BAB 5

IMPLEMENTASI

Bab ini menjelaskan mengenai proses implementasi sistem yang telah didesain sebelumnya. Proses-proses implementasi yang dibahas meliputi penjelasan entitas pada *database* dan implementasi proses pencatatan informasi barang dan transaksi pada program. Implementasi sistem dilakukan dengan memanfaatkan *framework* Laravel 8 dan *database* MySQL versi 8.0.

5.1. Implementasi *Database*

Bagian ini menjelaskan mengenai implementasi *database* di MySQL. *Database* yang digunakan terdiri dari 16 tabel yang berkorelasi satu sama lain. Penjelasan dari masing-masing tabel beserta data dapat dijabarkan seperti berikut:

1. Entitas *users*

   Tabel ini digunakan untuk menyimpan data pengguna untuk proses *login*. Terdapat 3 atribut yang digunakan pada tabel ini, yakni terdiri dari *id* sebagai *primary key*, *name* yang merupakan nama akun pengguna serta *password* untuk kata sandi pengguna untuk mengakses sistem. Struktur dan implementasi tabel *users* dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	bigint		UNSIGNED	No	*None*		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_unicode_ci		No	*None*		
3	email	varchar(255)	utf8mb4_unicode_ci		No	*None*		
4	email_verified_at	timestamp			Yes	*NULL*		
5	password	varchar(255)	utf8mb4_unicode_ci		No	*None*		
6	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	*NULL*		
7	created_at	timestamp			Yes	*NULL*		
8	updated_at	timestamp			Yes	*NULL*		

Gambar 5.1. Struktur entitas *users*

2. Entitas *inventory*

Tabel ini digunakan untuk menyimpan data barang perusahaan. Terdapat 8 atribut pada tabel ini yang terdiri dari *kode_brg* atau kode barang, *nama_brg*, *kode_divisi*, *kode_jenis*, *kode_type*, *id_satuan* untuk satuan barang, *hrg_jual* yakni harga jual, dan keterangan barang. Tabel ini berelasi *one to many* dengan tabel *invdivisi, invjenis, invtype,* dan satuan. Struktur dan implementasi dari tabel *inventory* ditunjukkan pada gambar berikut ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	kode_brg	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	nama_brg	varchar(255)	utf8mb4_0900_ai_ci		No	*None*		
3	kode_divisi	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
4	kode_jenis	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
5	kode_type	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
6	id_satuan	int			No	*None*		
7	hrg_jual	float			Yes	*NULL*		
8	keterangan	varchar(255)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.2. Struktur entitas *inventory*

3. Entitas *invdivisi*

Tabel ini digunakan untuk menyimpan daftar divisi barang. Terdapat 2 atribut pada tabel ini. Atribut yang digunakan terdiri dari kode dan nama divisi. Struktur dan implementasi dari tabel *invdivisi* dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	kode	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	divisi	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.3. Struktur entitas *invdivisi*

4. Entitas *invgudang*

Tabel ini digunakan untuk menyimpan daftar gudang pada perusahaan. Terdapat 4 atribut pada tabel ini yang terdiri dari kode, nama, alamat, dan keterangan gudang. Struktur dan implementasi dari tabel *invgudang* dapat dilihat pada gambar berikut ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	kode	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	nama	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
3	alamat	varchar(500)	utf8mb4_0900_ai_ci		Yes	*NULL*		
4	keterangan	varchar(255)	utf8mb4_0900_ai_ci		Yes	*NULL*		

Gambar 5.4. Struktur entitas *invgudang*

5. Entitas *invjenis*

Tabel ini digunakan untuk menyimpan data jenis barang. Terdapat 2 buah atribut pada tabel ini yang terdiri dari kode dan jenis barang. Struktur dan implementasi dari tabel *invjenis* ditampilkan pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	kode	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	jenis	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.5. Struktur entitas *invjenis*

6. Entitas *invtype*

Tabel ini digunakan untuk menyimpan daftar tipe barang. Terdapat 2 atribut yang digunakan pada tabel ini, yakni kode dan tipe barang. Struktur dan implementasi dari tabel *invtype* dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	kode	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	type	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.6. Struktur entitas *invtype*

7. Entitas beli

Tabel ini digunakan untuk menyimpan data transaksi pembelian dari *supplier*. Terdapat 12 atribut yang digunakan pada tabel ini. Atribut yang disediakan terdiri dari *no_bukti* atau nomor nota, tanggal transaksi, tanggal jatuh tempo, kode *supplier*, harga sub total, *persen_ppn* atau persentase PPN, harga total, lunas atau status pembayaran, *tgl_lunas*, status yang menandakan status pengiriman, *tgl_terkirim*, serta *author* atau penulis transaksi. Tabel ini berelasi *one to many* dengan tabel *supplier*. Struktur dan implementasi dari tabel beli dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	no_bukti	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	tanggal	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
3	jatuh_tempo	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
4	kode_supp	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
5	sub_total	decimal(17,2)			Yes	*NULL*		
6	persen_ppn	int			Yes	*NULL*		
7	total	decimal(17,2)			Yes	*NULL*		
8	lunas	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
9	tgl_lunas	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
10	status	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
11	tgl_terkirim	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
12	author	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.7. Struktur entitas beli

8. Entitas *beli_dtl*

Tabel ini digunakan untuk mencatat informasi *detail* masing-masing nota pembelian. Terdapat 8 atribut pada tabel ini yang terdiri dari *no_bukti, kode_brg, nama_brg, qty_order* yang mencatat kuantitas barang yang dibeli, *id_satuan, hrg_per_unit, hrg_total*, serta *kirim_gudang* atau tujuan pengiriman barang. Tabel ini berelasi *one to many* dengan tabel beli, *inventory*, satuan, dan *invgudang*. Struktur dan implementasi dari tabel *beli_dtl* dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	no_bukti	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	kode_brg	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
3	nama_brg	varchar(255)	utf8mb4_0900_ai_ci		Yes	*NULL*		
4	id_satuan	int			No	*None*		
5	qty_order	int			Yes	*NULL*		
6	hrg_per_unit	float			Yes	*NULL*		
7	hrg_total	float			Yes	*NULL*		
8	kirim_gudang	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.8. Struktur entitas *beli_dtl*

9. Entitas *customer*

Tabel ini digunakan untuk menyimpan data *customer* perusahaan. Terdapat 7 buah atribut pada tabel ini yang terdiri dari *kode_cust* yang menyimpan data kode *customer*, *nama_cust* atau nama customer, alamat customer, kontak, no_telp, email, dan kode *sales person*. Tabel ini berelasi *one to many* dengan tabel *sales_person*. Struktur dan implementasi dari tabel *customer* telah ditampilkan pada gambar berikut ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	kode_cust	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	nama_cust	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
3	alamat	varchar(255)	utf8mb4_0900_ai_ci		Yes	*NULL*		
4	kontak	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
5	no_telp	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
6	email	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
7	kode_sales	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.9. Struktur entitas *customer*

10. Entitas jual

Tabel ini digunakan untuk mencatat transaksi penjualan ke *customer*. Terdapat 12 atribut yang disediakan pada tabel ini. Atribut yang digunakan terdiri dari *no_bukti*, tanggal transaksi, tanggal jatuh tempo, *kode_cust*, harga sub total, *persen_ppn*, harga total, lunas, *tgl_lunas*, status, *tgl_terkirim*, serta *author*. Tabel ini berelasi *one to many* dengan tabel *customer*. Struktur dan implementasi dari tabel jual dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**no_bukti**	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	**tanggal**	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
3	**jatuh_tempo**	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
4	**kode_cust**	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
5	**sub_total**	decimal(17,2)			Yes	*NULL*		
6	**persen_ppn**	int			Yes	*NULL*		
7	**total**	decimal(17,2)			Yes	*NULL*		
8	**lunas**	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
9	**tgl_lunas**	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
10	**status**	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
11	**tgl_terkirim**	varchar(255)	utf8mb4_0900_ai_ci		No	*None*		
12	**author**	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.10. Struktur entitas jual

11. Entitas *jual_dtl*

Tabel ini digunakan untuk mencatat data *detail* masing-masing transaksi penjualan. Terdapat 6 atribut yang disediakan pada tabel ini. Atribut

yang digunakan terdiri dari *no_bukti, kode_brg, qty_order, hrg_per_unit, hrg_total*, dan *kode_gudang*. Tabel ini berelasi *one to many* dengan tabel jual, *inventory*, dan *invgudang*. Struktur dan implementasi dari tabel *jual_dtl* ditampilkan pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	no_bukti	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	kode_brg	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
3	qty_order	int			Yes	*NULL*		
4	hrg_per_unit	float			Yes	*NULL*		
5	hrg_total	float			Yes	*NULL*		
6	kode_gudang	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		

Gambar 5.11. Struktur entitas *jual_dtl*

12. Entitas *sales_person*

Tabel ini digunakan untuk menyimpan data *sales person* untuk masing-masing *customer*. Terdapat 2 atribut yang digunakan pada tabel ini. Atribut tersebut terdiri dari kode dan nama sales. Struktur dan implementasi dari tabel *sales_person* dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	kode_sales	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	nama_sales	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		

Gambar 5.12. Struktur entitas *sales_person*

13. Entitas satuan

Tabel ini digunakan untuk menyimpan data satuan barang. Terdapat 2 buah atribut pada tabel ini yang terdiri dari *id* dan nama satuan. Struktur dan implementasi dari tabel satuan dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int			No	*None*		AUTO_INCREMENT
2	satuan	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		

Gambar 5.13. Struktur entitas satuan

14. Entitas *supplier*

Tabel ini digunakan untuk menyimpan data *supplier* perusahaan. Terdapat 6 buah atribut pada tabel ini yang terdiri dari kode, nama dan alamat supplier, kontak, no_telp, serta email supplier. Struktur dan implementasi untuk tabel *supplier* telah ditampilkan pada gambar berikut ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	kode_supp	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
2	nama_supp	varchar(255)	utf8mb4_0900_ai_ci		Yes	*NULL*		
3	alamat	varchar(255)	utf8mb4_0900_ai_ci		Yes	*NULL*		
4	kontak	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
5	no_telp	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
6	email	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		

Gambar 5.14. Struktur entitas *supplier*

15. Entitas *mutasi_stok*

Tabel ini digunakan untuk menyimpan jumlah barang yang masuk dan keluar di gudang dan transaksi tertentu. Terdapat 10 atribut yang disediakan pada tabel ini. Atribut yang digunakan terdiri dari *id*, *no_bukti*, tanggal mutasi, *kode_brg*, *kode_gudang*, *stok_awal, qty_masuk, qty_keluar, qty_rusak_exp,* dan *stok_akhir*. Tabel ini memiliki relasi *one to many* dengan tabel *inventory* dan *invgudang*. Struktur dan implementasi dari tabel *mutasi_stok* ditunjukkan oleh gambar berikut ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int			No	*None*		AUTO_INCREMENT
2	no_bukti	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
3	tanggal	date			No	*None*		
4	kode_brg	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
5	kode_gudang	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
6	stok_awal	int			No	*None*		
7	qty_masuk	int			No	*None*		
8	qty_keluar	int			No	*None*		
9	qty_rusak_exp	int			No	*None*		
10	stok_akhir	int			No	*None*		

Gambar 5.15. Struktur entitas *mutasi_stok*

16. Entitas *opname_stok*

Tabel ini digunakan untuk menyimpan informasi stok opname yang dilakukan di suatu gudang. Terdapat 8 atribut yang disediakan pada tabel ini. Atribut yang digunakan terdiri dari *id*, tanggal opname, *kode_brg, kode_gudang, qty_sistem, qty_fisik*, selisih, dan keterangan

atau sebab terjadinya selisih. Tabel ini berelasi *one to many* dengan tabel *inventory* dan *invgudang*. Struktur dan implementasi untuk tabel *opname_stok* dapat dilihat pada gambar di bawah ini.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int			No	*None*		AUTO_INCREMENT
2	tanggal	varchar(45)	utf8mb4_0900_ai_ci		Yes	*NULL*		
3	kode_brg	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
4	kode_gudang	varchar(45)	utf8mb4_0900_ai_ci		No	*None*		
5	qty_sistem	int			Yes	*NULL*		
6	qty_fisik	int			Yes	*NULL*		
7	selisih	int			Yes	*NULL*		
8	keterangan	varchar(255)	utf8mb4_0900_ai_ci		Yes	*NULL*		

Gambar 5.16. Struktur entitas *opname_stok*

5.2. Implementasi Program

5.2.1. Proses pada Halaman *Login*

Pada halaman *login*, pengguna akan menginputkan *username* dan *password* sesuai dengan data yang telah terdaftar di tabel *users* pada *database*. Setelah melakukan *input*, data kemudian dikirim ke *server* untuk dilakukan proses validasi. Jika *username* dan *password* yang diinputkan telah terdaftar pada *database*, maka pengguna akan masuk ke halaman utama aplikasi. Sebaliknya jika tidak terdaftar, maka pengguna diarahkan kembali ke halaman *login* dan akan muncul notifikasi "Detail login tidak valid". Proses *login* dilakukan dengan memanfaatkan *library authentication* yang disediakan Laravel. Proses diatas diimplementasikan dalam sebuah fungsi bernama *loginPost* dengan sintaks seperti pada listing di bawah ini.

Listing 5.1. Proses *Login*

```
function loginPost(Request $request){
    $request->validate([
        'name' => 'required',
        'password' => 'required'
    ]);

    $credentials = $request->only('name','password');
    if(Auth::attempt($credentials)){
        return redirect()->intended(route('home'));
    }
    return redirect(route('login'))->with("error", "Login
      details are not valid");
}
```

5.2.2. Proses pada Halaman *Inventory*

Pada halaman *inventory*, ditampilkan daftar barang pada semua gudang perusahaan. Data yang ditampilkan diambil dari tabel *mutasi_stok*. Pada halaman ini disediakan fitur *filter* berdasarkan gudang, jenis barang, dan kode atau nama barang. Fitur ini digunakan untuk proses stok opname yang dapat dilakukan melalui halaman ini. Jika pengguna menggunakan salah satu atau semua *filter* yang disediakan, maka proses pengambilan data barang akan melibatkan *filter* tersebut agar informasi yang ditampilkan sesuai dengan kategori yang diinginkan pengguna. Proses pengambilan data barang akan dilakukan melalui fungsi *inventory* yang menerima parameter *request* untuk mendapatkan masing-masing kategori *filter* yang akan dikirim ke fungsi ini jika pengguna menggunakan fitur *filter* tertentu. Selanjutnya, dilakukan proses *query* untuk mengambil data barang terkini yang akan melibatkan semua data *filter* yang dipilih pengguna. Hasil proses

pengambilan data yang telah diterapkan proses *filter* kemudian dikirim menuju halaman *inventory* menggunakan fungsi *compact* untuk ditampilkan dalam bentuk tabel. Proses pengambilan serupa juga diterapkan pada halaman log mutasi. Perbedaannya hanya terletak pada adanya tambahan *filter* berdasarkan periode tanggal tertentu serta jenis transaksi. Sintaks dari proses pengambilan data *inventory* dapat dilihat pada listing di bawah ini.

Listing 5.2. Proses Pengambilan Data pada Halaman *Inventory*

```
public function inventory(Request $request){
    $selectedGudang = $request->get('gudang');
    $jenis = $request->get('jenis');
    $search = $request->get('search');

    $query = MutasiStok::whereIn('id', function ($query){
                $query->select(DB::raw('MAX(id)'))
                    ->from('mutasi_stok')
                    ->groupBy('kode_brg', 'kode_gudang');
                })->select('inventory.kode_brg as kode_brg'
                    ,'inventory.nama_brg as nama_brg',
                    'inventory.kode_divisi as kode_divisi',
                    'inventory.kode_jenis as kode_jenis',
                    'inventory.kode_type as kode_type',
                    'mutasi_stok.stok_akhir as quantity',
                    'inventory.id_satuan as id_satuan',
                    'inventory.hrg_jual as hrg_jual',
                    'inventory.keterangan as keterangan',
                    'mutasi_stok.kode_gudang as kode_gudang',
                    'invgudang.nama as nama_gudang')
                  ->join('inventory', 'mutasi_stok.kode_brg'
                    ,'=','inventory.kode_brg')
                  ->join('invgudang','mutasi_stok.kode_gudang'
                    ,'=','invgudang.kode');

    if($selectedGudang && $selectedGudang != 'All'){
        $query->where('invgudang.nama', $selectedGudang);
    }

    if($selectedGudang == 'All'){
        $query->select('inventory.kode_brg', 'inventory.
          nama_brg','inventory.kode_divisi','inventory.
          kode_jenis','inventory.kode_type','inventory.
          id_satuan', 'inventory.hrg_jual','inventory.
          keterangan')->selectRaw('SUM(mutasi_stok.stok_akhir)
          as quantity')
```

```
            ->groupBy('inventory.kode_brg', 'inventory.nama_brg',
              'inventory.kode_divisi', 'inventory.kode_jenis',
              'inventory.kode_type','inventory.id_satuan',
              'inventory.hrg_jual','inventory.keterangan');
    }

    if($jenis && $jenis != 'All'){
        $query->where('inventory.kode_jenis',$jenis);
    }

    if($search){
        $query->where('inventory.nama_brg', 'like', '%'.
          $search.'%')
          ->orWhere('inventory.kode_brg', 'like','%'.$search.
            '%');
    }

    $inventory = $query->paginate(10);

    $divisi = Divisi::all();
    $gudang = Gudang::all();
    $jenis = Jenis::all();
    $type = Type::all();
    $satuan = Satuan::all();
    $beli = BeliDetail::all();
    $jual = JualDetail::all();

    return view('inventory.inventory',compact('inventory',
      'divisi','gudang','jenis','type','satuan',
      'selectedGudang','search','jenis','beli','jual'));
}
```

Halaman ini menyediakan fitur stok opname barang yang dapat digunakan jika jumlah barang pada sistem tidak sesuai dengan jumlah barang fisik pada suatu gudang. Pengguna melakukan *filter* data *inventory* berdasarkan gudang terlebih dahulu untuk menampilkan semua barang yang disimpan pada suatu gudang. Setelah itu, pengguna dapat mengakses fitur ini melalui tombol “Opname” pada suatu baris tabel yang akan memunculkan sebuah dialog untuk menginputkan jumlah barang fisik serta keterangan yang merupakan sebab terjadinya selisih jumlah barang.

Penyebab selisih bisa karena rusak, kedaluwarsa, atau salah pencatatan. Setelah menginputkan data, informasi tersebut kemudian dikirim ke fungsi *opnameBarang* untuk dilakukan pencatatan informasi stok opname ke *database* untuk pembuatan laporan.

Setelah fungsi menerima seluruh informasi, dilakukan pengecekan jika penyebab terjadinya selisih adalah karena rusak atau kedaluwarsa, maka dilakukan penambahan data mutasi ke tabel *mutasi_stok* agar jumlah barang sistem sesuai dengan jumlah barang fisik yang masih layak dijual. Selain itu, jika penyebab terjadinya selisih adalah karena salah pencatatan, maka sistem akan menampilkan fitur koreksi transaksi pembelian dan penjualan agar pengguna dapat memilih transaksi yang ingin dikoreksi. Setelah itu, sistem secara otomatis akan menyesuaikan informasi transaksi tersebut sesuai dengan selisih yang telah dihitung sebelumnya. Data mutasi barang terkait juga akan disesuaikan secara otomatis oleh sistem. Kuantitas barang sistem kemudian disesuaikan dengan jumlah barang fisik dan dilakukan penghitungan kembali harga jual untuk barang tersebut jika telah terjadi perubahan data di transaksi pembelian. Informasi stok opname yang telah dilakukan untuk barang tersebut akan disimpan ke *database* untuk ditampilkan pada laporan stok opname perusahaan. Sintaks dari proses stok opname ini dapat dilihat pada listing berikut ini.

Listing 5.3. Proses Stok Opname

```
public function opnameBarang(Request $request){
    $kode_brg = $request->input('kode_brg');
```

```
    $nama_brg = $request->input('nama_brg');
    $quantity = $request->input('quantity');
    $qty_awal = $request->input('qty_awal');
    $kode_gudang = $request->input('kode_gudang');
    $keterangan = $request->input('keterangan');

    $qty_fisik = $request->input('qty_fisik');
    $transaction = $request->input('transaction');
    $no_bukti = $request->input('no_bukti');
    $qty_order = $request->input('qty_order');
    $hrg_total = $request->input('hrg_total');

    if($quantity != 0){
        DB::table('opname_stok')->insert([
            'tanggal' => Carbon::now()->format('d-m-Y'),
            'kode_brg' => $kode_brg,
            'kode_gudang' => $kode_gudang,
            'qty_sistem' => $qty_awal,
            'qty_fisik' => $qty_fisik,
            'selisih' => $qty_fisik - $qty_awal,
            'keterangan' => $keterangan
        ]);

        if($keterangan == "BARANG RUSAK" || $keterangan ==
          "BARANG EXPIRED"){
            $quantity = abs($quantity);

            DB::table('mutasi_stok')->insert([
                'no_bukti' => "-",
                'tanggal' => Carbon::now()->format('Y-m-d'),
                'kode_brg' => $kode_brg,
                'kode_gudang' => $kode_gudang,
                'stok_awal' => $qty_awal,
                'qty_masuk' => 0,
                'qty_keluar' => 0,
                'qty_rusak_exp' => $quantity,
                'stok_akhir' => $qty_awal - $quantity
            ]);

        } else{
            if($transaction == 'pembelian'){
                DB::table('beli_dtl')->where('no_bukti'
                  ,$no_bukti)
                  ->where('kode_brg', $kode_brg)
                  ->where('nama_brg', $nama_brg)
                  ->where('kirim_gudang', $kode_gudang)
                  ->update([
                      'qty_order' => $qty_order,
                      'hrg_total' => $hrg_total
                  ]);

                $sub_total = DB::table('beli_dtl')
                  ->where('no_bukti', $no_bukti)
```

```
                    ->sum('hrg_total');

                $beli_data = DB::table('beli')->where(
                  'no_bukti',$no_bukti)->first();
                $sub_total = floatval($sub_total);
                $total = $sub_total + ($sub_total
                  * ($beli_data->persen_ppn / 100));

                DB::table('beli')
                  ->where('no_bukti', $no_bukti)
                  ->update([
                      'sub_total' => $sub_total,
                      'total' => $total
                  ]);

                DB::table('mutasi_stok')
                  ->where('no_bukti', $no_bukti)
                  ->where('kode_brg', $kode_brg)
                  ->where('kode_gudang', $kode_gudang)
                  ->update([
                      'qty_masuk' => $qty_order,
                      'stok_akhir' => DB::raw('stok_awal + '
                        . $qty_order)
                  ]);

            } else{
                DB::table('jual_dtl')->where('no_bukti'
                  ,$no_bukti)
                  ->where('kode_brg', $kode_brg)
                  ->where('kode_gudang', $kode_gudang)
                  ->update([
                      'qty_order' => $qty_order,
                      'hrg_total' => $hrg_total
                  ]);

                $sub_total = DB::table('jual_dtl')
                  ->where('no_bukti', $no_bukti)
                  ->sum('hrg_total');

                $jual_data = DB::table('jual')->where(
                  'no_bukti',$no_bukti)->first();
                $sub_total = floatval($sub_total);
                $total = $sub_total + ($sub_total
                  * ($jual_data->persen_ppn / 100));

                DB::table('jual')
                  ->where('no_bukti', $no_bukti)
                  ->update([
                      'sub_total' => $sub_total,
                      'total' => $total
                  ]);

                DB::table('mutasi_stok')
                  ->where('no_bukti', $no_bukti)
```

```
                ->where('kode_brg', $kode_brg)
                ->where('kode_gudang', $kode_gudang)
                ->update([
                    'qty_keluar' => $qty_order,
                    'stok_akhir' => DB::raw('stok_awal - '
                      . $qty_order)
                ]);
            }
            $dataRow = DB::table('mutasi_stok')
              ->where('no_bukti', $no_bukti)
              ->where('kode_brg', $kode_brg)
              ->where('kode_gudang', $kode_gudang)
              ->get(['id', 'stok_akhir']);

            foreach($dataRow as $row){
                $rowId = $row->id;
                $rowStokAkhir = $row->stok_akhir;
            }

            $rowsToUpdate = DB::table('mutasi_stok')
              ->where('kode_brg', $kode_brg)
              ->where('kode_gudang', $kode_gudang)
              ->where('id', '>', $rowId)
              ->orderBy('id')
              ->get();

            $count = count($rowsToUpdate);

            if($count > 0){
                $firstRow = $rowsToUpdate[0];
                $stokAwal = $rowStokAkhir;

                DB::table('mutasi_stok')
                  ->where('id', $firstRow->id)
                  ->update([
                      'stok_awal' => $stokAwal
                  ]);

                for($i = 0; $i < $count; $i++){
                    $currentRow = $rowsToUpdate[$i];

                    $stokAwal = $rowStokAkhir;
                    $masuk = $currentRow->qty_masuk;
                    $keluar = $currentRow->qty_keluar;
                    $rusakExp = $currentRow->qty_rusak_exp;

                    $stokAkhir = $stokAwal + $masuk - $keluar
                      - $rusakExp;

                    DB::table('mutasi_stok')
                      ->where('id', $currentRow->id)
                      ->update([
                          'stok_akhir' => $stokAkhir
```

```
                        ]);

                    if($i < $count - 1){
                        DB::table('mutasi_stok')
                          ->where('id', $rowsToUpdate[$i + 1]
                            ->id)
                          ->update([
                              'stok_awal' => $stokAkhir
                          ]);
                    }
                    $rowStokAkhir = $stokAkhir;
                }
            }

            $transactions = DB::table('beli_dtl')
              ->where('kode_brg', $kode_brg)
              ->get();

            $totalCost = 0;
            $currentQuantity = 0;
            foreach($transactions as $transaction){
                $totalCost += $transaction->hrg_total;
                $currentQuantity += $transaction->qty_order;
            }

            $minSellPrice = $totalCost / $currentQuantity;
            $sellPrice = $minSellPrice + ($minSellPrice * 0.5);

            DB::table('inventory')
              ->where('kode_brg', $kode_brg)
              ->update(['hrg_jual' => $sellPrice]);
        }
        return response()->json(['success' => true]);
    }
}
```

Setelah melakukan stok opname, pengguna dapat melihat semua riwayat stok opname yang pernah dilakukan di sebuah laporan yang ditampilkan dalam bentuk PDF agar pengguna dapat mengunduh file laporan tersebut. Proses pembuatan PDF laporan memanfaatkan library *Dompdf* yang disediakan laravel. Jika pengguna menekan tombol “Lihat Laporan”, maka sistem akan menjalankan fungsi *cetak_pdf* yang menerima parameter *request* agar dapat melakukan *filter* data stok opname

berdasarkan gudang dan tanggal opname. Kemudian dilakukan proses pengambilan informasi yang akan ditampilkan di laporan menggunakan sintaks *query*. Informasi yang ditampilkan terdiri dari tanggal opname, kode barang, nama barang, satuan barang, gudang atau lokasi penyimpanan barang, jumlah kuantitas sistem saat stok opname dilakukan, jumlah kuantitas fisik, selisih, serta keterangan atau sebab terjadinya selisih. Untuk data barang yang diopname akan diambil dari tabel *inventory*. Setelah dilakukan proses pengambilan data, diterapkan proses *filter* berdasarkan gudang dan tanggal. Hasil proses diatas kemudian dikirim ke halaman *opnamepdf* agar informasi stok opname dapat ditampilkan dalam bentuk PDF. Proses pengambilan data laporan ini juga diterapkan untuk pembuatan laporan mutasi stok dengan pendekatan serupa. Sintaks dari proses diatas secara keseluruhan dapat dilihat pada listing berikut ini.

Listing 5.4. Proses Pengambilan Data untuk Laporan Stok Opname

```
public function cetak_pdf(Request $request){
    $selectedGudang = $request->input('selectedGudang');
    $selectedTanggal = $request->get('selectedTanggal');

    $query = OpnameStok::query()
                ->select('opname_stok.*','inventory.nama_brg
                  as nama_brg','satuan.satuan as nama_satuan',
                  'invgudang.nama as nama_gudang')
                ->join('inventory', 'opname_stok.kode_brg'
                  ,'=','inventory.kode_brg')
                ->join('invgudang', 'invgudang.kode', '=',
                  'opname_stok.kode_gudang')
                ->join('satuan', 'inventory.id_satuan', '=',
                  'satuan.id');

    if($selectedGudang && $selectedGudang != 'All'){
        $query->where('invgudang.nama', $selectedGudang);
    }
```

```
    if($selectedTanggal){
        $query->where('opname_stok.tanggal', $selectedTanggal);
    }

    $opname = $query->get();
    $data = $opname;
    $gudang = Gudang::all();

    $view = View::make('inventory.opnamepdf', ['data'=>$data,
      'selectedGudang'=>$selectedGudang, 'selectedTanggal'=>
      $selectedTanggal, 'gudang'=>$gudang]);
    $pdf = new Dompdf();
    $pdf->loadHtml($view->render());
    $pdf->setPaper('A4', 'portrait');
    $pdf->render();
    return response($pdf->output(), 200, [
        'Content-Type' => 'application/pdf',
        'Content-Disposition' => 'inline;',
    ]);
}
```

5.2.3. Proses pada Halaman Transaksi Pembelian

Pada halaman ini, ditampilkan daftar transaksi pembelian dari *supplier*. Data-data transaksi pembelian diambil menggunakan fungsi beli yang akan mengambil semua data dari tabel beli pada *database*. Pengguna dapat melakukan *filter* data transaksi berdasarkan tanggal melalui sebuah *date picker*. Saat pengguna memilih tanggal tertentu, data tersebut kemudian dikirim ke fungsi beli untuk dapat memfilter data transaksi berdasarkan tanggal. Setelah melakukan *filter*, dilakukan proses *sorting* secara *descending* sehingga data transaksi dengan tanggal yang terbaru akan ditampilkan terlebih dahulu. Setelah itu, diterapkan juga *paging* sebanyak 10 data ditampilkan untuk setiap halaman. Pada akhirnya, data transaksi pembelian akan dikirim ke halaman transaksi beli untuk ditampilkan dalam

bentuk tabel. Sintaks untuk menampilkan data transaksi pembelian dapat dilihat pada listing di bawah ini**.**

Listing 5.5. Proses Pengambilan Data Transaksi Pembelian

```
public function beli(Request $request){
    $selectedDate = $request->get('selectedDate');

    $query = Beli::query();

    if($selectedDate){
        $query->where('tanggal', $selectedDate);
    }

    $beli = $query->orderBy('tanggal', 'desc')->paginate(10);

    $supplier = Supplier::all();
    $gudang = Gudang::all();
    return view('transaksi.beli',compact('beli','supplier',
      'gudang','selectedDate'));
}
```

Pengguna dapat melakukan navigasi dari halaman transaksi pembelian ke halaman *detail* pembelian dengan menekan tombol “Details” pada suatu baris data transaksi. Halaman ini akan menampilkan data *detail* dari suatu transaksi dalam bentuk tabel. *Detail* yang ditampilkan adalah data barang, kuantitas pesanan, serta harga barang yang dipesan perusahaan dari *supplier* melalui transaksi tersebut, serta gudang tujuan pengiriman barang. Data *detail* transaksi pembelian diambil menggunakan fungsi *showDetail* berparameter *no_bukti* yang akan menjadi acuan untuk mencari data *detail* berdasarkan nomor nota. Setelah mendapatkan semua data *detail* dari suatu transaksi, data tersebut kemudian akan dikirim ke halaman *detail* transaksi

pembelian untuk ditampilkan dalam bentuk tabel. Sintaks dari proses diatas dapat dilihat pada listing di bawah ini.

Listing 5.6. Proses Pengambilan Data *Detail* Transaksi Pembelian

```
public function showDetail($no_bukti){
    $beliDetail = BeliDetail::where('no_bukti', $no_bukti)
      ->get();
    $satuan = Satuan::all();
    $gudang = Gudang::all();

    return view('transaksi.belidetail', compact('beliDetail',
      'satuan','gudang','no_bukti'));
}
```

Pada halaman ini, pengguna dapat mengakses halaman tambah transaksi dengan menekan tombol “Tambah Transaksi”. Pengguna akan menginputkan informasi untuk membuat transaksi baru beserta *detail* transaksi pada *input field* yang disediakan. Setelah mengisi semua *field*, informasi yang diinputkan kemudian dikirim ke fungsi *store* untuk disimpan ke *database*. Jika proses penambahan transaksi dan *detail* berhasil, maka pengguna diarahkan kembali ke halaman transaksi pembelian dan muncul notifikasi bahwa proses penambahan berhasil. Sintaks dari proses tambah transaksi dan *detail* pembelian telah dijabarkan pada listing di bawah ini.

Listing 5.7. Proses Tambah *Detail* dan Transaksi Pembelian

```
public function store(Request $request)
{
    $data = new Beli();
    $data->no_bukti = $request->get('no_bukti');
    $data->tanggal = $request->get('datepicker');
    $data->kode_supp = $request->get('select_supplier');
    $data->sub_total = $request->get('sub_total');
```

```
    $data->persen_ppn = $request->get('persen_ppn');
    $data->total = $request->get('total');
    $data->lunas = 'Belum Lunas';
    $data->status = 'Belum Terkirim';
    $data->author = auth()->user()->name;
    $data->jatuh_tempo = Carbon::parse($data->tanggal)
      ->addMonth()->format('d-m-Y');
    $data->tgl_lunas = '-';
    $data->tgl_terkirim = '-';
    $data->save();

    $kode_brg = $request->get('kode_brg');
    $nama_brg = $request->get('nama_brg');
    $qty_order = $request->get('qty_order');
    $id_satuan = $request->get('select_satuan');
    $hrg_per_unit = $request->get('hrg_per_unit');
    $hrg_total = $request->get('hrg_total');
    $kirim_gudang = $request->get('select_gudang');

    foreach($kode_brg as $key => $value){
        $detail = new BeliDetail();
        $detail->no_bukti = $data->no_bukti;
        $detail->kode_brg = $kode_brg[$key];
        $detail->nama_brg = $nama_brg[$key];
        $detail->qty_order = $qty_order[$key];
        $detail->id_satuan = $id_satuan[$key];
        $detail->hrg_per_unit = $hrg_per_unit[$key];
        $detail->hrg_total = $hrg_total[$key];
        $detail->kirim_gudang = $kirim_gudang[$key];
        $detail->save();
    }
    return redirect()->route('pembelian')->with('status',
      'Hooray!! Your new transaction is already inserted');
}
```

Pengguna dapat melakukan *update* status pembayaran menjadi lunas dengan menekan tombol “Belum Lunas” dan *update* status pengiriman menjadi “Sudah Terkirim” jika menekan tombol “Belum Terkirim”. Proses *update* status pembayaran dilakukan dalam fungsi *updateBayar* yang menerima data *no_bukti* dan tanggal pembayaran. *no_bukti* akan menjadi indikator transaksi yang perlu diubah status pembayarannya. Setelah menemukan transaksi yang ingin diupdate, status pembayaran pada

transaksi tersebut kemudian diubah dari “Belum Lunas” menjadi “Lunas” dan tanggal pembayaran akan disimpan juga pada transaksi tersebut. Hasil perubahan kemudian disimpan pada *database* dan muncul notifikasi bahwa *update* status pembayaran sukses dilakukan.

Proses *update* status pengiriman dilakukan dalam fungsi *updateKirim* yang juga menerima data *no_bukti* sebagai acuan untuk menemukan transaksi yang ingin diubah status pengirimannya dari “Belum Terkirim” menjadi “Sudah Terkirim” dan data tanggal terkirimnya barang. Data status dan tanggal akan diterima oleh sistem dan disimpan ke transaksi terkait di *database*. Setelah itu, dilakukan proses penambahan kuantitas barang jika pada sistem telah terdapat informasi mengenai barang yang dibeli. Jika barang belum ada sebelumnya pada gudang, maka dilakukan penambahan data barang baru. Selanjutnya, dilakukan proses perhitungan kembali harga jual untuk barang yang dibeli menggunakan metode *average cost* karena terjadi perubahan jumlah stok barang tersebut pada gudang.

Proses perhitungan harga jual dengan metode *average cost* dilakukan dengan cara menjumlahkan harga total dari semua pembelian barang terkait. Kemudian dilakukan pembagian antara total harga dengan jumlah barang yang pernah dibeli untuk mendapatkan rata-rata biaya pembelian per kuantitas barang. Harga jual barang ditentukan dengan menggunakan biaya rata-rata tersebut kemudian diterapkan *markup* sebesar 50%. Harga jual tersebut kemudian disimpan ke *database* dan berikutnya

dilakukan proses penyimpanan informasi barang ke tabel *mutasi_stok* untuk mencatat jumlah barang yang masuk ke gudang. Setelah itu, pengguna diarahkan kembali ke halaman transaksi pembelian dan dimunculkan notifikasi bahwa status pengiriman berhasil diubah. Sintaks dari proses pengubahan status pembayaran dan pengiriman dapat dilihat pada listing di bawah ini.

Listing 5.8. Proses Ubah Status Pembayaran dan Pengiriman pada Transaksi Pembelian

```
public function updateBayar(Request $request)
{
    $no_bukti = $request->input('no_bukti');
    $tgl_lunas = $request->input('tgl_lunas');

    $beli = Beli::where('no_bukti', $no_bukti)->firstOrFail();

    $beli->lunas = 'Lunas';
    $beli->tgl_lunas = $tgl_lunas;
    $beli->save();

    return response()->json(['success' => true]);
}

public function updateKirim(Request $request)
{
    $no_bukti = $request->input('no_bukti');
    $tgl_terkirim = $request->input('tgl_terkirim');

    $beli = Beli::where('no_bukti', $no_bukti)->firstOrFail();

    $beli->status = 'Sudah Terkirim';
    $beli->tgl_terkirim = $tgl_terkirim;
    $beli->save();

    $beliDetail = BeliDetail::where('no_bukti', $no_bukti)
      ->get();

    foreach($beliDetail as $detail){
        $mutasi = DB::table('mutasi_stok')
            ->where('kode_brg', $detail->kode_brg)
            ->where('kode_gudang', $detail->kirim_gudang)
```

```
            ->orderBy('id', 'desc')
            ->first();

        if($mutasi){
            $stok_awal = $mutasi->stok_akhir;

            $transactions = DB::table('beli_dtl')
                ->where('kode_brg', $detail->kode_brg)
                ->get();

            $totalCost = 0;
            $currentQuantity = 0;
            foreach($transactions as $transaction){
                $totalCost += $transaction->hrg_total;
                $currentQuantity += $transaction->qty_order;
            }

            $minSellPrice = $totalCost / $currentQuantity;
            $sellPrice = $minSellPrice + ($minSellPrice * 0.5);

            DB::table('inventory')
                ->where('kode_brg', $detail->kode_brg)
                ->update(['hrg_jual' => $sellPrice]);

            DB::table('mutasi_stok')->insert([
                'no_bukti' => $no_bukti,
                'tanggal' => Carbon::parse($tgl_terkirim)
                  ->format('Y-m-d'),
                'kode_brg' => $detail->kode_brg,
                'kode_gudang' => $detail->kirim_gudang,
                'stok_awal' => $stok_awal,
                'qty_masuk' => $detail->qty_order,
                'qty_keluar' => 0,
                'qty_rusak_exp' => 0,
                'stok_akhir' => $stok_awal + $detail->qty_order
            ]);

        } else{

            $transactions = DB::table('beli_dtl')
                ->where('kode_brg', $detail->kode_brg)
                ->get();

            $totalCost = 0;
            $currentQuantity = 0;
            foreach($transactions as $transaction){
                $totalCost += $transaction->hrg_total;
                $currentQuantity += $transaction->qty_order;
            }

            $minSellPrice = $totalCost / $currentQuantity;
            $sellPrice = $minSellPrice + ($minSellPrice * 0.5);
```

```
            DB::table('inventory')
                ->where('kode_brg', $detail->kode_brg)
                ->update(['hrg_jual' => $sellPrice]);

            DB::table('mutasi_stok')->insert([
                'no_bukti' => $no_bukti,
                'tanggal' => Carbon::parse($tgl_terkirim)
                  ->format('Y-m-d'),
                'kode_brg' => $detail->kode_brg,
                'kode_gudang' => $detail->kirim_gudang,
                'stok_awal' => 0,
                'qty_masuk' => $detail->qty_order,
                'qty_keluar' => 0,
                'qty_rusak_exp' => 0,
                'stok_akhir' => $detail->qty_order
            ]);
        }
    }
    return response()->json(['success' => true]);
}
```

Pengguna dapat melihat faktur beli dari masing-masing transaksi pembelian dengan mengakses tombol “Lihat Faktur” pada masing-masing baris data transaksi. Jika tombol tersebut ditekan, sistem akan mengirimkan data *no_bukti* dari baris data tersebut ke fungsi *cetak_pdf* yang menerima parameter ini agar sistem dapat mengambil data transaksi serta *detail* terkait untuk kemudian ditampilkan dalam sebuah dokumen PDF. Pembuatan dokumen faktur beli melibatkan sebuah *library* Dompdf yang disediakan oleh Laravel. Fungsi pengambilan data faktur beli ini dapat dilihat pada listing di bawah ini.

Listing 5.9. Proses Pengambilan Data Faktur Beli

```
public function cetak_pdf($no_bukti)
{
    $beli = Beli::find($no_bukti);
    $beliDetail = BeliDetail::where('no_bukti', $no_bukti)
      ->get();
```

```
    $supplier = Supplier::all();
    $gudang = Gudang::all();
    $satuan = Satuan::all();
    $data = $beli;
    $dataDetail = $beliDetail;

    $view = View::make('transaksi.belipdf', ['data'=>$data,
      'dataDetail'=>$dataDetail,'supplier'=>$supplier,
      'gudang'=>$gudang,'satuan'=>$satuan]);
    $pdf = new Dompdf();
    $pdf->loadHtml($view->render());
    $pdf->setPaper('A4', 'landscape');
    $pdf->render();
    return response($pdf->output(), 200, [
        'Content-Type' => 'application/pdf',
        'Content-Disposition' => 'inline;',
    ]);
}
```

5.2.4. Proses pada Halaman Transaksi Penjualan

Halaman ini menampilkan data transaksi penjualan ke *customer* dalam bentuk tabel. Disediakan juga fitur *filter* tabel berdasarkan tanggal, fitur tambah data transaksi beserta detailnya, fitur *detail* transaksi yang menampilkan informasi semua barang yang dijual pada suatu transaksi, fitur ubah status pembayaran, dan fitur *invoice* yang menampilkan file *invoice* dalam bentuk PDF. Proses implementasi dari fitur-fitur diatas hampir sama dengan halaman transaksi pembelian. Perbedaannya hanya terletak pada tabel yang dilibatkan dalam proses pengambilan dan penambahan transaksi penjualan beserta detailnya, yakni tabel jual dan *jual_dtl*. Sintaks *query* pada fungsi *showDetail* untuk fitur *detail* transaksi dan fungsi *cetak_pdf* pada fitur *invoice* juga berbeda karena harus mengambil informasi barang berupa kode, nama, dan satuan dari tabel *inventory* berdasarkan kode barang. Fitur ubah status pengiriman juga diimplementasikan pada halaman

ini dengan menerapkan fungsi *updateKirim* yang akan merubah data status pada suatu transaksi dari “Belum Terkirim” menjadi “Sudah Terkirim”. Fungsi ini memerlukan data *no_bukti* untuk dapat mencari transaksi penjualan yang ingin diubah statusnya. Setelah mengubah status pengiriman menjadi “Sudah Terkirim”, dilakukan penambahan jumlah barang yang keluar sesuai dengan setiap transaksi pada tabel *mutasi_stok* untuk mencatat perubahan jumlah stok setelah barang dijual. Sintaks dari proses diatas secara keseluruhan telah ditampilkan pada listing berikut ini.

Listing 5.10. Proses Ubah Status Pengiriman pada Transaksi Penjualan

```
public function updateKirim(Request $request)
{
    $no_bukti = $request->input('no_bukti');
    $tgl_terkirim = $request->input('tgl_terkirim');

    $jual = Jual::where('no_bukti', $no_bukti)->firstOrFail();

    $jual->status = 'Sudah Terkirim';
    $jual->tgl_terkirim = $tgl_terkirim;
    $jual->save();

    $jualDetail = JualDetail::where('no_bukti', $no_bukti)
      ->get();

    foreach ($jualDetail as $detail){
        $mutasi = DB::table('mutasi_stok')
            ->where('kode_brg', $detail->kode_brg)
            ->where('kode_gudang', $detail->kode_gudang)
            ->orderBy('id', 'desc')
            ->first();

        if($mutasi){
            $stok_awal = $mutasi->stok_akhir;

            DB::table('mutasi_stok')->insert([
                'no_bukti' => $no_bukti,
                'tanggal' => Carbon::parse($tgl_terkirim)
                  ->format('Y-m-d'),
                'kode_brg' => $detail->kode_brg,
                'kode_gudang' => $detail->kode_gudang,
```

```
                'stok_awal' => $stok_awal,
                'qty_masuk' => 0,
                'qty_keluar' => $detail->qty_order,
                'qty_rusak_exp' => 0,
                'stok_akhir' => $stok_awal - $detail->qty_order
            ]);
        }
    }
    return response()->json(['success' => true]);
}
```

5.2.5. Proses pada Halaman *Supplier*

Halaman ini digunakan untuk menampilkan data *supplier* perusahaan dalam bentuk tabel. Data *supplier* akan dikirim ke halaman ini melalui fungsi *supplier* yang menerima paramater *request* untuk mendapatkan *filter search* berdasarkan nama *supplier* yang akan dikirim ke fungsi ini jika pengguna menginputkan teks di *input field search* pada halaman *supplier*. *Filter search* tersebut akan dimanfaatkan dalam proses pengambilan data *supplier* sehingga data yang dikirim hanya yang sesuai dengan teks *search* yang diinputkan pengguna. Jika tidak menggunakan *filter*, maka fungsi akan mengambil semua data dari tabel *supplier*. *Paging* juga diterapkan dengan menampilkan 10 data per halaman. Data *supplier* kemudian dikirim ke halaman *supplier* untuk ditampilkan pada tabel. Proses pengambilan data pada halaman ini juga diterapkan pada halaman gudang, *customer*, dan *sales person* menggunakan pendekatan yang serupa. Sintaks dari proses pengambilan data *supplier* dapat dilihat pada listing di bawah ini.

Listing 5.11. Proses Pengambilan Data *Supplier*

```
public function supplier(Request $request){
    $search = $request->get('search');

    $query = Supplier::query();

    if($search){
        $query->where('nama_supp', 'like', '%'.$search.'%');
    }

    $supplier = $query->paginate(10);

    return view('supplier.supplier',compact('supplier',
      'search'));
}
```

Pengguna dapat melakukan penambahan data *supplier* melalui tombol “Tambah Supplier”. Pada halaman ini, disediakan beberapa *input field* yang bisa diberi informasi oleh pengguna untuk kemudian dikirim seluruh datanya ke fungsi *store* yang akan menyimpan informasi yang diinputkan pengguna ke *database*. Setelah berhasil *input* ke *database*, sistem akan mengarahkan pengguna kembali ke halaman *supplier* untuk diberi notifikasi bahwa data *supplier* berhasil ditambahkan. Fitur tambah data pada halaman gudang, *customer*, dan *sales person* juga menggunakan pendekatan yang sama seperti proses penambahan data pada halaman ini. Proses penambahan data *supplier* dapat dilihat pada listing berikut ini.

Listing 5.12. Proses Tambah *Supplier*

```
public function store(Request $request)
{
    $data = new Supplier();
    $data->kode_supp = $request->get('kode_supp');
    $data->nama_supp = $request->get('nama_supp');
```

```
    $data->alamat = $request->get('alamat');
    $data->kontak = $request->get('kontak');
    $data->no_telp = $request->get('no_telp');
    $data->email = $request->get('email');
    $data->save();
    return redirect()->route('supplier')->with('status',
      'Hooray!! Your new item is already inserted');
}
```

Halaman *supplier* dapat mengakses halaman *edit supplier* melalui tombol "Edit" yang berada di setiap baris data *supplier* pada tabel. Pengguna dapat mengubah informasi *supplier* sebelumnya dengan mengubah data *supplier* yang disajikan di setiap *input field*. Data yang berisikan informasi *supplier* didapat dari proses pengambilan data *supplier* yang dilakukan pada fungsi *edit* berparameter *kode_supp* yang digunakan untuk mencari *supplier* yang ingin diubah datanya. Data tersebut kemudian dikirim ke halaman *edit supplier* untuk ditampilkan di setiap *input field*. Setelah pengguna mengubah informasi *supplier*, seluruh data kemudian dikirim ke fungsi *update* untuk disimpan ke *database*. Setelah itu, pengguna diarahkan kembali ke halaman *supplier* untuk diberi notifikasi bahwa proses *edit supplier* berhasil dilakukan. Proses *edit* pada halaman ini juga diterapkan serupa dengan proses pada halaman *edit* gudang, *customer*, dan *sales person*. Sintaks dari proses *edit supplier* dapat dilihat pada listing berikut ini.

Listing 5.13. Proses *Edit Supplier*

```
public function edit($kode_supp){
    $objSupplier = Supplier::find($kode_supp);
```

```
    $data = $objSupplier;
    return view('supplier.formedit',compact('data'));
}

public function update(Request $request, $kode_supp){
    $objSupplier = Supplier::find($kode_supp);
    $objSupplier->kode_supp = $request->get('kode_supp');
    $objSupplier->nama_supp = $request->get('nama_supp');
    $objSupplier->alamat = $request->get('alamat');
    $objSupplier->kontak = $request->get('kontak');
    $objSupplier->no_telp = $request->get('no_telp');
    $objSupplier->email = $request->get('email');
    $objSupplier->save();
    return redirect()->route('supplier')->with('status',
      'Your item is up-to-date');
}
```

Pada halaman *supplier* disediakan tombol "Delete" pada setiap baris data untuk menghapus suatu *supplier*. Jika pengguna menekan tombol tersebut, maka data *kode_supp* dari *supplier* tersebut akan dikirim ke fungsi *destroy* sebagai acuan untuk menemukan data *supplier* tersebut dan kemudian dihapus dari *database*. Jika berhasil, maka pengguna diarahkan kembali ke halaman *supplier* dan muncul notifikasi bahwa proses penghapusan berhasil. Jika gagal, maka muncul pesan gagal hapus *supplier* karena datanya telah tercatat pada suatu transaksi pembelian sehingga masih diperlukan pada sistem. Proses penghapusan serupa juga diterapkan pada halaman gudang, *customer*, dan *sales person*. Sintaks dari proses hapus *supplier* dapat dilihat pada listing di bawah ini.

Listing 5.14. Proses Hapus *Supplier*

```
public function destroy($kode_supp){
    try{
        $objSupplier = Supplier::find($kode_supp);
        $objSupplier->delete();
        return redirect()->route('supplier')->with('status',
          'Deleted Successfully');
    }
```

```
    catch(\PDOException $ex){
        $msg = "Data tidak dapat dihapus karena diperlukan
          pada suatu transaksi pembelian";
        return redirect()->route('supplier')->with('status',
          $msg);
    }
}
```