BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

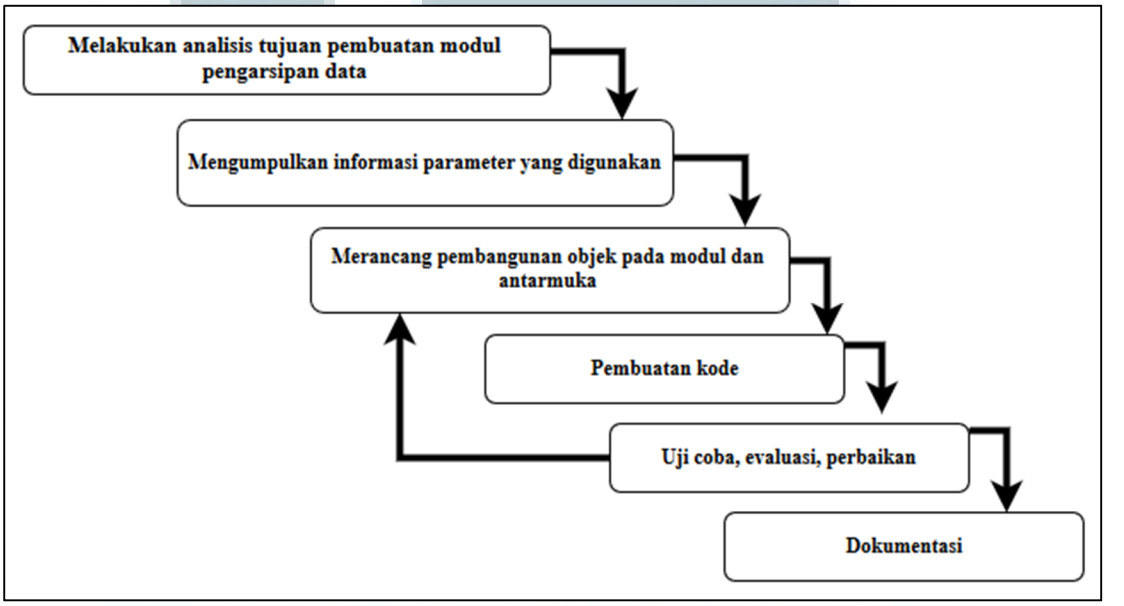
Kerja magang dilakukan pada Departemen Production, Research, and Development Center (PRDC), di bawah bimbingan Bapak Josua Napitupulu selaku Department Head PRDC. Kemudian, dalam pengembangan aplikasi dibantu oleh Muchammad Ikhsan Ariya Girinata selaku Project Leader bidang PSAK.

3.2 Tugas yang Dilakukan

Membangun modul pengarsipan data yang akan digunakan pada aplikasi Anabatic CR-One, PT Anabatic Technologies Tbk.

3.3 Uraian Pelaksanaan Kerja Magang

3.3.1 Proses Pelaksanaan



Gambar 3.1 Siklus Pengembangan Modul

Secara garis besar, modul pengarsipan data dikembangkan dengan menggunakan siklus *waterfall* karena model pengembangan piranti lunak tersebut merupakan model yang paling mudah dimengerti dan diimplementasikan (Munassar dan Govardhan, 2010). Implementasi model *waterfall* pada siklus pengembangan modul pengarsipan data aplikasi Anabatic CR-One diilustrasikan dengan Gambar 3.1. Adapun uraian mengenai siklus tersebut adalah sebagai berikut.

A. Melakukan Analisis Tujuan Pembuatan Modul Pengarsipan Data

Sebelum modul dibuat, analisis terhadap pembuatan modul harus dilakukan terlebih dahulu agar tujuan yang ingin dicapai dapat diketahui. Hal ini bertujuan agar pembangunan modul dapat terarah dan memberikan hasil yang maksimal. Informasi ini diperoleh dengan mewawancarai Muchammad Ikshan Ariya Girinata selaku Project Leader proyek PSAK. Proses bisnis peminjaman uang yang ditangani oleh aplikasi Anabatic CR-One akan menghasilkan ratusan bahkan ribuan data baru setiap terjadi transaksi. Namun, tidak semua data tersebut digunakan pada proses bisnis sehari-hari. Bahkan, terdapat banyak data yang sudah tidak digunakan karena sudah diperbarui oleh data baru, tetapi masih tersimpan dalam sistem basis data aplikasi.

Walaupun sudah digantikan oleh data baru, aplikasi tetap harus menyimpan data lama untuk menjaga integritas dan konsistensi dari laporan yang dihasilkan. Hal ini menyebabkan penumpukan data dan peningkatan beban muatan pada sistem basis data utama aplikasi. Oleh karena itu, modul ini bertujuan untuk meringankan beban jumlah data pada sistem basis data utama aplikasi Anabatic CR-One dengan memindahkan data-data yang sudah tidak digunakan tersebut ke sistem basis data lain.

B. Mengumpulkan Informasi Parameter yang Digunakan

Proses pengumpulan informasi mengenai parameter yang digunakan pada modul pengarsipan data dilakukan dengan cara mewawancarai Muchammad Ikhsan Ariya Girinata. Adapun parameter yang dibutuhkan yakni.

1) Nama data yang ingin diarsip, seperti Loan Cash Flow, Loan Amortization Result, dan Loan Fee Cost Amortization Result.
2) Tanggal batas kedaluwarsa data, digunakan untuk melakukan penyaringan pada data yang sudah kedaluwarsa di dalam sistem basis data utama aplikasi.

C. Merancang Perancangan Objek, Persiapan Struktur Table, dan Pembangunan Antarmuka

Setelah informasi mengenai tujuan pembuatan modul dan parameter yang dibutuhkan diketahui, pembangunan modul dilanjutkan ke tahap perancangan objek, persiapan struktur *table* History, dan pembuatan desain antarmuka. Namun, sebelum memasuki tahap perancangan, adapun gambaran umum mengenai objek-objek yang digunakan pada proyek pengembangan aplikasi Anabatic CR-One ditunjukkan pada Gambar 3.2.

Pengembangan aplikasi Anabatic CR-One menggunakan konsep Spring Model-View-Controller (MVC) yang diimplementasikan dalam *framework* Spring. Adapun penjelasan dari tiap-tiap komponen, yaitu sebagai berikut.

1) View, yaitu objek yang merepresentasikan halaman antarmuka yang ditampilkan kepada pengguna. Diaplikasikan dalam bentuk JavaServer Page (JSP).
2) Controller, yaitu objek yang berperan sebagai pengatur (*controller*) dalam konsep MVC, bertugas melakukan pemetaan terhadap *request* yang dilakukan

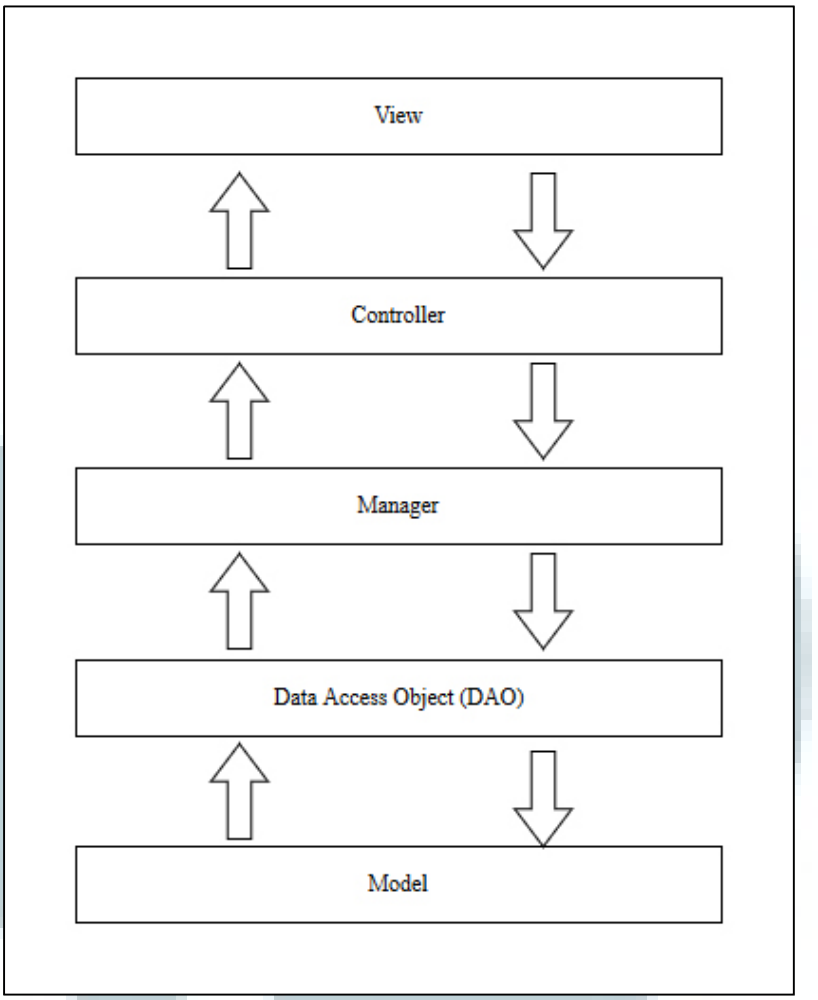
pengguna ke fungsi yang tepat sehingga aplikasi dapat menampilkan halaman antarmuka sesuai permintaan pengguna.

3) Model, yaitu objek yang merepresentasikan struktur fisik dari *table* pada sistem basis data. Berisi atribut-atribut yang mewakili *field* dalam *table* beserta fungsi *setter-getter* dari tiap atribut.

4) Data Access Object (DAO), merupakan layer yang berisi berbagai *query* untuk mengakses data dari objek Model yang diinginkan. Pada umumnya, layer ini digunakan untuk membuat fungsi Create, Read, Update, Delete (CRUD) terhadap suatu objek Model.

5) Manager, atau dikenal sebagai Service, adalah layer yang menghubungkan komponen-komponen dalam aplikasi, terutama Controller, dengan objek DAO. Objek ini berisi operasi-operasi independen terhadap objek Model yang digunakan (Evans, 2003). Salah satu operasi yang terdapat pada objek Manager dalam pengembangan aplikasi Anabatic CR-One adalah konversi data dari objek Model menjadi objek temporer yang akan digunakan oleh Controller untuk menampilkan data pada View.

Objek Manager dan DAO diaplikasikan dengan suatu *interface* sehingga pada objek ini hanya dilakukan pendeklarasian fungsi. Tiap-tiap fungsi didefinisikan pada *class* yang mengimplementasi *interface* tersebut. Komponen lain yang ingin menggunakan fungsi pada DAO ataupun Manager harus melalui *interface* yang bersangkutan. Setiap Manager ataupun DAO akan memiliki minimal satu buah *class* yang mengimplementasikan *interface* tersebut.



Gambar 3.2 Objek-Objek pada konsep Spring MVC

C.1 Perancangan Objek

Perancangan objek dilakukan dengan membuat lima diagram Unified Modeling Language (UML), yaitu *activity diagram, use case diagram, class diagram, sequence diagram,* dan *collaboration diagram*. Melalui kelima diagram ini, daftar objek yang dibutuhkan modul dan relasi antar objek tersebut dapat diketahui.

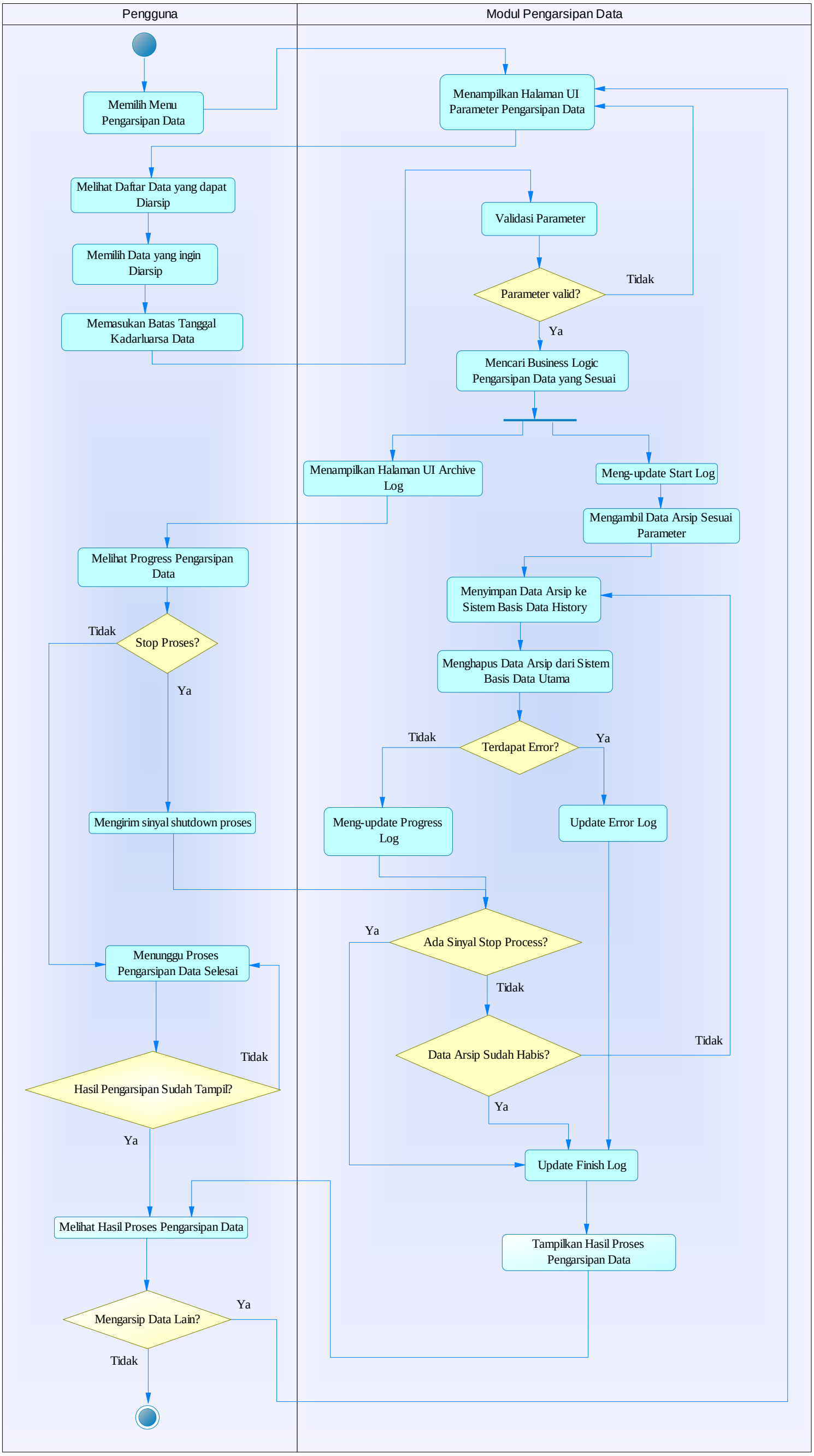
C.1.1 Activity Diagram

Actvitiy diagram dimulai ketika pengguna memilih menu pengarsipan data pada aplikasi Anabatic CR-One. Modul Pengarsipan data akan menampikan halaman antarmuka yang menampilkan daftar data yang dapat diarsip dan kolom untuk memasukkan parameter batas tanggal kedaluwarsa data. Kedua komponen ini adalah parameter yang dibutuhkan oleh modul dalam mengarisp data. Setelah pengguna memasukkan data parameter yang dibutuhkan, modul akan melakukan validasi terhadap parameter tersebut. Jika parameter tidak valid, maka modul akan kembali mengarahkan ke halaman awal antarmuka, tempat pengguna memasukkan parameter tersebut. Hal ini akan terus dilakukan sampai didapatkan parameter yang valid.

Parameter yang valid akan digunakan oleh modul untuk melanjutkan proses pengarsipan data dengan mengambil objek Business Logic sesuai parameter. Setelah itu, proses dibagi menjadi dua yang berjalan secara bersamaan, yaitu membuka halaman Log dan melakukan proses pengarsipan data. Hal ini bertujuan agar pengguna dapat melihat perkembangan atau *progress* kerja dari modul ketika mengarsip data.

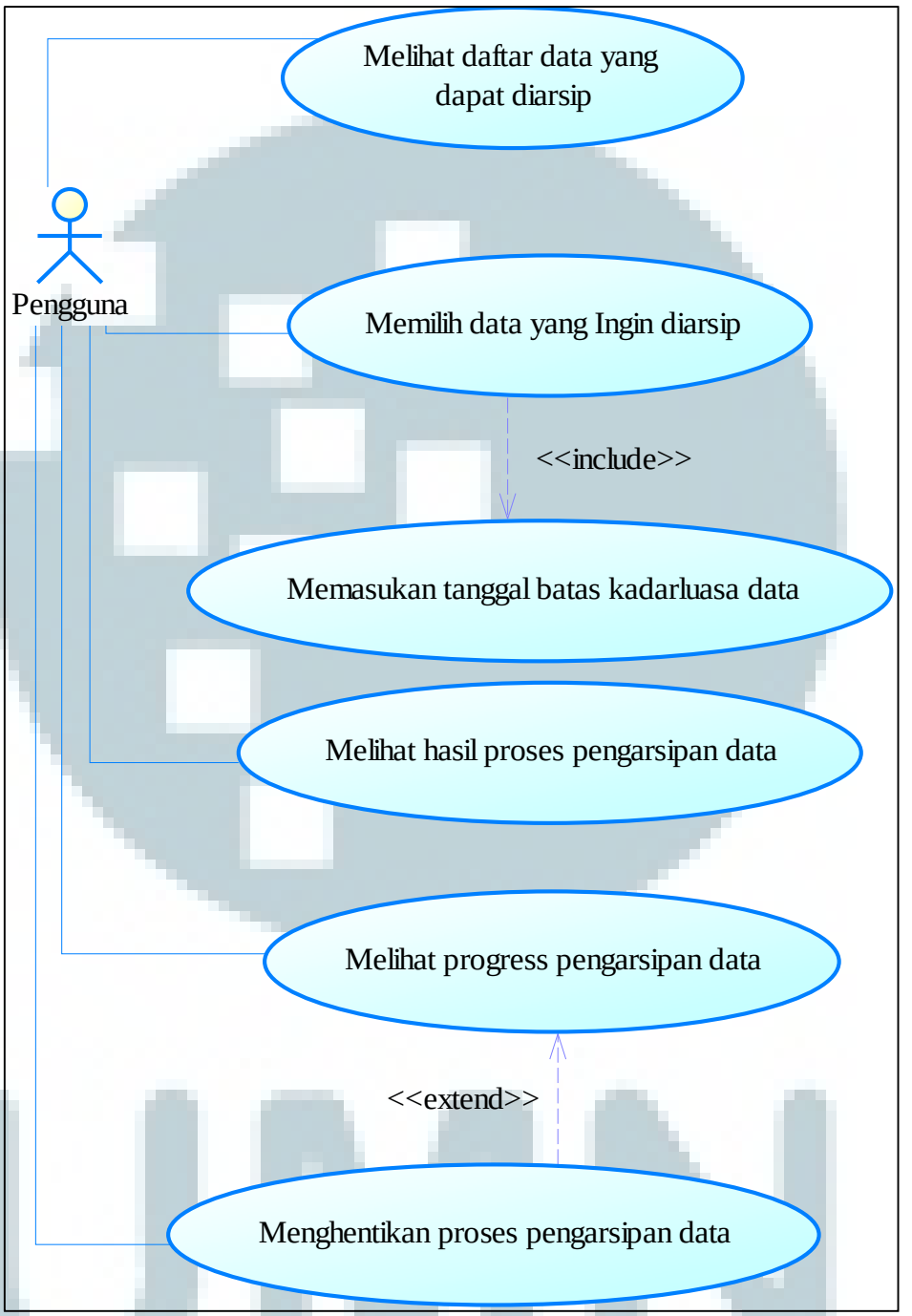
Proses pengarsipan diawali dengan meng-*update* Start Log untuk menunjukan bahwa proses pengasipan akan dimulai. Kemudian, proses dilanjutkan dengan mengambil data arsip sesuai parameter yang diberikan. Tiap-tiap data tersebut akan dimasukkan ke dalam sistem basis data History dan dihapus dari sistem basis data utama aplikasi Anabatic CR-One. Setiap proses pemindahan akan dilakukan *update* pada Progress Log. Hal ini bertujuan agar pengguna dapat melihat perkembangan dari proses kerja modul pengarsipan data. Proses pemindahan ini akan dilakukan secara terus-menerus selama masih terdapat data yang dapat diarsip dan pengguna tidak menghentikan proses pengarsipan data secara manual.

Pengguna yang tidak ingin menghentikan proses pengarsipan secara manual, dapat menunggu hingga hasil proses pengarsipan data ditampilkan. Ketika data selesai diarsip, maka modul akan meng-*update* Finish Log dan menampilkan hasil dari proses pengarsipan. Hasil yang ditampilkan berupa jumlah data yang berhasil diproses, total waktu pengarsipan data, dan jumlah kesalahan atau *error* yang terjadi ketika proses berlangsung. Modul juga akan menampilkan pilihan untuk kembali ke halaman utama pengarsipan data apabila pengguna ingin mengarsip data yang berbeda. Jika terdapat kesalahan pada saat melakukan pengarsipan, maka modul akan menghentikan proses dan meng-*update* Finish Log. Keseluruhan *activity diagram* modul pengarsipan data ditunjukkan pada Gambar 3.3.



Gambar 3.3 *Activity Diagram* Modul Pengarsipan Data

C.1.2 Use Case Diagram



Gambar 3.4 *Use Case Diagram* Modul Pengarsipan Data

Use case diagram pada Gambar 3.4 menunjukan proses-proses terkomputerisasi yang dilihat dari sudut pandang pengguna berdasarkan *activity diagram*. Pengguna dapat memasukan parameter yang modul butuhkan, yaitu memilih data yang ingin diarsip dan menentukan tanggal batas kedaluwarsa data. Perkembangan atau *progress* kerja dari modul saat melakukan pengarsipan data dapat dilihat oleh pengguna. Saat melihat *progress* kerja modul, pengguna juga dapat menghentikan proses pengarsipan data secara manual. Setelah data selesai diarsip, maka pengguna dapat melihat hasil dari proses pengarsipan data.

C.1.3 Sequence dan Collaboration Diagram

Sequence diagram menunjukan aktivitas modul dilihat dari sudut pandang urutan waktu, sedangkan *collaboration diagram* berfokus pada relasi yang terjadi antara objek yang satu dengan objek lainnya (Hoffer, 2007). Kedua diagram ini memiliki skenario aktivitas atau proses yang sama dan merupakan jenis dari *interaction diagram*. Terdapat tiga aktivitas utama dari modul pengarsipan data, yaitu validasi masukan pengguna, pengarsipan data, dan pemberhentian pengarsipan data secara manual.

1) Sequence dan Collaboration Diagram Proses Validasi Masukan Pengguna

Sequence diagram proses validasi masukan pengguna ditunjukkan pada Gambar 3.5. Adapun uraian skenario dari *sequence diagram* tersebut, yaitu sebagai berikut.

(1) Ketika pengguna memilih menu Archiving, maka Controller akan membentuk suatu objek *form* yang dapat digunakan untuk memasukan parameter pengarsipan data.

(2) Controller menampilkan halaman utama modul pengarsipan data.

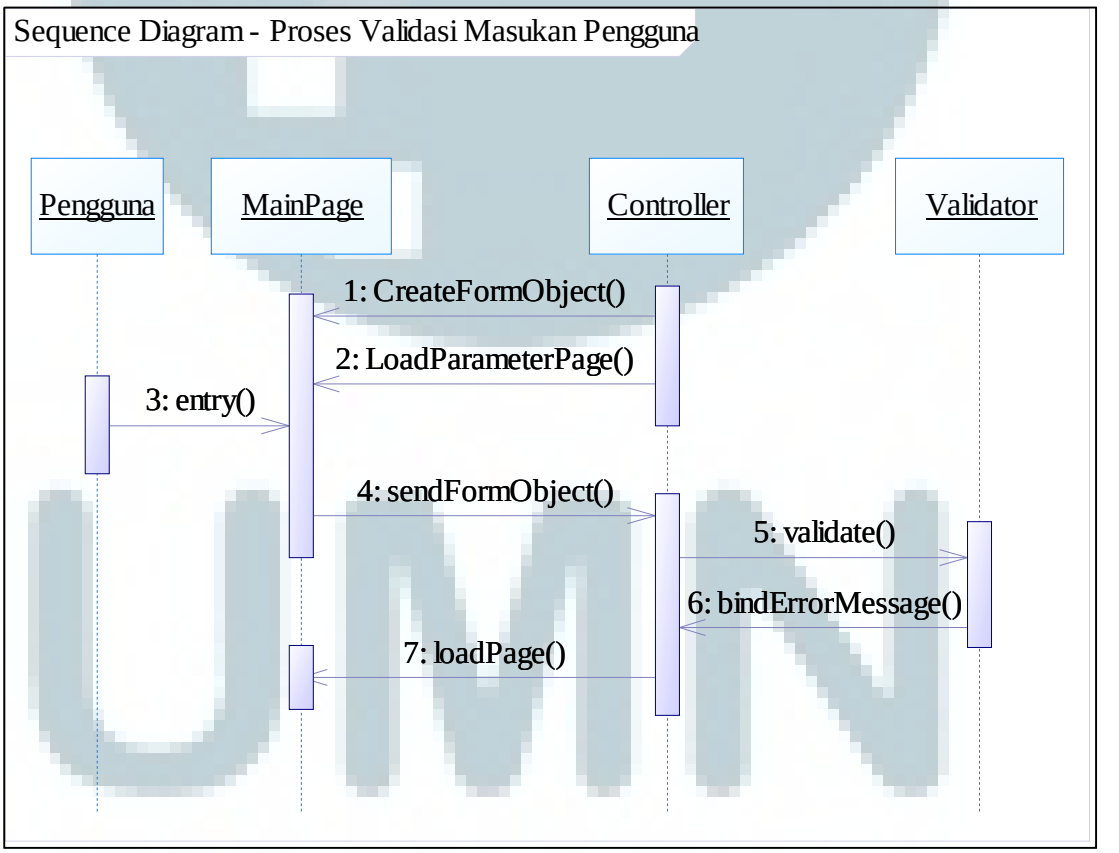
(3) Pengguna memasukan parameter yang dibutuhkan modul.

(4) Halaman utama mengirimkan objek yang berisi masukan dari pengguna ke Controller.

(5) Controller mengirimkan objek tersebut ke Validator untuk dilakukan validasi terhadap masukan pengguna.

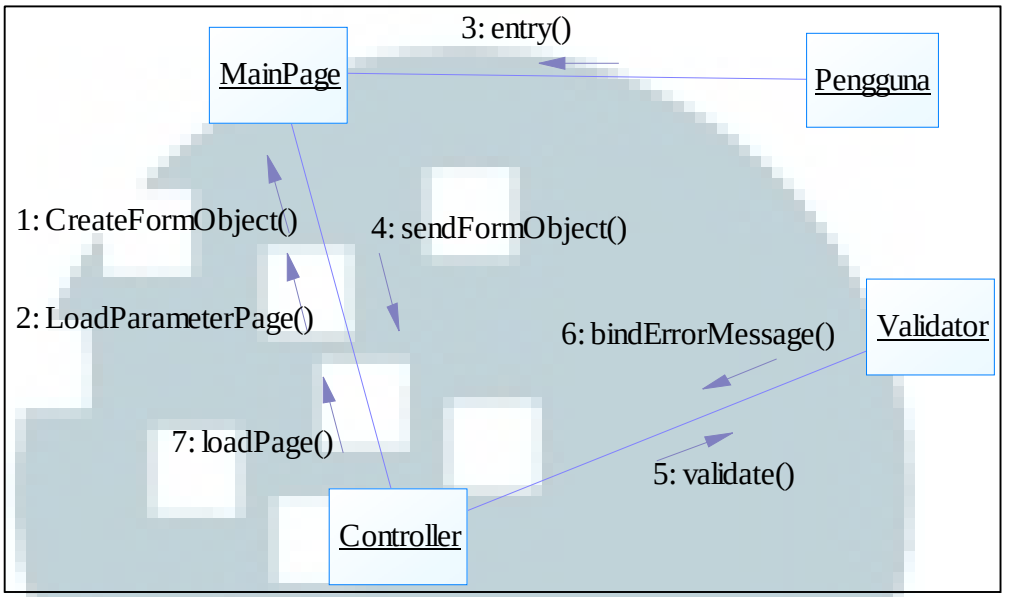
(6) Validator mengembalikan hasil validasi ke Controller beserta pesan *error* dari tiap masukan.

(7) Controller akan menampilkan kembali data masukan pengguna beserta pesan *error* dari tiap masukan ke halaman utama modul.



Gambar 3.5 *Sequence Diagram* Proses Validasi Masukan Pengguna

Adapun *collaboration diagram* yang dihasilkan berdasarkan skenario proses validasi masukan pengguna. Diagram tersebut ditunjukkan pada Gambar 3.6.



Gambar 3.6 *Collaboration Diagram* Proses Validasi Masukan Pengguna

2) Sequence dan Collaboration Diagram Proses Pengarsipan Data

Sequence diagram proses pengarsipan data ditunjukkan pada Gambar 3.7. Adapun uraian skenario dari *sequence diagram* tersebut, yaitu sebagai berikut.

(1) Setelah mendapat parameter yang valid, Controller akan meminta objek Business Logic pada Business Logic Factory sesuai parameter yang diterima.

(2) Controller memanggil fungsi Archive yang dimiliki oleh objek Business Logic.

(3) Controller menampilkan halaman Log, lalu Business Logic meng-*update* Start Log yang menunjukan bahwa proses pengarsipan akan dimulai.

(4) Business Logic akan meminta objek Table Manager dari Table Manager Factory dan objek History Manager dari History Manager Factory.

(5) Table Manager Factory dan History Manager factory akan mengembalikan objek Manager yang sesuai dengan parameter.

(6) Business Logic akan mengambil data arsip berdasarkan parameter tanggal kedaluwarsa dari objek Table Manager.

(7) Table Manager akan mengembalikan data arsip ke Business Logic.

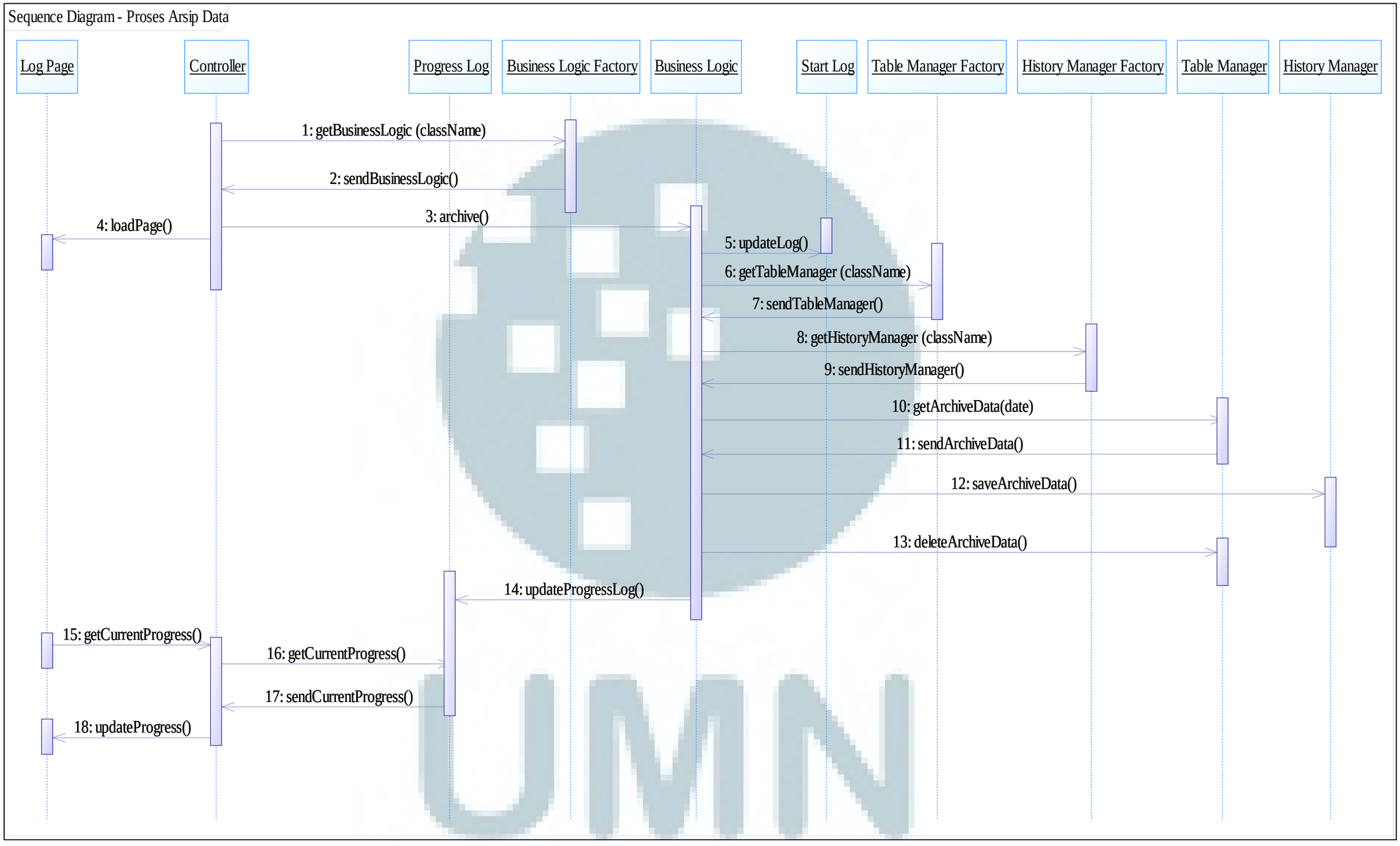
(8) Business Logic mengirimkan data arsip tersebut ke History Manager untuk disimpan ke dalam sistem basis data History.

(9) Business Logic menghapus data arsip dari sistem basis data utama aplikasi melalui Table Manager.

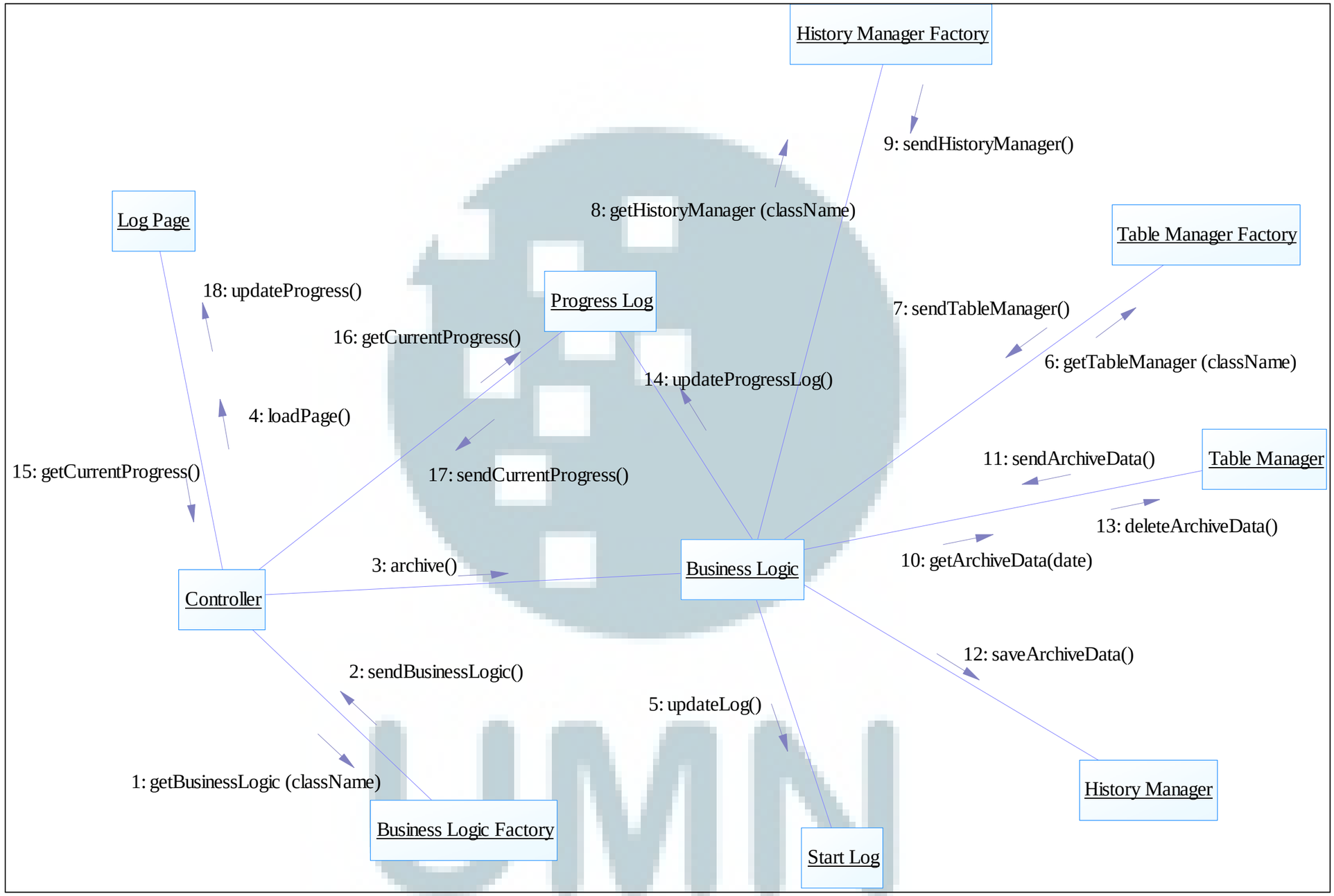
(10) Business Logic memperbarui jumlah data yang sudah diproses pada Progress Log.

(11) Halaman Log akan meminta Controller untuk mengirimkan data perkembangan kerja proses pengarsipan data dari Log Manager. Setelah itu, Controller akan mengembalikan data tersebut ke halaman Log untuk dilakukan *update* pada proses perkembangan kerja modul.

Adapun relasi-relasi yang terjadi antar objek berdasarkan skenario proses pengarsipan data direalisasikan ke dalam *collaboration diagram* pada Gambar 3.8.

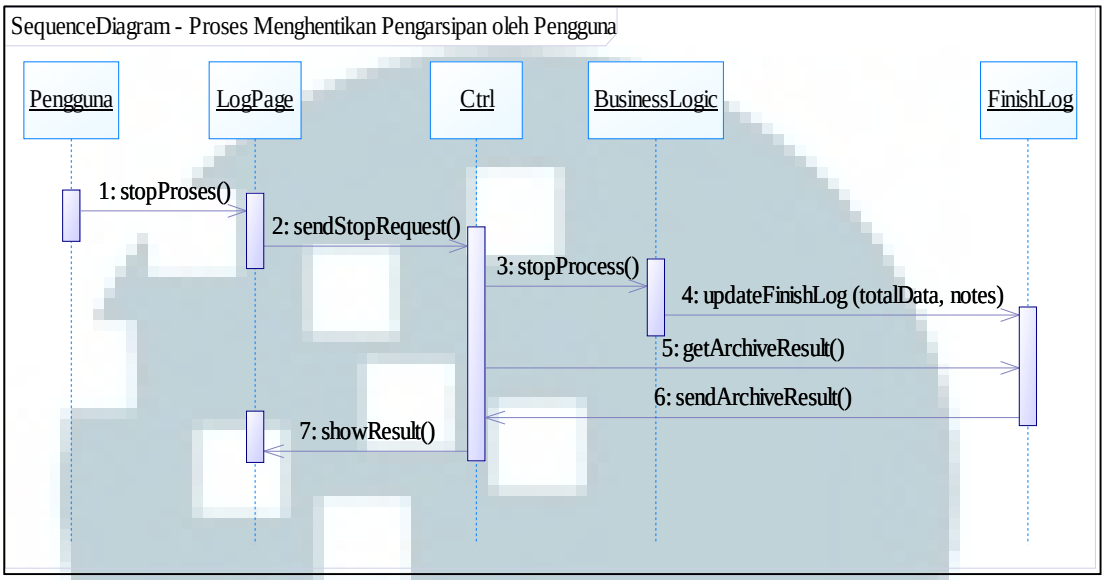


Gambar 3.7 *Sequence Diagram* Proses Pengarsipan Data



Gambar 3.8 *Collaboration Diagram* Proses Pengarsipan Data

3) Sequence dan Collaboration Diagram Proses Penghentian Pengarsipan oleh Pengguna



Gambar 3.9 *Sequence Diagram* Proses Pemberhentian Pengarsipan oleh Pengguna

Sequence diagram proses penghentian pengarsipan oleh pengguna ditunjukkan pada Gambar 3.9. Adapun skenario dari *sequence diagram* tersebut, yaitu sebagai berikut.

(1) Pengguna memilih menu untuk menghentikan proses pengarsipan data secara manual pada halaman Log.

(2) Permintaan pemberhentian proses arsip tersebut dikirim oleh halaman Log ke Controller.

(3) Controller akan memanggil fungsi Stop Process dari Business Logic untuk menghentikan proses pengarsipan data.

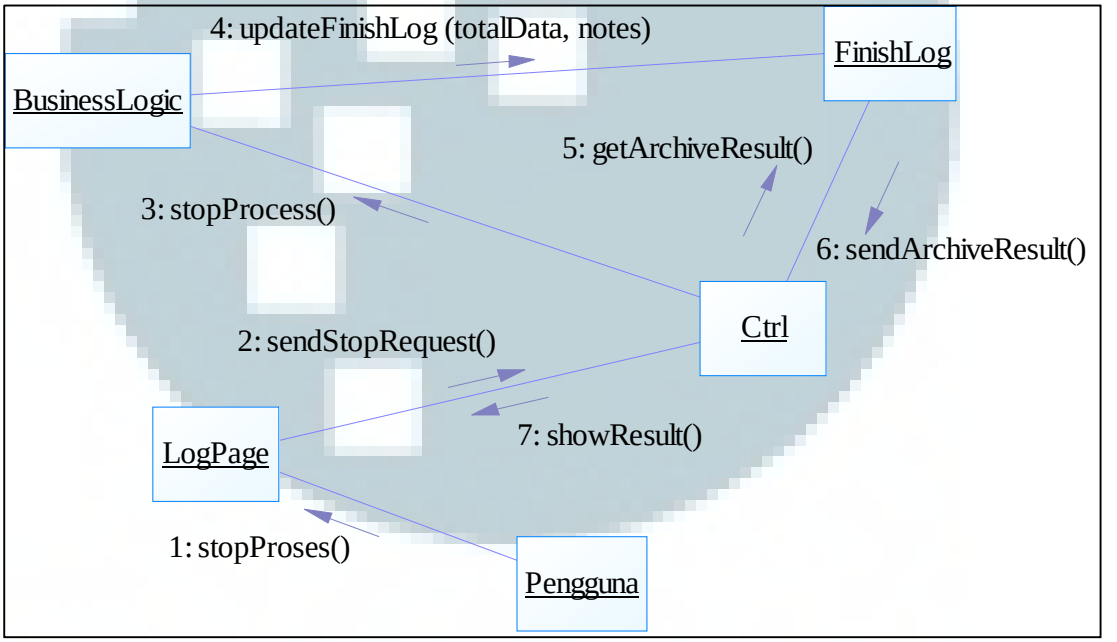
(4) Setelah Business Logic mengetahui bahwa proses pengarsipan telah dihentikan, maka dilakukan *update* ke Finish Log yang menyatakan bahwa proses telah selesai dilakukan. *Update* yang dilakukan juga

menyimpan total data yang berhasil diproses dan catatan khusus bahwa proses dihentikan oleh pengguna.

(5) Controller akan meminta hasil dari proses pengarsipan data ke Finish Log.

(6) Controller menampilkan data tersebut ke halaman Log.

Adapun relasi-relasi yang terjadi antar objek berdasarkan skenario proses penghentian pengarsipan data oleh pengguna direalisasikan ke dalam *collaboration diagram* pada Gambar 3.10.



Gambar 3.10 *Collaboration Diagram* Proses Penghentian Pengarsipan Data oleh Pengguna

C.1.4 Class Diagram

Class diagram modul pengarsipan data ditunjukkan melalui Gambar 3.11. *Class* Archive Controller berperan sebagai pengatur (*controller)* dalam konsep Model-View-Controller (MVC) yang digunakan dalam pengembangan aplikasi Anabatic CR-One. *Class* ini memiliki satu buah atribut, yaitu Process, yang bertipe

data Thread. Atribut Process digunakan untuk menjalankan proses pengarsipan data pada *thread* yang berbeda. *Class* ini memiliki hubungan *one-to-one* dengan *class* Archive Process Result Dto, Archive Dto, dan Archiving Business Logic Factory. Archive Process Result Dto berfungsi sebagai *class* temporer untuk menampilkan hasil dari proses pengarsipan data pada halaman antar muka pengguna. Data yang ditampilkan sebagai hasil dari proses pengarsipan data adalah total waktu proses, total data yang berhasil diproses, dan total *error* yang terjadi selama pengarsipan data. Archive Dto berfungsi sebagai *class* temporer untuk menampung data parameter dari masukan pengguna pada halaman antarmuka.

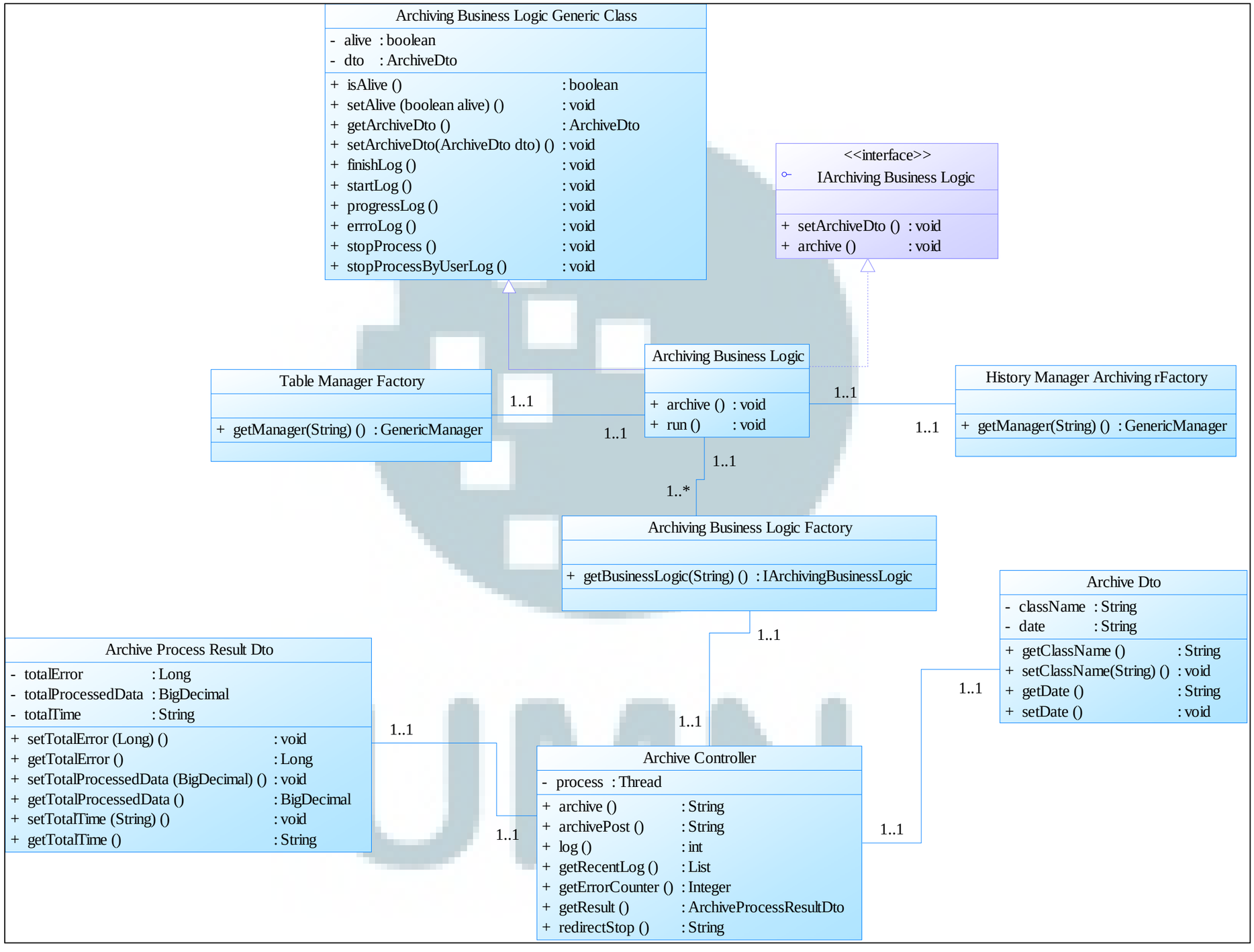
Archiving Business Logic Factory berfungsi sebagai *class* Factory yang mengembalikan objek Business Logic sesuai dengan parameter yang dimasukan oleh pengguna. *Class* Factory adalah *class* hasil implementasi *factory design pattern* yang merupakan salah satu bagian dari pola desain pada konsep Java Design Pattern. Java Design Pattern merupakan pola desain yang banyak digunakan pada pemrograman Java (Tutorials Point, 2015). Dengan menggunakan konsep ini, pembuatan objek dapat dilakukan dengan hanya menggunakan *interface* yang merujuk ke objek baru tersebut sehingga dapat menyembunyikan logika pembuatan objek kepada pengguna.

Class Archiving Business Logic Factory memiliki hubungan *one-to-many* terhadap *class* Archiving Business Logic karena dalam implementasi modul, *class* Archiving Business Logic akan memiliki banyak *instance* tergantung dari jumlah jenis data yang ingin diarsip. Setiap *instance* dari Archiving Business Logic akan berhubungan *one-to-one* dengan Table Manager Factory dan History Manager Factory. Kedua *class factory* ini berperan untuk mengembalikan objek Manager yang tepat sesuai parameter.

Pembuatan *instance* dari Archiving Business Logic dilakukan dengan mengimplementasi *interface* IArchiving Business Logic dan menurunkan (*inherit*) Archiving Business Logic Generic Class. IArchiving Business Logic merupakan *interface* yang menurunkan *class* Runnable sehingga setiap *instance* dari *interface* ini dapat dijalankan pada *thread* berbeda. Penggunaan *thread* yang berbeda dari proses utama bertujuan untuk menampilkan perkembangan kerja dari modul pengarsipan data pada halaman Log.

Archiving Business Logic Generic Class adalah *class* yang berguna sebagai dasar dari pembuatan *instance* objek Business Logic. Class ini memiliki dua buah atribut, yaitu Alive dan Dto. Atribut Alive merupakan atribut yang menunjukan *state* dari proses pengarsipan data. Atribut inilah yang akan diubah ketika pengguna menghentikan proses pengarsipan data secara manual. Atribut Dto merupakan atribut penampung dari objek parameter pengarsipan data yang diisikan oleh pengguna pada halaman utama modul.

Pada Archiving Business Logic Generic Class terdapat berbagai operasi *logging*, seperti startLog(), progressLog(), errorLog(), stopProcessByUserLog(), dan finishLog(). Operasi-operasi *logging* ini didefinisikan pada *generic class* agar setiap *instance* dapat mengakses operasi *logging* yang sama sehingga dapat memudahkan pengembang aplikasi dalam menetapkan suatu mekanisme *logging* yang universal pada setiap *instance*, terutama pada pembuatan *instance* baru. Hal ini dapat membuat proses pengembangan modul menjadi lebih efisien dan efektif.



Gambar 3.11 *Class Diagram* Modul Pengarsipan Data

C.2 Persiapan Struktur Table

Setelah kelima diagram UML selesai dibuat, pembangunan modul dilanjutkan ke tahap persiapan struktur *table* pada sistem basis data History sebagai tempat penyimpanan data arsip. Struktur *table* yang digunakan pada sistem basis data History disesuaikan dengan struktur *table* pada sistem basis data utama aplikasi dan hanya memiliki perbedaan di penamaan saja. Setiap nama *table* pada sistem basis data History diberikan kode "_hist" pada akhir penamaan sebagai tanda bahwa *table* tersebut merupakan *table* History dari *table* utama aplikasi yang bersangkutan, contohnya adalah asa_trx_loan_cash_flow_hist merupakan *table* History dari *table* asa_trx_loan_cash_flow yang ada pada sistem basis data utama aplikasi.

Pembuatan jumlah *table* pada sistem basis data History disesuaikan dengan jumlah jenis data yang dapat diarsip. Beberapa *table* pada sistem basis data utama aplikasi telah memiliki table History. Namun, *table-table* History tersebut belum mencakup keseluruhan jenis data yang dapat diarsip. Oleh karena itu, pembuatan *table* pada sistem basis data History juga dilakukan terhadap *table* dari sistem basis data utama yang belum memiliki *table* History.

Terdapat enam jenis data yang dapat diarsip, yaitu Transaction Loan Cash Flow, Transaction Journal Extract, Transaction Journal Detail, Transaction Loan Amortization Result, Transaction Loan Fee and Cost Amortization Result, dan Transaction Loan Impairment Result. Berikut ini adalah penjelasan singkat mengenai kegunaan dan struktur *table* dari tiap jenis data yang dapat diarsip.

C.2.1 Transaction Loan Cash Flow

Table Transaction Loan Cash Flow digunakan untuk menyimpan informasi *cash flow*, yaitu jumlah pembayaran yang harus dibayarkan pada setiap periode waktu tertentu, dari peminjaman yang dilakukan. Struktur *table* Transaction Loan Cash Flow yang digunakan dalam pembangunan modul pengarsipan data ditunjukkan pada Tabel 3.1.

Tabel 3.1 Struktur *Table* Transaction Loan Cash Flow

No.	Nama Field	Tipe Data
1	sk_loan_id	Integer
2	sk_payment_schedule_id	Integer
3	sk_payment_interest_id	Integer
4	sk_flow_time_id	Integer
5	sk_calculation_time_id	Integer
6	deal_id	Varchar (100 Bytes)
7	source_system	Varchar (100 Bytes)
8	entity	Varchar (100 Bytes)
9	flow_type	Varchar (100 Bytes)
10	amount_type	Varchar (100 Bytes)
11	start_validity_date	Datetime
12	end_validity_date	Datetime
13	calculation_date	Datetime
14	flow_date	Datetime

Tabel 3.2 Struktur *Table* Transaction Loan Cash Flow (Lanjutan)

No.	Nama Field	Tipe Data
15	currency_id	Char (3 Bytes)
16	principal_amount	Numeric (30, 8)
17	interest_amount	Numeric (30, 8)
18	balance_amount	Numeric (30, 8)
19	char_cust_element1	Varchar (100 Bytes)
20	…	Varchar (100 Bytes)
29	char_cust_element10	Varchar (100 Bytes)
30	num_cust_element1	Numeric (30, 8)
31	…	Numeric (30, 8)
39	num_cust_element10	Numeric (30, 8)
40	last_modified	Datetime
41	modified_by	Varchar (100 Bytes)
42	archive_date	Datetime
43	authoriser	Varchar (25 Bytes)
44	authorize_time	Datetime
45	created_by	Varchar (25 Bytes)
46	created_time	Datetime
47	new_value	Varbinary (8000 Bytes)
48	status	Varchar (10 Bytes)
49	updated_by	Varchar (25 Bytes)
50	updated_time	Datetime

Tabel 3.3 Struktur *Table* Transaction Loan Cash Flow (Lanjutan)

No.	Nama Field	Tipe Data
51	version	Integer
52	use_unuse	Integer
53	rest_balance	Numeric (30, 8)

Data arsip pada Transaction Loan Cash Flow tidak dapat dipilih hanya berdasarkan batas tanggal kedaluwarsa saja. Hal ini disebabkan karena terdapat data yang melewati tanggal batas kedaluwarsa, tetapi masih digunakan pada proses bisnis sehari-hari. Oleh karena itu, selain dibandingkan dengan tanggal kedaluawarsa, dibutuhkan kondisi tambahan pada pemilihan data arsip dari *table* Transaction Loan Cash Flow, yaitu membandingkan nilai *field* Calculation Time Id dengan jumlah dan nilai maksimum dari tiap jenis data. Adapun contoh *query* yang dibuat untuk mengambil data arsip pada *table* Transaction Loan Cash Flow ditunjukkan pada Gambar 3.12.

```
select  sk_loan_id, sk_payment_schedule_id, sk_payment_interest_id,
        sk_flow_time_id, sk_calculation_time_id,
        ...
from asa_trx_loan_cash_flow a
where (select COUNT(distinct sk_calculation_time_id)
       from asa_trx_loan_cash_flow c
       where c.deal_id = a.deal_id) > 1
and a.sk_calculation_time_id != (select MAX(sk_calculation_time_id)
                                 from asa_trx_loan_cash_flow b
                                 where b.deal_id = a.deal_id)
and a.sk_calculation_time_id <= :calculateTimeId
order by a.deal_id asc, a.sk_calculation_time_id desc
```

Gambar 3.12 Contoh *Query* Pemilihan Data Arsip Transaction Loan Cash Flow

C.2.2 Transaction Journal Extract

Table Transaction Journal Extract digunakan sebagai *header* dalam penyimpanan informasi jurnal yang telah dikelompokkan berdasarkan tipe *event* jurnal, seperti Amortization Below Market, Amortization Cost, dan Amortization Fee. Detail dari data pada *table* ini akan dijabarkan pada Transaction Journal Detail. Struktur *table* Transaction Journal Extract yang digunakan dalam pembangunan modul pengarsipan data ditunjukkan pada Tabel 3.4.

Tabel 3.4 Struktur *Table* Transaction Journal Extract

No.	Nama Field	Tipe Data
1	no	Integer
2	entity	Varchar (5 Bytes)
3	journal_event_type	Varchar (500 Bytes)
4	valid_date	Datetime
5	posting_date	Datetime
6	branch	Varchar (100 Bytes)
7	original_ccy	Varchar (5 Bytes)
8	original_amount	Numeric (38, 16)
9	division	Varchar (100 Bytes)
10	product_code	Varchar (100 Bytes)
11	dc_indicator	Varchar (100 Bytes)
12	dimension1	Varchar (100 Bytes)
13	…	Varchar (100 Bytes)

Tabel 3.5 Struktur *Table* Transaction Journal Extract (Lanjutan)

No.	Nama Field	Tipe Data
16	dimension5	Varchar (100 Bytes)
17	created_date	Datetime
18	created_by	Varchar (100 Bytes)
19	archive_date	Datetime
20	authoriser	Varchar (25 Bytes)
21	authorize_time	Datetime
22	created_by	Varchar (25 Bytes)
23	created_time	Datetime
24	new_value	Varbinary (8000 Bytes)
25	status	Varchar (10 Bytes)
26	updated_by	Varchar (25 Bytes)
27	updated_time	Datetime
28	version	Integer

Data arsip pada Transaction Journal Extract dapat diperoleh dengan membandingkan *field* Posting Date dengan tanggal batas kedaluwarsa yang dimasukkan pengguna. Adapun contoh *query* yang dibuat untuk memperoleh data arsip Transaction Journal Extract ditunjukkan pada Gambar 3.13.

```
select  no, entity, coa, journal_event_type, valid_date, posting_date,
        branch, original_ccy, original_amount, division, product_code,
        dc_indicator, Dimension1, Dimension2, Dimension3, Dimension4,
        Dimension5, CreatedDate, CreatedBy
from asa_trx_journal_extract a
where a.posting_date <= :postingDate;
```

Gambar 3.13 Contoh *Query* Pemilihan Data Arsip Transaction Journal Extract

C.2.3 Transaction Journal Detail

Table Transaction Journal Detail digunakan dalam penyimpanan informasi jurnal secara detail untuk setiap rekening yang telah didefinisikan pada Transaction Journal Extract. Struktur *table* Transaction Journal Detail yang digunakan dalam pembangunan modul pengarsipan data ditunjukkan pada Tabel 3.6.

Tabel 3.6 Struktur *Table* Transaction Journal Detail

No.	Nama Field	Tipe Data
1	id	Integer
2	journal_type	Varchar (100 Bytes)
3	header_no	Varchar (100 Bytes)
4	journal_no	Varchar (100 Bytes)
5	entity	Varchar (100 Bytes)
6	customer_no	Varchar (100 Bytes)
7	coa_code	Varchar (100 Bytes)
8	sk_loan_id	Integer
9	deal_id	Varchar (100 Bytes)

Tabel 3.7 Struktur *Table* Transaction Journal Detail (Lanjutan)

No.	Nama Field	Tipe Data
10	sk_fee_cost_id	Integer
11	fee_cost_id	Varchar (100 Bytes)
12	product_group_code	Varchar (100 Bytes)
13	product_code	Varchar (100 Bytes)
14	branch	Varchar (100 Bytes)
15	industry_code	Varchar (100 Bytes)
16	valid_date	Datetime
17	sk_posting_time_id	Integer
18	posting_date	Datetime
19	original_ccy	Varchar (5 Bytes)
20	original_amount	Numeric (38, 16)
21	local_amount	Numeric (38, 16)
22	sk_journal_event_type	Integer
23	journal_event_type	Varchar (100 Bytes)
24	description	Varchar (250 Bytes)
25	dc_indicator	Varchar (100 Bytes)
26	journal_event_id	Varchar (100 Bytes)
27	valuation_type	Varchar (100 Bytes)
28	custom_detail1	Varchar (100 Bytes)
29	…	Varchar (100 Bytes)
32	custom_detail5	Varchar (100 Bytes)

Tabel 3.8 Struktur *Table* Transaction Journal Detail (Lanjutan)

No.	Nama Field	Tipe Data
33	source_ref	Varchar (100 Bytes)
34	last_modified	Datetime
35	modified_by	Varchar (100 Bytes)
36	archive_date	Datetime
37	authoriser	Varchar (25 Bytes)
38	authorize_time	Datetime
39	created_by	Varchar (25 Bytes)
40	created_time	Datetime
41	new_value	Varbinary (8000 Bytes)
42	status	Varchar (10 Bytes)
43	updated_by	Varchar (25 Bytes)
44	updated_time	Datetime
45	version	Integer

Data arsip pada Transaction Journal Detail dapat diperoleh dengan membandingkan *field* Sk Posting Time Id dengan tanggal batas kedaluwarsa yang dimasukkan pengguna. Adapun contoh *query* yang dibuat untuk memperoleh data arsip Transaction Journal Detail ditunjukkan pada Gambar 3.14.

```
select  id, journal_type, header_no, journal_no, entity, customer_no,
        coa_code, sk_loan_id, deal_id, sk_fee_cost_id, fee_cost_id,
        product_group_code, product_code, branch, industry_code,
        valid_date, sk_posting_time_id, posting_date, original_ccy,
        original_amount, local_amount, sk_journal_event_type,
        journal_event_type, description, dc_indicator, journal_event_id,
        valuation_type, custom_detail1, custom_detail2, custom_detail3,
        custom_detail4, custom_detail5, source_ref, last_modified,
        modified_by
from asa_trx_journal_detail a
where a.sk_posting_time_id <= :calculateTimeId
order by a.deal_id asc, a.sk_posting_time_id desc;
```

Gambar 3.14 Contoh *Query* Pemilihan Data Arsip Transaction Journal Detail

C.2.4 Transaction Loan Amortization Result

Table Transaction Loan Amortization Result digunakan untuk menyimpan informasi mengenai hasil amortisasi terhadap peminjaman yang dilakukan pada tiap-tiap rekening. Struktur *table* Transaction Loan Amortization Result yang digunakan dalam pembangunan modul pengarsipan data ditunjukkan pada Tabel 3.9.

Tabel 3.9 Struktur *Table* Loan Amortization Result

No.	Nama Field	Tipe Data
1	sk_loan_id	Integer
2	sk_time_id	Integer
3	deal_id	Varchar (20 Bytes)
4	entity	Varchar (10 Bytes)
5	amortization_date	Datetime
6	currency_id	Varchar (3 Bytes)
7	ac_00	Decimal (30, 8)

Tabel 3.10 Struktur *Table* Loan Amortization Result (Lanjutan)

No.	Nama Field	Tipe Data
8	ac_99	Decimal (30, 8)
9	ac_adj_00	Decimal (30, 8)
10	ac_adj_99	Decimal (30, 8)
11	effective_interest_99	Decimal (25, 16)
12	effective_interest_00	Decimal (25, 16)
13	eir_00	Decimal (19, 16)
14	eir_99	Decimal (25, 16)
15	amort	Decimal (30, 8)
16	additional_amort	Decimal (30, 8)
17	theo_balance	Decimal (30, 8)
18	cummulative_amort	Decimal (30, 8)
19	unamortized_amount	Decimal (30, 8)
20	effective_interest_00_per_diem	Decimal (19, 16)
21	effective_interest_99_per_diem	Decimal (19, 16)
22	amort_per_diem	Decimal (30, 8)
23	day_diff	Integer
24	cummulative_effective_interest_00	Decimal (25, 16)
25	cummulative_effective_interest_99	Decimal (25, 16)
26	archive_date	Datetime

Tabel 3.11 Struktur *Table* Loan Amortization Result (Lanjutan)

No.	Nama Field	Tipe Data
27	authoriser	Varchar (25 Bytes)
28	authorize_time	Datetime
29	created_by	Varchar (25 Bytes)
30	created_time	Datetime
31	new_value	Varbinary (8000 Bytes)
32	status	Varchar (10 Bytes)
33	updated_by	Varchar (25 Bytes)
34	updated_time	Datetime
35	version	Integer

Data arsip pada Transaction Loan Amortization Result dapat diperoleh dengan membandingkan *field* Sk Time Id dengan tanggal batas kedaluwarsa yang dimasukkan pengguna. Adapun contoh *query* yang dibuat untuk memperoleh data arsip Transaction Loan Amortization Result ditunjukkan pada Gambar 3.15.

```
select  sk_loan_id, sk_time_id, deal_id, entity, amortization_date,
        currency_id, ac_00, ac_99, ac_adj_00, ac_adj_99,
        effective_interest_99, effective_interest_00, eir_00, eir_99,
        amort, additional_amort, theo_balance, cummulative_amort,
        unamortized_amount, effective_interest_00_per_diem,
        effective_interest_99_per_diem, amort_per_diem, day_diff,
        cummulative_effective_interest_00,
        cummulative_effective_interest_99
from asa_trx_loan_amortization_result a
where a.sk_time_id <= :calculationTimeId
order by deal_id asc, sk_time_id desc;
```

Gambar 3.15 Contoh *Query* Pemilihan Data Arsip Transaction Loan Amortization Result

C.2.5 Transaction Loan Fee and Cost Amortization Result

Table Transaction Loan Fee and Cost Amortization Result digunakan untuk menyimpan informasi mengenai hasil amortisasi terhadap *fee* dan *cost* dari peminjaman yang dilakukan. Struktur *table* Transaction Loan Fee and Cost Amortization Result yang digunakan dalam pembangunan modul pengarsipan data ditunjukkan pada Tabel 3.12.

Tabel 3.12 Struktur *Table* Transaction Loan Fee and Cost Amortization Result

No.	Nama Field	Tipe Data
1	sk_fee_cost_id	Integer
2	sk_time_id	Integer
3	fee_cost_id	Varchar (20 Bytes)
4	entity	Varchar (10 Bytes)
5	amortization_date	Datetime
6	currency_id	Varchar (3 Bytes)
7	ac	Decimal (26, 8)
8	ac_adj	Decimal (26, 8)
9	eir	Decimal (19, 16)
10	amort	Decimal (26, 8)
11	additional_amort	Decimal (26, 8)
12	theo_balance	Decimal (26, 8)
13	cummulative_amort	Decimal (26, 8)
14	unamortized_amount	Decimal (26, 8)
15	effective_interest_per_diem	Decimal (20, 2)

Tabel 3.13 Struktur *Table* Transaction Loan Fee and Cost Amortization Result (Lanjutan)

No.	Nama Field	Tipe Data
16	amort_per_diem	Decimal (20, 2)
17	day_diff	Integer
18	cummulative_effective_interest	Decimal (25, 16)
19	effective_interest	Decimal (25, 16)
20	archive_date	Datetime
21	authoriser	Varchar (25 Bytes)
22	authorize_time	Datetime
23	created_by	Varchar (25 Bytes)
24	created_time	Datetime
25	new_value	Varbinary (8000 Bytes)
26	status	Varchar (10 Bytes)
27	updated_by	Varchar (25 Bytes)
28	updated_time	Datetime
29	version	Integer

Data arsip pada Transaction Loan Fee and Cost Amortization Result dapat diperoleh dengan membandingkan *field* Sk Time Id dengan tanggal batas kedaluwarsa yang dimasukkan pengguna. Adapun contoh *query* yang dibuat untuk memperoleh data arsip Transaction Loan Fee and Cost Amortization Result ditunjukkan pada Gambar 3.16.

```
select  sk_fee_cost_id, sk_time_id, fee_cost_id, entity,
        amortization_date, currency_id, ac, ac_adj, eir,
        amort, additional_amort, theo_balance, cummulative_amort,
        unamortized_amount, effective_interest_per_diem,
        amort_per_diem, day_diff, cummulative_effective_interest,
        effective_interest
from asa_trx_loan_fee_cost_amortization_result a
where a.sk_time_id <= :calculationTimeId
order by sk_fee_cost_id asc, sk_time_id desc;
```

Gambar 3.16 Contoh *Query* Pemilihan Data Arsip Transaction Loan Fee and Cost Amortization Result

C.2.6 Transaction Loan Impairment Result

Table Transaction Loan Impairment Result digunakan untuk menyimpan informasi *impairment*, yaitu pencadangan dana atas kerugian yang mungkin dihasilkan dari proses peminjaman, terhadap tiap-tiap rekening. Struktur *table* Transaction Loan Impairment Result yang digunakan dalam pembangunan modul pengarsipan data ditunjukkan pada Tabel 3.14.

Tabel 3.14 Struktur *Table* Transaction Loan Impairment Result

No.	Nama Field	Tipe Data
1	sk_loan_id	Integer
2	sk_time_id	Varchar (20 Bytes)
3	entity	Varchar (10 Bytes)
4	impairment_date	Datetime
5	currency_id	Varchar (3 Bytes)
6	recov_amount	Decimal (20, 2)
7	tech_interest	Decimal (20, 2)
8	accum_tech_interest	Decimal (20, 2)

Tabel 3.15 Struktur *Table* Transaction Loan Impairment Result (Lanjutan)

No.	Nama Field	Tipe Data
9	recov_amount_99	Decimal (20, 2)
10	impairment	Decimal (20, 2)
11	impairment_delta	Decimal (20, 2)
12	archive_date	Datetime
13	authoriser	Varchar (25 Bytes)
14	authorize_time	Datetime
15	created_by	Varchar (25 Bytes)
16	created_time	Datetime
17	new_value	Varbinary (8000 Bytes)
18	status	Varchar (10 Bytes)
19	updated_by	Varchar (25 Bytes)
20	updated_time	Datetime
21	version	Integer

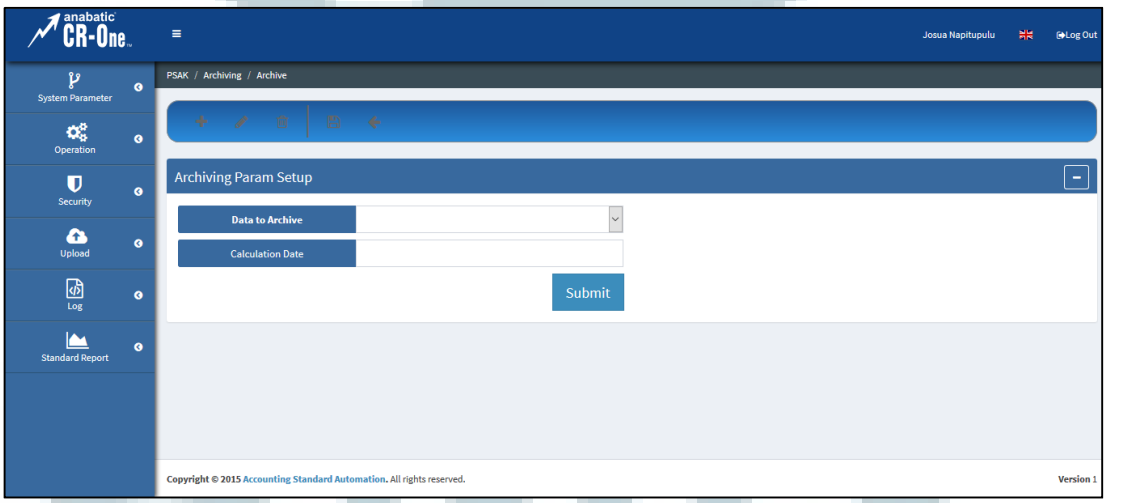
Data arsip pada Transaction Loan Impairment Result dapat diperoleh dengan membandingkan *field* Sk Time Id dengan tanggal batas kedaluwarsa yang dimasukkan pengguna. Adapun contoh *query* yang dibuat untuk memperoleh data arsip Transaction Loan Impairment Result ditunjukkan pada Gambar 3.17.

```
select  sk_loan_id, sk_time_id, deal_id, entity,
        impairment_date, currency_id, recov_amount,
        tech_interest, accum_tech_interest,
        recov_amount_99, impairment, impairment_delta
from asa_trx_loan_impairment_result a
where a.sk_time_id <= :calculationTimeId
order by deal_id asc, sk_time_id desc;
```

Gambar 3.17 Contoh *Query* Pemilihan Data Arsip Transaction Loan Impairment Result

C.3 Pembuatan Antarmuka

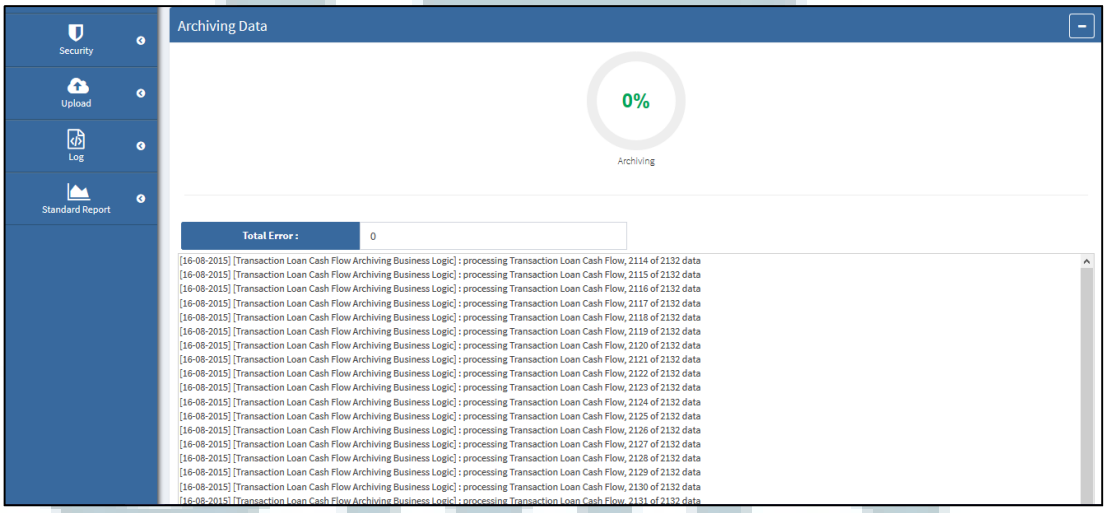
Setelah perancangan objek dan persiapan sistem basis data History telah selesai dilakukan, pembangunan modul dilanjutkan ke tahap pembuatan tampilan antarmuka modul yang akan diimplementasikan pada aplikasi Anabatic CR-One. Antarmuka modul pengarsipan data mengikuti tata letak dan standar yang digunakan dalam pengembangan aplikasi Anabatic CR-One. Adapun tampilan antarmuka halaman utama modul pengarsipan data ditunjukkan pada Gambar 3.18.



Gambar 3.18 Halaman Utama Modul Pengarsipan Data

Halaman utama modul pengarsipan data memiliki satu objek *form* yang memiliki dua kolom untuk masukan parameter. Kolom pertama berupa *combobox*, digunakan untuk memilih data yang ingin diarsip. Kolom kedua berupa *text field* yang dikombinasikan dengan JQuery Datepicker, digunakan untuk menentukan tanggal batas kedaluwarsa dari data yang ingin diarsip. Pada halaman ini juga telah disediakan validasi terhadap masukan pengguna, baik secara *client-side*, maupun *server-side*. Secara *client-side*, validasi dilakukan dengan menggunakan Javascript yang disimpan dalam *file* terpisah, sedangkan validasi pada *server-side* menggunakan objek Validator.

Setelah parameter yang dimasukkan pengguna tervalidasi dan tombol Submit diklik, maka modul akan mengarahkan pengguna ke halaman Log untuk melihat perkembangan kerja dari proses pengarsipan data. Adapun tampilan antarmuka halaman Log modul pengarsipan data ditunjukkan pada Gambar 3.19.



Gambar 3.19 Halaman Log Modul Pengasipan Data

Halaman Log terdiri dari tiga komponen utama, yaitu *progress bar*, *text field*, dan *text area*. *Progress bar* digunakan untuk merepresentasikan perkembangan kerja proses pengarsipan dalam angka persentase, dibuat dengan menggunakan JQuery Knob yang telah disediakan pada aplikasi. *Text field* digunakan untuk memberikan informasi mengenai jumlah kesalahan yang terjadi selama proses pengarsipan data. *Text area* digunakan untuk memberikan informasi mengenai detail *log*, yaitu informasi mengenai total data yang telah diproses, nama jenis data yang sedang diarsip, dan status dari proses pengarsipan data dalam suatu waktu, serta menampilkan catatan khusus apabila proses diberhentikan secara manual oleh pengguna.

D. Pembuatan Kode

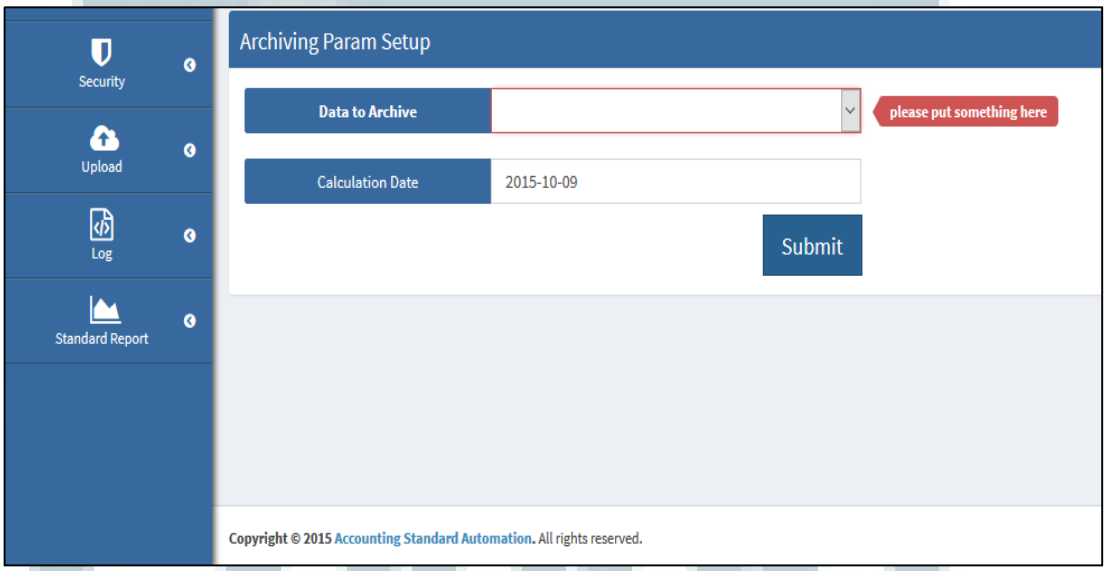
Modul dibangun dengan menggunakan bahasa pemrograman Java yang diimplementasikan dalam JavaServer Page (JSP) dan menggunakan sistem basis data Microsoft SQL Server 2008 R2 sesuai dengan proyek pengembangan aplikasi Anabatic CR-One yang telah ada sebelumnya. Kenyamanan yang sama terhadap penggunaan modul selalu diupayakan dengan penggunaan fitur-fitur JQuery yang telah digunakan sebelumnya pada pengembangan aplikasi, seperti fitur JQuery Knob dan JQuery Datepicker.

Framework yang digunakan pada pengembangan modul adalah Spring Tool Suite versi 3.6.3.RELEASE. Adapun beberapa *tools* yang digunakan dalam *framework* ini guna membantu pengembangan modul, yakni Apache Maven 3.0.3 sebagai tempat penyimpanan *dependencies* sehingga modul dapat dibuat atau di-*build* secara otomatis, Spring Beans untuk memberikan anotasi pada setiap objek dalam

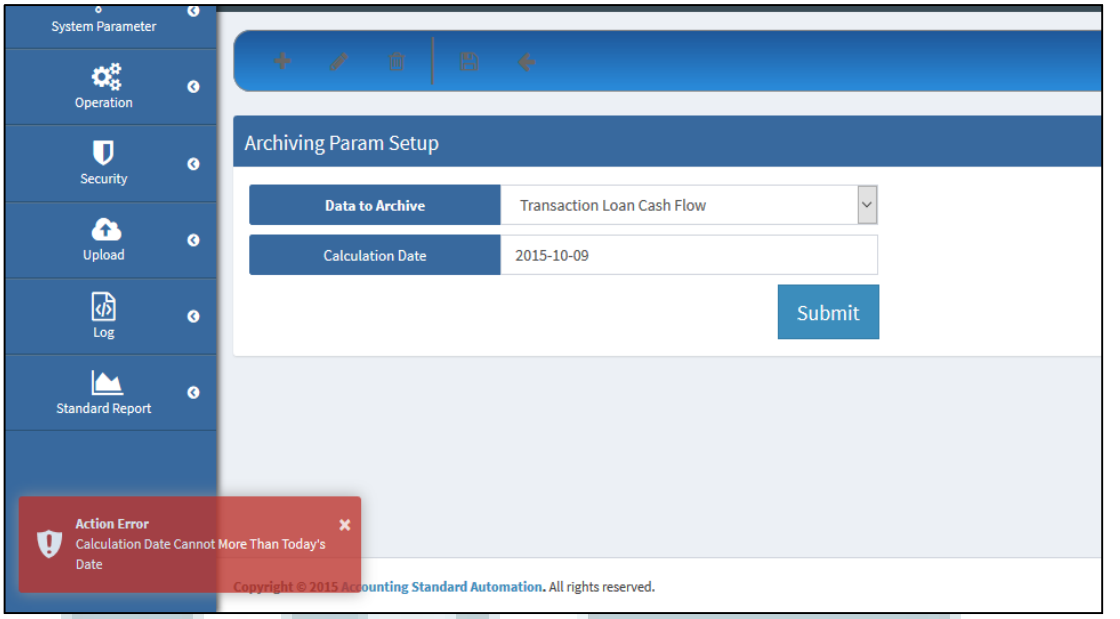
modul, dan Apache Tomcat 7.0.54 sebagai teknologi yang digunakan untuk mengimplementasikan JSP dari modul dan aplikasi pada *server*.

E. Uji coba, evaluasi, dan perbaikan

Uji coba dilakukan pada setiap jenis data yang diintegrasikan dengan modul pengarsipan data. Proses uji coba dimulai dari tahap memasukkan parameter, yakni jenis data yang ingin diarsip dan tanggal kedaluwarsa data. Tahap ini akan terus diulang sampai didapatkan parameter yang valid. Proses validasi dilakukan, baik secara *client-side* (Gambar 3.20), maupun *server-side* (Gambar 3.21). Setelah itu, modul akan membuka halaman Log untuk menampilkan perkembangan kerja dari proses pengarsipan data.



Gambar 3.20 Validasi *Client-Side* Menggunakan Javascript

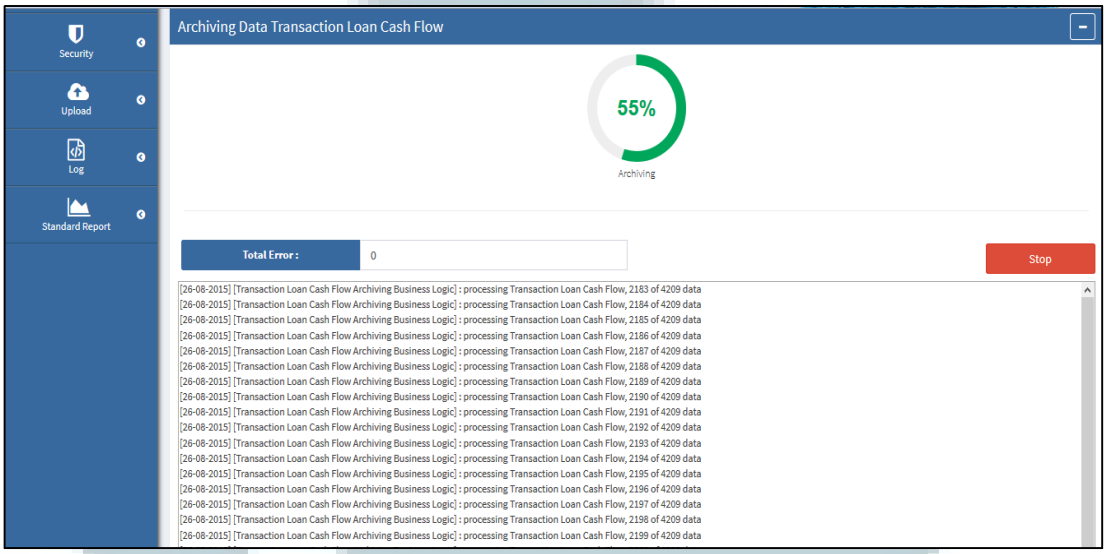


Gambar 3.21 Validasi *Server-Side* dengan Objek Validator

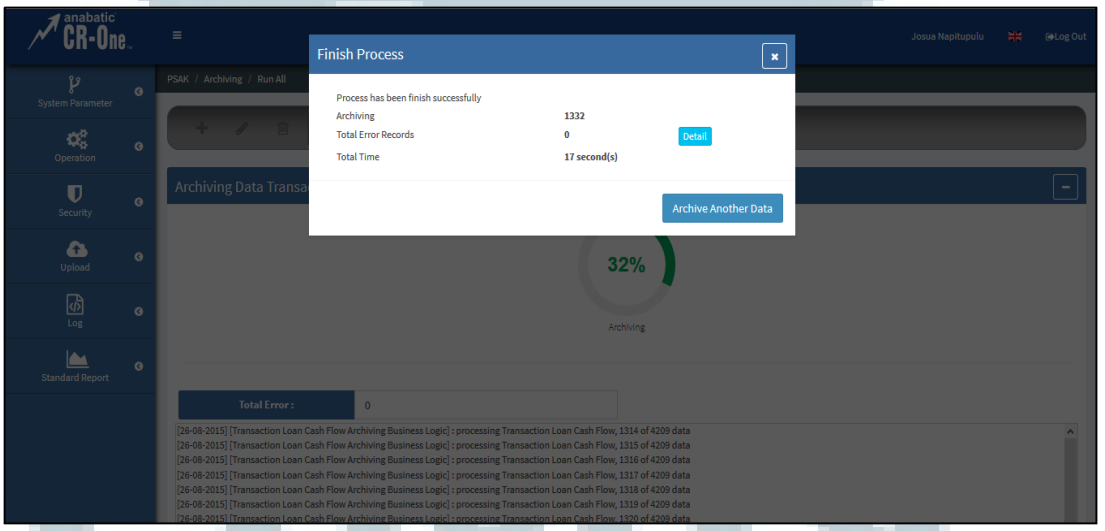
Perkembangan kerja proses pengarsipan data direpresentasikan dalam angka persentase pada halaman Log melalui sebuah *progress bar*. Persentase perkembangan kerja akan terus di-*update* sampai proses selesai. Di bawah *progress bar*, terdapat *text field* untuk menghitung jumlah kesalahan yang terjadi ketika proses pengarsipan data. Pada bagian akhir halaman Log, terdapat *text area* berisi detail *log* untuk menunjukan informasi mengenai jumlah data yang telah diproses pada suatu waktu. Gambar 3.22 menunjukan tampilan dari halaman Log ketika proses pengarsipan data sedang berlangsung.

Uji coba yang dilakukan juga meliputi pemberhentian proses pengarsipan data secara manual. Ketika pengguna menekan tombol Stop dan memilih tombol Yes pada menu konfirmasi, maka proses akan dihentikan dan hasil dari pengarsipan data akan ditampilkan. Proses yang diberhentikan secara manual akan diberikan catatan khusus berupa keterangan bahwa proses diberhentikan secara manual oleh pengguna di

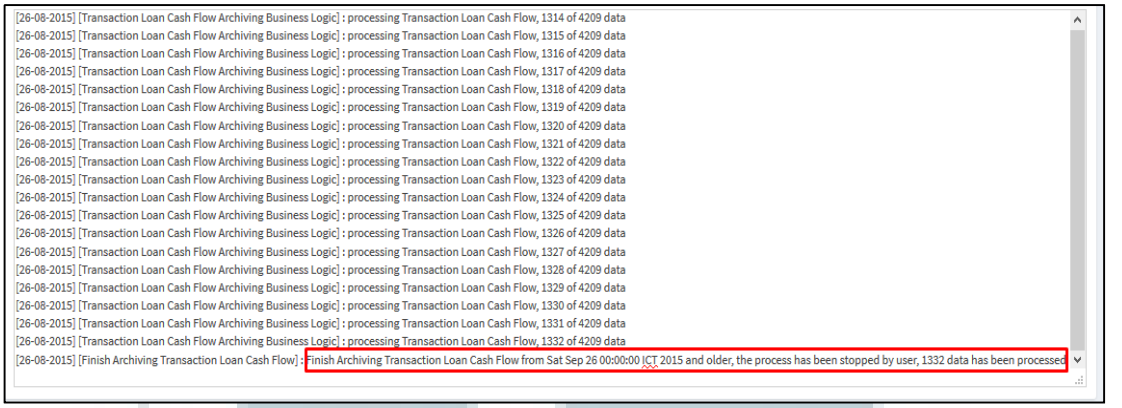
bagian detail *log*. Gambar 3.23 menunjukan tampilan dari hasil proses pengarsipan data dan Gambar 3.24 menunjukan catatan khusus yang diberikan ketika proses diberhentikan secara manual oleh pengguna.



Gambar 3.22 Halaman Log Ketika Menjalankan Proses Pengarsipan Data



Gambar 3.23 Tampilan Hasil dari Proses Pengarsipan Data



Gambar 3.24 Detail Log dengan Catatan Khusus Ketika Proses Pengarsipan Dihentikan secara Manual

Ketika terdapat kesalahan, maka evaluasi dan perbaikan akan langsung dilakukan pada modul. Evaluasi bertujuan untuk menentukan langkah yang akan diambil untuk mengatasi masalah yang ditemukan pada saat uji coba. Proses evaluasi juga dilakukan bersama Muchammad Ikhsan Girinata, selaku Project Leader dari proyek PSAK. Setelah evaluasi selesai dilakukan, perbaikan terhadap modul akan dilakukan sesuai dengan hasil evaluasi sebelumnya. Uji coba, evaluasi, dan perbaikan dilakukan terus menerus sampai tidak ada kesalahan yang ditemukan lagi pada modul.

F. Dokumentasi

Dokumentasi dari modul pengarsipan data berisikan daftar dari objek baru, baik *class, interface,* maupun *enumeration*, yang ditambahkan pada aplikasi Anabatic CR-One. Informasi objek mencakup letak penempatan objek pada aplikasi, kegunaan objek, dan daftar fungsi yang ada pada objek tersebut. Daftar fungsi mencakup nama fungsi, penjelasan mengenai fungsi tersebut, dan tipe data yang dikembalikan oleh fungsi.

Selain daftar objek baru, catatan mengenai daftar objek yang diubah untuk keperluan penyelesaian modul pengarsipan data juga diberikan agar implementasi modul pada aplikasi Anabatic CR-One dapat dilakukan dengan benar. Informasi mengenai mekanisme pembuatan *instance* baru ketika terdapat data tambahan yang ingin diarsip dan berbagai catatan singkat dalam implementasi modul juga disertakan dalam dokumentasi.

3.3.2 Kendala yang Ditemukan

Berikut adalah beberapa kendala yang ditemukan selama proses kerja magang, yakni.

1) *Framework* dan berbagai *tools* yang dipakai pada pengembangan aplikasi belum pernah dipelajari maupun digunakan sebelumnya.
2) Daftar data yang dapat diarsip belum lengkap, terdapat data yang masih menimbulkan pertimbangan akan kebutuhan pengarsipan.
3) Adanya perbedaan antara objek Model pada aplikasi dengan *physical table* yang digunakan pada sistem basis data aplikasi. Hal ini menyebabkan kesulitan dalam pemindahan data.
4) Terdapat data yang memiliki nilai identifikasi sama. Hal ini menyebabkan modul tidak dapat dibuat dengan menggunakan Hibernate Query Language (HQL) yang memiliki kelebihan dapat berjalan di berbagai *platform* sistem basis data tanpa harus mengubah *query* aplikasi.
5) Proyek yang diberikan bukanlah proyek asli, melainkan hasil enkripsi yang berjalan secara independen. Hal ini menyebabkan keberhasilan implementasi yang dilakukan di proyek enkripsi, belum tentu berjalan di proyek sungguhan.

3.3.3 Solusi atas Kendala yang Ditemukan

Berdasarkan kendala yang ditemukan, berikut adalah solusi-solusi yang dilakukan untuk mengurangi dampak dari kendala tersebut, yakni.

1) Melakukan pelatihan selama satu minggu dengan membuat proyek pribadi menggunakan *framework* yang digunakan perusahaan agar dapat membiasakan diri terhadap mekanisme kerja *framework*. Proyek pribadi tersebut dipresentasikan pada akhir batas waktu training dihadapan pembimbing lapangan.
2) Membuat modul menjadi mudah untuk dikembangkan, yakni dengan mengimplementasikan *factory design pattern* yang diberitahukan oleh pembimbing lapangan dan memanfaatkan pendekatan pemrograman berorientasi objek dengan lebih baik lagi, terutama dalam hal penggunaan objek *interface*, *generic class*, dan *enumeration*. Hal ini bertujuan agar pengembangan aplikasi dapat dilakukan dengan efisien dan efektif, serta tidak menimbulkan perubahan yang signifikan ketika terdapat data baru yang ingin diintegrasikan dengan modul pengarsipan data.
3) Mengubah dan menyamakan struktur objek, baik struktur pada objek Model, maupun struktur pada *physical table*, berdasarkan data yang telah tersimpan dalam sistem basis data.
4) Menggunakan Native Query Language pada pembuatan *query* dan fungsi atau *function* yang umum dimiliki oleh berbagai sistem basis data, seperti Count dan Sum. Pada Native Query Language, data dikembalikan dengan tipe data objek *array* dua dimensi (*two dimensional array of objects*). Oleh karena itu,

pembuatan fungsi konversi dilakukan pada layer *service* dari setiap objek Model yang ingin diarsip sehingga pemindahan data dapat dilakukan dengan baik dan lancar.

5) Membuat dokumentasi mengenai daftar objek yang diubah. Dokumentasi yang dibuat juga mencakup daftar objek baru, letak objek, tujuan penggunaan objek, dan daftar fungsi yang terdapat pada setiap objek baru tersebut secara lengkap dan jelas.