adalah modul yang berisi *source code* atau *binary code*, yang ada *library* atau *excutetabel* yang muncul pada *compile time*, *link time* ataupun pada *runtime*. *Component Diagram* ditunjukan pada gambar 3.20.



Gambar 3.18 *Component* Diagram

G. *Deployment* Diagram

Diagram*Deployment* menggambarkan rinci bagaimana komponen di-deploy dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin *Server* atau perangkat keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *Server* dan hal-hal yang lain bersifat fisikal.



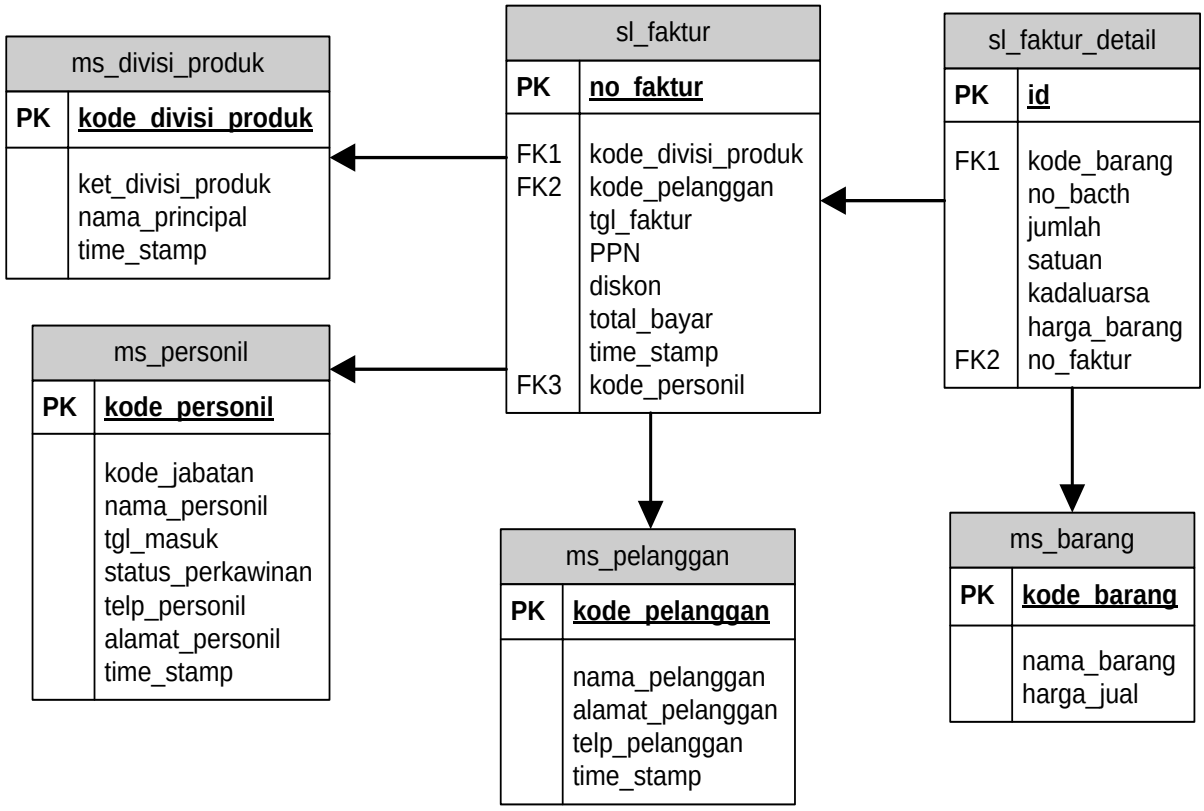
Gambar 3.19 *Deployment* Diagram

3.2 Perancangan sistem

Perancangan sistem adalah suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada fase analisis. Adapun tahapan yang dilakukan dalam perancangan sistem ini membahas mengenai tujuan perancangan sistem, dan arsitektur antar muka.

3.2.1 Diagram Relasi

Relasi antar tabel merupakan gabungan antar *file* yang mempunyai kunci utama yang sama, sehingga *file-file* tersebut menjadi satu kesatuan yang dihubungkan oleh *field* kunci (*Primary Key*). Pada proses ini elemen-elemen data dikelompokkan menjadi satu *file database* beserta *entitas* dan hubungannya. Diagram relasi ini dapat dilihat pada gambar 3.20.



Gambar 3.20 *Diagram Relasi*

3.2.2 Struktur Tabel

Perancangan struktur tabel adalah perancangan tabel-tabel yang akan digunakan pada *database*. Tabel-tabel yang terdapat dalam *database* yang digunakan dalam sistem informasi penilaian pegawai ini adalah:

1. Tabel sl_faktur

Tabel 3.8 Struktur Tabel Faktur

Nama Field	Tipe	Panjang	Extra	Kunci	Atribut
no_faktur	Varchar	10		*Primary Key*	Not Null
Kode_divisi_produk	Varchar	10		*Foreign Key*	Not Null
Kode_pelanggan	Varchar	50			Not Null
Tgl_faktur	Varchar	50			Not Null
PPN	Date				Not Null
Diskon	Varchar	20			
Total_bayar	Char				Not Null
Time_stamp	Varchar	10			Not Null
Kode_personil	Varchar	20			Not Null

2. Tabel ms_divisi_produk

Tabel 3.9 Struktur Tabel Divisi

Nama Field	Tipe	Panjang	Extra	Kunci	Atribut
Kode_divisi_produk	Varchar	10	*Auto Increment*	*Primary Key*	Not Null
Ket_divisi_produk	Varchar	50			Not Null
Nama_principal	Varchar	10		*Foreign Key*	Not Null
Time_stamp	Varchar	10		*Foreign Key*	Not Null

3. Tabel ms_personil

Tabel 3.10 Struktur Tabel Personil

Nama Field	Tipe	Panjang	Extra	Kunci	Atribut
Kode_personil	Varchar	10	*Auto Increment*	*Primary Key*	Not Null
Kode_jabatan	Varchar	50			Not Null
Nama_personil	Varchar	200			Not Null
Tgl_masuk	Varchar	150			Not Null
Status_perkawinan	Number				Not Null
Telp_personil	Varchar	150			Not Null
Alamat_personil	Varchar	50			Not Null
Time_stamp	Enum	10			Not Null

4. Tabel ms_pelanggan

Tabel 3.11 Struktur Tabel Penilaian

Nama Field	Tipe	Panjang	Extra	Kunci	Atribut
Kode_pelanggan	Varchar	10	*Auto Increment*	*Primary Key*	Not Null
Nama_pelanggan	Varchar	15		*Foreign Key*	Not Null
Alamat_ pelanggan	Number				Not Null
Telp_ pelanggan	Date				
Time_stamp	Varchar	20			Not Null

5. Tabel sl_faktur_detail

Tabel 3.12 Struktur Faktur Detail

Nama Field	Tipe	Panjang	Extra	Kunci	Atribut
Kode_barang	Varchar	10	*Auto Increment*	*Primary Key*	Not Null
No_faktur	Varchar	50			Not Null
No_bacht	Varchar	200			Not Null
Jumlah	Varchar	150			Not Null
Satuan	Number				Not Null
Kadaluarsa	Varchar	150			Not Null
Harga_barang	Varchar	50			Not Null

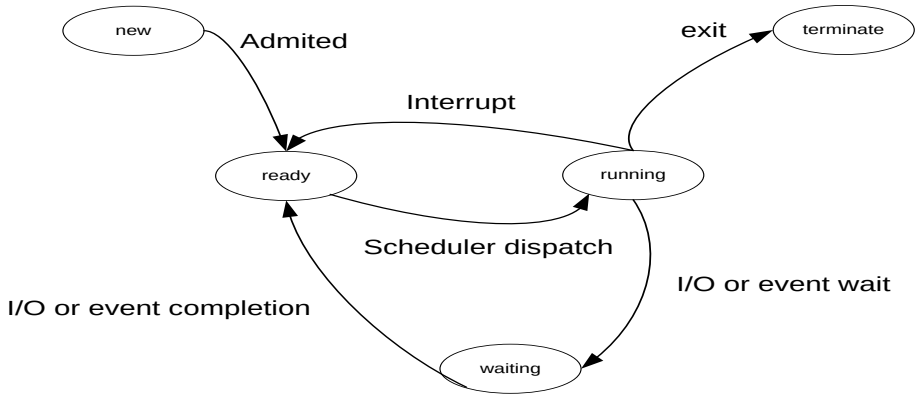
6. Tabel ms_barang

Tabel 3.13 Barang

Nama Field	Tipe	Panjang	Extra	Kunci	Atribut
Kode_barang	Varchar	10	*Auto Increment*	*Foreign Key*	Not Null
Harga_jual	Varchar	10	*Auto Increment*	*Foreign Key*	Not Null
Nama_barang	Number	20			Not Null

3.2.3 Perancangan pesan

Proses perancangan pesan adalah suatu proses yang terjadi dalam pesan dari awal pembuatan pesan sampai menjadi laporan setelah dibuat. Status yang dimiliki sebelum pesan terkirim yaitu *new*, *ready*, *running*, *waiting* dan *terminate*. Proses pengiriman pesan mempuyai beberapa tahap *new* awal pembuatan pesan, *ready* keadaan pesan yang siap untuk dijalankan, *running* keadaan pesan yang berjalan, *waiting* keadaan pesan menunggu dan *terminate* proses selesai dikirim. Berikut ini adalah contoh yang apa yang terjadi didalam sebuah proses pembuatan sebuah pesan seperti gambar 3.21.

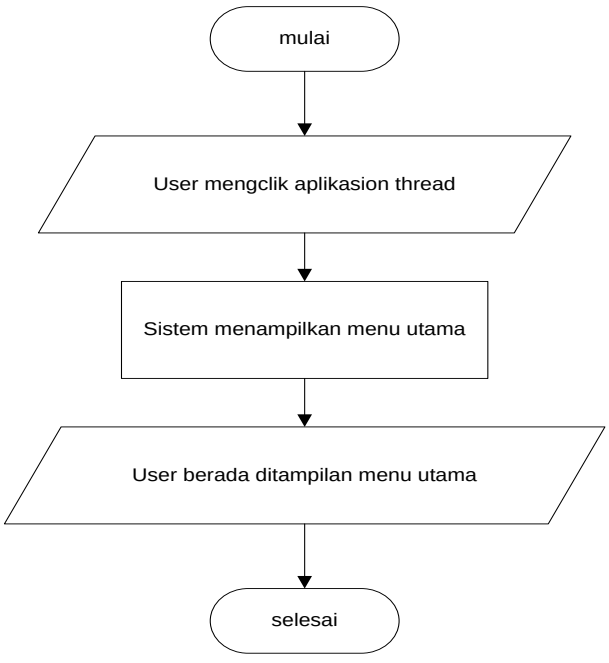


Gambar 3.21 Proses pengiriman pesan

3.2.4 Perancangan Method

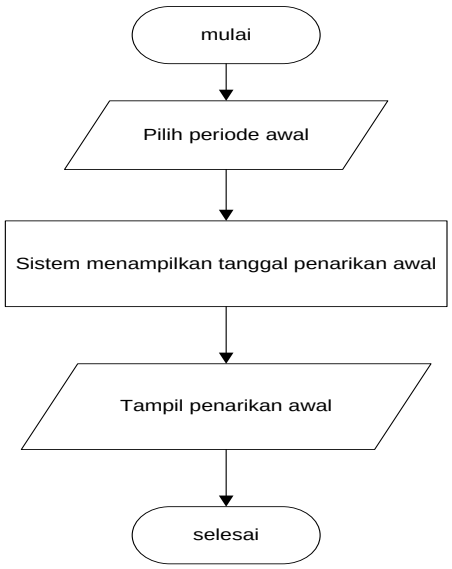
Perancangan *method* menjelaskan penjabaran alur proses-proses yang ada pada sistem yang dibangun menggunakan diagram *flowchart*. Adapun perancangan *method* untuk aplikasi yang akan dibangun yaitu:

1. Method Tampil Menu



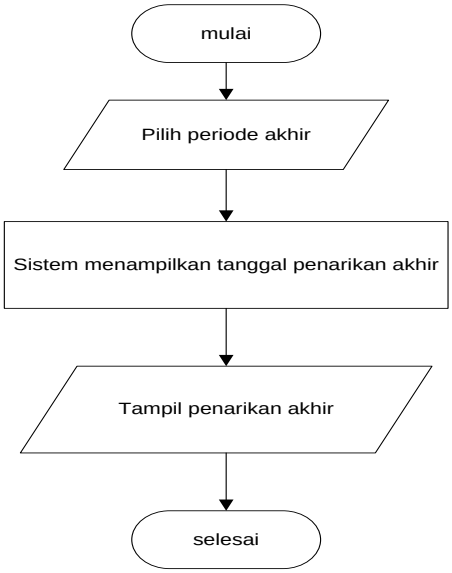
Gambar 3.22 Method Tampil Menu

2. Method periode awal



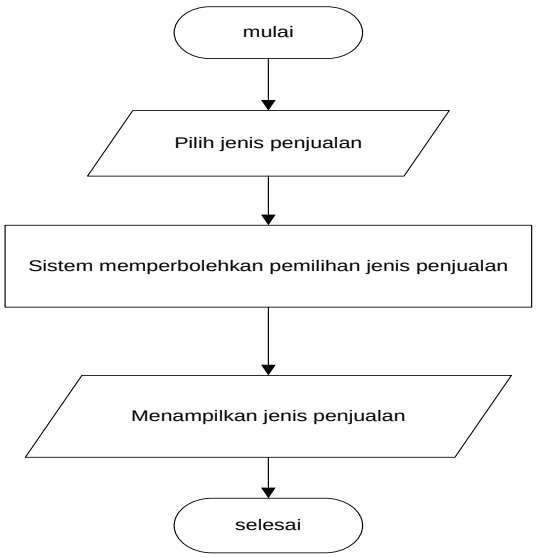
Gambar 3.23 Method periode awal

3. Method periode akhir



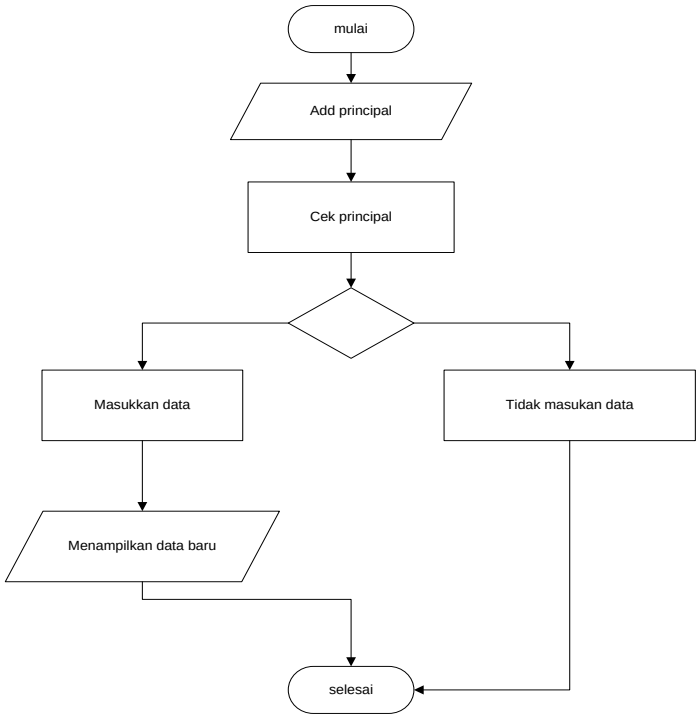
Gambar 3.24 Method periode awal

5. Method jenis penjualan



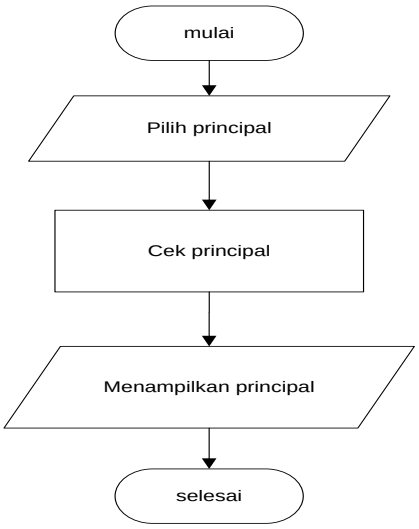
Gambar 3.25 Method jenis penjualan

6. Method add principal



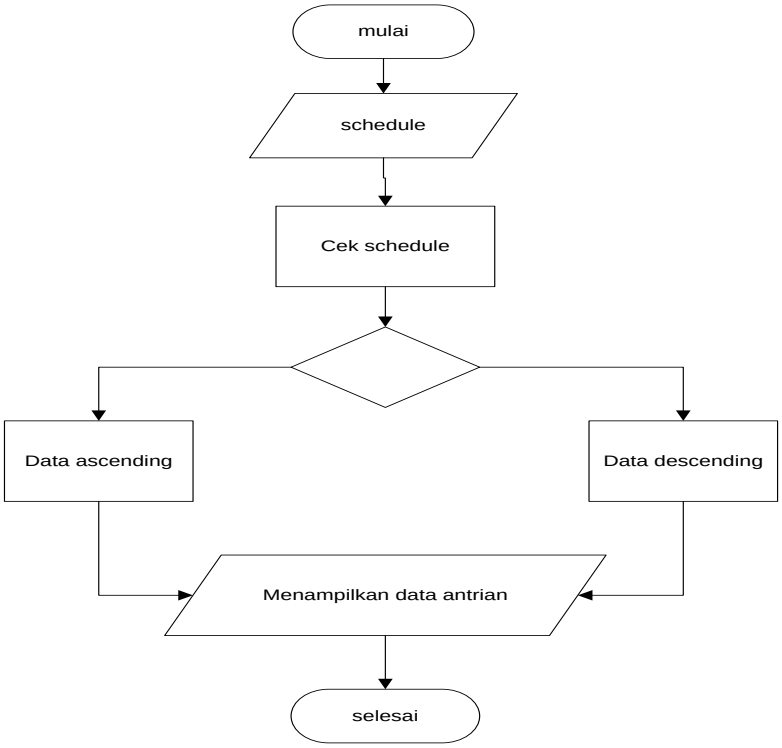
Gambar 3.26 Method add principal

7. Method principal



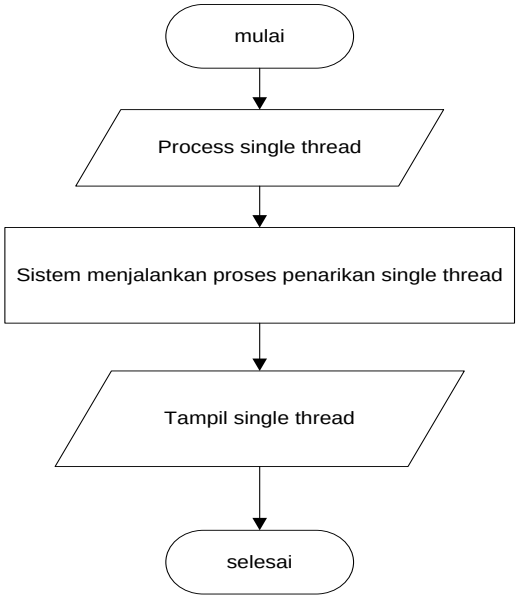
Gambar 3.27 Method principal

8. Method schedule



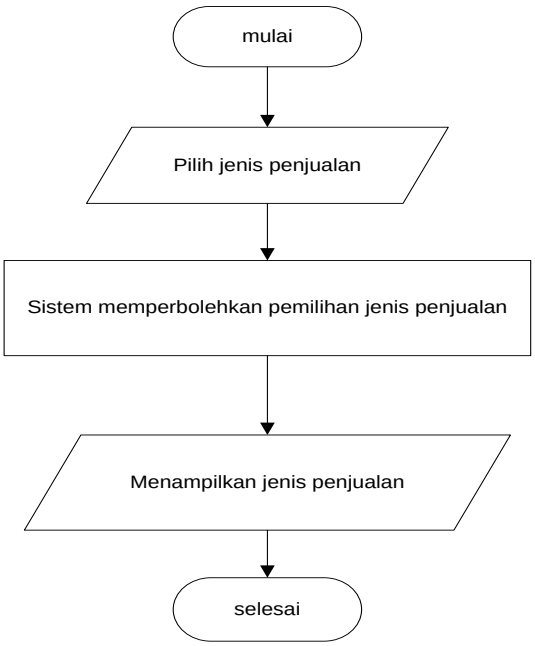
Gambar 3.28 Method schedule

9. Method process single thread



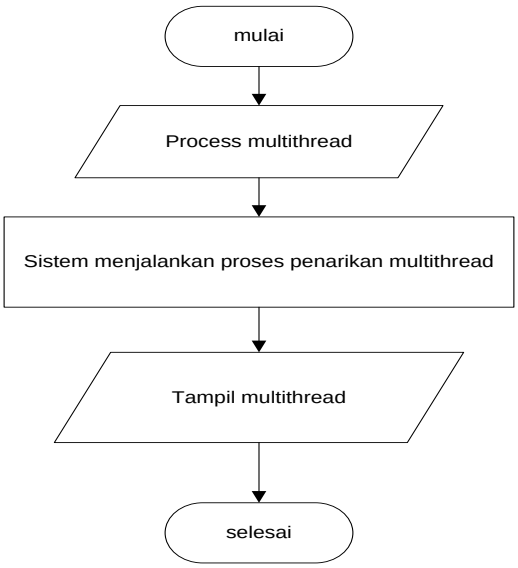
Gambar 3.29 Method process single thread

10. Method process multithread



Gambar 3.30 Method Tampil Menu

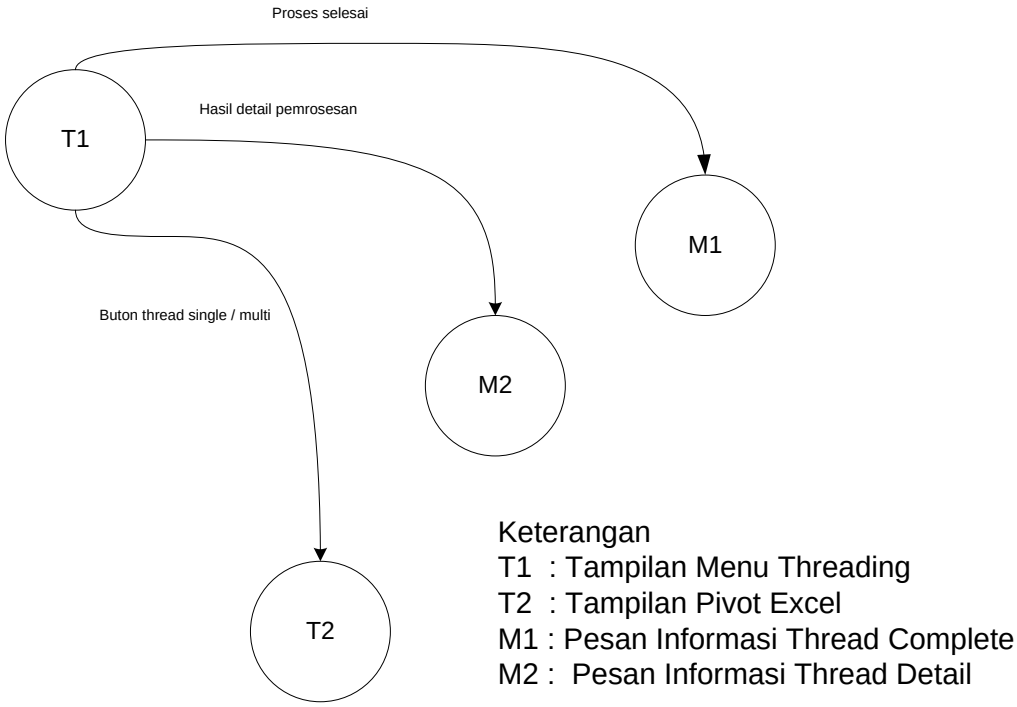
11. Method Eksekusi Data



Gambar 3.31 Metode Eksekusi Data

3.2.5 Jaringan Semantik

Jaringan semantik adalah gambaran pengetahuan grafis yang menunjukkan hubungan antar berbagai objek, terdiri dari lingkaran-lingkaran yang dihubungkan dengan anak panah yang menunjukkan objek dan informasi tentang objek-objek tersebut.Jaringan semantik yang terjadi dalam penarikan data adalah seperti gambar 3.32.

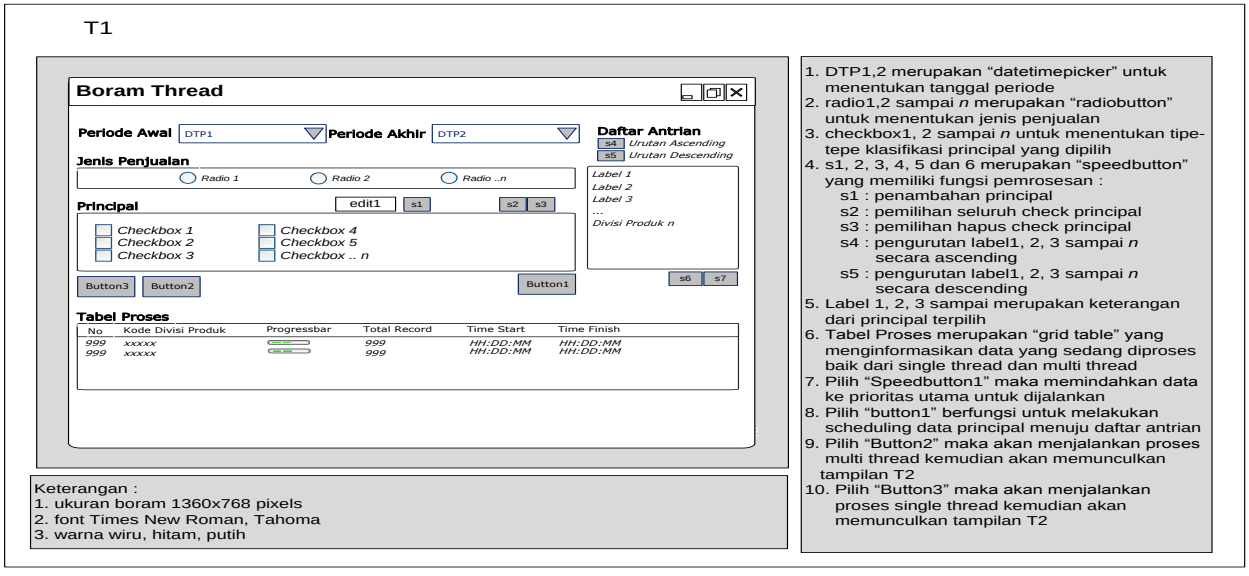


Gambar 3.32 Jaringan semantik Penarikan pesan

3.2.6 Arsitektur Antar Muka

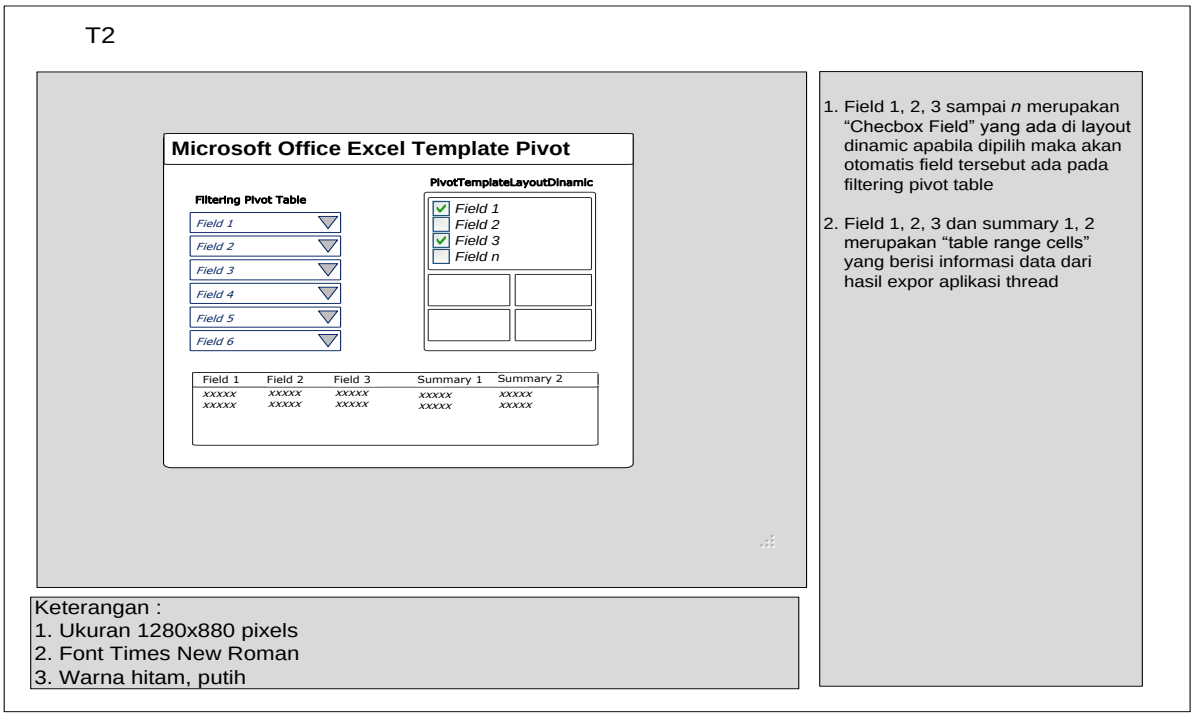
Arsitektur antarmuka dibutuhkan untuk mewakili keadaan sebenarnya dari aplikasi yang akan dibangun, berikut akan disajikan perancangan antarmuka dari aplikasi yang akan dibangun:

1. Tampilan Menu aplikasi penarikan data



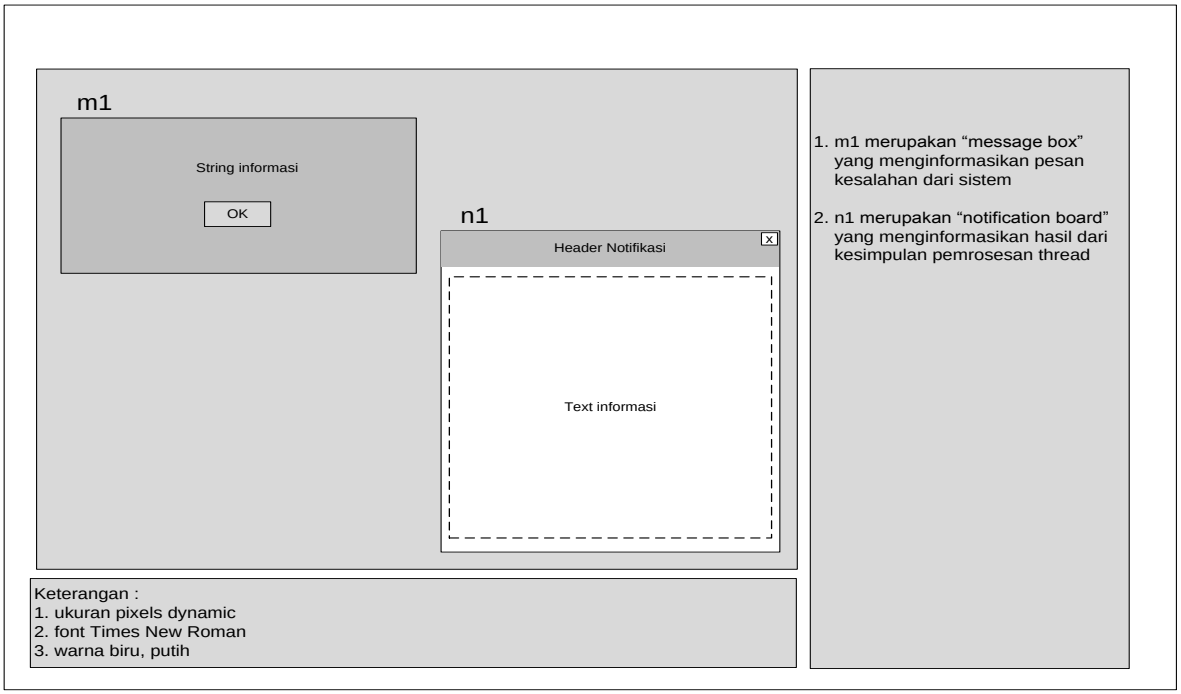
Gambar 3.33 Tampilan Menu aplikasi penarikan data

2. Tampilan Informasi data



Gambar 3.34 Tampilan informasi data

3. Tampil Informasi pesan



Gambar 3.35 Tampilan informasi data