BAB 5

IMPLEMENTASI

Bab ini menjelaskan mengenai proses implementasi sistem yang telah didesain sebelumnya. Proses-proses implementasi yang dibahas meliputi penjelasan data pada *database* dan implementasi program. Implementasi sistem dilakukan dengan memanfaatkan *framework* Laravel dan *database* MySQL.

5.1. Implementasi *Database*

Bagian ini menjelaskan mengenai implementasi database di MySQL. Database yang digunakan terdiri dari 15 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. Semua atribut beserta fungsinya dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
id	*bigint (primary key)*	*Id* pengguna
name	*varchar*	Nama pengguna
password	*varchar*	*Password* akun pengguna

Tabel 5.1. Entitas *users*

2. Entitas *invmaster*

Tabel ini digunakan untuk menyimpan data barang perusahaan. Terdapat 12 atribut pada tabel ini. Semua atribut beserta penjelasan dari tabel ini dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
id	*varchar*	*Id* Barang
kode_brg	*varchar*	Kode Barang
nama_brg	*varchar*	Nama Barang
kode_divisi	*varchar*	Kode Divisi Barang
kode_jenis	*varchar*	Kode Jenis Barang
kode_type	*varchar*	Kode Tipe Barang
packing	*varchar*	*Packing* Barang
quantity	*int*	Kuantitas Barang
id_satuan	*int*	*Id* Satuan Barang
hrg_jual	*float*	Harga Jual Per Pack
kode_gudang	*varchar*	Kode Gudang Barang
keterangan	*varchar*	Keterangan

Tabel 5.2. Entitas *invmaster*

3. Entitas *invdivisi*

Tabel ini digunakan untuk menyimpan daftar divisi barang. Terdapat 2 atribut pada tabel ini. Atribut beserta penjelasan dari tabel ini dapat dilihat pada tabel berikut ini.

Nama Atribut	Tipe Data	Keterangan
kode	*varchar*	Kode Divisi
divisi	*varchar*	Nama Divisi

Tabel 5.3. Entitas *invdivisi*

4. Entitas *invgudang*

   Tabel ini digunakan untuk menyimpan daftar gudang pada perusahaan. Terdapat 4 atribut pada tabel ini. Atribut serta penjelasan dapat dilihat pada tabel berikut ini.

Nama Atribut	Tipe Data	Keterangan
kode	*varchar*	Kode Gudang
nama	*varchar*	Nama Gudang
alamat	*varchar*	Alamat Gudang
keterangan	*varchar*	Keterangan

Tabel 5.4. Entitas *invgudang*

5. Entitas *invjenis*

   Tabel ini digunakan untuk menyimpan data jenis barang. Terdapat 2 buah atribut pada tabel ini. Atribut serta penjelasan dari tabel ini telah dijabarkan pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
kode	*varchar*	Kode Jenis
jenis	*varchar*	Nama Jenis

Tabel 5.5. Entitas *invjenis*

6. Entitas *invtype*

   Tabel ini digunakan untuk menyimpan daftar tipe barang. Terdapat 2 atribut yang digunakan pada tabel ini. Untuk atribut serta penjelasan dapat dilihat pada tabel berikut ini.

Nama Atribut	Tipe Data	Keterangan
kode	*varchar*	Kode Tipe
type	*varchar*	Nama Tipe

Tabel 5.6. Entitas *invtype*

7. Entitas beli

Tabel ini digunakan untuk menyimpan data transaksi pembelian dari *supplier*. Terdapat 12 atribut yang digunakan pada tabel ini. Atribut serta penjelasan telah dijabarkan pada tabel berikut ini.

Nama Atribut	Tipe Data	Keterangan
no_bukti	*varchar*	Nomor Nota
tanggal	*varchar*	Tanggal Nota
jatuh_tempo	*varchar*	Tanggal Jatuh Tempo
kode_supp	*varchar*	Kode *Supplier*
sub_total	*float*	Harga Sub Total
persen_ppn	*int*	Persentase PPN
total	*float*	Harga Total
lunas	*varchar*	Status Pembayaran
tgl_lunas	*varchar*	Tanggal Pelunasan
status	*varchar*	Status Pengiriman Barang
tgl_terkirim	*varchar*	Tanggal Terkirimnya Barang
author	*varchar*	Nama Pembuat Nota

Tabel 5.7. Entitas beli

8. Entitas *beli_dtl*

Tabel ini digunakan untuk mencatat informasi *detail* masing-masing nota pembelian. Terdapat 9 atribut pada tabel ini. Atribut serta penjelasan dari tabel ini telah dijabarkan pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
no_bukti	*varchar*	Nomor Nota
kode_brg	*varchar*	Kode Barang
nama_brg	*varchar*	Nama Barang
qty_order	*int*	Kuantitas Barang yang dipesan

packing	*varchar*	Packing Barang
id_satuan	*int*	*Id* Satuan Barang
hrg_per_unit	*float*	Harga Per Unit
hrg_total	*float*	Harga Total
kirim_gudang	*varchar*	Gudang Tujuan Barang

Tabel 5.8. Entitas *beli_dtl*

9. Entitas *customer*

   Tabel ini digunakan untuk menyimpan data *customer* perusahaan. Terdapat 11 buah atribut pada tabel ini. Atribut tabel serta penjelasan dapat dilihat pada tabel berikut ini.

Nama Atribut	Tipe Data	Keterangan
kode_cust	*varchar*	Kode Perusahaan Customer
nama_cust	*varchar*	Nama Perusahaan *Customer*
type_cust	*varchar*	Tipe Perusahaan *Customer*
alm_1	*varchar*	Alamat 1 Perusahaan *Customer*
alm_2	*varchar*	Alamat 2 Perusahaan *Customer*
alm_3	*varchar*	Alamat 3 Perusahaan *Customer*
kota	*varchar*	Kota Asal *Customer*
kontak	*varchar*	Nama Kontak *Customer*
no_telp	*varchar*	Nomor Telepon Kontak
email	*varchar*	Email Kontak
kode_sales	*varchar*	Kode *Sales Person*

Tabel 5.9. Entitas *customer*

10. Entitas jual

    Tabel ini digunakan untuk mencatat transaksi penjualan ke *customer*. Terdapat 12 atribut yang disediakan pada tabel ini. Untuk atribut tabel serta penjelasan dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
no_bukti	*varchar*	Nomor Nota
tanggal	*varchar*	Tanggal Nota
jatuh_tempo	*varchar*	Tanggal Jatuh Tempo
kode_cust	*varchar*	Kode *Customer*
sub_total	*float*	Harga Sub Total
persen_ppn	*int*	Persentase PPN
total	*float*	Harga Total
lunas	*varchar*	Status Pembayaran
tgl_lunas	*varchar*	Tanggal Pelunasan
status	*varchar*	Status Pengiriman Barang
tgl_terkirim	*varchar*	Tanggal Terkirimnya Barang
author	*varchar*	Nama Pembuat Nota

Tabel 5.10. Entitas jual

11. Entitas *jual_dtl*

Tabel ini digunakan untuk mencatat data *detail* masing-masing transaksi penjualan. Terdapat 9 atribut yang disediakan pada tabel ini. Atribut serta penjelasan untuk tabel ini dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
no_bukti	*varchar*	Nomor Nota
id_brg	*int*	Id Barang
kode_brg	*varchar*	Kode Barang
nama_brg	*varchar*	Nama Barang
qty_order	*int*	Kuantitas Barang Pesanan
id_satuan	*int*	*Id* Satuan Barang
hrg_per_unit	*float*	Harga Per Unit
hrg_total	*float*	Harga Total

kode_gudang	*varchar*	Gudang Asal Barang

Tabel 5.11. Entitas *jual_dtl*

12. Entitas *sales_person*

    Tabel ini digunakan untuk menyimpan data *sales person* untuk masing-masing *customer*. Terdapat 3 atribut yang digunakan pada tabel ini. Atribut tabel serta penjelasan telah dijabarkan pada tabel berikut ini.

Nama Atribut	Tipe Data	Keterangan
kode_sales	*varchar*	Kode *Sales Person*
nama_sales	*varchar*	Nama *Sales Person*
divisi	*varchar*	Divisi *Sales Person*

Tabel 5.12. Entitas *sales_person*

13. Entitas satuan

    Tabel ini digunakan untuk menyimpan data satuan barang. Terdapat 2 buah atribut pada tabel ini. Atribut tabel serta penjelasan dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
id	*int*	*Id* Satuan
satuan	*varchar*	Nama Satuan

Tabel 5.13. Entitas satuan

14. Entitas *supplier*

    Tabel ini digunakan untuk menyimpan data *supplier* perusahaan. Terdapat 12 buah atribut pada tabel ini. Atribut tabel serta penjelasan telah dijabarkan pada tabel berikut ini.

Nama Atribut	Tipe Data	Keterangan
kode_supp	*varchar*	Kode Perusahaan *Supplier*
nama_supp	*varchar*	Nama Perusahaan *Supplier*
bank	*varchar*	Nama Bank
acc_bank	*varchar*	Nomor Rekening Perusahaan *Supplier*
alm_1	*varchar*	Alamat 1 Perusahaan *Supplier*
alm_2	*varchar*	Alamat 2 Perusahaan *Supplier*
kota	*varchar*	Kota Asal Perusahaan *Supplier*
negara	*varchar*	Negara Asal Perusahaan *Supplier*
kontak	*varchar*	Nama Kontak *Supplier*
jabatan	*varchar*	Jabatan Kontak
no_telp	*varchar*	Nomor Telepon Kontak
email	*varchar*	Email Kontak

Tabel 5.14. Entitas *supplier*

15. Entitas *mutasi_stok*

Tabel ini digunakan untuk menyimpan jumlah barang yang masuk dan keluar di gudang tertentu pada suatu transaksi. Terdapat 6 atribut yang disediakan pada tabel ini. Atribut serta penjelasan untuk entitas ini dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
id	*int*	Id Mutasi
no_bukti	*varchar*	Nomor Nota
kode_brg	*varchar*	Kode Barang
kode_gudang	*varchar*	Kode Gudang
qty_masuk	*int*	Jumlah Kuantitas Masuk

qty_keluar	*int*	Jumlah Kuantitas Keluar

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 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 *Master*

Pada halaman *master*, ditampilkan daftar barang pada perusahaan. Data yang ditampilkan diambil dari tabel *invmaster*. Informasi dari tabel *master* akan ditampilkan hanya 5 buah barang untuk setiap halamannya. Fitur ini diimplementasikan menggunakan fungsi *paginate* setelah proses *filter* dilakukan. *Filter* berdasarkan gudang, jenis barang, maupun kode atau nama barang akan dilakukan terlebih dahulu jika pengguna menginputkan nama gudang melalui *combo box,* nama jenis barang melalui sebuah *combo box* juga, atau mengisi nama barang pada sebuah *input field*. Untuk informasi divisi, jenis, gudang, dan tipe barang masing-masing akan diambil dari tabel *invdivisi, invjenis, invgudang*, dan *invtype*. Sintaks pengambilan data dari masing-masing tabel tersebut untuk ditampilkan pada halaman *master* dapat dilihat pada listing berikut ini.

Listing 5.2. Proses Pengambilan Data Halaman *Master*

```
public function master(Request $request){
    $selectedGudang = $request->get('gudang');
    $jenis = $request->get('jenis');
    $search = $request->get('search');
    $query = Master::query();

    if($selectedGudang && $selectedGudang != 'All'){
        $query->whereHas('gudang', function ($q)
        use ($selectedGudang) {
            $q->where('nama', $selectedGudang);
        });
    }

    if($jenis && $jenis != 'All'){
        $query->whereHas('jenis', function ($j) use ($jenis){
            $j->where('jenis',$jenis);
```

```
        });
    }

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

    $master = $query->paginate(5);
    $divisi = Divisi::all();
    $gudang = Gudang::all();
    $jenis = Jenis::all();
    $type = Type::all();
    $satuan = Satuan::all();

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

Pada halaman *master*, pengguna dapat melakukan *filter* data barang berdasarkan gudang, jenis, dan juga nama barang. Jika pengguna menginputkan *combo box* gudang, *combo box* jenis, atau *input field search* yang tersedia di halaman *master*, maka data yang diinputkan akan diambil dan dikirim ke fungsi *master* menggunakan fungsi *updateTableData* untuk kemudian dilakukan *filter* pada *query master* agar dapat mengambil data barang dengan gudang, jenis, atau nama barang yang sesuai. Setelah itu, data hasil *filter* kemudian dikirim ke halaman *master* dan *paging* akan diupdate untuk memperbarui tampilan. Sintaks dari proses pengiriman data untuk *filter* data barang berdasarkan gudang dan nama barang dapat dilihat pada listing berikut ini.

Listing 5.3. Proses Pengiriman Data untuk *Filter* Barang

```
function updateTableData(page){
      var selectedGudang = $('#filterGudang').val();
      var selectedJenis = $('#filterJenis').val();
      var searchText = $('#searchItem').val();

      $.ajax({
         url: "{{route('master')}}",
         type: "GET",
         data: {gudang: selectedGudang, jenis: selectedJenis,
         search: searchText, page: page},
         success: function(data){
            $('.table tbody').html($(data).find('.table tbody').html());
            $('.text-center').html($(data).find('.text-center').html());
         }
      });
}

$('#filterGudang,#filterJenis,#searchItem').on('changekeyup', function(){
      updateTableData(1);
});
```

Ketika ingin melakukan opname stok barang jika terdapat suatu barang yang rusak atau *expired* pada gudang, pengguna dapat menekan tombol “Opname” yang terletak pada kolom “Actions” pada tabel *master*. Jika ditekan, maka akan muncul sebuah dialog yang dapat digunakan untuk menginputkan jumlah barang serta keterangan yang menandakan bahwa barang tersebut rusak atau *expired*. Setelah menginputkan data diatas, informasi tersebut kemudian dikirim ke fungsi *opnameBarang* untuk dilakukan pengurangan jumlah barang yang layak dijual oleh perusahaan. Sintaks untuk proses pengiriman data untuk dilakukan proses opname barang dapat dilihat pada listing di bawah ini.

Listing 5.4. Proses Pengiriman Data untuk Opname Barang

```
$(document).on('click', '.btn-opname', function(e) {
    e.preventDefault();
    var kodeBrg = $(this).data('kode');
    var namaBrg = $(this).data('nama');
    var divisiBrg = $(this).data('divisi');
    var jenisBrg = $(this).data('jenis');
    var tipeBrg = $(this).data('tipe');
    var packingBrg = $(this).data('packing');
    var quantityBrg = $(this).data('quantity');
    var satuanBrg = $(this).data('satuan');
    var gudangBrg = $(this).data('gudang');

    $('#kode-barang').val(kodeBrg);
    $('#nama-barang').val(namaBrg);
    $('#qty_sistem').val(quantityBrg);
    $('#selisih').attr('max', quantityBrg);
    $('#opnameModal').modal('show');

    $('#simpanOpname').click(function() {
      var selisih = $('#selisih').val();
      var keterangan = $('#keterangan').val();
      $.ajax({
        url: "{{route('OpnameBarang')}}",
        type: 'GET',
        data: {kode_brg: kodeBrg, nama_brg: namaBrg,
          kode_divisi: divisiBrg, kode_jenis: jenisBrg, kode_type:
          tipeBrg, packing: packingBrg,
          quantity: selisih, id_satuan: satuanBrg, hrg_jual: 0,
          kode_gudang: gudangBrg, keterangan: keterangan},
        success: function(response) {
          $('#opnameModal').modal('hide');

          $('#kode-barang').val('');
          $('#nama-barang').val('');
          $('#qty_sistem').val('');
          $('#selisih').val('');
          $('.modal-backdrop').remove();
        },
```

```
          error: function(xhr, status, error) {
            console.error(xhr.responseText);
          }
        });
        $('#opnameModal').modal('hide');
      });
});
```

Informasi yang dikirimkan melalui dialog akan diterima oleh fungsi *opnameBarang* untuk digunakan dalam menjalankan proses opname barang. Setelah menerima seluruh informasi, dilakukan proses pencarian apakah barang dan keterangan yang sama telah tercatat pada sistem. Jika sudah tercatat, maka dilakukan penambahan kuantitas khusus untuk baris data tersebut. Jika belum tercatat, maka dilakukan penambahan data barang dengan kuantitas dan keterangan yang diinputkan pengguna untuk membentuk baris data baru. Kuantitas barang yang sama dan tidak rusak atau *expired* akan dikurangi sesuai dengan jumlah barang yang diinputkan pengguna pada dialog.

Berikutnya dilakukan proses pembaharuan harga jual menggunakan metode FIFO. Metode ini melakukan proses penghitungan harga jual dengan mengalikan jumlah barang yang dibeli dengan harga beli per satuan terlebih dahulu, untuk kemudian dijumlahkan secara keseluruhan agar mendapatkan biaya total pembelian untuk barang tersebut. Selanjutnya, biaya total yang didapat akan dibagi dengan jumlah stok barang yang tidak rusak atau *expired* untuk mendapatkan harga penjualan minimum. Harga jual barang akan ditetapkan sebesar 50% lebih tinggi dari harga jual

minimum. Harga jual tersebut akan diinputkan ke baris data barang terkait yang tidak memiliki keterangan rusak atau *expired*. Sintaks dari proses opname barang ini dapat dilihat pada listing berikut ini.

Listing 5.5. Proses Opname Barang

```
public function opnameBarang(Request $request){
        $kode_brg = $request->input('kode_brg');
        $nama_brg = $request->input('nama_brg');
        $kode_divisi = $request->input('kode_divisi');
        $kode_jenis = $request->input('kode_jenis');
        $kode_type = $request->input('kode_type');
        $packing = $request->input('packing');
        $quantity = $request->input('quantity');
        $id_satuan = $request->input('id_satuan');
        $hrg_jual = $request->input('hrg_jual');
        $kode_gudang = $request->input('kode_gudang');
        $keterangan = $request->input('keterangan');

        $master = Master::where('kode_brg', $kode_brg)
            ->where('nama_brg', $nama_brg)
            ->where('kode_gudang', $kode_gudang)
            ->where('keterangan', $keterangan)
            ->first();

        if($master){
            DB::table('invmaster')
            ->where('kode_brg', $kode_brg)
            ->where('nama_brg', $nama_brg)
            ->where('kode_gudang', $kode_gudang)
            ->where('keterangan', [$keterangan])
            ->increment('quantity', $quantity);

        }else{
            DB::table('invmaster')->insert([
                'kode_brg' => $kode_brg,
                'nama_brg' => $nama_brg,
                'kode_divisi' => $kode_divisi,
                'kode_jenis' => $kode_jenis,
```

```
            'kode_type' => $kode_type,
            'packing' => $packing,
            'quantity' => $quantity,
            'id_satuan' => $id_satuan,
            'hrg_jual' => $hrg_jual,
            'kode_gudang' => $kode_gudang,
            'keterangan' => $keterangan,
        ]);
    }
    $keteranganArray = ["BARANG RUSAK",
      "BARANG EXPIRED", "BARANG RUSAK & EXPIRED"];

    DB::table('invmaster')
        ->where('kode_brg', $kode_brg)
        ->where('nama_brg', $nama_brg)
        ->where('kode_gudang', $kode_gudang)
        ->whereNotIn('keterangan', $keteranganArray)
        ->decrement('quantity', $quantity);

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

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

    $currentQuantity = DB::table('invmaster')
        ->where('kode_brg', $kode_brg)
        ->whereNotIn('keterangan', $keteranganArray)
        ->sum('quantity');

    $minSellPrice = $totalCost / $currentQuantity;
    $markup = $minSellPrice * 0.5;
    $sellPrice = $minSellPrice + $markup;

    DB::table('invmaster')
        ->where('kode_brg', $kode_brg)
```

```
            ->whereNotIn('keterangan', $keteranganArray)
            ->update(['hrg_jual' => $sellPrice]);

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

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 5 data ditampilkan untuk setiap halaman. Untuk informasi *supplier* dan gudang di setiap transaksi, dilakukan pengambilan seluruh data pada tabel supplier dan *invgudang*. Pada akhirnya, data transaksi beli, *supplier*, dan gudang akan dikirim ke halaman transaksi beli untuk ditampilkan pada tabel. Sintaks untuk menampilkan data transaksi pembelian dapat dilihat pada listing di bawah ini.

Listing 5.6. 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(5);

    $supplier = Supplier::all();
    $gudang = Gudang::all();

    return view('transaksi.beli',compact('beli','supplier','gudang',
    'selectedDate'));
}
```

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->create_time = Carbon::now()->format('d-m-Y');
        $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');
        $packing = $request->get('packing');
        $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->packing = $packing[$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. Perubahan tersebut dilakukan pada *database* dan berikutnya dilakukan proses penyimpanan informasi barang masuk ke tabel *mutasi_stok*. 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 pembaharuan harga jual untuk barang yang dibeli menggunakan metode FIFO karena terjadi perubahan jumlah stok barang tersebut pada gudang. Setelah menyimpan harga jual terbaru di

database, pengguna diarahkan kembali ke halaman transaksi pembelian dan dimunculkan notifikasi bahwa status pengiriman berhasil dirubah. Sintaks dari proses pengubahan status pembayaran dan pengiriman dapat dilihat pada listing di bawah ini.

Listing 5.8. Proses *Update* 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) {
            DB::table('mutasi_stok')->insert([
                'no_bukti' => $no_bukti,
```

```
            'kode_brg' => $detail->kode_brg,
            'kode_gudang' => $detail->kirim_gudang,
            'qty_masuk' => $detail->qty_order,
            'qty_keluar' => 0,
        ]);

        $master = DB::table('invmaster')
            ->where('kode_brg', $detail->kode_brg)
            ->where('nama_brg', $detail->nama_brg)
            ->where('kode_gudang', $detail->kirim_gudang)
            ->first();

        $keteranganArray = ["BARANG RUSAK",
          "BARANG EXPIRED", "BARANG RUSAK & EXPIRED"];
        if ($master) {
            DB::table('invmaster')
                ->where('kode_brg', $detail->kode_brg)
                ->where('nama_brg', $detail->nama_brg)
                ->where('kode_gudang', $detail->kirim_gudang)
                ->increment('quantity', $detail->qty_order);

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

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

            $currentQuantity = DB::table('invmaster')
            ->where('kode_brg', $detail->kode_brg)
            ->whereNotIn('keterangan', $keteranganArray)
            ->sum('quantity');

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

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

        } else {
            DB::table('invmaster')->insert([
                'kode_brg' => $detail->kode_brg,
                'nama_brg' => $detail->nama_brg,
                'packing' => $detail->packing,
                'quantity' => $detail->qty_order,
                'id_satuan' => $detail->id_satuan,
                'kode_gudang' => $detail->kirim_gudang,
                'keterangan' => '-',
            ]);

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

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

            $currentQuantity = DB::table('invmaster')
            ->where('kode_brg', $detail->kode_brg)
            ->sum('quantity');

            $minSellPrice = $totalCost / $currentQuantity;
            $markup = $minSellPrice * 0.5;
            $sellPrice = $minSellPrice + $markup;

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

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

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, 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. Untuk data satuan dan gudang barang pada *detail* akan diambil dari tabel satuan dan *invgudang*. Sintaks dari proses diatas dapat dilihat pada listing di bawah ini.

Listing 5.9. 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'));
    }
```

5.2.4. Proses pada Halaman Transaksi Penjualan

Halaman transaksi penjualan akan menampilkan data transaksi penjualan ke *customer*. Data transaksi diambil menggunakan fungsi jual yang berparameter *request* untuk menerima data *filter* berdasarkan tanggal yang akan dikirim ke fungsi ini jika pengguna menginputkan suatu tanggal pada *date picker* di halaman transaksi penjualan. Jika tanggal *filter* tersebut didapatkan, maka data tersebut akan disertakan pada *query* tabel jual agar pada proses pengambilan data transaksi, hanya diambil data yang memiliki *value* kolom tanggal yang sama dengan tanggal *filter*. Jika tidak ada *request* dari halaman transaksi penjualan, maka proses *filter* tidak akan dijalankan sehingga fungsi *query* akan mengambil seluruh data transaksi penjualan. Data transaksi akan diurutkan berdasarkan tanggal secara *descending* sehingga transaksi dengan tanggal terbaru akan ditampilkan terlebih dahulu pada tabel. Fitur *paging* juga diterapkan dengan menampilkan 5 data transaksi untuk setiap halaman. Untuk data *customer* akan diambil dari tabel *customer*. Data transaksi penjualan dan *customer* akan dikirim ke halaman transaksi penjualan untuk ditampilkan pada tabel. Sintaks dari proses pengambilan data transaksi ini telah dijabarkan pada listing di bawah ini.

Listing 5.10. Proses Pengambilan Data Transaksi Penjualan

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

    $query = Jual::query();

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

```
        }

        $jual = $query->orderBy('tanggal', 'desc')->paginate(5);

        $customer = Customer::all();

       return view('transaksi.jual',compact('jual','customer','selectedDate'));
    }
```

Pada halaman ini, disediakan juga tombol "Tambah Transaksi" yang akan mengarahkan pengguna menuju halaman tambah transaksi. Pengguna dapat menginputkan informasi transaksi serta *detail* penjualan baru dengan memasukkan data ke kumpulan *input field*. Setelah proses *input* selesai dilakukan, informasi-informasi tersebut kemudian dikirim ke fungsi *store* yang akan menerima semua data untuk dimasukkan ke *database*. Setelah itu, pengguna akan diarahkan kembali ke halaman transaksi penjualan dan muncul notifikasi bahwa proses penambahan transaksi serta *detail* berhasil dilakukan. Sintaks dari proses diatas dapat dilihat pada listing di bawah ini.

Listing 5.11. Proses Tambah *Detail* dan Transaksi Penjualan

```
public function store(Request $request)
    {
        $data = new Jual();
        $data->no_bukti = $request->get('no_bukti');
        $data->tanggal = $request->get('datepicker');
        $data->kode_cust = $request->get('select_customer');
        $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->create_time = Carbon::now()->format('d-m-Y');
        $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();

        $id_brg = $request->get('id_brg');
        $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');
        $kode_gudang = $request->get('select_gudang');

        foreach($kode_brg as $key => $value) {
            $detail = new JualDetail();
            $detail->id_brg = $id_brg[$key];
            $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->kode_gudang = $kode_gudang[$key];
            $detail->save();
        }

        return redirect()->route('penjualan')->with('status','Hooray!!
          Your new transaction is already inserted');
    }
```

Halaman ini juga menyediakan fitur *update* status pembayaran dan pengiriman. Proses *update* status pembayaran dilakukan pada fungsi *updateBayar* yang akan merubah data lunas pada suatu transaksi dari yang sebelumnya dinyatakan “Belum Lunas” menjadi “Lunas”. Proses *update*

status pengiriman dilakukan pada fungsi *updateKirim* yang akan merubah data status pada suatu transaksi dari "Belum Terkirim" menjadi "Sudah Terkirim". Kedua fungsi membutuhkan data *no_bukti* untuk dapat mencari transaksi penjualan yang ingin diupdate statusnya. Setelah mengubah status pengiriman menjadi "Sudah Terkirim", dilakukan penambahan jumlah barang yang keluar sesuai dengan setiap transaksi pada tabel *mutasi_stok*. Berikutnya, dilakukan pengurangan jumlah stok barang yang dijual pada gudang tertentu. Selanjutnya dilakukan proses pengubahan harga jual untuk setiap barang yang dijual menggunakan metode FIFO. Harga jual barang yang terbaru akan ditambahkan ke masing-masing barang terkait. Sintaks dari proses diatas secara keseluruhan telah ditampilkan pada listing berikut ini.

Listing 5.12. Proses *Update* Status Pembayaran dan Pengiriman pada Transaksi Penjualan

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

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

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

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

    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) {
   DB::table('mutasi_stok')->insert([
      'no_bukti' => $no_bukti,
      'kode_brg' => $detail->kode_brg,
      'kode_gudang' => $detail->kode_gudang,
      'qty_masuk' => 0,
      'qty_keluar' => $detail->qty_order,
   ]);

   $master = Master::find($detail->id_brg);

   $keteranganArray = ["BARANG RUSAK",
     "BARANG EXPIRED", "BARANG RUSAK & EXPIRED"];
   if ($master) {
      $master->quantity -= $detail->qty_order;
      $master->save();

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

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

      $currentQuantity = DB::table('invmaster')
      ->where('kode_brg', $detail->kode_brg)
```

```
            ->whereNotIn('keterangan', $keteranganArray)
            ->sum('quantity');

            $minSellPrice = $totalCost / $currentQuantity;
            $markup = $minSellPrice * 0.5;
            $sellPrice = $minSellPrice + $markup;

            DB::table('invmaster')
               ->where('kode_brg', $detail->kode_brg)
               ->whereNotIn('keterangan', $keteranganArray)
               ->update(['hrg_jual' => $sellPrice]);
         }
      }

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

Pengguna dapat mengakses halaman *detail* untuk setiap transaksi penjualan agar dapat melihat data *detail* berupa informasi mengenai barang yang dijual ke *customer*. Sistem akan menjalankan fungsi *showDetail* yang menerima parameter *no_bukti* yang akan digunakan untuk menemukan semua *detail* yang terkait dengan suatu transaksi. Untuk data satuan dan gudang akan diambil dari tabel satuan dan *invgudang*. Seluruh data *detail* dan satuan kemudian dikirim ke halaman *detail* penjualan untuk ditampilkan pada tabel. Sintaks dari proses ini dapat dilihat pada listing berikut ini.

Listing 5.13. Proses Pengambilan Data *Detail* Transaksi Penjualan

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

```
        return view('transaksi.jualdetail', compact('jualDetail','satuan',
          'gudang','no_bukti'));
}
```

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 5 data per halaman. Data *supplier* kemudian dikirim ke halaman *supplier* untuk ditampilkan pada tabel. Sintaks dari proses diatas dapat dilihat pada listing di bawah ini.

Listing 5.14. 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(5);
```

```
    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. Proses ini dapat dilihat pada listing berikut ini.

Listing 5.15. 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->bank = $request->get('bank');
        $data->acc_bank = $request->get('acc_bank');
        $data->alm_1 = $request->get('alm_1');
        $data->alm_2 = $request->get('alm_2');
        $data->kota = $request->get('kota');
        $data->negara = $request->get('negara');
        $data->kontak = $request->get('kontak');
        $data->jabatan = $request->get('jabatan');
        $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. Sintaks dari proses ini dapat dilihat pada listing berikut ini.

Listing 5.16. 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->bank = $request->get('bank');
        $objSupplier->acc_bank = $request->get('acc_bank');
        $objSupplier->alm_1 = $request->get('alm_1');
```

```
        $objSupplier->alm_2 = $request->get('alm_2');
        $objSupplier->kota = $request->get('kota');
        $objSupplier->negara = $request->get('negara');
        $objSupplier->kontak = $request->get('kontak');
        $objSupplier->jabatan = $request->get('jabatan');
        $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*. Sintaks dari proses ini dapat dilihat pada listing di bawah ini.

Listing 5.17. 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){
            $msg = "Delete data failed. Make sure there is no correlated
            data before deleting!";
            return redirect()->route('supplier')->with('status',$msg);
```

```
    }
}
```

5.2.6. Proses pada Halaman *Customer*

Halaman ini digunakan untuk menampilkan data *customer* perusahaan dalam bentuk tabel. Data *customer* akan dikirim ke halaman ini melalui fungsi *customer* berparamater *request* untuk mendapatkan data *search* yang digunakan untuk *filter* berdasarkan nama *customer* yang akan dikirim ke fungsi ini jika pengguna menginputkan teks di *input field search* pada halaman *customer*. Data *search* tersebut akan dimanfaatkan dalam proses pengambilan data *customer* sehingga data yang diambil hanya yang sesuai dengan teks *search* yang telah diinputkan pengguna. Jika tidak menggunakan fitur *search* tersebut, maka fungsi akan mengambil semua data dari tabel *customer*. Fitur *paging* akan menampilkan 5 data per halaman. Data *customer* kemudian dikirim ke halaman *customer* untuk ditampilkan pada tabel. Sintaks dari proses diatas dapat dilihat pada listing berikut ini.

Listing 5.18. Proses Pengambilan Data *Customer*

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

    $query = Customer::query();

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

    $customer = $query->paginate(5);
```

```
        $salesPerson = SalesPerson::all();

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

Pengguna dapat melakukan penambahan data *customer* melalui tombol “Tambah Customer”. 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* data ke *database*, sistem akan mengarahkan pengguna kembali ke halaman *customer* dan ditampilkan notifikasi bahwa data *customer* berhasil ditambahkan. Proses ini dapat dilihat pada listing berikut ini.

Listing 5.19. Proses Tambah *Customer*

```
public function store(Request $request)
    {
        $data = new Customer();
        $data->kode_cust = $request->get('kode_cust');
        $data->nama_cust = $request->get('nama_cust');
        $data->type_cust = $request->get('type_cust');
        $data->alm_1 = $request->get('alm_1');
        $data->alm_2 = $request->get('alm_2');
        $data->alm_3 = $request->get('alm_3');
        $data->kota = $request->get('kota');
        $data->kontak = $request->get('kontak');
        $data->no_telp = $request->get('no_telp');
        $data->kode_sales = $request->get('select_sales');
        $data->save();
        return redirect()->route('customer')->with('status','Hooray!!
          Your new item is already inserted');
    }
```

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

Listing 5.20. Proses Edit *Customer*

```
public function edit($kode_cust)
    {
        $objCustomer = Customer::find($kode_cust);
        $salesPerson = SalesPerson::all();

        $data = $objCustomer;
        return view('customer.formedit',compact('data','salesPerson'));
    }

    public function update(Request $request, $kode_cust)
    {
        $objCustomer = Customer::find($kode_cust);
        $objCustomer->kode_cust = $request->get('kode_cust');
        $objCustomer->nama_cust = $request->get('nama_cust');
        $objCustomer->type_cust = $request->get('type_cust');
        $objCustomer->alm_1 = $request->get('alm_1');
```

```
        $objCustomer->alm_2 = $request->get('alm_2');
        $objCustomer->alm_3 = $request->get('alm_3');
        $objCustomer->kota = $request->get('kota');
        $objCustomer->kontak = $request->get('kontak');
        $objCustomer->no_telp = $request->get('no_telp');
        $objCustomer->kode_sales = $request->get('select_sales');
        $objCustomer->save();
        return redirect()->route('customer')->with('status','Your item is
          up-to-date');
    }
```

Pada halaman *customer* juga disediakan tombol "Delete" pada setiap baris data *customer* untuk menghapus suatu data *customer*. Jika pengguna menekan tombol tersebut, maka data *kode_cust* dari *customer* tersebut akan dikirim ke fungsi *destroy* sebagai acuan untuk menemukan data *customer* berdasarkan kode *customer* agar bisa dihapus dari *database*. Jika berhasil, maka pengguna diarahkan kembali ke halaman *customer* dan muncul notifikasi bahwa proses penghapusan berhasil. Jika gagal, maka muncul notifikasi bahwa hapus *customer* gagal dilakukan. Sintaks dari proses ini dapat dilihat pada listing di bawah ini.

Listing 5.21. Proses Hapus *Customer*

```
public function destroy($kode_cust)
    {
        try{
            $objCustomer = Customer::find($kode_cust);
            $objCustomer->delete();
            return redirect()->route('customer')->with('status',
            'Deleted Successfully');
        }
        catch(\PDOException $ex){
            $msg = "Delete data failed. Make sure there is no correlated
            data before deleting!";
```

```
            return redirect()->route('customer')->with('status',$msg);
        }
    }
```