BAB VI
IMPLEMENTASI

Pada bab ini akan dijelaskan beberapa segmen program yang menjadi bagian penting pada aplikasi website manajemen bisnis penjualan ban pada PT. Goldfinger Wheels Indonesia ini. Segmen program yang diberikan akan disertai dengan penjelasan dari cara kerja segmen program tersebut, setiap segmen program memiliki fungsi dan peran yang penting dalam keberlangsungan fitur yang ada pada aplikasi website manajemen bisnis penjualan ban ini. Setiap pembaca diharapkan dapat mengerti cara kerja dari setiap segmen program yang akan dijelaskan dibawah ini.

6.1 Koneksi Database

Pada bagian ini akan dijelaskan mengenai peran penting dari koneksi database. Hal ini diperlukan agar website manajemen bisnis penjualan ban pada PT. Goldfinger Wheels Indonesia dapat bekerja dengan baik karena dapat mengakses database yang ada. Database yang digunakan pada program website manajemen bisnis penjualan ban ini adalah MySQL, pembuatan koneksi dilakukan sekali saja dengan mengakses file ".env" yang terdapat pada folder project program ini.

Segmen Program 6.1 Koneksi Database

```
01:       DB_CONNECTION=mysql
02:       DB_HOST=127.0.0.1
03:       DB_PORT=3306
04:       DB_DATABASE=nanaspos_gwi
05:       DB_USERNAME=nanaspos_gwi
06:       DB_PASSWORD=calvinadhikang02
```

Segmen program 6.1 di atas adalah cara untuk melakukan koneksi pada database tugas akhir ini. Saat program website manajemen bisnis penjualan ban pada PT. Goldfinger Wheels Indonesia dijalankan, maka program akan menghubungkan dengan database yang bersifat public sehingga dapat saling berhubungan dengan database pada masing-masing halaman. Seluruh baris akan

terbentuk pada saat membuat project baru pada laravel hanya saja pada baris 4 akan dilakukan perubahan kode yaitu mengganti nama database sesuai nama database yang dibuat dalam kasus ini nama databasenya adalah nanaspos_gwi, perubahan ini terjadi di awal saat proses pembuatan program website manajemen bisnis penjualan ban ini.

6.2 Pengecekan Login

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk melakukan pengecekan login dari user yang telah terdaftar pada website sistem informasi peternakan ikan ini. Segmen program ini bertujuan untuk melakukan pemeriksaan pengguna yang akan masuk kedalam program website manajemen bisnis penjualan ban ini sesuai dengan role masing-masing dan yang terpenting adalah juga merupakan staff dari PT. Goldfinger Wheels Indonesia. Sedangkan apabila belum terdaftar maka user akan diarahkan ke halaman login kembali dengan pesan yang ada.

Segmen Program 6.2 Pengecekan Login

```
01:         $username = $request->input('username');
02:          $password = $request->input('password');
03:
04:          $users                                                  =
   Karyawan::where('username','=',$username)->get();
05:          if (count($users) > 0) {
06:              //cari user yang sesuai dengan bcrypt
07:              foreach ($users as $key => $value) {
08:                  if                    (Hash::check($password,
   $value->password)) {
09:                      if ($value->status == "Aktif") {
10:                          Session::put('user', $value);
11:                          return redirect('/dashboard');
12:                      }else{
13:                          toast("User tidak aktif", "error");
14:                          return back();
15:                      }
16:                  }
17:              }
18:
19:              toast('Password Salah !', 'error');
20:              return back();
21:          } else {
22:              toast("Username tidak ditemukan", "error");
23:              return back();
```

```
24:         }
```

Segmen program 6.2 di atas adalah cara untuk melakukan pengecekan login pada website manajemen bisnis penjualan ban ini. Pada baris 1 akan meminta data yang berasal dari inputan user, data yang diminta adalah username user dan password user. Setelah itu program akan mencari data user yang diberikan dan mencarinya di model karyawan, data yang diambil adalah username dan password, jika hasil data yang ditemukan tidak lebih dari 0 maka akan muncul *alert* bahwa username atau password salah.

Jika validasi data berhasil maka akan masuk ke baris 8 yaitu pengecekan password yang di inputkan dengan password yang disimpan di database, bila password sama maka akan lanjut ke baris 9 dimana dilakukan pengecekan status karyawan, jika status karyawan tidak aktif maka program akan kembali membawa pesan yaitu user ini tidak aktif, user tidak aktif dikarenakan dipecat, atau berhenti bekerja maka dari itu user tidak dapat masuk kedalam program. Jika status user aktif maka program akan lanjut ke baris berikutnya.

Pada baris 10, program akan menyimpan data pengguna yang berhasil login, data yang disimpan seperti role, nama, password, dan username. Kemudian, program akan mengarahkan user ke halaman dashboard website, jika autentikasi gagal maka pada baris 22 dan 23 akan menampilkan pesan gagal login dan user dikembalikan ke halaman login.

6.3 Menampilkan Data Master

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk menampilkan data master. Segmen program ini bertujuan untuk mencari data untuk ditampilkan pada halaman menggunakan datatable. Sebagai contoh, akan digunakan data master karyawan.

Segmen Program 6.3 Menampilkan Data Master

```
01:             $data = Karyawan::latest()->get();
02:             return view('master.karyawan.view',[
03:                 'data' => $data
04:             ]);
```

Segmen program 6.3 di atas adalah cara program mencari dan menampilkan data karyawan. Pertama, seluruh data karyawan yang terdapat pada model karyawan akan diambil menggunakan perintah *Karyawan::latest()->get()*. Setelah itu, program akan mengembalikan seluruh data karyawan, diurutkan dari yang paling baru dan mengembalikan tampilan halaman atau view dengan data karyawan yang terdapat pada variable *$data* yang telah didapatkan. Fungsi ini sama dengan fungsi master lainnya untuk menampilkan data master pada website ini. Seperti data barang, kategori, dan customer.

6.4 Menambahkan Data Master

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk menambahkan data master. Segmen program ini bertujuan untuk menambah data dan menyimpannya ke database. Sebagai contoh, akan digunakan data master karyawan.

Segmen Program 6.4 Menambahkan Data Master

```
01:      $data          =          Karyawan::where('username',
   $request->input('username'))->get();
02:              if (count($data) > 0) {
03:                  toast('Username     sudah     digunakan',
   'error');
04:                  return redirect()->back();
05:              }
06:
07:              DB::beginTransaction();
08:              try {
09:                  $lastId                                  =
   DB::table('karyawan')->insertGetId([
10:                      'nama' => $request->input('nama'),
11:                      'username'                           =>
   $request->input('username'),
12:                      'password'                           =>
   Hash::make($request->input('password')),
13:                      'telp' => $request->input('telp'),
14:                      'role' => $request->input('role'),
15:                      'status' => 1,
16:                      'created_at' => Carbon::now()
17:                  ]);
18:
19:                  if       ($request->input('role')       ==
   'Stakeholder') {
20:                      DB::table('shares')->insert([
21:                          'karyawan_id' => $lastId,
```

```
22:                         'shares' => 0,
23:                         'created_at' => Carbon::now()
24:                     ]);
25:                 }
26:
27:                 DB::commit();
28:                 toast("Berhasil    Menambah    Karyawan",
    "success");
29:                 return
    redirect('/karyawan/detail/'.$lastId);
30:             } catch (\Exception $ex) {
31:                 dd($ex->getMessage());
32:                 DB::rollBack();
33:                 return redirect('/karyawan');
34:             }
35:
```

Segmen program 6.4 di atas merupakan cara untuk menambahkan data master. Pertama program akan mengambil inputan user berupa username dan melakukan pengecekan di database apakah ada karyawan yang sudah terdaftar dengan username yang di inputkan user, bila ada maka program akan membatalkan proses penambahan dan meminta user untuk memberikan karyawan username yang unik melalui pesan error.

Selanjutnya jika telah melewati proses verifikasi username kembar maka akan dilakukan penambahan data karyawan pada baris 9. Data yang disimpan seperti nama, username, password, nomor telepon dan role. Bila data yang dimasukan merupakan karyawan dengan role *stakeholder* maka program akan melakukan penambahan data saham dengan nilai awal 0 persen kepemilikan saham. Hasil proses penambahan data berhasil atau tidak akan ditampilkan menggunakan pesan. Bila berhasil, user akan diarahkan ke halaman detail karyawan yang barusan ditambahkan, bila gagal maka akan dikembalikan ke halaman master karyawan. Fungsi ini sama dengan fungsi master lainnya untuk menambah data master pada website ini. seperti data barang, kategori, dan customer.

6.5 Mengubah Data Master

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk mengubah data master. Segmen program ini bertujuan untuk mengubah data dan menyimpannya ke database. Sebagai contoh, akan digunakan data master karyawan.

Segmen Program 6.5 Mengubah Data Master

```
01:  $karyawan = Karyawan::find($id);
02:
03:          if ($karyawan == null) {
04:              toast("Karyawan Tidak Ditemukan", "error");
05:              return redirect('karyawan');
06:          }
07:
08:          $status   =   $request->input('status')   ==   "on"   ?
     "Aktif" : "Non-Aktif";
09:
10:          $karyawan->nama = $request->input('nama');
11:          $karyawan->username = $request->input('username');
12:          $karyawan->telp = $request->input('telp');
13:          $karyawan->role = $request->input('role');
14:          $karyawan->status = $status;
15:          $karyawan->save();
16:
17:          toast("Berhasil Update Karyawan", "success");
18:          return redirect()->back();
19:
```

Segmen program 6.5 di atas merupakan cara untuk mengubah data master karyawan pada website manajemen bisnis penjualan ban ini. Pertama program akan menerima seluruh inputan user dari variabel *$request*, lalu program akan mengecek apakah data karyawan yang ingin diubah ada pada database, jika tidak maka program akan berhenti, menampilkan pesan error, dan mengarahkan user ke halaman master karyawan. Jika data karyawan ada, maka data karyawan akan diubah sesuai dengan inputan dari user seperti pada bari 8 hingga 15. Setelah itu, website akan menampilkan pesan berhasil, dan user akan dikembalikan ke halaman sebelumnya. Fungsi ini sama dengan fungsi master lainnya untuk mengubah data master pada website ini. Seperti data barang, kategori, dan customer.

6.6 Menghapus Data Master

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk menghapus data master. Segmen program ini bertujuan untuk menghapus data dan menyimpannya ke database. Sebagai contoh, akan digunakan data master karyawan.

Segmen Program 6.6 Menghapus Data Master

```
01:      $karyawan = Karyawan::find($id);
02:              if ($karyawan == null) {
03:                  toast("Karyawan     Tidak     Ditemukan",
   "error");
04:                  return redirect('karyawan');
05:              }
06:
07:              $karyawan->delete();
08:              toast('Berhasil     Menghapus     Karyawan',
   'success');
09:              return redirect('karyawan');
10:
```

Segmen program 6.6 di atas merupakan cara untuk menghapus data master karyawan yang dipilih pada website manajemen bisnis penjualan ban ini. Pertama program akan mengambil *request* id yang dikirim melalui *route* lalu program akan melakukan penghapusan data berdasarkan id karyawan yang dikirim, jika *delete* berhasil maka program akan memberi pesan berhasil menghapus dan mengembalikan user ke halaman master. Fungsi ini sama dengan fungsi master lainnya untuk menghapus data master pada website ini. Seperti data barang, kategori, dan customer.

6.7 Melihat Transaksi Pembelian

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk melihat transaksi pembelian ban. Segmen program ini bertujuan untuk menampilkan data pembelian perusahaan. Sebagai contoh, akan digunakan potongan program dari daftar transaksi pembelian.

Segmen Program 6.7 Melihat Transaksi Pembelian

```
01:      $type = $request->query('type', 'all');
02:              if ($type == "deleted") {
03:                  $data                                    =
   HeaderPurchase::onlyTrashed()->get();
04:              }else{
05:                  $data = HeaderPurchase::latest()->get();
06:              }
07:
08:              return view('master.po.view', [
09:                  'data' => $data,
10:                  'type' => $type
11:              ]);
```

Segmen program 6.7 diatas merupakan cara untuk menampilkan data transaksi pembelian ban kepada vendor. Pertama, program akan mengambil data query yang di inputkan user dari variabel *request*. Dari data query program akan menentukan apakah akan menampilkan data transaksi pembelian yang dihapus atau tidak. Data transaksi akan ditampilkan dari yang paling baru seperti pada baris 5. Selanjutnya program akan mengirimkan data transaksi pembelian ke halaman transaksi pembelian.

6.8 Membuat Transaksi Pembelian

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk membuat transaksi pembelian. Segmen program ini bertujuan untuk membuat transaksi pembelian perusahaan kepada pihak vendor. Sebagai contoh, akan digunakan potongan program dari pembuatan transaksi pembelian yang akan dibagi menjadi 3 langkah.

6.8.1 Membuat Transaksi Pembelian Step 1

Pada bagian akan dijelaskan tentang potongan program mengenai langkah pertama ketika admin ingin membuat transaksi pembelian ban kepada vendor. Pada tahap ini admin perusahaan akan melakukan pemilihan terhadap produk ban yang akan dibeli beserta jumlahnya. Berikut potongan programnya.

Segmen Program 6.8.1 Membuat Transaksi Pembelian Step 1

```
12:      $qty = $request->input('qty');
13:              $part = $request->input('part');
14:
15:              $list = [];
16:              for ($i=0; $i < count($qty); $i++) {
17:                  if ($qty[$i] > 0) {
18:                      $obj = Barang::find($part[$i]);
19:                      $obj->qty = $qty[$i];
20:
21:                      $list[] = $obj;
22:                  }
23:              }
24:
25:              if(count($list) <= 0){
```

```
26:                     return back()->withErrors([
27:                         'msg' => 'Minimal Beli 1 Barang'
28:                     ]);
29:                 }
30:
31:                 $oldPO = Session::get('po_cart');
32:                 // Object Creation for Purchase
33:                 $PO = new stdClass();
34:                 if ($oldPO != null) {
35:                     $PO->vendor = $oldPO->vendor;
36:                     $PO->grandTotal = $oldPO->grandTotal;
37:                 }else{
38:                     $PO->vendor = null;
39:                     $PO->grandTotal = null;
40:                 }
41:                 $PO->PPN                                        =
   DB::table('settings')->select('ppn')->first()->ppn;
42:                 $PO->PPN_value = 0;
43:                 $PO->list = $list;
44:
45:                 Session::put('po_cart', $PO);
46:                 return redirect('/po/vendor');
```

Segmen program 6.8.1 diatas merupakan cara untuk memasukan barang yang akan di beli oleh admin beserta jumlahnya. Program akan menerima data inputan user berupa kode part number ban dan jumlahnya, selanjutnya program akan mengambil data barang pada database sesuai dengan part number yang dipilih admin. Tujuannya adalah untuk menerima informasi lebih detail mengenai barang yang akan dibeli. Selanjutnya pada baris 14 akan dilakukan pengecekan apakah jumlah barang yang dibeli tidak kosong, bila kosong maka website akan berhenti dan mengembalikan user dengan pesan error. Pada baris 20 website akan mengambil data session po_cart, bila tidak ada data session maka akan dibuat data session yang baru, namun bila sudah ada maka akan data session yang lama akan digunakan. Selanjutnya akan diambil data nilai persentase PPN paling baru pada website seperti pada bari 30. Selanjutnya data Session akan disimpan dan user diarahkan ke step / halaman berikutnya.

6.8.2 Membuat Transaksi Pembelian Step 2

Pada bagian akan dijelaskan tentang potongan program mengenai langkah

kedua ketika admin ingin membuat transaksi pembelian. Pada tahap ini admin perusahaan akan melakukan pemilihan terhadap vendor yang menjual produk yang sudah dipilih admin pada step sebelumnya. Berikut potongan programnya.

Segmen Program 6.8.2 Membuat Transaksi Pembelian Step 2

```
01:      $oldPO = Session::get('po_cart');
02:
03:              $vendorId = $request->input('vendor');
04:              $total = $request->input('total');
05:
06:              $oldPO->total = $total;
07:              $oldPO->vendor = Vendor::find($vendorId);
08:
09:              //set po->list subtotal
10:              foreach ($oldPO->list as $key => $value) {
11:                  $objBarang                                  =
   BarangVendor::where('vendor_id',                             '=',
   $vendorId)->where('barang_id', '=', $value->part)->get();
12:                  $value->harga = $objBarang[0]->harga;
13:                  $value->subtotal    =    $value->harga    *
   $value->qty;
14:              }
15:
16:              //set PPN from Setting
17:              $oldPO->PPN_value  =  ($oldPO->total  /  100  *
   $oldPO->PPN);
18:              $oldPO->grandTotal   =   $oldPO->PPN_value   +
   $oldPO->total;
19:
20:              Session::put('po_cart', $oldPO);
21:              return redirect('/po/confirmation');
22:
```

Segmen program 6.8.2 diatas merupakan cara admin memilih vendor. Pertama program akan mengambil data Session, data vendor yang dipilih dan total harga dari vendor melalui variabel *request*. Selanjutnya program akan menambah data pada Session, yaitu informasi harga barang dari vendor ke dalam tiap barang yang mau dibeli admin serta menghitung harga subtotalnya, harga ppn tiap barang, dan grand total dari pembelian seperti pada bari 10 hingga 18. Selanjutnya data Session yang sudah diubah akan disimpan dan user akan diarahkan ke step / halaman berikutnya.

6.8.3 Membuat Transaksi Pembelian Step 3

Pada bagian akan dijelaskan tentang potongan program mengenai langkah ketiga dan terakhir ketika admin ingin membuat transaksi pembelian. Pada tahap ini admin perusahaan akan melakukan konfirmasi terhadap data pembelian yang akan dibuat, dan input data seperti tanggal jatuh tempo, dan mengganti nilai PPN bila perlu. Selanjutnya admin akan menekan tombol buat Purchase Order untuk membuat pembelian. Berikut potongan programnya.

Segmen Program 6.8.3 Membuat Transaksi Pembelian Step 3

```
01:      $po = Session::get('po_cart');
02:              $user = Session::get('user');
03:              $jatuhTempo                                    =
   Carbon::parse($request->input('jatuhTempo'));
04:
05:              //cek apakah transaksi lama
06:              $timePembayaran                                =
   $request->input('timeValuePembayaran') ?? null;
07:              $timeCreation                                  =
   $request->input('timePembuatan') ?? Carbon::now();
08:              $timePenerimaan                                =
   $request->input('timeValuePenerimaan') ?? null;
09:
10:              if ($timePembayaran == null && $timePenerimaan
   == null) {
11:                  if ($jatuhTempo->isPast()) {
12:                      toast('Tanggal Jatuh Tempo minimal hari
   ini', 'error');
13:                      return redirect()->back();
14:                  }
15:              }
16:
17:              DB::beginTransaction();
18:              try {
19:                  //insert header
20:                  $lastId                                    =
   DB::table('hpurchase')->insertGetId([
21:                      'vendor_id' => $po->vendor->id,
22:                      'kode'                                 =>
   Util::generatePurchaseCode(),
23:                      'karyawan_id' => $user->id,
24:                      'total' => $po->total,
25:                      'grand_total' => $po->grandTotal,
26:                      'ppn_value' => $po->PPN_value,
27:                      'ppn' => $po->PPN,
28:                      'jatuh_tempo' => $jatuhTempo,
29:                      'created_at' => $timeCreation,
30:                      'paid_at' => $timePembayaran,
31:                      'paid_by' => $timePembayaran != null ?
   Session::get('user')->id : null,
32:                      'recieved_at' => $timePenerimaan,
```

```
33:                     'recieved_by'  =>  $timePenerimaan  !=
   null ? Session::get('user')->id : null
34:                 ]);
35:
36:                 foreach ($po->list as $key => $value) {
37:                     DB::table('dpurchase')->insert([
38:                         'hpurchase_id' => $lastId,
39:                         'part' => $value->part,
40:                         'nama' => $value->nama,
41:                         'harga' => $value->harga,
42:                         'qty' => $value->qty,
43:                         'subtotal' => $value->subtotal,
44:                     ]);
45:                 }
46:
47:                 Session::remove('po_cart');
48:                 toast('Berhasil  Membuat  Purchase  Order',
   'success');
49:                 DB::commit();
50:                 return redirect('/po');
51:             } catch (\Exception $ex) {
52:                 toast($ex->getMessage(), 'error');
53:                 DB::rollBack();
54:                 return back();
55:             }
```

Segmen program 6.8.3 diatas merupakan cara admin membuat transaksi pembelian atau purchase order. Pertama program akan mengambil data Session yang berisi informasi pembelian, data user atau admin yang sedang login, dan data dari inputan user seperti tanggal jatuh tempo, apakah termasuk pembelian lama melalui variabel *request*. Bila pembelian merupakan pembelian lama, maka program akan mengambil inputan user seperti tanggal pembayaran, metode pembayaran. Bila pembelian merupakan pembelian saat ini, maka program akan mengecek bahwa jatuh tempo tidak boleh kurang dari hari ini seperti pada bari 11 hingga 14. Selanjutnya program akan memulai pembuatan transaksi, dimulai dari memasukan data header transaksi pembelian seperti, siapa vendornya, berapa biayanya, ppn, tanggal jatuh tempo, dan membuat kode purchase order seperti pada baris 20 hingga 34. Selanjutnya program akan memasukan data detail transaksi ke database seperti part number barang, harga beli, jumlah pembelian, dan subtotal. Bila semua proses pemasukan data ke database berhasil, maka data Session akan dihapus, user akan diarahkan ke halaman pembelian, dan diberikan pesan sukses. Bila ada salah satu proses pemasukan data tidak berhasil, maka semua data akan

dibatalkan, user akan diberikan pesan error dan dikembalikan ke halaman sebelumnya.

6.9 Melihat Transaksi Penjualan

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk melihat transaksi penjualan ban. Segmen program ini bertujuan untuk menampilkan data penjualan perusahaan. Sebagai contoh, akan digunakan potongan program dari daftar transaksi penjualan.

Segmen Program 6.9 Melihat Transaksi Penjualan

```
01:      $type = $request->query('type', 'all');
02:              if ($type == 'all') {
03:                  $data                                          =
   HeaderInvoice::latest()->whereIn('status', [1, 2])->get();
04:              }else if($type == 'deleted') {
05:                  $data                                          =
   HeaderInvoice::onlyTrashed()->get();
06:              }else if($type == 'unconfirmed'){
07:                  $data                                          =
   HeaderInvoice::latest()->where('status', 0)->get();
08:              }else if($type == 'canceled'){
09:                  $data                                          =
   HeaderInvoice::latest()->where('status', -1)->get();
10:              }
11:
12:              $countNotConfirm                                   =
   HeaderInvoice::where('status', 0)->count();
13:
14:              return view('master.invoice.view', [
15:                  'type' => $type,
16:                  'data' => $data,
17:                  'countNotConfirm' => $countNotConfirm
18:              ]);
19:
```

Segmen program 6.9 diatas merupakan cara untuk menampilkan data transaksi penjualan ban kepada customer. Pertama, program akan mengambil data query yang di inputkan user dari variabel *request*. Dari data query program akan menentukan apakah akan menampilkan data transaksi pembelian yang dihapus, belum terkonfirmasi, dibatalkan atau penjualan yang aktif. Data transaksi akan ditampilkan dari yang paling baru seperti pada baris 5. Selanjutnya program akan mengirimkan data transaksi pembelian ke halaman transaksi pembelian.

6.10 Membuat Transaksi Penjualan

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk membuat transaksi penjualan ban. Segmen program ini bertujuan untuk membuat transaksi penjualan perusahaan kepada pihak customer. Sebagai contoh, akan digunakan potongan program dari pembuatan transaksi penjualan yang akan dibagi menjadi 3 langkah.

6.10.1 Membuat Transaksi Penjualan Step 1

Pada bagian akan dijelaskan tentang potongan program mengenai langkah pertama ketika admin ingin membuat transaksi penjualan. Pada tahap ini admin perusahaan akan melakukan pencatatan terhadap customer yang terlibat transaksi penjualan di perusahaan. Berikut potongan programnya.

Segmen Program 6.10.1 Membuat Transaksi Pembelian Step 1

```
01:             $customer                                       =
   Customer::find($request->input('id'));
02:             $invoice = Session::get('invoice_cart');
03:
04:             $invoice->customer = $customer;
05:
06:             toast('Berhasil memilih customer', 'success');
07:             return redirect()->back();
08:
```

Segmen program 6.10.1 di atas merupakan cara admin untuk memilih customer yang akan terlibat dalam transaksi penjualan perusahaan. Program akan mengambil data customer sesuai dengan id yang di dapat dari inputan user pada variabel request. Selanjutnya program akan menyimpan data customer pada Session invoice_cart, program akan menampilkan pesan sukses dan user akan diarahkan ke halaman berikutnya.

6.10.2 Membuat Transaksi Penjualan Step 2

Pada bagian akan dijelaskan tentang potongan program mengenai langkah

kedua ketika admin ingin membuat transaksi penjualan. Pada tahap ini admin perusahaan akan melakukan pencatatan terhadap data barang yang akan dijual seperti kode barang, jumlah penjualan, dan mengganti harga barang bila diperlukan. Berikut potongan programnya.

Segmen Program 6.10.2 Membuat Transaksi Pembelian Step 2

```
01:             $barangs                                        =
   json_decode($request->input('barang') ?? "[]");
02:
03:             $ppn_include = $request->input('ppn-include');
04:             $invoice = Session::get('invoice_cart');
05:
06:             $total = 0;
07:             $cleanTotal = 0; //total harga bersih, untuk
   perhitungan harga non ppn
08:             $list_barang = []; //simpan data barang lengkap
09:             $total_list_barang = 0;
10:
11:             foreach ($barangs as $key => $barang) {
12:                 $obj = Barang::find($barang->part);
13:                 $obj->qty = $barang->qty;
14:                 $price                                      =
   Util::parseNumericValue($barang->harga);
15:                 $obj->clean_price = $price;
16:
17:                 if ($ppn_include != null) {
18:                     //kalau harga termasuk PPN
19:                     $ppnItem = $price - ($price / 100 *
   $invoice->PPN);
20:                     $obj->harga = $ppnItem;
21:                     $cleanTotal += $price * $barang->qty;
22:                 }else{
23:                     $obj->harga = $price;
24:                 }
25:
26:                 $subtotal = $obj->harga * $barang->qty;
27:                 $obj->subtotal = $subtotal;
28:
29:                 $list_barang[] = $obj;
30:                 $total_list_barang += $subtotal;
31:                 $total += $subtotal;
32:             }
33:
34:             if (count($barangs) <= 0) {
35:                 return redirect()->back()->withErrors([
36:                     'msg' => 'Minimal membeli 1 barang /
   paket !'
37:                 ]);
38:             }
39:
40:             $invoice->list_barang = $list_barang;
```

```
41:             $invoice->total_list_barang                 =
   $total_list_barang;
42:             $invoice->total = $total;
43:
44:             if ($ppn_include != null){
45:                 //kalau harga termasuk ppn
46:                 $invoice->PPN_value = ($cleanTotal / 100)
   * $invoice->PPN;
47:                 $invoice->PPN_included = true;
48:             }else{
49:                 $invoice->PPN_value  =  ($total  /  100)  *
   $invoice->PPN;
50:                 $invoice->PPN_included = false;
51:             }
52:             $invoice->grandTotal      =      $total      +
   $invoice->PPN_value;
53:
54:             Session::put('invoice_cart', $invoice);
55:             return redirect('/invoice/confirmation');
```

Segmen program 6.10.2 di atas merupakan cara admin untuk memilih menambahkan barang yang dibeli custome, program akan mencari barang yang telah diinputkan admin, lalu melakukan kalkulasi perhitungan pajak, dan terdapat pengecekan apabila tidak ada barang yang dipilih maka program akan berhenti dan menampilkan pesan error. Bila proses kalkulasi sudah selesai maka program akan menyimpan data pesanan, lalu akan menampilkan pesan sukses dan user akan diarahkan ke halaman berikutnya.

6.10.3 Membuat Transaksi Penjualan Step 3

Pada bagian akan dijelaskan tentang potongan program mengenai langkah kedua ketika admin ingin membuat transaksi penjualan. Pada tahap ini admin perusahaan akan melakukan konfirmasi terhadap data transaksi penjualan yang akan dibuat. Bila data sudah benar maka program akan membuat transaksi penjualan. Berikut potongan programnya.

Segmen Program 6.10.3 Membuat Transaksi Pembelian Step 3

```
01:             $komisiJumlah = 0;
02:             $komisiPenerima = "-";
03:             $karyawan = Session::get('user');
04:             $jatuhTempo = $request->input('jatuhTempo');
05:             $invoice = Session::get('invoice_cart');
06:
07:             //cek apakah transaksi lama
08:             $timePembayaran                             =
   $request->input('timeValuePembayaran');
```

```
09:             $timeCreation                                   =
   $request->input('timePembuatan') ?? Carbon::now();
10:
11:             if ($timePembayaran == null) {
12:                 if  (Carbon::parse($jatuhTempo)->isPast())
   {
13:                     toast('Tanggal  Jatuh  Tempo  Minimal
   Besok', 'error');
14:                     return back()->withErrors([
15:                         'msg'  =>  'Tanggal  Jatuh  Tempo
   Minimal Besok !'
16:                     ]);
17:                 }
18:             }
19:
20:             $komisiStatus = $request->input('komisi');
21:             if ($komisiStatus) {
22:                 $komisiJumlah                               =
   Util::parseNumericValue($request->input('komisiJumlah'));
23:                 $komisiPenerima                             =
   $request->input('komisiPenerima');
24:             }
25:
26:             DB::beginTransaction();
27:             try {
28:                 $kode = Util::generateInvoiceCode();
29:                 $suratJalan                                 =
   Util::generateSuratJalanCodeFromInvoiceCode($kode);
30:
31:                 $currentDateTime = Carbon::now();
32:                 //insert header
33:                 $lastId                                     =
   DB::table('hinvoice')->insertGetId([
34:                     'customer_id'                          =>
   $invoice->customer->id,
35:                     'karyawan_id' => $karyawan->id,
36:                     'kode' => $kode,
37:                     'surat_jalan' => $suratJalan,
38:                     'total' => $invoice->total,
39:                     'contact_person' => $komisiPenerima,
40:                     'komisi' => $komisiJumlah,
41:                     'ppn' => $invoice->PPN,
42:                     'ppn_value' => $invoice->PPN_value,
43:                     'grand_total' => $invoice->grandTotal,
44:                     'po' => $request->input('po'),
45:                     'jatuh_tempo' => $jatuhTempo,
46:                     'created_at' => $timeCreation,
47:                     'paid_at' => $timePembayaran,
48:                     'paid_by' => $timePembayaran != null ?
   Session::get('user')->id : null,
49:                     'confirmed_at' => $timeCreation,
50:                     'confirmed_by' => $karyawan->id,
51:                     'status'=> 1,
52:                 ]);
53:
```

```
54:                     foreach ($invoice->list_barang as $key =>
   $value) {
55:                         DB::table('dinvoice')->insert([
56:                             'hinvoice_id' => $lastId,
57:                             'part' => $value->part,
58:                             'nama' => $value->nama,
59:                             'harga' => $value->harga,
60:                             'qty' => $value->qty,
61:                             'subtotal' => $value->subtotal,
62:                             'type' => 'barang',
63:                             'created_at' => $currentDateTime
64:                         ]);
65:                     }
66:
67:                     foreach ($invoice->list_paket as $key =>
   $value) {
68:                         DB::table('dinvoice')->insert([
69:                             'hinvoice_id' => $lastId,
70:                             'part' => $value->id,
71:                             'nama' => $value->nama,
72:                             'harga' => $value->harga,
73:                             'qty' => $value->qty,
74:                             'subtotal' => $value->subtotal,
75:                             'type' => 'paket',
76:                             'created_at' => $currentDateTime
77:                         ]);
78:                     }
79:
80:                     Session::remove('invoice_cart');
81:                     toast("Transaksi                 Customer:
   ".$invoice->customer->nama.", Berhasil dibuat", 'success');
82:                     DB::commit();
83:                     return redirect('/invoice');
84:                 } catch (\Exception $ex) {
85:                     DB::rollBack();
86:                     return back()->withErrors([
87:                         'msg' => $ex->getMessage()
88:                     ]);
89:                 }
90:
```

Segmen program 6.10.3 di atas merupakan cara admin untuk membuat transaksi penjualan dan disimpan pada sistem. Pertama program akan mengecek apakah data penjualan yang dimasukan admin merupakan data penjualan yang sudah terjadi. Ini bertujuan untuk membedakan transaksi penjualan yang masih aktif atau tidak. Selanjutnya proses pemasukan data ke database akan dilakukan dimulai dari data header transaksi. Data header transaksi berupa karyawan yang bertanggung jawab atas transaksi, data customer, kode invoice, total harga, status transaksi, jatuh tempo, dan lainnya. Bila proses pemasukan data gagal, maka

aplikasi akan membatalkan proses dan mengembalikan user ke halaman sebelumnya. Bila prose pemasukan berhasil maka data akan disimpan di database dan program akan menampilkan pesan sukses dan user akan diarahkan ke halaman detail invoice / detail transaksi..

6.11 Membuat Penawaran

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan customer untuk membuat penawaran pada website. Segmen program ini bertujuan untuk membuat dan menyimpan data penawaran customer. Sebagai contoh akan digunakan potongan program dari pembuatan penawaran customer.

Segmen Program 6.11 Membuat Penawaran

```
01:             $user = Session::get('user');
02:             $penawaran = Session::get('penawaran');
03:             $total = $request->input('total');
04:             $grand_total = $request->input('grand_total');
05:             $ppn = $request->input('ppn');
06:             $total_ppn = $request->input('total_ppn');
07:
08:             DB::beginTransaction();
09:             try {
10:                 // Insert Header Penawaran
11:                 $lastId                                    =
   DB::table('hpenawaran')->insertGetId([
12:                     'customer_id' => $user->id,
13:                     'total' => $total,
14:                     'ppn' => $ppn,
15:                     'ppn_value' => $total_ppn,
16:                     'grand_total' => $grand_total,
17:                     'created_at' => Carbon::now(),
18:                 ]);
19:
20:                 foreach ($penawaran as $key => $value) {
21:                     $barang    =    Barang::where('part',
   $value->part)->first();
22:
23:                     DB::table('dpenawaran')->insert([
24:                         'hpenawaran_id' => $lastId,
25:                         'part' => $value->part,
26:                         'qty' => $value->qty,
27:                         'harga_penawaran'                =>
   $value->subtotal / $value->qty,
28:                         'subtotal' => $value->subtotal,
29:                         'created_at' => Carbon::now(),
30:                     ]);
31:                 }
```

```
32:
33:                     Session::forget('penawaran');
34:                     DB::commit();
35:                     return redirect('/penawaran/view')->with([
36:                         'title'    =>    'Berhasil    membuat
   penawaran!',
37:                         'msg' => 'Penawaran berhasil dibuat!'
38:                     ]);
39:                 } catch (\Exception $ex) {
40:                     DB::rollBack();
41:                     return
   redirect('/penawaran')->withErrors([
42:                         'msg'  =>  'Gagal  membuat  penawaran,
   silahkan coba lagi!'
43:                     ]);
44:                 }
45:
```

Segmen program 6.11 di atas merupakan cara untuk menyimpan dan membuat penawaran yang dibuat oleh customer pada website, pada baris 1 sampai baris 6 dijelaskan mengenai bagaimana cara program mengambil data dari inputan user pada website melalui variabel request. Setelah mengambil data, program akan melakukan proses pembuatan dan penyimpanan penawaran customer, dimulai dari proses memasukan data header penawaran seperti data customer, total biaya, nilai ppn, dan tanggal pembuatan. Lalu dilanjutkan dengan memasukan data detail penawaran seperti kode barang yang dibeli, harga penawarannya berapa, jumlah penawaran, dan harga subtotal. Bila berhasil maka data akan disimpan pada database, dan user akan diarahkan ke halaman lihat penawaran. Bila gagal, proses akan dihentikan dan user diarahkan ke halaman sebelumnya. Baik gagal maupun berhasil user akan menerima pesan dari website.

6.12 Melihat Laporan Laba Bersih

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk melihat laporan laba bersih menggunakan FIFO. Segmen program ini bertujuan untuk melihat laporan bersih perusahaan menggunakan FIFO. Sebagai contoh akan digunakan potongan program dari pembuatan laporan laba bersih pada website.

Segmen Program 6.12 Melihat Laporan Laba Bersih

```
01:             $mulai = $request->input('mulai', null);
02:             $akhir = $request->input('akhir', null);
03:
04:             $data    =    DFIFO::whereBetween('created_at',
   [$mulai, $akhir])->get();
05:
06:             $totalBersih = 0;
07:             foreach ($data as $key => $value) {
08:                 $totalBersih += $value->profit_total;
09:                 $value->invoice                          =
   HeaderInvoice::find($value->hinvoice_id);
10:             }
11:
12:             return view('laporan.laba_bersih', [
13:                 'mulai' => $mulai,
14:                 'akhir' => $akhir,
15:                 'data' => $data,
16:             ]);
```

Segmen program 6.12 di atas merupakan potongan program untuk melihat data laporan laba bersih. Pertama program akan menerima inputan user berupa tanggal awal data dan tanggal akhir data dari variabel request. Selanjutnya program akan mengambil data tabel DFIFO yang menyimpan data FIFO penjualan seperti berapa jumlah yang keluar, kode barangnya, kode transaksinya, berapa profit total dan profit per barang yang dijual. Setelah itu program akan mengembalikan tampilan laporan kepada user melalui variabel data.

6.13 Membuat Dokumen

Pada bagian ini akan dijelaskan tentang potongan program yang digunakan untuk membuat dokumen. Segmen program ini bertujuan untuk melihat bagaimana cara program membuat dokumen. Sebagai contoh akan digunakan potongan program dari pembuatan dokumen invoice.

Segmen Program 6.13 Membuat Dokumen

```
01:             $data = HeaderInvoice::find($id);
02:
03:             $pdf = Pdf::loadView('template.pdf.invoice', [
04:                 'data' => $data
05:             ]);
06:             return
   $pdf->download('invoice_'.$data->kode.'.pdf');
```

```
07:
```

Segmen program 6.13 di atas merupakan potongan program untuk melihat membuat dokumen invoice. Pertama program akan mengambil data id invoice yang telah diinputkan user dari variabel request. Selanjutnya program akan mengambil data invoice berdasarkan id dari inputan user. Berikutnya program akan menggunakan library Pdf untuk membuat file dokumen berdasarkan file tampilan yang diberikan, dalam kasus ini file yang diberikan adalah file invoice. Setelah itu program memberikan file dokumen dan user bisa menerima file tersebut dengan cara mendownloadnya.