

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 14 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. Atribut yang disediakan beserta fungsinya dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
<i>id</i>	<i>bigint (primary key)</i>	<i>Id</i> pengguna
<i>name</i>	<i>varchar</i>	Nama pengguna
<i>email</i>	<i>varchar</i>	Email pengguna
<i>password</i>	<i>varchar</i>	<i>Password</i> akun pengguna

Tabel 5.1. Entitas *users*

2. Entitas *invmaster*

Tabel ini digunakan untuk menyimpan data barang perusahaan. Atribut beserta penjelasan dari tabel ini dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
<i>kode_brg</i>	<i>varchar</i>	Kode Barang
<i>nama_brg</i>	<i>varchar</i>	Nama Barang
<i>kode_divisi</i>	<i>varchar</i>	Kode Divisi Barang
<i>kode_jenis</i>	<i>varchar</i>	Kode Jenis Barang
<i>kode_type</i>	<i>varchar</i>	Kode Tipe Barang
<i>packing</i>	<i>varchar</i>	<i>Packing</i> Barang
<i>Quantity</i>	<i>int</i>	Kuantitas Barang
<i>id_satuan</i>	<i>int</i>	<i>Id</i> Satuan Barang
<i>hrg_jual_item</i>	<i>float</i>	Harga Jual Per Item
<i>hrg_jual</i>	<i>float</i>	Harga Jual Per Pack
<i>hrg_jual_total</i>	<i>decimal(18,2)</i>	Harga Jual Total
<i>kode_gudang</i>	<i>varchar</i>	Kode Gudang Barang
<i>keterangan</i>	<i>varchar</i>	Keterangan

Tabel 5.2. Entitas *invmaster*3. Entitas *invdivisi*

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

Nama Atribut	Tipe Data	Keterangan
<i>kode</i>	<i>varchar</i>	Kode Divisi
<i>divisi</i>	<i>varchar</i>	Nama Divisi

Tabel 5.3. Entitas *invdivisi*

4. Entitas *invgudang*

Tabel ini digunakan untuk menyimpan daftar gudang pada perusahaan.

Atribut serta penjelasan dapat dilihat pada tabel berikut ini.

Nama Atribut	Tipe Data	Keterangan
kode	<i>varchar</i>	Kode Gudang
nama	<i>varchar</i>	Nama Gudang

Tabel 5.4. Entitas *invgudang*

5. Entitas *invjenis*

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

Nama Atribut	Tipe Data	Keterangan
kode	<i>varchar</i>	Kode Jenis
jenis	<i>varchar</i>	Nama Jenis

Tabel 5.5. Entitas *invjenis*

6. Entitas *invtype*

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

Nama Atribut	Tipe Data	Keterangan
kode	<i>varchar</i>	Kode Tipe
<i>type</i>	<i>varchar</i>	Nama Tipe

Tabel 5.6. Entitas *invtype*

7. Entitas *beli*

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

Nama Atribut	Tipe Data	Keterangan
<i>no_bukti</i>	<i>varchar</i>	Nomor Nota
<i>tanggal</i>	<i>varchar</i>	Tanggal Nota
<i>jatuh_tempo</i>	<i>varchar</i>	Tanggal Jatuh Tempo
<i>kode_supp</i>	<i>varchar</i>	Kode <i>Supplier</i>
<i>mata_uang</i>	<i>varchar</i>	Mata Uang Pembayaran
<i>irim_gudang</i>	<i>varchar</i>	Alamat Kirim Gudang
<i>sub_total</i>	<i>float</i>	Harga Sub Total
<i>persen_ppn</i>	<i>int</i>	Persentase PPN
<i>total</i>	<i>float</i>	Harga Total
<i>lunas</i>	<i>varchar</i>	Status Pembayaran
<i>status</i>	<i>varchar</i>	Status Pengiriman Barang
<i>create_time</i>	<i>varchar</i>	Tanggal <i>Input</i> Nota
<i>author</i>	<i>varchar</i>	Nama Pembuat Nota

Tabel 5.7. Entitas *beli*8. Entitas *beli_dtl*

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

Nama Atribut	Tipe Data	Keterangan
<i>no_bukti</i>	<i>varchar</i>	Nomor Nota
<i>kode_brg</i>	<i>varchar</i>	Kode Barang
<i>nama_brg</i>	<i>varchar</i>	Nama Barang
<i>qty_order</i>	<i>int</i>	Kuantitas Barang yang dipesan
<i>id_satuan</i>	<i>int</i>	<i>Id</i> Satuan Barang
<i>hrg_per_unit</i>	<i>float</i>	Harga Per Unit
<i>hrg_total</i>	<i>float</i>	Harga Total

Tabel 5.8. Entitas *beli_dtl*

9. Entitas *customer*

Tabel ini digunakan untuk menyimpan data *customer* perusahaan.

Atribut tabel serta penjelasan dapat dilihat pada tabel berikut ini.

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

Tabel 5.9. Entitas *customer*

10. Entitas jual

Tabel ini digunakan untuk mencatat transaksi penjualan ke *customer*.

Untuk atribut tabel serta penjelasan dapat dilihat pada tabel di bawah ini.

Nama Atribut	Tipe Data	Keterangan
<i>no_bukti</i>	<i>varchar</i>	Nomor Nota
<i>tanggal</i>	<i>varchar</i>	Tanggal Nota
<i>jatuh_tempo</i>	<i>varchar</i>	Tanggal Jatuh Tempo
<i>kode_cust</i>	<i>varchar</i>	Kode <i>Customer</i>
<i>mata_uang</i>	<i>varchar</i>	Mata Uang Pembayaran
<i>sub_total</i>	<i>float</i>	Harga Sub Total

<i>persen_ppn</i>	<i>int</i>	Persentase PPN
<i>total</i>	<i>float</i>	Harga Total
<i>lunas</i>	<i>varchar</i>	Status Pembayaran
<i>status</i>	<i>varchar</i>	Status Pengiriman Barang
<i>create_time</i>	<i>varchar</i>	Tanggal <i>Input</i> Nota
<i>author</i>	<i>varchar</i>	Nama Pembuat Nota

Tabel 5.10. Entitas *jual*11. Entitas *jual_dtl*

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

Nama Atribut	Tipe Data	Keterangan
<i>no_bukti</i>	<i>varchar</i>	Nomor Nota
<i>kode_brg</i>	<i>varchar</i>	Kode Barang
<i>nama_brg</i>	<i>varchar</i>	Nama Barang
<i>qty_order</i>	<i>int</i>	Kuantitas Barang Pesanan
<i>id_satuan</i>	<i>int</i>	<i>Id</i> Satuan Barang
<i>hrp_per_unit</i>	<i>float</i>	Harga Per Unit
<i>hrp_total</i>	<i>float</i>	Harga Total

Tabel 5.11. Entitas *jual_dtl*12. Entitas *sales_person*

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

Nama Atribut	Tipe Data	Keterangan
<i>kode_sales</i>	<i>varchar</i>	Kode <i>Sales Person</i>
<i>nama_sales</i>	<i>varchar</i>	Nama <i>Sales Person</i>

divisi	<i>varchar</i>	Divisi <i>Sales Person</i>
--------	----------------	----------------------------

Tabel 5.12. Entitas *sales_person*

13. Entitas satuan

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

Nama Atribut	Tipe Data	Keterangan
<i>id</i>	<i>int</i>	<i>Id</i> Satuan
satuan	<i>varchar</i>	Nama Satuan

Tabel 5.13. Entitas satuan

14. Entitas *supplier*

Tabel ini digunakan untuk menyimpan data *supplier* perusahaan.

Atribut tabel serta penjelasan telah dijabarkan pada tabel berikut ini.

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

Tabel 5.14. Entitas *supplier*

5.2. Implementasi Program

5.2.1. Proses pada Halaman *Login*

Pada halaman *login*, pengguna akan memasukkan *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 atau nama barang akan dilakukan terlebih dahulu jika pengguna menginputkan nama gudang melalui *combo box* 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');
    $search = $request->get('search');
    $query = Master::query();

    if($selectedGudang && $selectedGudang != 'All'){
        $query->whereHas('gudang', function ($q)
        use ($selectedGudang) {
            $q->where('nama', $selectedGudang);
        });
    }
    if($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 dan juga nama barang. Jika pengguna menginputkan *combo box* gudang 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 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 searchText = $('#searchItem').val();

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

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

Pengguna dapat melakukan penambahan data barang melalui halaman *master* dengan menekan tombol “Tambah Barang” dan pengguna akan diarahkan menuju halaman tambah barang. Pada halaman ini, pengguna akan menginputkan seluruh informasi yang dibutuhkan melalui *input field* yang disediakan. Setelah itu, data-data yang diinputkan akan dimasukkan ke tabel *master* di database. Jika data berhasil masuk ke *database*, maka pengguna akan diarahkan kembali ke halaman *master* dan muncul notifikasi bahwa data berhasil diinputkan. Sintaks dari proses penambahan barang diatas dapat dilihat pada listing di bawah ini.

Listing 5.4. Proses Tambah Barang

```
public function store(Request $request)
{
    $data = new Master();
    $data->kode_brg = $request->get('kode_brg');
    $data->nama_brg = $request->get('nama_brg');
    $data->kode_divisi = $request->get('select_divisi');
    $data->kode_jenis = $request->get('select_jenis');
    $data->kode_type = $request->get('select_type');
    $data->packing = $request->get('packing');
    $data->quantity = $request->get('quantity');
    $data->id_satuan = $request->get('select_satuan');
    $data->hrg_jual_item = $request->get('hrg_jual_item');
    $data->hrg_jual = $request->get('hrg_jual');
    $data->hrg_jual_total = $request->get('hrg_jual_total');
    $data->kode_gudang = $request->get('select_gudang');
    $data->keterangan = $request->get('keterangan');
    $data->save();
    return redirect()->route('master')->with('status','Hooray!! Your
    new item is already inserted');
}
```

Pada halaman *master*, pengguna juga dapat melakukan *edit* informasi barang dengan menekan tombol “Edit” pada suatu baris di tabel barang. Setelah menekan tombol tersebut, pengguna akan diarahkan menuju halaman *edit* barang untuk dapat mengubah informasi suatu barang. Ketika pengguna menekan tombol “Edit”, data kode barang akan dikirim ke fungsi *edit* untuk mendapatkan informasi barang dengan kode yang sama. Informasi yang didapat kemudian dikirim ke halaman *edit* barang untuk ditampilkan pada masing-masing *input field*. Setelah pengguna mengubah informasi barang pada *input field*, seluruh data pada *input field* akan dikirim ke fungsi *update* untuk melakukan *update* data barang dengan kode barang yang sesuai. Jika berhasil melakukan *update*, pengguna akan diarahkan menuju halaman *master* dan muncul notifikasi bahwa data barang berhasil diupdate. Sintaks dari proses *edit* barang secara keseluruhan dapat dilihat pada listing di bawah ini.

Listing 5.5. Proses *Edit* Barang

```
public function edit($kode_brg)
{
    $objMaster = Master::find($kode_brg);
    $divisi = Divisi::all();
    $gudang = Gudang::all();
    $jenis = Jenis::all();
    $type = Type::all();
    $satuan = Satuan::all();

    $data = $objMaster;
    return view('master.formedit',compact('data','divisi','gudang',
    'jenis','type','satuan'));
}
```

```
public function update(Request $request, $kode_brg)
{
    $objMaster = Master::find($kode_brg);
    $objMaster->kode_brg = $request->get('kode_brg');
    $objMaster->nama_brg = $request->get('nama_brg');
    $objMaster->kode_divisi = $request->get('select_divisi');
    $objMaster->kode_jenis = $request->get('select_jenis');
    $objMaster->kode_type = $request->get('select_type');
    $objMaster->packing = $request->get('packing');
    $objMaster->quantity = $request->get('quantity');
    $objMaster->id_satuan = $request->get('select_satuan');
    $objMaster->hrg_jual_item = $request->get('hrg_jual_item');
    $objMaster->hrg_jual = $request->get('hrg_jual');
    $objMaster->hrg_jual_total = $request->get('hrg_jual_total');
    $objMaster->kode_gudang = $request->get('select_gudang');
    $objMaster->keterangan = $request->get('keterangan');
    $objMaster->save();
    return redirect()->route('master')->with('status','Your item is
up-to-date');
}
```

Pada tabel *master* di masing-masing baris data barang terdapat tombol “Delete” yang jika ditekan oleh pengguna akan menjalankan proses penghapusan data barang pada *database* sesuai dengan barang pada baris tersebut. Proses penghapusan dilakukan menggunakan fungsi *destroy* yang menerima parameter kode barang yang didapat jika pengguna menekan tombol “Delete”. Berdasarkan data kode barang yang didapat, dilakukan pencarian terhadap barang dengan kode yang sama pada tabel *master* dan data tersebut akan dihapus dari *database*. Jika proses diatas berhasil, maka pengguna akan diarahkan kembali ke halaman *master* dan muncul notifikasi bahwa proses penghapusan berhasil dilakukan. Jika gagal, maka notifikasi

yang dimunculkan adalah proses penghapusan gagal. Sintaks dari proses penghapusan barang dapat dilihat pada listing di bawah ini.

Listing 5.6. Proses Hapus Barang

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

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 5.7. 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 5.8. 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->mata_uang = $request->get('mata_uang');
    $data->kirim_gudang = $request->get('select_gudang');
    $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->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');

    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->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 parameter *no_bukti* yang 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”. 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 parameter *no_bukti* sebagai acuan untuk menemukan transaksi yang ingin diubah status pengirimannya dari “Belum Terkirim” menjadi “Sudah Terkirim”. Perubahan tersebut dilakukan pada *database* dan dimunculkan notifikasi bahwa status pengiriman berhasil dirubah. Sintaks dari proses pengubahan status pembayaran dan pengiriman dapat dilihat pada listing di bawah ini.

Listing 5.9. Proses *Update* Status Pembayaran dan Pengiriman pada Transaksi Pembelian

```
public function updateBayar($no_bukti)
```

```
{
    $beli = Beli::where('no_bukti', $no_bukti)->firstOrFail();
    $beli->lunas = 'Lunas';
    $beli->save();

    return redirect()->route('pembelian')->with('status','Sukses
update status pembayaran');
}

public function updateKirim($no_bukti)
{
    $beli = Beli::where('no_bukti', $no_bukti)->firstOrFail();
    $beli->status = 'Sudah Terkirim';
    $beli->save();

    return redirect()->route('pembelian')->with('status','Sukses
update status pengiriman');
}
```

Pada halaman ini, disediakan juga fitur hapus transaksi pembelian yang akan menghapus transaksi beserta semua *detail* yang terlibat. Proses penghapusan transaksi dilakukan menggunakan fungsi *destroy* yang menerima parameter *no_bukti* yang berisikan nomor nota dari transaksi yang ingin dihapus. Langkah pertama dalam menghapus transaksi adalah mencari *detail* transaksi berdasarkan data nomor nota dan kemudian dilakukan penghapusan *detail* terlebih dahulu. Setelah itu, dilakukan pencarian data transaksi berdasarkan data nomor nota, lalu diterapkan fungsi *delete* untuk menghapus transaksi. Jika proses penghapusan berhasil, maka pengguna diarahkan kembali ke halaman transaksi pembelian dan muncul notifikasi bahwa transaksi sukses dihapus. Jika gagal, maka muncul pesan error bahwa proses penghapusan gagal. Sintaks dari proses

penghapusan transaksi serta *detail* pembelian dapat dilihat pada listing di bawah ini.

Listing 5.10. Proses Hapus *Detail* dan Transaksi Pembelian

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

Pengguna dapat melakukan navigasi 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. 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 barang

pada *detail* akan diambil dari tabel satuan. Sintaks dari proses diatas dapat dilihat pada listing di bawah ini.

Listing 5.11. Proses Pengambilan Data *Detail* Transaksi Pembelian

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

    return view('transaksi.belidetil', compact('beliDetail','satuan',
    '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 5.12. 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 5.13. 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->mata_uang = $request->get('mata_uang');
    $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->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');

    foreach($kode_brg as $key => $value) {
        $detail = new JualDetail();
        $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->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 parameter *no_bukti* untuk dapat mencari transaksi penjualan yang ingin diupdate statusnya. Sintaks dari proses diatas telah ditampilkan pada listing berikut ini.

Listing 5.14. Proses *Update* Status Pembayaran dan Pengiriman pada Transaksi Penjualan

```
public function updateBayar($no_bukti)
{
    $jual = Jual::where('no_bukti', $no_bukti)->firstOrFail();
    $jual->lunas = 'Lunas';
    $jual->save();

    return redirect()->route('penjualan')->with('status','Sukses
update status pembayaran');
}

public function updateKirim($no_bukti)
{
    $jual = Jual::where('no_bukti', $no_bukti)->firstOrFail();
    $jual->status = 'Sudah Terkirim';
    $jual->save();

    return redirect()->route('penjualan')->with('status','Sukses
update status pengiriman');
}
```

Pengguna dapat menghapus suatu data transaksi penjualan melalui tombol “Delete” yang disediakan di setiap baris transaksi pada tabel. Jika pengguna menekan tombol tersebut, maka sistem akan menjalankan fungsi *destroy* berparameter *no_bukti* yang digunakan sebagai acuan untuk menemukan transaksi serta *detail* yang ingin dihapus. Setelah mendapatkan *detail*, maka *detail* tersebut akan dihapus dari *database*. Kemudian data transaksi penjualan juga akan dihapus dan pengguna akan diarahkan kembali ke halaman transaksi penjualan dan diberi notifikasi bahwa proses penghapusan transaksi berhasil dilakukan. Sintaks dari proses ini telah dijabarkan pada listing di bawah ini.

Listing 5.15. Proses Hapus Detail dan Transaksi Penjualan

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

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 akan diambil dari tabel satuan. 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 5.16. Proses Pengambilan Data *Detail* Transaksi Penjualan

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

    return view('transaksi.jualdetail', compact('jualDetail','satuan',
        '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 parameter *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 5.17. 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 5.18. 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->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 5.19. Proses *Edit Supplier*

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

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

public function update(Request $request, $kode_suppl)
{
    $objSupplier = Supplier::find($kode_suppl);
    $objSupplier->kode_suppl = $request->get('kode_suppl');
    $objSupplier->nama_suppl = $request->get('nama_suppl');
    $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_suppl* 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 5.20. 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 memasukkan 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 5.21. 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 5.22. 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 5.23. 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 5.24. 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);
    }
}
```