# BAB III

# ANALISA DAN PERANCANGAN

## 3.1 Analisa

Analisa secara umum adalah tahap dalam pembangunan atau pengembangan sistem yaitu menemukan dan mempelajari permasalahan-permasalahan yang ada dalam suatu lingkungan kerja atau proses bisnis. Proses analisis dilakukan dengan cara mempelajari sistem yang sedang berjalan saat ini, lalu melakukan studi permasalahan dengan membandingkan proses kerja sistem saat ini dan temuan data di lapangan. Perbandingan antara proses kerja sistem dengan temuan data di lapangan dilakukan untuk menemukan permasalahan yang selanjutnya akan dipelajari untuk ditentukan solusi dan kebutuhan penunjang solusi tersebut.

### 3.1.1 Analisa Masalah

Proses pencatatan transaksi dan *maintenance service* dari perangkat yang sudah terpasang di *client* untuk saat ini dilakukan secara *manual*, dimana tim *sales* dan tim *maintenance* yang datang ke-*client* membawa sebuah *form* yang dimana *form* tersebut digunakan untuk melakukan pengecekan dan pencatatan transaksi yang dilakukan. Setelah tim *sales* dan *maintenance* melakukan tugasnya di client, *form-form* tersebut diserahkan kepada tim *admin* dan tim *admin* akan *menginputkan* secara manual dari hasil *form* tersebut di dalam sebuah *file excel*. Setelah *admin* menginputkan hasil transaksi dan *maintenance* ke *file excel* , *file* tersebut akan dikirimkan kepada tim *management* untuk proses *reporting*.

Oleh karena proses yang semua masih serba *manual*, sering terjadi kekeliruan data yang didapat oleh tim sales dan *maintenance* dan *diinputkan* oleh tim *admin*. Selain sering terjadi kesalahan data, data yang sudah diinputkan oleh *admin* rawan terjadi rusak pada *file* nya, karena penyimpanan *form* fisik yang masih kurang baik. Untuk tim *management* pun cukup kesulitan untuk melihat laporan penjualan dan *maintenance* yang dilakukan oleh tim nya karena data tidak bisa dilihat secara *real time* saat itu juga.

### 3.1.2 Pemecahan Masalah

Permasalahan yang dijelaskan diatas dapat teratasi dengan menggunakan sistem *sales tracking activity.* Sistem *sales activity tracking* memungkinkan tim *sales* dan *maintenance* untuk menginputkan data transaksi atau aktivitas yang dilakukan saat berada di tempat client dengan lebih mudah dan *real time.* Sistem *sales activity tracking* dapat mempermudah kerja dari tim Admin, dimana admin sudah tidak perlu menginputkan manual hasil pekerjaan dari tim *sales* dan *maintenance* secara manual. Dan tim *management* dapat melihat proses bisnis nya berjalan secara real time tanpa takut dengan kesalahan data yang sudah diinputkan oleh tim *admin.*

Sistem yang dirancang memberikan solusi terhadap permasalahan yang ada. Pengolahan data tidak lagi menggunakan cara manual, sistem yang dirancang adalah solusi untuk mempermudah proses bisnis yang dilakukan oleh PT. iClean. Sistem juga tidak memerlukan waktu yang lama dari pengolahan data untuk memberikan *output* berupa reporting pekerjaan yang dilakukan oleh tim *sales* dan *maintenance.* Pada akhirnya perusahaan dapat mengambil keputusan dengan lebih cepat dan efisien terhadap kinerja bisnisnya.

### 3.1.3 Analisa Kebutuhan

Hal pertama yang perlu dilakukan dalam analisa kebutuhan sistem adalah menentukan kebutuhan sistem. Kebutuhan sistem terbagi menjadi dua yaitu: kebutuhan fungsional dan kebutuhan non-fungsional.

#### 3.1.3.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan yang terkait langsung dengan fungsi-fungsi atau fitur-fitur yang harus ada dalam suatu sistem atau aplikasi. Kebutuhan fungsional menggambarkan apa yang sistem atau aplikasi tersebut harus lakukan dan bagaimana sistem tersebut harus berperilaku dalam berbagai situasi.:

a. Kebutuhan tim *Sales* dan *Maintenance*:

   Dapat menginputkan proses penjualan dan proses maintenance yang dilakukan secara online, tanpa harus membawa *form - from* yang selama ini dibawa.

b. Kebutuhan tim *Admin*:

   Tidak perlu menginputkan hasil dari tim *sales* dan *maintenance* secara *manual* lagi, tim *admin* hanya perlu memelihara *system* dan melakukan proses *monitoring* terhadap *system.*

c. Kebutuhan tim *Management* :

   Tim *management* dapat melakukan *monitoring* hasil transaksi dan *maintenance* dari timnya secara *real time* dan data yang akurat.

#### 3.1.3.2 Kebutuhan Non Fungsional

Kebutuhan non-fungsional adalah kebutuhan yang tidak berkaitan secara langsung dengan fitur-fitur spesifik dalam sistem atau aplikasi, tetapi lebih berkaitan dengan aspek-aspek umum dari sistem tersebut.

a. Kebutuhan Perangkat Keras :

   Untuk membangun aplikasi, baik perangkat keras administrator sistem maupun perangkat keras pengguna, perlu dipilih dengan cermat untuk memastikan aplikasi berjalan dengan baik dan optimal. Berikut adalah contoh spesifikasi perangkat keras laptop untuk administrator sistem:

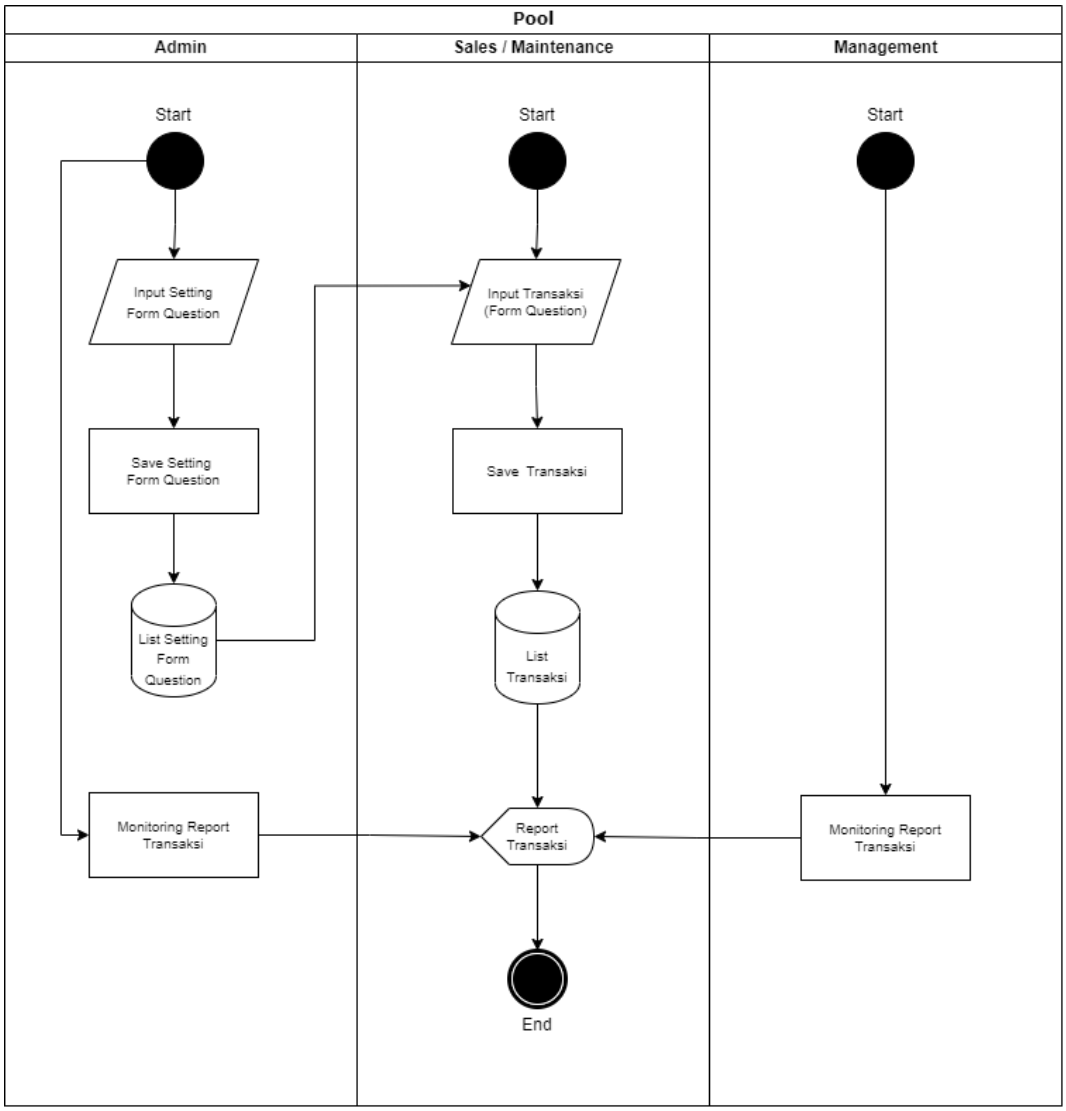
   i. Prosesor Intel (R) Core (TM) i5-1135G7 CPU
   ii. RAM 12 GB
   iii. Harddisk 1 TB
   iv. VGA Intel iRIS xe
   v. Perangkat standar input dan output

b. Kebutuhan Perangkat Lunak

   i. Browser Internet (Google Chrome atau Mozilla Firefox).
   ii. XAMPP Control Panel v7.3.9.
   iii. Laravel 8.
   iv. MySQL.
   v. Text Editor Notepad++.

## 3.2 Perancangan

Dalam melakukan perancangan sistem, model perancangan yang digunakan adalah perancangan berorientasi proses menggunakan *FlowChart*.



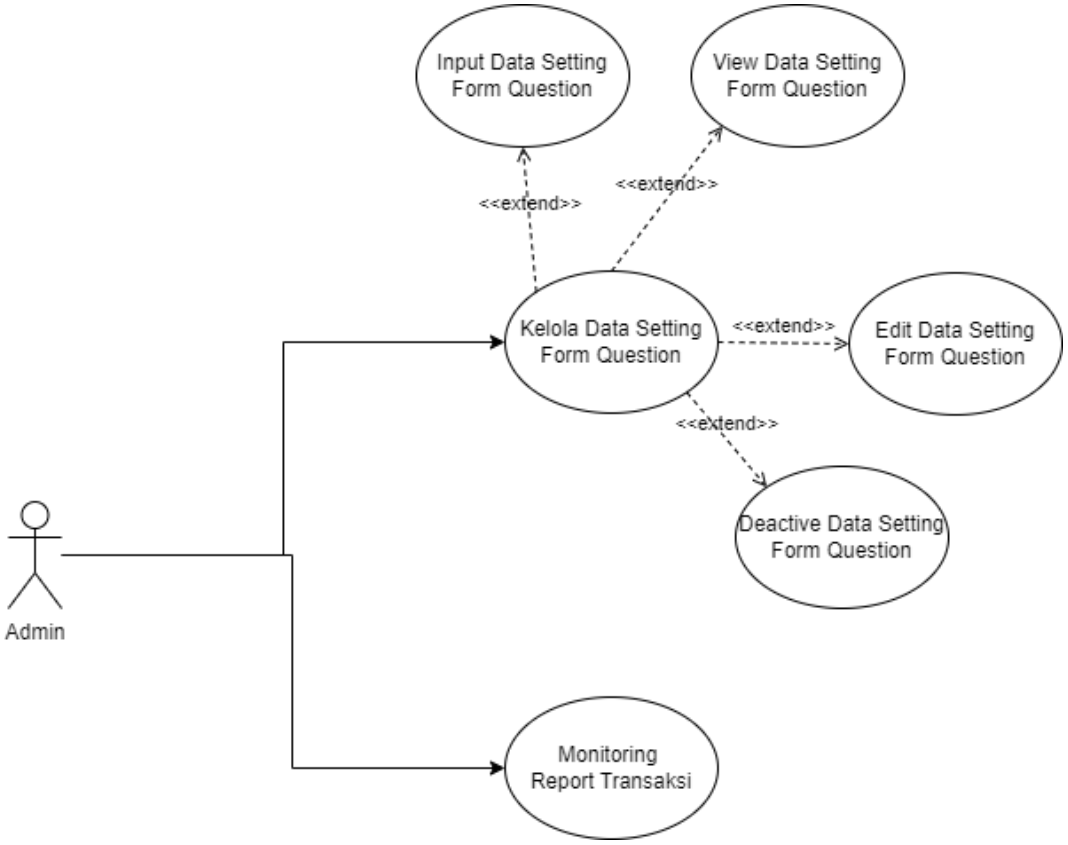

Gambar 3.1.1 *Flow Chart*

### 3.2.1 Perancangan Sistem

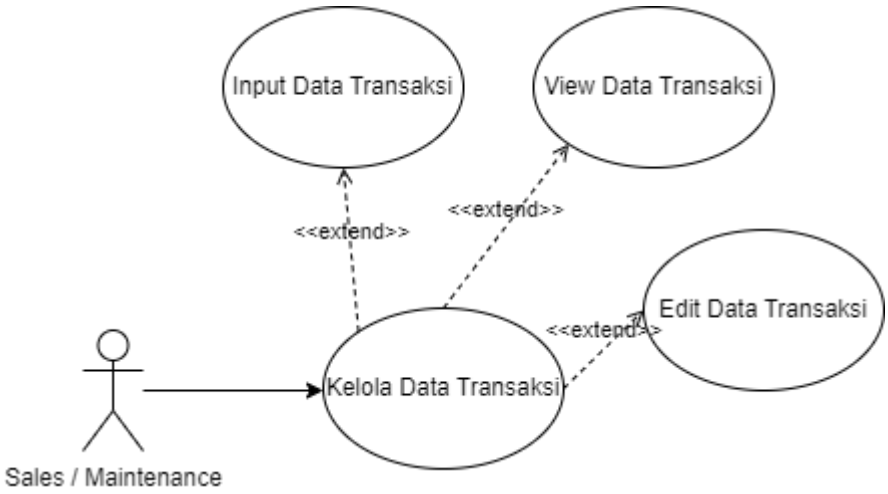
Perancangan sistem menggunakan UML *(Unified Modeling Language)*, yang merupakan metode pemodelan berorientasi objek. Diagram UML yang digunakan untuk perancangan ini adalah *use case diagram, sequence diagram* dan *class diagram.*

#### 3.2.1.1 Use Case Diagram

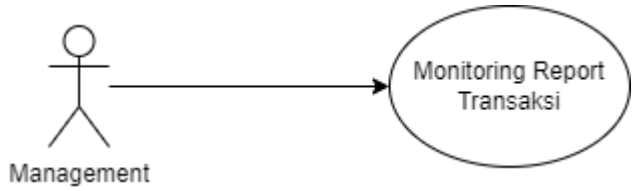
*Use case diagram* merupakan pemodelan yang digunakan untuk menggambarkan secara singkat hubungan antara *use case*, aktor dan sistem. Di dalam *use case* akan diketahui fungsi-fungsi yang ada di dalam sistem.




Gambar 3.1.2 *Use Case Diagram Admin*




Gambar 3.1.3 *Use Case Sales / Maintenance*



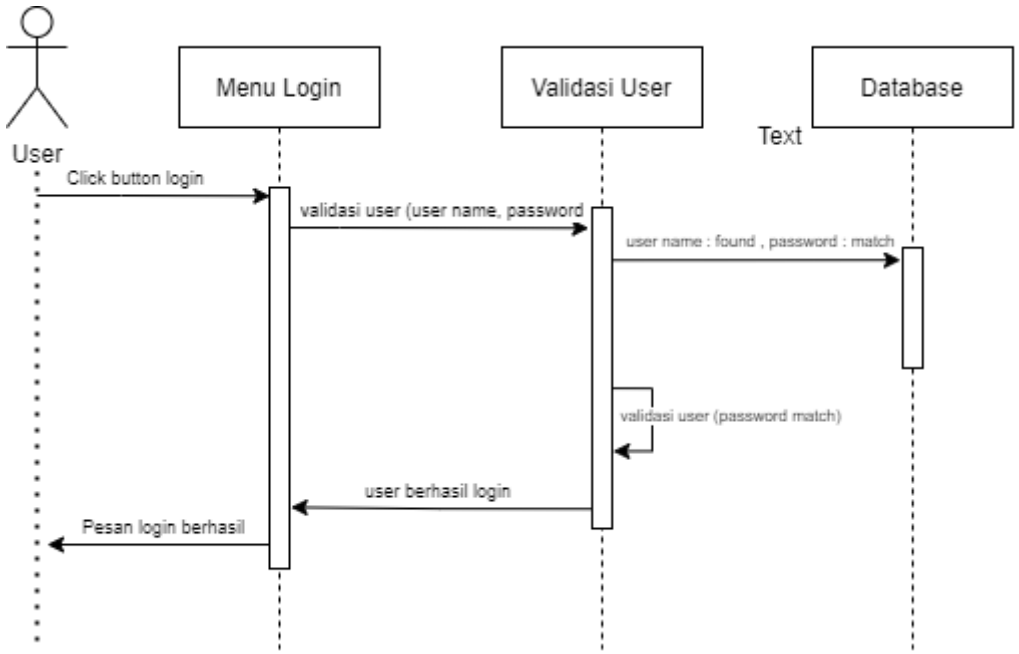

Gambar 3.1.4 *Use Case Management*

#### 3.2.1.2 Sequence Diagram

Pada *sequence diagram* dibawah ini menampilkan interaksi antar objek di dalam aplikasi pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut termasuk *user*, *display* dan sebagainya berupa pesan / *message.*

##### 3.2.1.2.1 Login

Di bawah ini merupakan *sequence diagram* untuk melakukan *login.* Proses *login* dilakukan oleh *user.*

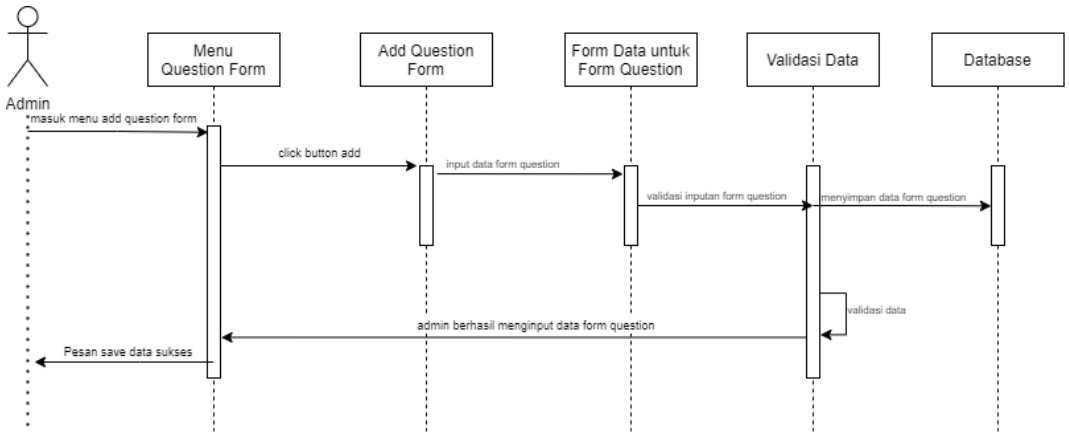



Gambar 3.1.4 *Sequence diagram login*

Gambar 3.1.4 Merupakan gambar *sequence login.* Proses *login* dilakukan oleh *user* ketika akan masuk ke dalam aplikasi. Pada saat membuka aplikasi, halaman yang ditampilkan adalah halaman *login.* Halaman *login* menampilkan kolom *username* dan *password.* Kemudian *user* akan meng*input*kan *username* dan *password,* dan sistem akan mem-validasi hasil *input.* Jika data valid maka akan masuk ke halaman utama dan jika tidak valid maka akan kembali ke *form login.*

#### 3.2.1.2.2 Add Question Form

Di bawah ini merupakan *sequence diagram* untuk melakukan aktivitas *add question form.* Proses tambah *add question form* dilakukan oleh *Admin.*

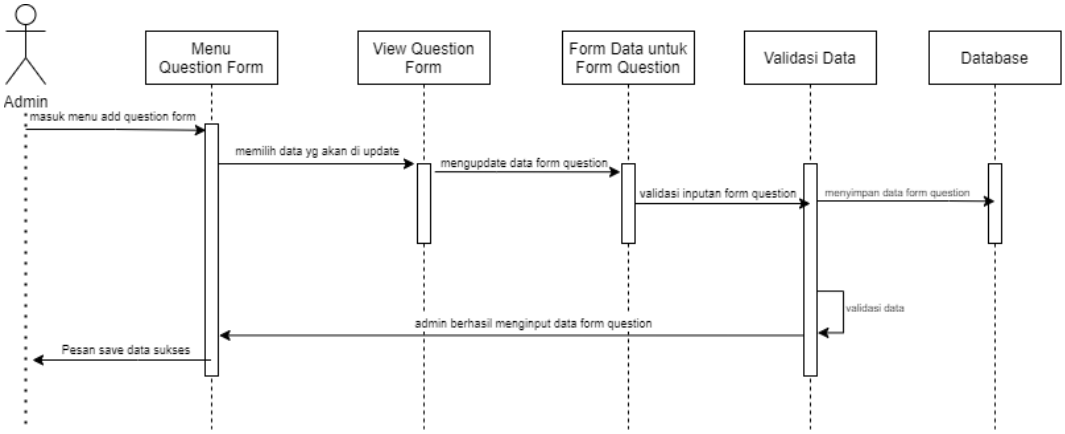



Gambar 3.1.5 *sequence diagram add question form*

Gambar 3.15 Merupakan gambar *sequence diagram add question form.* Proses *add question form* dilakukan oleh *admin* ketika berada pada menu *question form,* setelah itu menekan tombol *add question form* dan akan muncul *form* data untuk *form question* yang akan diisi oleh *admin* sesuai dengan data *form* yang akan di *input* kan. Apabila *inputan* data benar maka akan di-validasi dan disimpan ke dalam *database* dan akan kembali ke menu *form question.* Sebaliknya jika inputan data salah maka akan kembali ke menu *form* data *form question*

#### 3.2.1.2.3 Update Data Question Form

Di bawah ini merupakan *sequence diagram* untuk melakukan aktivitas mengubah data *form question.* Proses ubah *form question* dilakukan oleh *admin.*

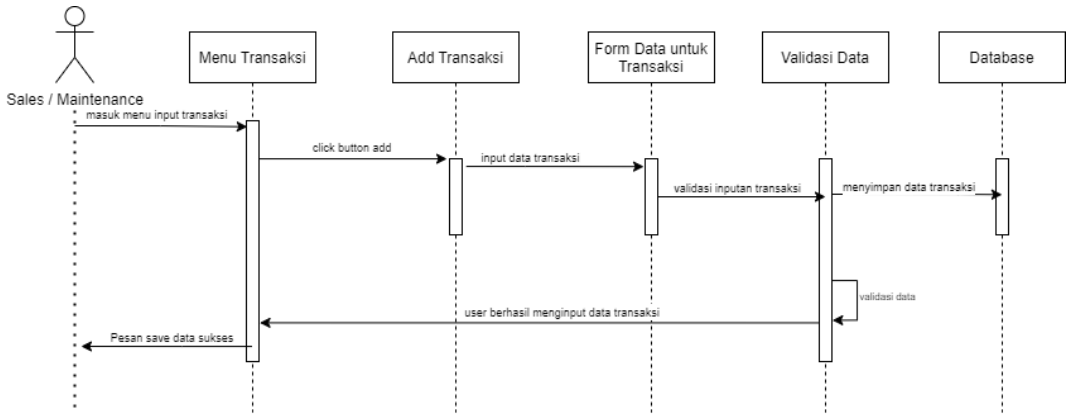



Gambar 3.1.6 *Sequence update data form question*

Gambar 3.1.6 Merupakan gambar *sequence diagram* ubah data *form question.* Proses ubah data *form question* dilakukan oleh *admin* ketika berada pada menu *question form*, setelah itu memilih *question form* yang akan diubah datanya. Apabila telah dipilih, maka akan muncul *form* data *question form* yang akan diubah oleh *admin*, kemudian *admin* akan mengubah data sesuai dengan yang diinginkan. Apabila *inputan* data benar maka akan tervalidasi dan disimpan ke dalam *database* dan akan kembali ke menu *question form.* Sebaliknya jika inputan data salah maka akan kembali ke menu *form* data *form question.*

#### 3.2.1.2.4 Input Data Transaksi

Di bawah ini merupakan *sequence diagram* untuk melakukan aktivitas input data transaksi. Proses input data transaksi dilakukan oleh *Sales / Maintenance.*

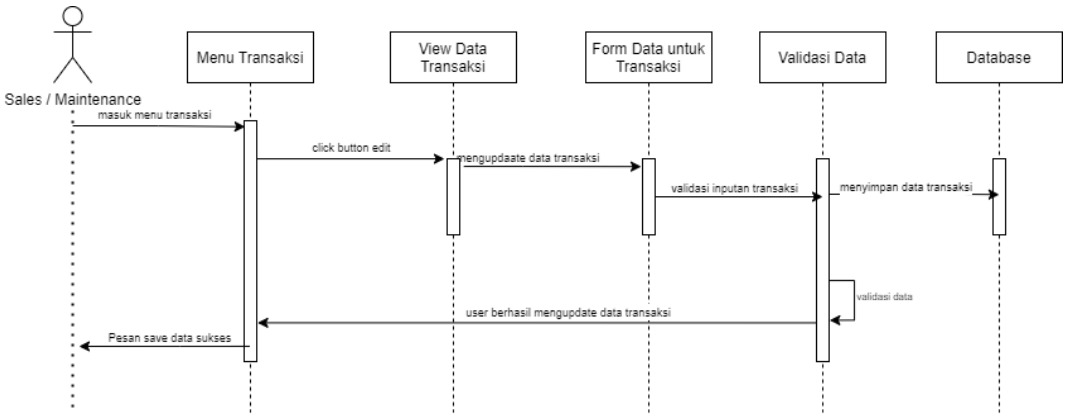



Gambar 3.1.7 *Sequence diagram input* data transaksi

Gambar 3.1.7 Merupakan gambar *sequence diagram input* data transaksi. Proses *input* data transaksi dilakukan oleh *sales / maintenance* ketika berada pada menu transaksi, setelah itu menekan tombol *add* transaksi dan akan muncul *form* data untuk transaksi yang akan diisi oleh *user* sesuai dengan data *transaksi* yang akan di *input* kan. Apabila *inputan* data benar maka akan di-validasi dan disimpan ke dalam *database* dan akan kembali ke menu transaksi. Sebaliknya jika inputan data salah maka akan kembali ke menu *form* data transaksi.

#### 3.2.1.2.5 Update Data Transaksi

Di bawah ini merupakan *sequence diagram* untuk melakukan aktivitas merubah data transaksi. Proses merubah data transaksi dilakukan oleh *Sales / Maintenance.*

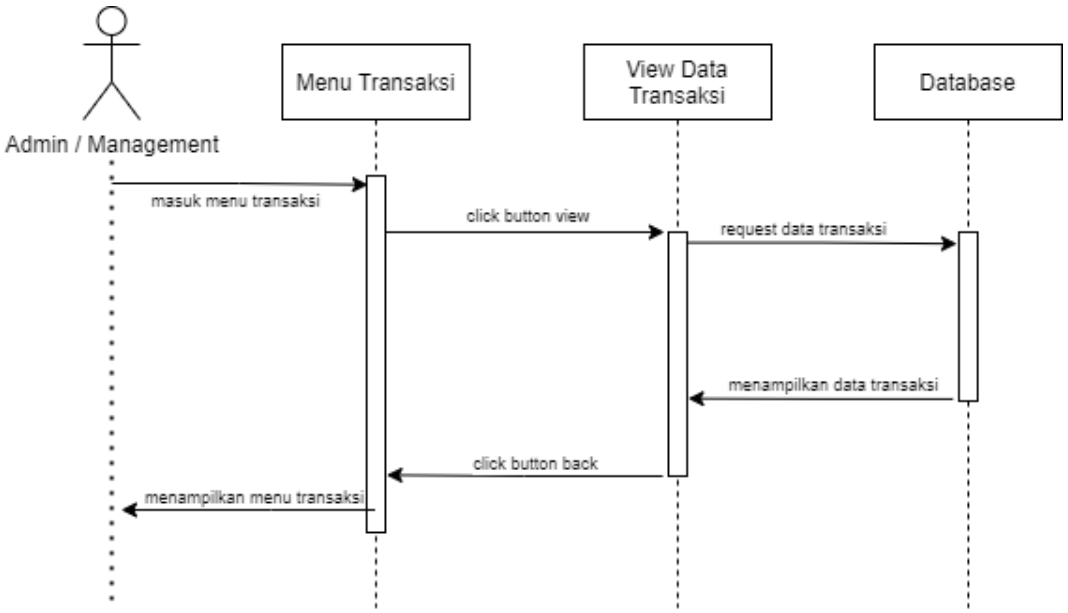



Gambar 3.1.8 *Sequence diagram update* data transaksi

Gambar 3.1.8 Merupakan gambar *sequence diagram* ubah data transaksi. Proses ubah data transaksi dilakukan oleh *sales / maintenance* ketika berada pada menu transaksi, setelah itu memilih data transaksi yang akan diubah datanya. Apabila telah dipilih, maka akan muncul *form* data transaksi yang akan diubah oleh *sales / maintenance*, kemudian *sales / maintenance* akan mengubah data sesuai dengan yang diinginkan. Apabila *inputan* data benar maka akan tervalidasi dan disimpan ke dalam *database* dan akan kembali ke menu transaksi. Sebaliknya jika inputan data salah maka akan kembali ke menu *form* data transaksi.

#### 3.2.1.2.6 View Transaksi

Di bawah ini merupakan *sequence diagram* untuk melakukan aktivitas *view* data transaksi. Proses *view* transaksi dilakukan oleh *Admin* dan *Management.*




Gambar 3.1.9 *Sequence Diagram view* transaksi

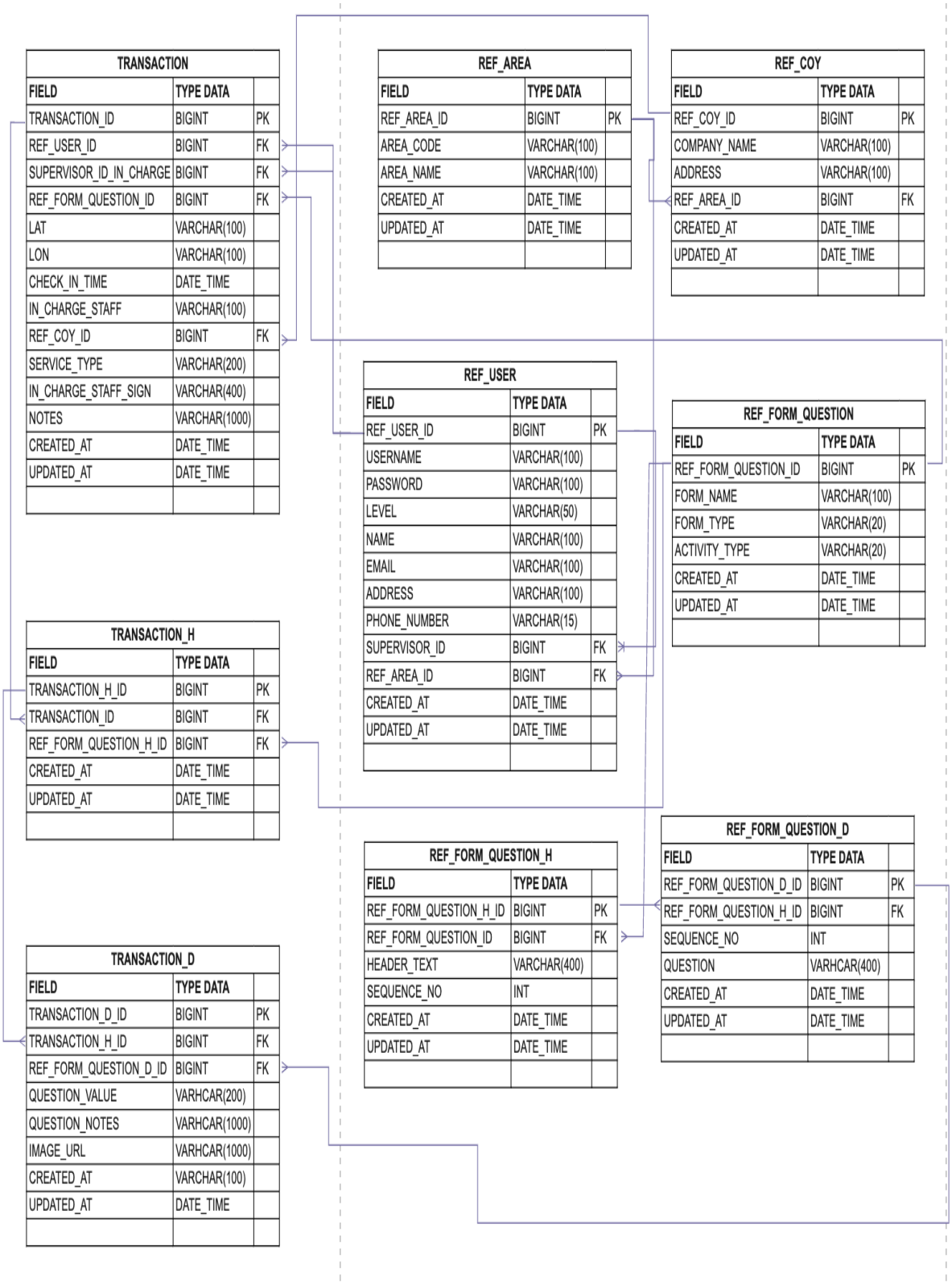
Gambar 3.1.9 Merupakan gambar *sequence view* transaksi. Proses *view* transaksi dilakukan oleh *Admin* dan *Management* ketika berada pada menu transaksi, setelah itu menekan button *view*. Setelah dipilih maka akan masuk ke menu *view* data transaksi di mana di menu tersebut menampilkan seluruh data transaksi yang ditambahkan dan tersimpan dalam *database*.

### 3.2.2 Perancangan Basis Data

Perancangan basis data menggunakan ERD (*Entity Relationship Diagram*) dan struktur tabel.

#### 3.2.2.1 Entity Relationship Diagram

Gambar 3.25 Dibawah ini merupakan gambar *entity relationship diagram* pada “Implementasi *Dynamic Form* Pada aplikasi *Sales Activity Tracking* Menggunakan Laravel”.




Gambar 3.2.0 *Entity Relationship Diagram*

#### 3.2.2.2 Struktur Tabel

Kebutuhan tabel dalam sistem informasi digambarkan pada Gambar 3.25 dengan hubungan antara tabel. Rincian tabel yang dibutuhkan pada sistem informasi adalah sebagai berikut :

1. REF_AREA, berfungsi sebagai penyimpanan data pembagian area.

Tabel 3.1 Struktur tabel REF_AREA

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|
| REF_AREA_ID | INT | PK |
| AREA_CODE | VARCHAR(100) | |
| AREA_NAME | VARCHAR(100) | |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |

2. REF_COY, berfungsi sebagai tabel penyimpanan data client.

Tabel 3.2 Struktur tabel REF_COY

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|
| REF_COY_ID | INT | PK |
| COMPANY_NAME | VARCHAR(100) | |
| ADDRESS | VARCHAR(100) | |
| REF_AREA_ID | INT | FK |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |

3. REF_USER, berfungsi sebagai tabel penyimpanan user.

Tabel 3.3 Struktur tabel REF_USER.

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|
| REF_USER_ID | INT | PK |
| USERNAME | VARCHAR(100) | |
| PASSWORD | VARCHAR(100) | |
| LEVEL | VARCHAR(50) | |
| NAME | VARCHAR(100) | |
| EMAIL | VARCHAR(100) | |
| ADDRESS | VARCHAR(100) | |
| PHONE_NUMBER | VARCHAR(15) | |
| SUPERVISOR_ID | INT | FK(REF_USER) |
| REF_AREA_ID | INT | FK |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |

4. REF_FORM_QUESTION, berfungsi sebagai tabel penyimpanan data *form question*.

Tabel 3.4 Struktur tabel REF_FORM_QUESTION.

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|
| REF_FORM_QUESTION_ID | INT | PK |
| FORM_NAME | VARCHAR(100) | |
| FORM_TYPE | VARCHAR(20) | |
| ACTIVITY_TYPE | VARCHAR(20) | |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |

5. REF_FORM_QUESTION_H, berfungsi sebagai tabel penyimpanan data *header* dari *form question*.

Tabel 3.5 Struktur tabel REF_FORM_QUESTION_H.

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|
| REF_FORM_QUESTION_H_ID | INT | PK |
| REF_FORM_QUESTION_ID | INT | FK |
| HEADER_TEXT | VARCHAR(400) | |
| SEQUENCE_NO | INT | |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |

6. REF_FORM_QUESTION_D, berfungsi sebagai tabel penyimpanan data *detail* dari *form question*.

Tabel 3.6 Struktur tabel REF_FORM_QUESTION_D.

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|
| REF_FORM_QUESTION_D_ID | INT | PK |
| REF_FORM_QUESTION_H_ID | INT | FK |
| SEQUENCE_NO | INT | |
| QUESTION | VARCHAR(400) | |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |

7. TRANSACTION, berfungsi sebagai tabel penyimpanan data *transaction*.

Tabel 3.7 Struktur tabel TRANSACTION.

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|

| | | |
|---|---|---|
| TRANSACTION_ID | INT | PK |
| REF_USER_ID | INT | FK |
| SUPERVISOR_ID_IN_CHARGE | INT | FK |
| REF_FORM_QUESTION_ID | INT | FK |
| LAT | VARCHAR(100) | |
| LON | VARCHAR(100) | |
| CHECK_IN_TIME | DATE_TIME | |
| IN_CHARGE_STAFF | VARCHAR(100) | |
| REF_COY_ID | INT | FK |
| NOTES | VARCHAR(1000) | |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |

8. TRANSACTION_H, berfungsi sebagai tabel penyimpanan data *header* dari *transaction*.

Tabel 3.8 Struktur tabel TRANSACTION_H.

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|
| TRANSACTION_H_ID | INT | PK |
| TRANSACTION_ID | INT | FK |
| REF_FORM_QUESTION_H_ID | INT | FK |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |

9. TRANSACTION_D, berfungsi sebagai tabel penyimpanan data *detail* dari *transaction*.

Tabel 3.9 Struktur tabel TRANSACTION_D.

| **FIELD** | **TYPE DATA** | **NOTES** |
|---|---|---|
| TRANSACTION_D_ID | INT | PK |

| | | |
|---|---|---|
| TRANSACTION_H_ID | INT | FK |
| REF_FORM_QUESTION_D_ID | INT | FK |
| QUESTION_VALUE | VARCHAR(200) | |
| QUESTION_NOTES | VARCHAR(1000) | |
| IMAGE_URL | VARCHAR(1000) | |
| CREATED_AT | DATE_TIME | |
| UPDATED_AT | DATE_TIME | |