BUKU PANDUAN

APLIKASI *MOBILE* BERBASIS ANDROID PENJUALAN SEMBAKO DI CV NURHALIM JAYA TEKSTIL

2024

KATA PENGANTAR

Puji dan syukur kita panjatkan ke hadirat Allah SWT atas segala rahmat dan karunia-Nya yang telah diberikan sehingga buku panduan penggunaan aplikasi penjualan sembako dapat selesai dibuat. Panduan ini disusun untuk membantu CV Nurhalim Jaya Tekstil dalam menggunakan aplikasi *mobile* berbasis Android penjualan sembako. Aplikasi ini diharapkan mampu membantu tim penjualan dalam melakukan kegiatan operasionalnya dalam menawarkan dan menjual barang sembako.

Kami menyadari bahwa panduan ini mungkin tidak sempurna. Oleh karena itu, saran dan kritik yang membangun sangat kami harapkan guna perbaikan dan penyempurnaan di masa yang akan datang. Terima kasih kepada seluruh pihak yang telah berkontribusi dalam penyusunan panduan ini. Semoga buku panduan ini bermanfaat bagi seluruh pengguna.

Tim Penyusun

Dimas Wahyu Ardiyanto

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Bab pendahuluan ini membahas mengenai deskripsi umum aplikasi, aktor pengguna aplikasi, spesifikasi perangkat, dan fitur aplikasi.

1.1 Deskripsi Umum

Aplikasi penjualan sembako di CV Nurhalim Jaya Tekstil memiliki nama JJ Sembako. Aplikasi JJ Sembako terdiri atas dua platform. Kedua platform tersebut adalah aplikasi berbasis *website* dan aplikasi *mobile* berbasis Android. Setiap platform memiliki kegunaannya masing-masing.

Aplikasi *mobile* berbasis Android penjualan sembako adalah aplikasi yang dibuat untuk membantu tim penjualan dari CV Nurhalim Jaya Tekstil dalam melakukan tugas operasionalnya menawarkan dan menjual sembako. Aplikasi ini dapat membantu tim penjualan dalam mengetahui keadaan stok gudang secara *real-time*. Selain itu, data pesanan pelanggan juga terintegrasi secara langsung sehingga memudahkan dalam urusan dokumentasi.

1.2 Aktor Pengguna

Aplikasi JJ Sembako secara umum memiliki tiga aktor utama, yaitu *owner* (pemilik usaha), *admin* (administrator gudang alias operator), dan *sales* (tim penjualan). Aktor tersebut didapatkan dengan menyesuaikan kebiasaan penyebutan dalam kegiatan penjualan sembako di CV Nurhalim Jaya Tekstil. Aplikasi *mobile* berbasis Android penjualan sembako hanya memiliki akses untuk satu jenis aktor pengguna. Aktor tersebut adalah *sales*.

1.3 Spesifikasi Perangkat

Aplikasi *mobile* berbasis Android penjualan sembako memiliki spesifikasi minimum untuk dapat digunakan. Aplikasi ini hanya dapat dijalankan untuk *smartphone* atau *tablet* dengan sistem operasi Android. Sistem operasi Android yang digunakan minimal versi 5.0 (Lollipop), tetapi disarankan menggunakan versi 8.0

(Oreo) ke atas untuk pengalaman penggunaan yang lebih baik. Selain itu, aplikasi ini hanya memerlukan izin penggunaan koneksi internet saja.

1.4 Fitur Aplikasi

Aplikasi *mobile* berbasis Android penjualan sembako memiliki beberapa fitur untuk menunjang kegiatan operasional tim penjualan. Fitur aplikasi ini dirangkum pada Tabel 1.1. Berikut adalah fitur aplikasi ini:

Tabel 1.1 Fitur Aplikasi

No.	Fitur Secara Umum
1.	Mengelola akun
2.	Mengetahui informasi barang di gudang
3.	Mengetahui informasi performa
4.	Mengelola data pelanggan
5.	Mengelola pesanan

BAB II
MANUAL PENGGUNAAN

Bab panduan penggunaan ini membahas mengenai cara menggunakan aplikasi *mobile* berbasis Android penjualan sembako untuk CV Nurhalim Jaya Tekstil. Pada bab ini hanya memberikan beberapa panduan secara umum.

2.1 *Login*

Aplikasi ini hanya dapat diakses menggunakan akun khusus. Akun tersebut memiliki peran sebagai *sales* yang hanya dapat dibuat oleh *admin* melalui aplikasi pada platform *website.* Langkah-langkah untuk melakukan *login* sebagai berikut:

1. Buka aplikasi dan untuk pertama kalinya akan diarahkan ke tampilan *login.* Tampilan *login* dapat dilihat pada Gambar 2.1.

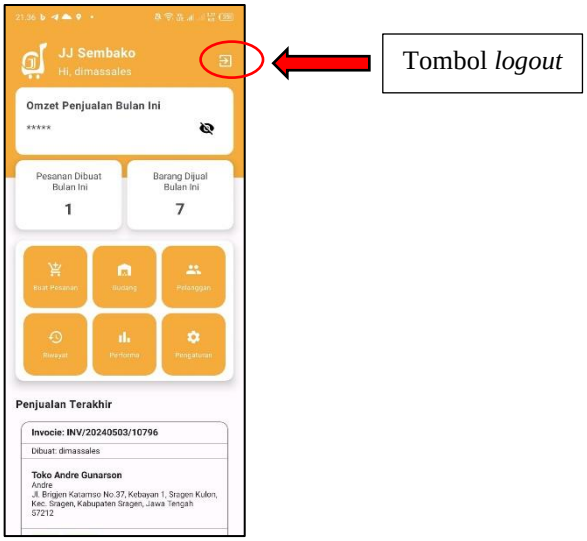
Gambar 2.1 Tampilan *Login*

2. Isi *username* dan password.
3. Tekan tombol masuk. Jika benar, pengguna akan diarahkan ke tampilan *home*. Jika salah, pesan peringatan *error* akan muncul. Tampilan *home* dapat dilihat pada Gambar 2.2.

2.2 *Logout*

Aplikasi ini tidak memiliki batasan waktu untuk dapat menggunakan aplikasi. Oleh karena itu, pengguna harus melakukan *logout* jika ingin keluar dari aplikasi atau untuk berganti akun. Langkah-langkah untuk melakukan *logout* sebagai berikut:

1. Pastikan berada di tampilan *home*.



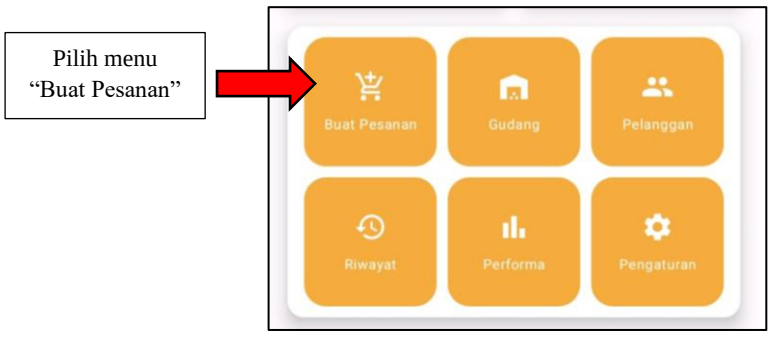
Gambar 2.2 Tampilan *Home*

2. Tekan tombol dengan ikon keluar yang terletak pada bagian kanan atas.
3. Pengguna akan langsung diarahkan ke tampilan *login*.

2.3 Membuat Pesanan Baru

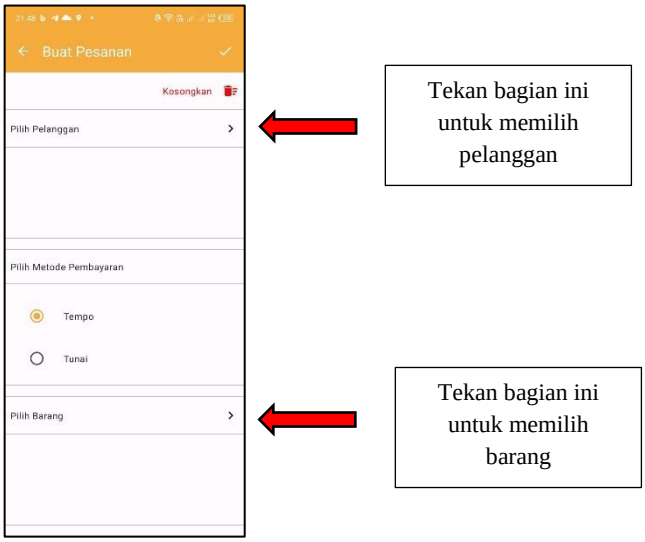
Pembuatan pesanan baru di aplikasi ini memerlukan beberapa langkah-langkah tahapan yang harus diikuti. Langkah-langkah tersebut sebagai berikut:

1. Pastikan berada di tampilan *home*.



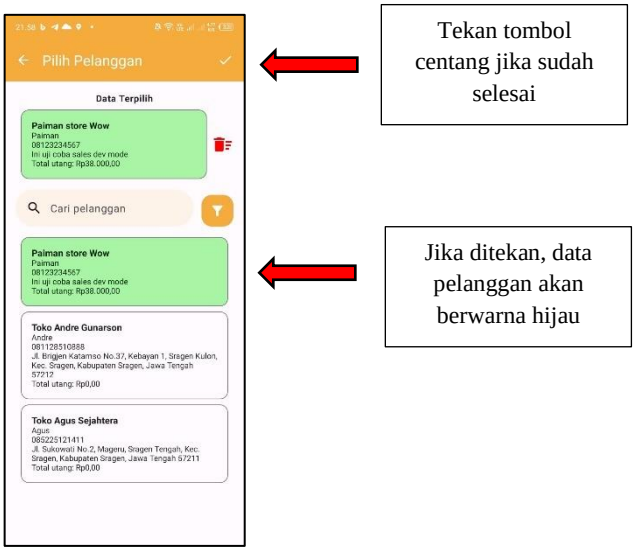
Gambar 2.3 Pilih Menu "Buat Pesanan"

2. Pilih menu “Buat Pesanan” sesuai petunjuk Gambar 2.3. Pengguna akan diarahkan ke tampilan buat pesanan. Tampilan buat pesanan dapat dilihat pada Gambar 2.4.



Gambar 2.4 Tampilan Buat Pesanan

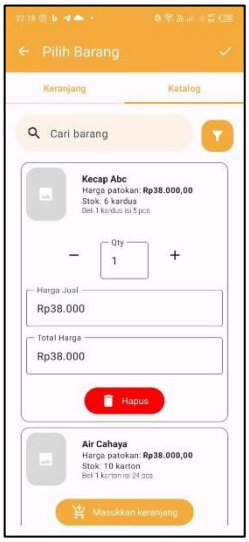
3. Pilih bagian “Pilih Pelanggan”. Pengguna akan diarahkan ke tampilan pilih pelanggan. Tampilan pilih pelanggan dapat dilihat pada Gambar 2.5.



Gambar 2.5 Tampilan Pilih Pelanggan

4. Pilih salah satu pelanggan dengan cara ditekan. Jika sudah, tombol centang pada bagian pojok kanan atas dapat ditekan untuk menyimpan data pelanggan. Selain itu, pengguna akan diarahkan kembali ke tampilan buat pesanan.

5. Pada tampilan buat pesanan, pilih bagian “Pilih Barang”. Pengguna akan diarahkan ke tampilan pilih barang. Tampilan pilih barang dapat dilihat pada Gambar 2.6.



Gambar 2.6 Tampilan Pilih Barang

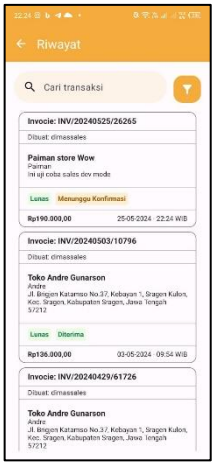
6. Arahkan ke *tab* katalog. Pilih barang dengan cara menekan tombol “Masukkan keranjang”. Lengkapi data pesanan barang. Jika sudah, tombol centang pada bagian pojok kanan atas dapat ditekan untuk menyimpan data barang pesanan. Selain itu, pengguna akan diarahkan kembali ke tampilan buat pesanan.
7. Pilih metode pembayaran. Pembayaran pesanan hanya bisa dilakukan secara tunai maupun tempo (utang). Pembayaran secara tempo hanya bisa dilakukan jika pelanggan tidak memiliki utang.
8. Tekan tombol centang pada halaman buat pesanan. Jika sukses, pengguna akan diarahkan ke tampilan detail transaksi. Jika gagal, pesan peringatan *error* akan muncul.

2.4 Melihat Daftar Riwayat Pesanan Yang Telah Dibuat

Melihat daftar riwayat pesanan yang telah dibuat di aplikasi ini memerlukan beberapa langkah-langkah tahapan yang harus diikuti. Langkah-langkah tersebut sebagai berikut:

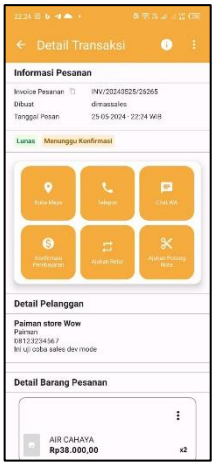
Gambar 2.7 Pilih Menu "Riwayat"

1. Pastikan berada di tampilan *home*.



Gambar 2.8 Tampilan Riwayat

2. Pilih menu “Riwayat” sesuai petunjuk Gambar 2.7. Pengguna akan diarahkan ke tampilan riwayat. Tampilan riwayat dapat dilihat pada Gambar 2.8.

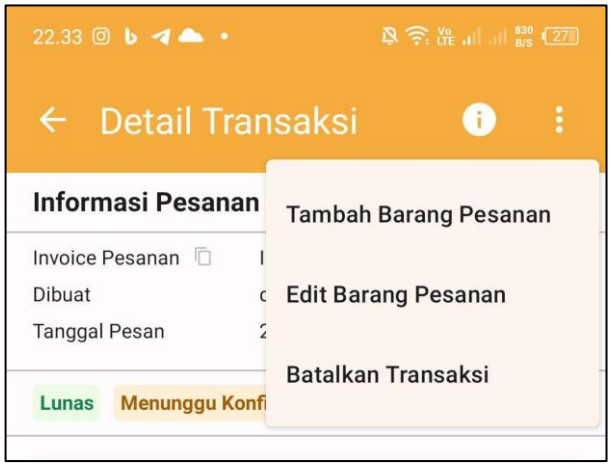


Gambar 2.9 Tampilan Detail Transaksi

3. Jika ingin melihat detail pesanan, tekan salah satu data pesanan. Pengguna akan diarahkan ke tampilan detail transaksi. Tampilan detail transaksi dapat dilihat pada Gambar 2.9.

2.5 Menambah Barang Baru Ke Pesanan Yang Telah Dibuat

Menambah barang baru ke pesanan yang telah dibuat di aplikasi ini memerlukan beberapa langkah-langkah tahapan yang harus diikuti. Langkah-langkah tersebut sebagai berikut:



Gambar 2.10 Isi Menu Titik Tiga Di Tampilan Detail Transaksi

1. Pastikan berada di tampilan detail transaksi.
2. Pilih menu titik tiga yang berada di bagian pojok kanan atas. Opsi menu seperti pada Gambar 2.10 akan muncul.

Gambar 2.11 Tampilan Tambah Barang Pesanan

3. Kemudian, pilih menu “Tambah Barang Pesanan”. Pengguna akan diarahkan ke tampilan tambah barang pesanan. Tampilan tambah barang pesanan dapat dilihat pada Gambar 2.11.
4. Arahkan ke *tab* katalog. Pilih salah satu barang dengan cara menekan tombol “Pilih”.
5. Lengkapi data barang pesanan yang baru.
6. Jika sudah, tombol centang pada bagian pojok kanan atas dapat ditekan untuk menyimpan data. Jika berhasil, pengguna akan diarahkan kembali ke tampilan detail transaksi.

2.6 Mengubah Data Barang Pesanan Yang Telah Dibuat

Mengubah data barang pesanan yang telah dibuat di aplikasi ini memerlukan beberapa langkah-langkah tahapan yang harus diikuti. Langkah-langkah tersebut sebagai berikut:

1. Pastikan berada di tampilan detail transaksi.
2. Pilih menu titik tiga yang berada di bagian pojok kanan atas. Opsi menu seperti pada Gambar 2.10 akan muncul.
3. Kemudian, pilih menu “Edit Barang Pesanan”. Pengguna akan diarahkan ke tampilan edit barang pesanan. Tampilan edit barang pesanan dapat dilihat pada Gambar 2.12.

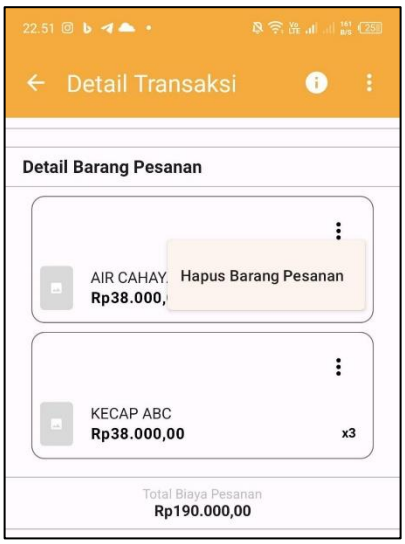
Gambar 2.12 Tampilan Edit Barang Pesanan

4. Arahkan ke tab katalog. Pilih salah satu barang dengan cara menekan tombol “Pilih”.
5. Perbarui data barang pesanan.
6. Jika sudah, tombol centang pada bagian pojok kanan atas dapat ditekan untuk menyimpan data. Jika berhasil, pengguna akan diarahkan kembali ke tampilan detail transaksi

2.7 Menghapus Barang Pesanan Yang Telah Dibuat

Menghapus barang pesanan yang telah dibuat di aplikasi ini memerlukan beberapa langkah-langkah tahapan yang harus diikuti. Langkah-langkah tersebut sebagai berikut:

1. Pastikan berada di tampilan detail transaksi.
2. Gulir tampilan hingga bagian detail barang pesanan. Bagian ini dapat dilihat pada Gambar 2.13.
3. Pilih menu titik tiga yang berada di bagian pojok kanan atas item barang. Opsi menu seperti pada Gambar 2.13 akan muncul.



Gambar 2.13 Tampilan Konfirmasi Hapus Pesanan

4. Kemudian, pilih menu “Hapus Barang Pesanan”. Tampilan konfirmasi akan muncul. Tampilan konfirmasi dapat dilihat pada Gambar 2.14.

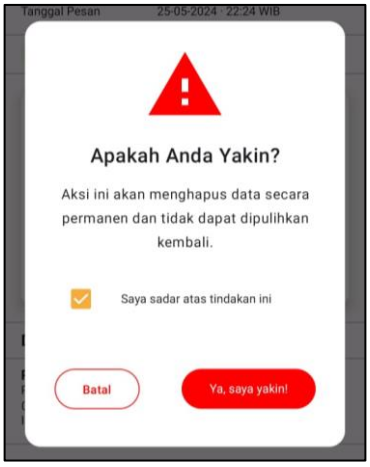
Gambar 2.14 Tampilan Konfirmasi Hapus Barang Pesanan

5. Lakukan konfirmasi tindakan dan tekan tombol “Ya, saya yakin”.
6. Jika berhasil, pesan sukses akan muncul.

2.8 Menghapus Pesanan Yang Telah Dibuat

Menghapus pesanan yang telah dibuat di aplikasi ini memerlukan beberapa langkah-langkah tahapan yang harus diikuti. Langkah-langkah tersebut sebagai berikut:

1. Pastikan berada di tampilan detail transaksi.
2. Pilih menu titik tiga yang berada di bagian pojok kanan atas. Opsi menu seperti pada Gambar 2.15 akan muncul.



Gambar 2.15 Tampilan Konfirmasi Hapus Pesanan

3. Kemudian, pilih menu “Batalkan Transaksi”. Tampilan konfirmasi akan muncul. Tampilan konfirmasi dapat dilihat pada Gambar 2.15.
4. Lakukan konfirmasi tindakan dan tekan tombol “Ya, saya yakin”.
5. Jika berhasil, pengguna akan diarahkan ke tampilan riwayat jika pesanan yang dihapus berasal dari data yang dipilih dari tampilan riwayat.

BAB III
SOURCE CODE

Bab *source code* ini membahas mengenai beberapa bagian kode program yang terdapat dalam aplikasi. Kode program yang dijabarkan merupakan bagian penting dalam aplikasi yang dihasilkan. Selain itu, bab ini juga membahas mengenai sekilas gambaran mengenai pengembangan aplikasi.

3.1 Pengembangan Aplikasi

Aplikasi *mobile* berbasis Android penjualan sembako dibuat menggunakan *Integrated Development Environment* (IDE) Android Studio. Tampilan antarmuka aplikasi ini dikembangkan menggunakan *toolkit user interface* (UI) Jetpack Compose yang berbasis bahasa pemrograman Kotlin. Selain itu, aplikasi ini juga menggunakan pola desain *Model-View-ViewModel* (MVVM) untuk membuat pengembangan aplikasi menjadi lebih terstruktur dan mudah dipelihara.

3.2 Kebutuhan Fungsional Aplikasi

Kebutuhan fungsional dari aplikasi dari aplikasi *mobile* berbasis Android penjualan sembako dapat dilihat pada Tabel 3.1.

Tabel 3.1 Kebutuhan Fungsional Aplikasi

No.	ID SRS	Fitur Secara Umum
1.	SRS-PSM-01	Mengelola akun
2.	SRS-PSM-02	Mengetahui informasi barang di gudang
3.	SRS-PSM-03	Mengetahui informasi performa
4.	SRS-PSM-04	Mengelola data pelanggan
5.	SRS-PSM-05	Mengelola pesanan

3.3 *Source Code* Bagian *Model*

Model dalam pengembangan aplikasi ini lebih berkaitan dengan tipe data abstrak. *Model* ini dibuat menggunakan *keyword data class* milik bahasa pemrograman Kotlin. Berikut adalah *source code* salah satu *model* yang telah dibuat:

```
data class Order(
    @field:SerializedName("id")
    val id: String,
    @field:SerializedName("invoice")
    val invoice: String,
    @field:SerializedName("order_status")
    val orderStatus: Int,
    @field:SerializedName("payment_status")
    val paymentStatus: Int,
    @field:SerializedName("total_price")
    val totalPrice: Long,
    @field:SerializedName("actual_total_price")
    val actualTotalPrice: Long,
    @field:SerializedName("created_at")
    val createdAt: String,
    @field:SerializedName("updated_at")
    val updatedAt: String,
    @field:SerializedName("deliver_at")
    val deliverAt: String? = null,
    @field:SerializedName("finished_at")
    val finishedAt: String? = null,
    @field:SerializedName("account")
    val account: Account,
    @field:SerializedName("customer")
    val customer: Customer,
    @field:SerializedName("orderToProducts")
    val orderToProducts: List<OrderedProduct>,
    @field:SerializedName("retur")
    val retur: Int,
    @field:SerializedName("canceled")
    val canceled: Int
)
```

3.4 *Source Code* Bagian *View Model*

View Model dalam pengembangan aplikasi ini berkaitan dengan pengelolaan manipulasi data yang ditampilkan ke tampilan antarmuka. *View Model* ini dibuat menggunakan *keyword class* milik bahasa pemrograman Kotlin. Berikut adalah cuplikan bagian *view model* yang telah dibuat:

```
@HiltViewModel
class DetailTransaksiViewModel @Inject constructor(
    private val canceledStore: DataStore<CanceledStore>,
    private val returStore: DataStore<ReturStore>,
    private val substituteStore: DataStore<SubstituteStore>,
    private val fetchOrderUseCase: FetchOrderUseCase,
    ...
    private val handleDeleteOrderUseCase: HandleDeleteOrderUseCase,
    private val handleDeleteCanceledUseCase: HandleDeleteCanceledUseCase,
    private val handleDeleteReturUseCase: HandleDeleteReturUseCase
) : ViewModel() {
    ...
    fun fetchOrder(id: String) {
        viewModelScope.launch {
            fetchOrderUseCase.fetchOrder(id).collect {
                when (it) {
```

```
                    is Resource.Loading -> {
                        if (orderData.value?.id.isNullOrEmpty())
_stateFirst.value =
                            StateResponse.LOADING
                        else _stateRefresh.value = StateResponse.LOADING
                    }
                    is Resource.Success -> {
                        if (orderData.value?.id.isNullOrEmpty())
_stateFirst.value =
                            StateResponse.SUCCESS
                        else _stateRefresh.value = StateResponse.SUCCESS
                        _message.value = it.message
                        _statusCode.value = it.status
                        _orderData.value = it.data
                    }
                    is Resource.Error -> {
                        if (orderData.value?.id.isNullOrEmpty())
_stateFirst.value =
                            StateResponse.ERROR
                        else _stateRefresh.value = StateResponse.ERROR
                        _message.value = it.message
                        _statusCode.value = it.status
                    }
                }
            }
        }
    }
    ...
}
```

3.5 *Source Code* Bagian *API Service*

API Service dalam pengembangan aplikasi ini berkaitan dengan layanan koneksi ke *endpoint* dari *Application Programming Interface* (API). Bagian ini dibuat menggunakan *keyword interface* milik bahasa pemrograman Kotlin. Berikut adalah cuplikan bagian ini yang telah dibuat:

```
interface OrderApiService {

    @GET("order")
    suspend fun fetchOrders(
        @Query("search") search: String? = null,
        @Query("page") page: Int? = null,
        @Query("limit") limit: Int? = null,
        @Query("minDate") minDate: String? = null,
        @Query("maxDate") maxDate: String? = null,
        @Query("me") me: Int? = null,
        @Query("customerId") customerId: String? = null
    ): GetFetchOrdersResponse
```

```
@GET("order/{id}")
suspend fun fetchOrder(
    @Path("id") id: String
): GetFetchOrderResponse

@POST("order")
suspend fun handleCreateOrder(
    @Body orderRequest: OrderRequest
): PostHandleCreateOrderResponse

@FormUrlEncoded
@POST("order/{id}")
suspend fun handleAddProductOrder(
    @Path("id") id: String,
    @Field("productId") productId: String,
    @Field("amountInUnit") amountInUnit: Int,
    @Field("pricePerUnit") pricePerUnit: Long
): PostHandleAddProductOrderResponse

@FormUrlEncoded
@PATCH("order/{id}/product/{productId}")
suspend fun handleUpdateProductOrder(
    @Path("id") id: String,
    @Path("productId") productId: String,
    @Field("amountInUnit") amountInUnit: Int,
    @Field("pricePerUnit") pricePerUnit: Long
): PatchHandleUpdateProductOrderResponse

@PATCH("order/{id}/payment")
suspend fun handleUpdatePaymentStatus(
    @Path("id") id: String
): PatchHandleUpdatePaymentStatusResponse

@DELETE("order/{id}/product/{productId}")
suspend fun handleDeleteProductOrder(
    @Path("id") id: String,
    @Path("productId") productId: String
): DeleteHandleDeleteProductOrderResponse
```

```
    @DELETE("order/{id}")
    suspend fun handleDeleteOrder(
        @Path("id") id: String
    ): DeleteHandleDeleteOrderResponse
}
```