BAB III

PELAKSANAAN KERJA MAGANG

3.1. Kedudukan dan Posisi

Pada proyek ini, penulis melaksanakan kerja magang di Sekretariat Jenderal Dewan Perwakilan Rakyat Republik Indonesia yang berlokasi di Gedung Nusantara I Komplek DPR/MPR RI, Jalan Jenderal Gatot Subroto Senayan, Jakarta selama lebih dari 2 bulan dengan 42 hari masuk kerja. Penulis ditempatkan dibagian Bidang Data dan Sarana Informasi (BDSI) yang berada di bawah Kepala Pusat Pengkajian, Pengolahan Data dan Informasi. Sesuai dengan namanya, BDSI memegang kendali atas data dan informasi yang berada di Sekretariat Jenderal DPR RI dan melayani serta memberikan bantuan dengan menjadi *IT support* untuk membantu pegawai-pegawai Setjen DPR RI dalam hal-hal yang berkaitan dengan sistem, data dan teknologi informasi.

Penulis bekerja didalam sebuah tim yang terdiri dari 3 orang yang merupakan rekan mahasiswa dari Universitas Multimedia Nusantara yang terdiri dari Penulis Deandrha Agusta sebagai *programmer* dengan judul laporan magang PERANCANGAN SISTEM INFORMASI PENJUALAN *ONLINE* MODUL *ITEM* DAN *LAYOUT* PADA KOPERASI SETJEN DPR RI, Fachrin Hafizh Fauzan sebagai *programmer* yang mengerjakan modul *login* dan jenis produk, dan Noer Muhammad Ghozali sebagai *programmer* yang mengerjakan bagian yang

terkait dengan proses transaksi dan pencarian produk. Masing-masing modul saling terhubung satu sama lainnya sehingga apabila ada salah satu modul yang bermasalah, maka akan menimbulkan masalah juga pada modul lainnya.

Dalam melakukan pembuatan perancangan sistem ini, penulis bersama tim dibimbing langsung oleh Bapak Airlangga Eka Wardhana,S.Kom.,M.T.I. yang menjabat sebagai *Staff IT* di Bidang Data dan Sarana Informasi Sekretariat Jenderal DPR RI dan di bawah pengawasan Bapak Juhartono, S.Sos. yang menjabat sebagai Kepala Bidang Data dan Sarana Informasi.

3.2. Tugas yang dilakukan

Dalam pelaksanaan kerja magang yang dilakukan di Badan Data dan Sarana Informasi atau yang biasa disebut BDSI, penulis melaksanakan beberapa tugas yang diberikan oleh pembimbing lapangan yang merupakan *staff* teknologi informasi, diantaranya adalah :

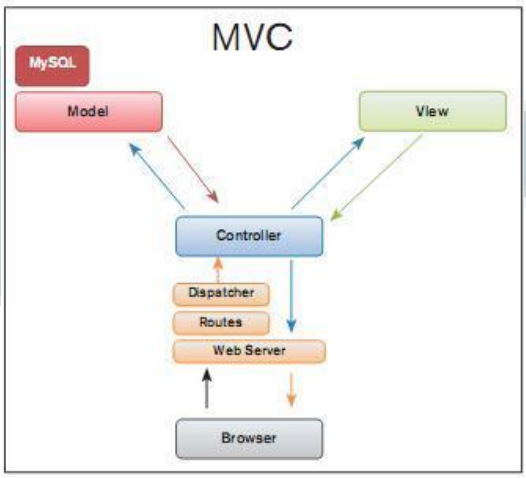
1. Melakukan pencarian informasi tentang *zend framework.*
2. Melakukan perancangan sistem informasi penjualan *online* berbasis *web* dengan menggunakan *zend framework*
3. Melakukan perancangan dan penyesuaian *user interface* dengan menggunakan *bootstrap* yang telah disediakan.
4. Menganalisa *database* yang diperlukan untuk kepentingan sistem penjualan tersebut.

Pada tahap perancangan aplikasi penulis hanya diberikan tugas untuk merancang bagian *front-end.* Sedangkan untuk bagian *back-end* yang merupakan modul *admin* diadaptasi dari aplikasi pencatatan penjualan koperasi yang sudah ada sebelumnya sehingga penulis harus menyesuaikan bagian *front-end* dan *back-end* agar dapat terhubung dengan baik.

3.3. Uraian Pelaksanaan Kerja Magang

3.3.1. Proses Pelaksanaan

Pada tahap awal, penulis melakukan pencarian informasi tentang *zend framework.* Dikutip dari situs resmi *zend framework, zend framework* atau yang disingkat *ZF* merupakan sebuah *framework* yang digunakan untuk pembuatan aplikasi *web* berorientasi *object* yang mengimplementasikan *MVC* (*model, view, controller*) dengan arsitektur *use-at-will* yang memungkinkan programer menggunakan ulang (*reuse*) sebuah komponen kapanpun diinginkan tanpa membutuhkan komponen *ZF* lainnya, sehingga meminimalisasi ketergantungan.



Gambar 3.1. Skema MVC

Setelah mengetahui sedikit tentang *zend framework*, penulis melakukan perancangan tampilan untuk bagian *front-end* sistem penjualan *online* dengan menggunakan *bootstrap* yang telah disediaka. *Bootstrap* merupakan *framework* untuk *css* yang digunakan untuk membantu menyediakan *library* dalam pembuatan *layout website*. Lalu dilanjutkan dengan menganalisa *database* sistem penjualan *online* tersebut.

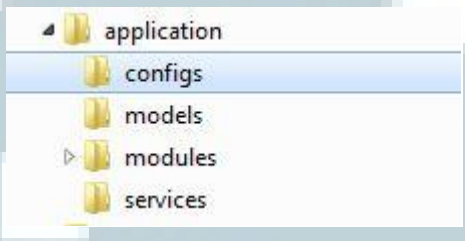
Setelah menganalisa *database*, tahap selanjutnya adalah melakukan pembuatan *script* yang sudah ditetapkan pembuatannya oleh Pembimbing Lapangan berdasarkan modul-modul yang ada sehingga anggota tim memiliki tanggungjawabnya masing-masing.

3.3.1.1. Penjelasan Aplikasi yang dibuat

Pembuatan aplikasi ini dilakukan oleh sebuah tim yang terdiri dari 3 orang yaitu penulis dan 2 orang rekan mahasiswa Universitas Multimedia Nusantara. Aplikasi yang dibuat adalah aplikasi penjualan *online* berbasis *web* yang menggunakan *zend framefork* sebagai pendukungnya.

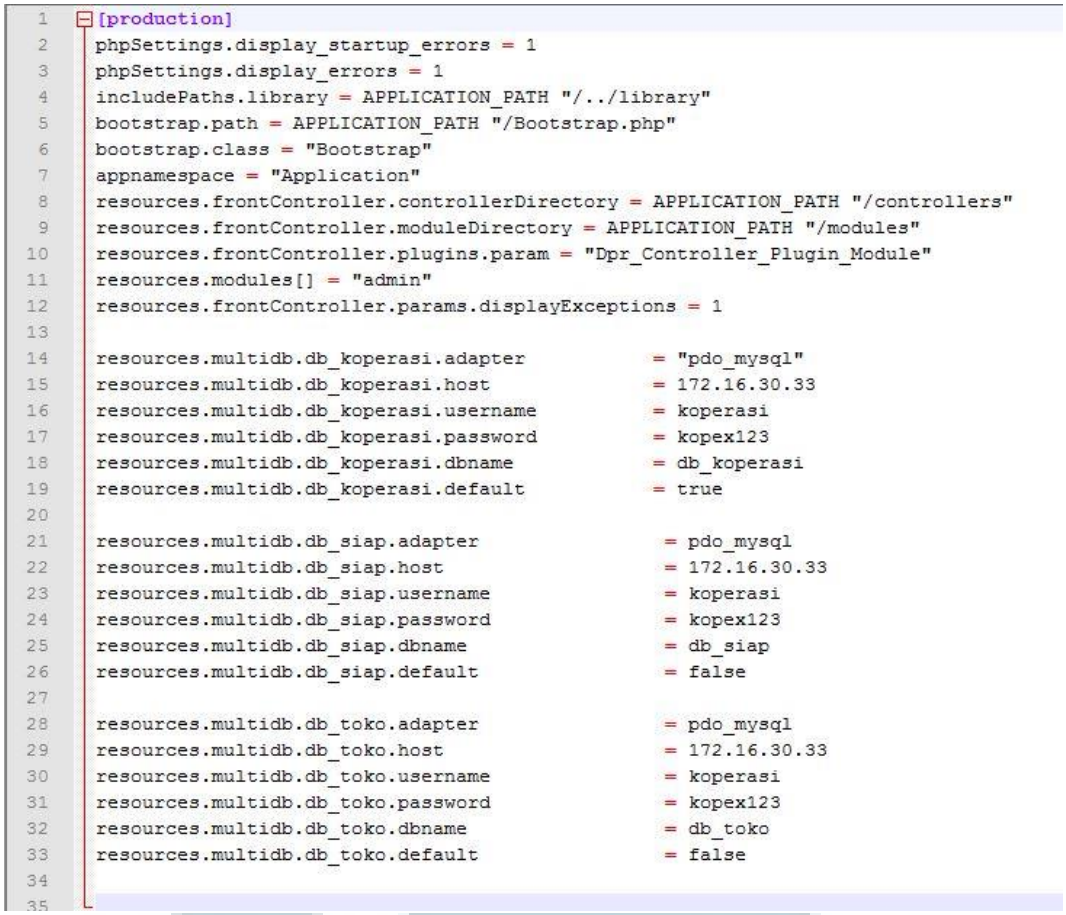
Aplikasi penjualan *online* ini dibuat untuk memudahkan pegawai dan anggota koperasi dalam melakukan pembelian produk. Pegawai tidak harus datang ke koperasi, melainkan hanya perlu memesannya melalui *website* toko.dpr.go.id, mengisi formulir dan data yang di perlukan lalu barang akan di kirimkan ke ruangannya secara langsung.

Pada tahap awal pembuatan aplikasi, penulis melakukan konfigurasi awal terhadap penempatan-penempatan *file* dengan menggunakan metode *MVC*. Metode *MVC* membagi *file* berdasarkan fungsinya masing-masing.



Gambar 3.2. Susunan *Folder* Aplikasi

Pada diatas terlihat susunan dari folder *application* yang berisi *configs, models, modules*, dan *services*. *File* yang berada di dalam *folder* tersebut memiliki fungsinya masing-masing dibagi sesuai dengan nama *folder*-nya. Pada *folder configs* berisi *file* yang berfungsi untuk mengkonfigurasi *database, host, database username, database password* serta *adapter* yang digunakan untuk menghubungkan aplikasi dengan basis datanya.



```
[production]
phpSettings.display_startup_errors = 1
phpSettings.display_errors = 1
includePaths.library = APPLICATION_PATH "/../library"
bootstrap.path = APPLICATION_PATH "/Bootstrap.php"
bootstrap.class = "Bootstrap"
appnamespace = "Application"
resources.frontController.controllerDirectory = APPLICATION_PATH "/controllers"
resources.frontController.moduleDirectory = APPLICATION_PATH "/modules"
resources.frontController.plugins.param = "Dpr_Controller_Plugin_Module"
resources.modules[] = "admin"
resources.frontController.params.displayExceptions = 1

resources.multidb.db_koperasi.adapter          = "pdo_mysql"
resources.multidb.db_koperasi.host             = 172.16.30.33
resources.multidb.db_koperasi.username         = koperasi
resources.multidb.db_koperasi.password         = kopex123
resources.multidb.db_koperasi.dbname           = db_koperasi
resources.multidb.db_koperasi.default          = true

resources.multidb.db_siap.adapter               = pdo_mysql
resources.multidb.db_siap.host                  = 172.16.30.33
resources.multidb.db_siap.username              = koperasi
resources.multidb.db_siap.password              = kopex123
resources.multidb.db_siap.dbname                = db_siap
resources.multidb.db_siap.default               = false

resources.multidb.db_toko.adapter               = pdo_mysql
resources.multidb.db_toko.host                  = 172.16.30.33
resources.multidb.db_toko.username              = koperasi
resources.multidb.db_toko.password              = kopex123
resources.multidb.db_toko.dbname                = db_toko
resources.multidb.db_toko.default               = false
```

Gambar 3.3. *file application.ini*

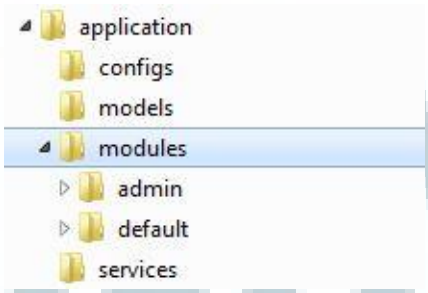
Pada *file application.ini* terdapat banyak *database* dengan *host* dan *adapter* yang sama. Dalam proyek ini, *database* yang digunakan adalah *database* "db_toko" dengan *username* "koperasi" dan *password* "kopex123".

Pada folder *models* hanya berisi *file-file* berformat .php yang membuat *class* sesuai dengan modulnya dan menghubungkannya dengan *database* dan *host* yang ada. Perlu diingat model dalam proyek ini hanya sebagai penghubung aplikasi dengan tabel yang ada di *database* sedangkan untuk *function* pemanggilannya dilakukan oleh *services*.

```php
<?php
class Stok extends Zend_Db_Table {
    protected $_name;
    protected $_schema;
    protected $_adapter;
    protected $_db;

    public function init()
    {
        $this->_name = 'stok';
        $this->_schema  = 'db_toko';
        $this->_adapter = 'db_toko';
        $this->_db = Zend_Registry::get('db_toko');
    }
}
```

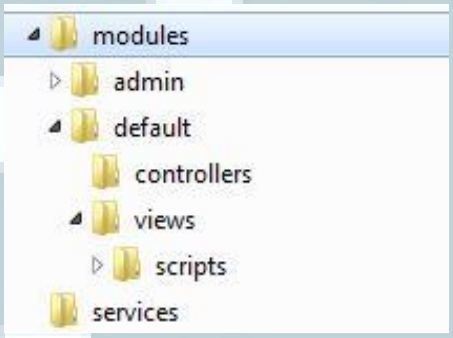
Gambar 3.4. Contoh *file stok.php* di *folder models*

Selanjutnya adalah *folder modules.* Didalam folder ini, terdapat 2 bagian yaitu bagian untuk *admin* yang bernama folder *admin* dan untuk *front-end* yang bernama *folder default.* Pada proyek ini, penulis hanya melakukan perancangan dibagian *front-end* saja namun penulis diberi hak akses untuk mengakses halaman *admin*.



Gambar 3.5. Susunan *folder modules*

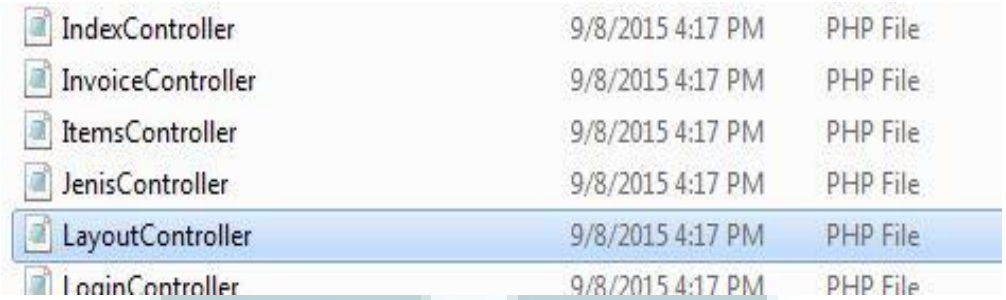
Folder default dibagi menjadi 2 bagian yaitu *folder default/controllers* dan *folder default/views/scripts*. Masing-masing folder memiliki fungsinya tersendiri.



Gambar 3.6. Susunan *folder modules/default*

Controller berfungsi sebagai pengatur dan penggabung modul *model* dan modul *view* untuk memastikan bahwa data yang benar akan tampil pada halaman *web*.

Dalam *zend framework, controllers* merupakan sebuah *class* yang dipanggil dengan metode (*Controller name*) . Cara penulisannya pun juga harus diperhatikan, bahwa dalam penulisan (*Controller name*) harus dimulai dengan huruf besar dan *class* ini harus dinamai dengan bentuk (*Controller name*)*Controller*.php. Penamaan *file* (*Controller name*) harus diawali dengan huruf besar dan selanjutnya ditulis dengan huruf kecil. Tidak boleh memakai tanda *strip* atau semacamnya.



Gambar 3.7. Contoh *file controllers*

Komponen *view* merupakan modul yang berhubungan dengan tampilan. Dalam proyek ini, komponen *view* menggunakan *file* berbentuk *phtml. Phtml* merupakan halaman *html* yang didalamnya terdapat *script php*. Sebenarnya tidak ada perbedaan yang signifikan antara *phtml* dengan *php*, yang membedakannya hanyalah *file php* tidak memiliki kode untuk *view* sedangkan *phtml* memiliki *data logic* dan sebagian besarnya merupakan kode untuk *view*.

```
<div class="row">

  <!-- Sidebar -->
  <div class="col-md-3 col-sm-3 hidden-xs">

    <h5 class="title">Categories</h5>
    <!-- Sidebar navigation -->
      <nav>
        <!-- Main menu. Use the class "has_sub" to "li" tag if it has submenu. -->
          <?
          $peran = Zend_Auth::getInstance()->getIdentity()->peran;
          $username = Zend_Auth::getInstance()->getIdentity()->nama;
          $nip = Zend_Auth::getInstance()->getIdentity()->nip;
          ?>
        <ul>
         <?  if (count($this->RowJenisByItem) > 0) {
          foreach($this->RowJenisByItem as $row)
          {
              echo '<li class="has_sub"><a href="/jenis/index/id/' . $row->id_jenis .
              <ul>';
              foreach($this->RowJenisKategori as $row)
              {
                      echo '  <li class="has_sub"><a href="/items/index/id/' . $row->i
              }
              echo '</ul>
              </li>';
          }
      }   else {
```

Gambar 3.7. Contoh *script* phtml

Selain ketiga metode *MVC* tersebut, ada *folder* tambahan yaitu *folder service.* Fungsi dari *folder service* adalah untuk menampung *query* dari setiap *class* yang dapat dipanggil oleh *controller* sesuai dengan kebutuhan. Isi dari *file service* adalah berupa kumpulan *function sql* yang terdiri dari *query select, insert, delete* maupun *update*.

Komponen *service* terhubung dengan komponen *controller* dan dapat dipanggil langsung oleh komponen *controller*. *Controller* sebagai pengatur hanya memanggil *function* yang ada pada *service* sesuai dengan kebutuhan kelasnya masing-masing.

Sistem Penjualan *Online* bagian *front-end* ini memiliki beberapa modul. Setiap modul tersebut memiliki *folder* masing-masing dan didalam *folder* tersebut terdapat *file phtml* yang merupakan *index* dari tiap *modules*. Penulis hanya mengerjakan 4 bagian modul saja, dan modul tersebut hanya mencakup bagian *front-end* saja dikarenakan penulis tidak mendapatkan akses untuk mengubah halaman *admin*. Modul yang penulis kerjakan adalah :

- Modul *Items*
- Modul *Single Item*
- Modul *Feedback*
- Modul *Layout*

Terdapat modul-modul lain yang dikerjakan oleh rekan penulis dalam perancangan sistem informasi penjualan ini. Berikut adalah modul-modul yang dikerjakan oleh rekan penulis :

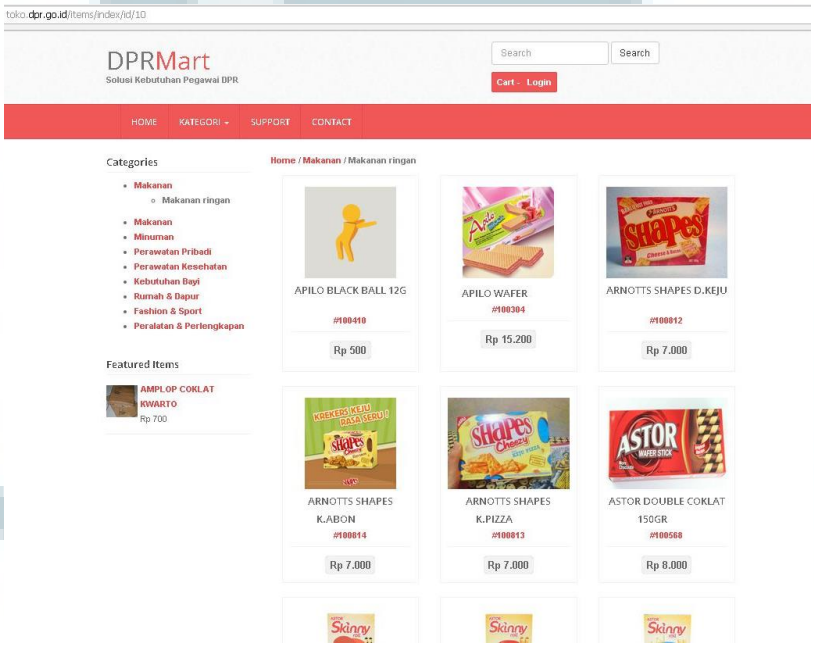
- Modul *Login*
- Modul *Index*
- Modul Jenis
- Modul *View Cart*
- Modul *Invoice*
- Modul *Order History*
- Modul *Search*
- Modul *Pagination*

Dari semua modul yang telah disebutkan diatas, tiap modul memiliki fungsinya masing-masing dan saling berkaitan satu sama lain. Apabila ada salah satu modul yang bermasalah, maka modul lain pun akan ikut terlibat karena saling terhubung.

Berikut adalah penjelasan secara rinci tentang proses pembuatan bagian *front-end* dari empat modul yang penulis kerjakan :

A. Pembuatan Modul *Items*

Modul *items* merupakan modul kategori yang merupakan *sub-modul* dari *modul* jenis. Modul ini berada di *folder items/index.phtml.* Berbeda dengan modul jenis, modul *item* menampilkan produk yang lebih spesifik, seperti untuk kategori *item* makanan ringan dari jenis makanan hanya menampilkan makanan ringan saja sedangkan jika hanya melihat dari jenis makanan, produk yang akan tampil adalah produk semua makanan yang dijual.



Gambar 3.8. Halaman *items* jenis Makanan dengan kategori Makanan Ringan

Modul *item* terhubung dengan modul lain yaitu modul jenis dan modul *layout*. Dalam hubungannya dengan modul jenis, modul *item* membutuhkan modul jenis untuk mendeklarasikan jenis dari masing-masing *item*, sedangkan modul *layout* dipakai untuk bagian *navigasi, header, footer* dan *recent item*.

Dalam pembuatan *user interface* modul *items*, dibutuhkan *file phtml* yang berisi kode *html* dan *php* dengan dibantu oleh *bootstrap* yang berisi *file css* dan *jquery*. *File phtml* tersebut berada di folder *items/index.phtml* yang berfungsi sebagai penampil *user interface* dan membutuhkan *controller* sebagai *file* pendukungnya. *File html* berisi kode-kode untuk pengaturan penempatan posisi dan letak sesuai dengan desain yang telah ditentukan. Berikut potongan *source code html* dari *file items/index.phtml* :

```
<div class="items">
 <div class="container">
  <div class="row">
   <div class="col-md-3 col-sm-3 hidden-xs">
    <h5 class="title">Categories</h5>
     <nav>
```

Untuk memanggil gambar, produk dan jenis dibutuhkan kode *php* yang ditempatkan diantara kode-kode *html* dengan diawali tanda “<?php” atau cukup tanda “<?” dan di tutup dengan tanda “?>”. Berikut adalah contoh kode *php* yang merupakan potongan kode dari *file items/index.phtml* :

```
<? if (count($this->RowJenisByItem) > 0) {
                    foreach($this->RowJenisByItem as $row)
                    {
                    echo '<li class="has_sub">
                    <a href="/jenis/index/id/' . $row->id_jenis . '">' .
                    $row->jenis . '</a></li>';
                    }
```

Kode tersebut digunakan untuk melakukan pemanggilan "jenis" produk berdasarkan "*RowJenisByItem*" yang merupakan perintah yang berasal dari *controller/ItemsController.php*. Untuk menampilkan jenis produknya digunakan kode *"$row->jenis"*. Untuk memanggil nama produk bisa menggunakan kode "*$row->nama"*. "*$row"* digunakan untuk memanggil produk berdasarkan *row* yang ada pada tabel dan dibuat dengan menggunakan metode "*foreach"*.

Seperti yang dijelaskan sebelumnya, *controller* berfungsi sebagai pengatur dan penggabung modul *model* dan modul *view* untuk memastikan bahwa data yang benar akan tampil pada halaman *web*. Perlu diperhatikan bahwa nama kelas harus sama dengan nama *file*.

Dalam *file controller/ItemsController.php* terdapat beberapa fungsi yang digunakan seperti "*public function init"* yang merupakan function untuk mengatur dalam memberikan hak akses *user*, "*public function preDispatch"* untuk mendefinisikan *new service* dan *"public function indexAction"* untuk mendefinisikan fungsi-fungsi yang digunakan dalam halaman *index*. Dalam hal

ini, "*function indexAction"* melakukan *getRequest* dari *id* yang ada di *url* juga membuat *function view* dengan memanggil *function* dari *services*. Berikut contoh dari kode *controller/ItemsController.php* yang digunakan untuk melakukan pemanggilan *view* dari *file JenisService.php* :

```
$this->view->RowJenisByItem=$this->JenisService-
>getAllJenisByKategori('$id_kategori');
```

Kode tersebut digunakan untuk melakukan pendeklarasian untuk *view* atau tampilan yang akan digunakan pada *file phtml* dengan menggunakan fungsi yang sudah dibuat didalam *file JenisService.php* yang berada dalam *folder services*. Metode *services* digunakan untuk menyimpan fungsi-fungsi *query sql* yang terhubung dengan tabel-tabel yang ada pada basis data. Fungsi yang digunakan adalah *"function getAllJenisByKategori('$id_kategori');"*. Berikut adalah kode dari fungsi tersebut :

```
function getAllJenisByKategori($id_kategori)
{
$id_kategori = Zend_Controller_Front::getInstance()->getRequest()-
>getParam('id');
$sql = "SELECT a.id, a.jenis, b.id, b.kategori, b.id_jenis
FROM db_toko.jenis a
JOIN db_toko.kategori b ON a.id = b.id_jenis
WHERE b.id = '" . $id_kategori . "'";
```

```
        $db = Zend_Registry::get('db');
        $stmt = $db->query($sql);
        $stmt->setFetchMode(Zend_Db::FETCH_OBJ);
        $result = $stmt->fetchAll();
        return $result;
}
```

Fungsi tersebut digunakan untuk memanggil data yang ada pada tabel jenis dan kategori dari *database* "db_toko" dengan melakukan *join* antara kedua tabel. Dalam modul *items* terdapat juga produk dan deskripsinya yang di panggil melalui *file services/StokService.php.*

Dalam *file* ini terdapat beberapa function seperti *function getAllData(), getAllDataByStokXRows()* dan *getAllDataByCategoryLimit().* Berikut adalah penjelasan dari masing-masing fungsi yang hanya digunakan untuk modul items :

1) ***Function getAllData*()** : memanggil data dari tabel stok berdasarkan tanggal *update* secara *descending,*
2) ***Function getAllDataByStokXRows*()** : memanggil data dari *database* berdasarkan *id* kategorinya.
3) ***Function getAllDataByCategoryLimit*()** *:* memanggil data dari *database* berdasarkan *id* kategori namun dibatasi hanya 4 data saja yang tampil.

Dalam pembuatan modul *items* dibutuhkan juga *file services/KategoriService.php*. Peran *KategoriServices.php* adalah untuk memanggil nama kategori yang berhubungan dengan *item* atau produk. Dalam proyek ini, jenis merupakan *parent* sedangkan kategori merupakan *child*-nya. Berikut adalah kode dari fungsi-fungsi yang ada pada *services/KategoriService.php* yang digunakan pada modul *items* :

```
function getAllData()
{
$select = $this->kategori->select()
                ->setIntegrityCheck(false)
                ->from(array('a' => 'kategori'), array('*'))
                ->joinLeft(array('b' => 'jenis'), 'a.id_jenis = b.id', array('jenis'))
                ->where('a.status = 1');
                $result = $this->kategori->fetchAll($select);
                return $result;
}
```

Fungsi *getAllData()* adalah untuk memanggil seluruh data yang berada pada tabel kategori dan tabel jenis. Dalam *sql* untuk dapat memanggil data dari 2 tabel secara langsung dapat dilakukan dengan menggunakan metode *join.*

```
function getAllDataByCategoryXRows($id)
{
```

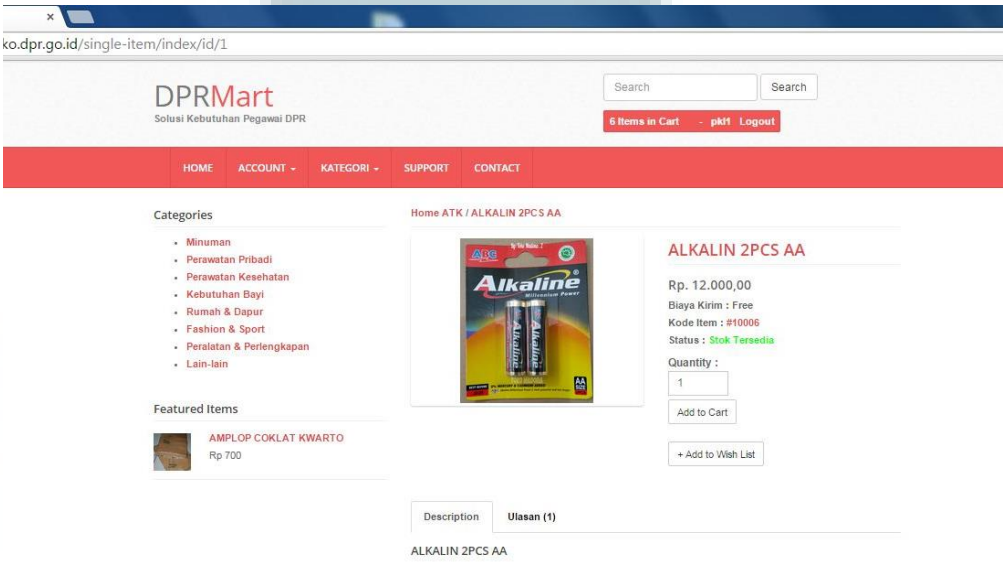
```
$id = Zend_Controller_Front::getInstance()->getRequest()->getParam('id');
$select = $this->kategori->select()
                ->setIntegrityCheck(false)
                ->from('kategori', array('*'))
                ->where('id = ?', $id)
                ->where('status = 1');
$result = $this->kategori->fetchAll($select);
return $result;
}
```

Fungsi getAllDataByCategoryXRows($id) adalah untuk memanggil data dari tabel kategori berdasarkan id kategori yang dipilih. Id kategori yang dipilih oleh *user* dapat ditemukan melalui *url* menggunakan fungsi "*Zend_Controller_Front::getInstance()->getRequest()->getParam('id');*" dan mendeklarasikannya sebagai *variable* "$id". Fungsi ini digunakan pada saat *user* melakukan pemilihan *item* berdasarkan kategori yang dipilih. Modul *items* akan menampilkan produk berdasarkan kategori yang dipilih menggunakan fungsi ini.

B. Modul *Single Item*

Modul *single item* berisi tentang *detail* dari tiap-tiap produk yang ada pada katalog. *Detail* tersebut adalah berupa nama produk, harga produk, kode produk serta ulasan dari masing-masing produk. Didalam modul ini juga terdapat "*field text box"* yang digunakan untuk melakukan *input quantity*, tombol "*add to cart*" yang berfungsi menyimpan data pembelian kedalam *shopping cart* dan

tombol "*add to wish list*" yang fungisnya adalah untuk menyimpan data produk yang ingin dibeli. Hal tersebut merupakan tanggungjawab dari rekan tim penulis, Noer Muhammad Ghozali karena berkaitan dengan proses transaksi.



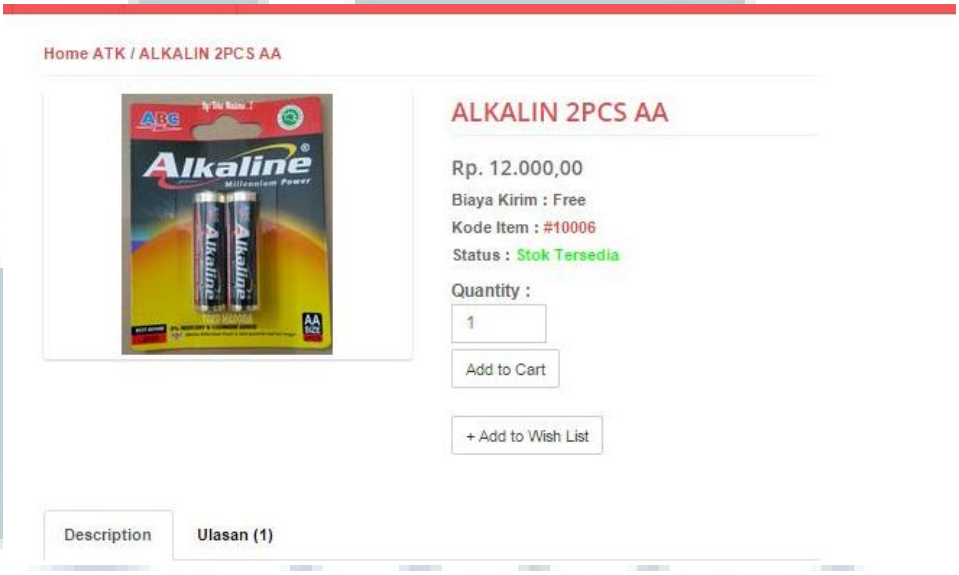
Gambar 3.9. Halaman *Single item*

Hal-hal yang akan di bahas pada modul *single item* ini adalah berkaitan dengan *detail* dari produk dan ulasan produk. Dalam membuat modul *single item*, dibutuhkan *file phtml* untuk membuat *user interface a*tau tampilan untuk modul *single item*. Dalam proyek ini, *file phtml* untuk *user interface* modul *single item* berada di *SingleItem/index.php*. Dalam tersebut terdapat *script* berbentuk *javascript* yang mengatur bilangan dibelakang angka untuk harga produk. Jadi dibelakang harga produk ada bilangan desimal yang banyaknya bisa diatur. Berikut adalah kode dari *script* tersebut :

```
<script type="text/javascript">
    var DecimalSeparator = ",";
    var ThousandSeparator = ".";
```

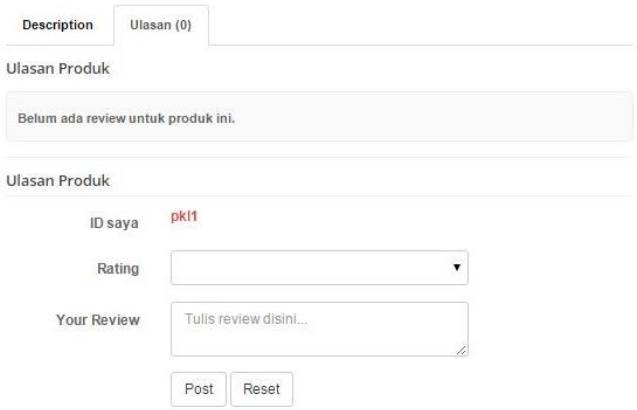
```
var CurrencyDecimalDigits = 0;

</script>
```

Dibawah *script* tersebut terdapat script *php* dan *html* yang digunakan untuk menampilkan data produk yang diambil dari tabel *stok* dan tabel *pos_data*. *Script* tersebut merupakan bagian *detail* produk. *Detail* produk berisi tentang penjelasan secara *detail* dari produk yang ada seperti nama produk, harga jual dan kode *item* dari produk tersebut. Untuk melakukan penambahan, perubahan atau penghapusan terhadap *detail* produk, *user* harus mengisi *form* yang ada pada halaman *admin*. Seperti yang telah dijelaskan sebelumnya, penulis hanya membuat bagian *front-end* saja, tidak termasuk bagian *admin* karena penulis tidak diberikan hak untuk melakukan perubahan pada bagian *admin*.



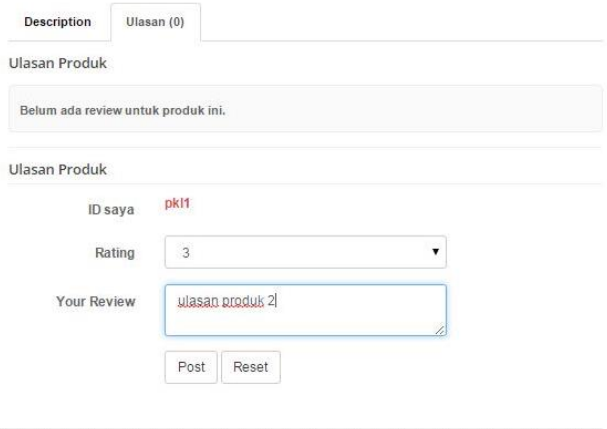
Gambar 3.10. *Detail* Produk

Dalam halaman *single item* juga terdapat ulasan produk. Ulasan produk digunakan oleh *user* untuk mengulas produk. Fungsinya adalah agar produk tersebut bisa dinilai kualitasnya secara langsung oleh pengguna.



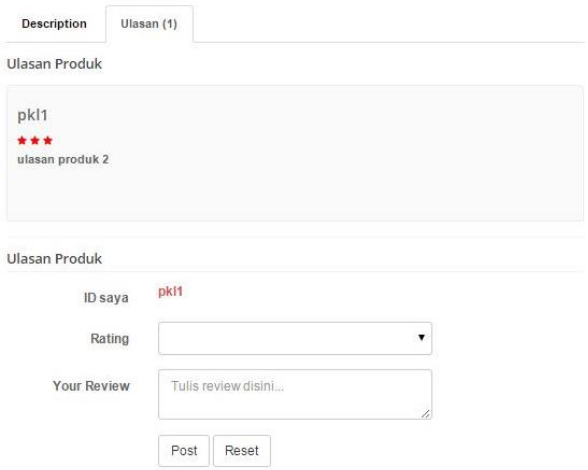
Gambar 3.11. Ulasan produk 1

Pada gambar diatas terlihat belum ada yang mengulas produk. Namun pengguna sudah melakukan *login* menggunakan *userid* "pkl1".



Gambar 3.12. Ulasan Produk 2

Pada gambar diatas dijelaskan bahwa pengguna dengan *userid* “pkl1” mengulas suatu produk dengan memilih *rating* dan menulis ulasannya pada kolom yang tersedia.



Gambar 3.13. Ulasan produk 3

Pada gambar diatas juga dijelaskan bahwa ulasan sudah diisi oleh pengguna dengan *userid* “pkl1” dengan *rating* produk adalah 3. Ulasan tersebut hanya bisa diisi oleh pengguna yang sudah terdaftar dan harus melakukan *login* terlebih dahulu.

Data yang diisi oleh *userid* “pkl1” masuk kedalam tabel *review* yang ada pada *database* “db_toko” melalui *file controller/SingleItemController.php*. *File* tersebut digunakan untuk membuat *action* yang kemudian dipanggil oleh modul *view* untuk dijalankan sesuai dengan *syntax* yang diberikan. Jika pengguna belum *login*, maka pengguna tersebut diperintahkan untuk *login* terlebih dahulu jika ingin menulis ulasan.

Selain bagian *review items*, pada modul *view* juga terdapat bagian deskripsi produk yang menjelaskan secara rinci tentang sebuah produk. Bagian ini memberikan informasi kepada pengguna terhadap spesifikasi barang seperti warna, berat, ukuran atau bentuk dari suatu barang. Proses pemanggilan deskripsi barang tersebut dilakukan menggunakan bahasa pemrograman *php*, berikut adalah potongan dari *source code file SingleItem/index.phtml* yang ada pada modul *view* :

```
<div class="tab-pane active" id="tab1">
        <h5><?php echo "$row->nama"; ?></h5>
        <p><?php echo "$row->deskripsi"; ?></p>
</div>
```

Sama seperti "*$rows"* yang ada pada modul *Items*, *variable* tersebut digunakan untuk menandakan urutan dari sebuah tabel sesuai dengan *id* yang ada pada produk dengan cara melakukan pendeklarasian *id* produk yang ada pada *database* dengan *id* produk yang ada pada *url* dengan melakukan *getParameter()* berdasarkan *id*. Fungsi tersebut dibuat pada modul *services* dengan perantara modul *controller* sebagai pengaturnya yaitu berada pada *file controller/SingleItemController.php.*

File controller/SingleItemController berisi tentang fungsi yang digunakan untuk melakukan pengaturan terhadap *action* yang akan digunakan di modul *view* dan datanya diambil dari *file* data yang ada pada *folder services*. Dalam *file controller* terdapat *script* "*$id = $this->getRequest()-*

>getParam('id');" yang menyatakan bahwa "*$id"* adalah *id* yang ada pada produk yang dipilih. Jadi jika sebuah produk di klik maka produk tersebut akan menampilkan rincian dari produk tersebut. Fungsi inilah yang digunakan untuk mendeklarasikan "*$row* "yang merupakan bagian dari fungsi "*foreach*".

Terdapat banyak fungsi yang digunakan oleh modul *view*, salah satunya adalah fungsi pemanggilan data dari sebuah *item* yang berada pada tabel stok sesuai dengan produk yang dipilih oleh pengguna. *Controller* melakukan *action view* menggunakan *script* "$this->view->StokRowsId = $this->StokService->getAllDataById('$id_produk');" yang berarti *controller* memanggil fungsi *getAllDataById* yang ada pada modul *services* dengan menggunakan "*StokRowsId*" sebagai variabel untuk melakukan pemanggilan *view* dengan metode "*foreach*".

Seperti yang telah dijelaskan sebelumnya bahwa proses pemanggilan *item* dilakukan dengan cara melakukan pendeklarasian atas *id* produk yang ada pada *database* dengan *id* produk yang berada pada *url* dengan melakukan metode "*getParameter*." Pendeklarasian ini dilakukan di modul *services* tepatnya *pada file services/StokServices.php* karena semua data produk yang ditampilkan ada pada tabel stok. Untuk lebih jelasnya bisa dilihat pada *function getAllDataById* yang ada pada *file services/StokService.php* sebagai berikut :

```
function getAllDataById($id)
{
$id_produk = Zend_Controller_Front::getInstance()->getRequest()->getParam('id');
```

```
$select = $this->stok->select()
                ->setIntegrityCheck(false)
                ->from('stok', array('*'))
                ->where('id = ?', $id_produk)
                ->where('status = 1');
$result = $this->stok->fetchAll($select);
return $result;
}
```

Dalam function tersebut terdapat *script "$id_produk = Zend_Controller_Front::getInstance()->getRequest()->getParam('id');"* yang menyatakan bahwa *"$id_produk"* dipanggil melalui fungsi *getRequest()* dengan metode *getParam()* berdasarkan *'id'*. Dalam *script* ini juga di jelaskan *"->where('id = ?', $id_produk)"* yang menyatakan bahwa *id* yang ada pada tabel stok harus sama dengan *id* yang di panggil oleh metode *getParameter* yang merupakan *id* produk yang di klik.

Dalam modul *view* juga terdapat bagian yang digunakan untuk menampilkan kategori dari sebuah produk yang dipilih. Hal ini digunakan untuk mendefinisikan bahwa produk tersebut merupakan bagian dari suatu kategori yang ada. Fungsi tersebut terdapat pada *file services/KategoriService.php* karena berkaitan dengan kategori. Terdapat beberapa fungsi dalam *file KategoriService.php* yaitu :

1) *getAllData()* : mengakses data yang ada pada tabel kategori dengan melakukan *join* dengan tabel jenis yang merupakan *parent* dari tabel kategori.
2) *getAllDataByCategoryXRows($id)* : mengakses data dari tabel kategori dengan menyetarakan *id* kategori dengan *id* produk yang dipilih.
3) *getAllDataByFeatured()* : mengakses data dari tabel kategori dimana kategorinya adalah kategori *featured.*
4) *getAllDataById($id)* : mengakses data dari tabel kategori berdasarkan *id*.
5) *getDataByCategory($kategori)* : mengakses data dari tabel kategori berdasarkan kategori.

C. Modul Layout

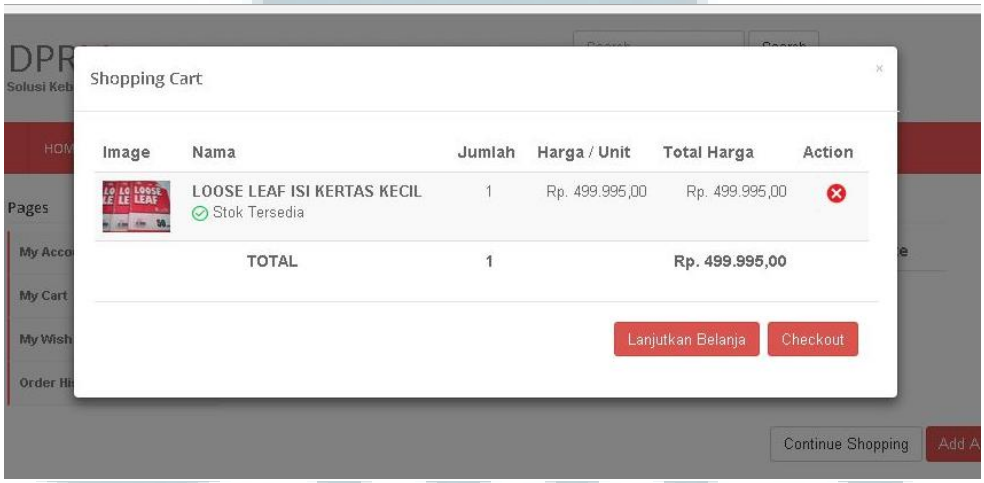
Layout digunakan sebagai *file* dasar yang dipakai pada setiap halaman tanpa harus dibuat kembali, seperti fungsi *root* pada bahasa pemrograman VB. Modul *layout* berada didalam modul *view* yang bernama *layout.phtml* namun berada diatas satu tingkat dari *file index.phtml* pada tiap halaman yang membuat *file* ini dapat digunakan oleh semua halaman. Pada *file layout.phtml* terdapat beberapa bagian yaitu bagian *header, shopping cart, navigation, recent item, login, search* dan *footer*. Penulis hanya mengerjakan beberapa bagian saja yaitu bagian *header, shopping cart, navigation, recent item dan footer.*

Header merupakan bagian paling atas *layout* yang berisi logo, menu *search,* menu *login/logout* dan menu *shopping cart*. Bagian ini berisi bahasa *html*

dan *php*. Bahasa pemrograman *php* digunakan karena ada fitur *search* dan *login* didalamnya.

Gambar 3.14. *Header Layout*

Pada bagian *header* juga terdapat menu *shopping cart* yang merupakan sebuah tempat untuk menampung barang yang ingin dibeli oleh pengguna. Pada halaman *single item* terdapat *button* "*add to cart*" yang digunakan untuk memasukkan produk tersebut kedalam "*shopping cart".* Dalam pembuatan *shopping cart*, penulis hanya melakukan pembuatan untuk bagian layoutnya saja, tidak termasuk fungsi-fungsi didalamnya. Dengan adanya *shopping cart* , *user* dapat membatalkan produk yang ingin dibeli dengan melakukan klik pada tanda "X" yang ada pada kolom *action.*



Gambar 3.15. *Shopping Cart Layout*

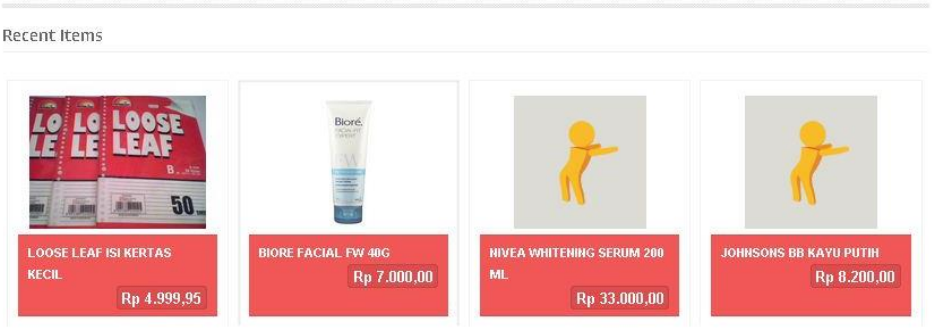
Menu ini berisi sebuah tabel yang didalamnya terdapat produk yang telah dipilih oleh pengguna dengan keterangan jumlah produk dan total biaya

yang harus dibayar. Tombol "*checkout*" digunakan untuk melanjutkan proses transaksi ke tahap selanjutnya yaitu tahap pembayaran. Proses pembayaran dilakukan menggunakan sistem "*Cash On Del*ivery". Pegawai koperasi akan mengantarkan produknya langsung ke ruangan pemesan. Jika dilakukan klik pada tombol "lanjutkan belanja" halaman *web* akan mengarah langsung ke tempat sebelumnya.

Selanjutnya adalah menu *navigation* yang berisi *link navigasi* seperti *menu* "*home*" yang berfungsi untuk kembali ke halaman *home*, *menu* "*account*" yang digunakan untuk melihat rincian transaksi maupun rincian dari pengguna, menu "kategori" yang digunakan untuk menuju ke halaman kategori yang dipilih, menu "*support*" yang berisi tentang data koperasi dan menu "*contact*" yang digunakan oleh pengguna untuk mengirimkan *feedback.*

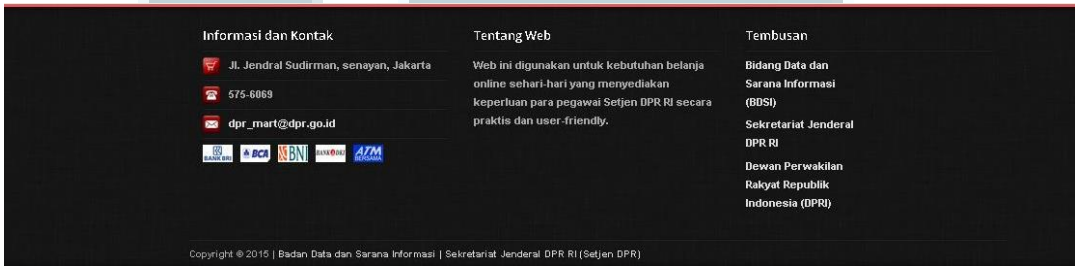
Gambar 3.16. *Navigation Layout*

Pada bagian bawah terdapat bagian "*recent items"* yang berisi tampilan produk diurutkan berdasarkan tanggal *input* produk. Dari bagian ini bisa diketahui produk mana yang merupakan produk terbaru. Jika produk tersebut diklik, maka halaman yang dituju adalah halaman *single item.*



Gambar 3.17. *Recent Items Layout*

Selanjutnya adalah bagian yang terakhir yaitu bagian "*footer"*. Bagian ini berisi tautan-tautan dan informasi tentang Setjen dan Koperasi DPR yang terhubung dengan situs ini serta menandakan hak cipta dari situs ini. Semua bagian yang ada pada modul *layout* digunakan disetiap halaman. Jadi jika kita mengubah salah satu bagian pada modul *layout*, maka setiap halaman yang terhubung juga akan ikut berubah.



Gambar 3.18. *Footer layout*

Dalam modul *layout* juga terdapat *controller* yang digunakan sebagai pengatur dalam melakukan pemanggilan data dari modul *services*. Modul ini digunakan untuk melakukan "*action*" pemanggilan kategori, *items* dan lainnya yang berada pada modul *layout*.

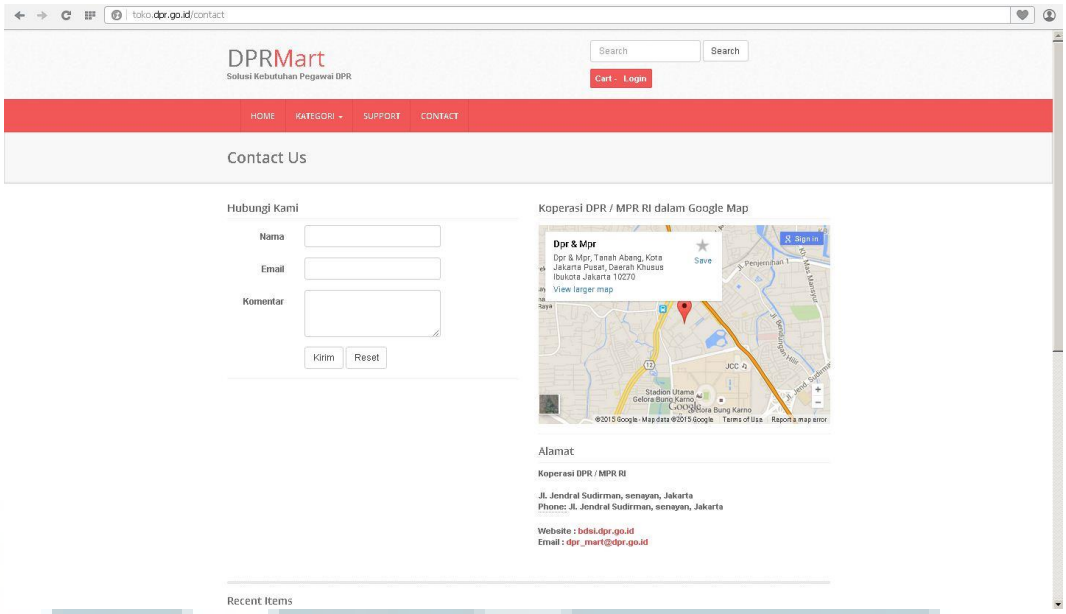
Dalam modul *layout* terdapat *function getAllData* yang merupakan fungsi yang dimiliki oleh setiap *file service* dan berada di *folder services*. Fungsi

ini berfungsi untuk memanggil semua data dari tabel yang ada di dalam suatu *database*. Dalam modul *layout* ini, fungsi yang digunakan sama persis hanya berbeda sumber tabelnya saja. Berikut adalah penjelasan dari tiap-tiap *view* :

1) *$this->PosService->getAllData();* : melakukan *select* data dari tabel pos dan berada pada *file controller*.
2) *$this->KategoriService->getAllData*(); : melakukan *select* data dari tabel pos dan berada pada *file controller*.
3) *$this->JenisService->getAllData();* : melakukan *select* data dari tabel jenis dan berada pada *file controller*.
4) *$this->StokService->getAllData();* : melakukan *select* data dari tabel stok dan berada pada *file controller*.

D. Modul Feedback

Modul ini berisi formulir untuk mengirimkan *feedback* yang digunakan oleh pengguna untuk mengirimkan informasi berupa pertanyaan, pernyataan ataupun komplain. Namun, pengguna harus melakukan *login* terlebih dahulu agar bisa mengirimkan *feedback.*



Gambar 3.19. Halaman *Feedback*

Pada gambar 3.19. terdapat peta lokasi Koperasi Setjen DPR dengan menggunakan "*Google Maps*" yang telah di *embed* menggunakan fungsi "*iframe*". Selain itu, terdapat juga formulir pengisian *feedback* yang berisi nama, *email* dan komentar. Isi dari formulir tersebut akan disimpan pada tabel "formulir_kontak" menggunakan fungsi *Insert.* Fungsi tersebut berada pada *file services* tepatnya pada *file services/FormulirKontakService.php* dan dipanggil oleh *controller*.

Untuk mendapatkan data yang sudah dimasukkan maka harus dibuat fungsi untuk mengambil data dari formulir yang telah dimasukkan tersebut dengan melakukan *getParameter* berdasarkan *name* yang ada pada *form input* tersebut. Berikut adalah potongan *source code* dari *file controller* tersebut:

```
if ( $this->getRequest()->isPost() )
{
        $email = $this->getRequest()->getParam('email');
        $komentar = $this->getRequest()->getParam('komentar');
        $this->FormulirKontakService->addData($email, $komentar);
        $this->_redirect('/contact');
}
```

Kode tersebut merupakan sebuah fungsi yang digunakan apabila tombol "kirim" di klik. *Variable "$email*" dan "$komentar" diambil dari formulir berdasarkan *name* yang ada pada *form input* masing-masing. Setelah melakukan pengambilan data dari *form*, maka function "*addData"* yang ada pada *file services/FormulirKontakService.php* akan dijalankan dan akan dikembalikan ke halaman "*contact*".

File services/FormulirKontakService.php berisi fungsi *addData* yang berfungsi untuk melakukan *query insert* data yang sebelumnya telah dipanggil oleh *controller*. Data yang dimasukkan adalah data *email*, komentar, *user_input*, tanggal_*input*, *user_update*, tanggal_*update* ke dalam tabel formulir_kontak. Data *user_input* dan *user_update* diambil dari *session* pengguna yang *login*, sedangkan tanggal_*input* dan tanggal_*update* diambil dari data pada tanggal dilakukannya *posting* . Berikut adalah fungsi *addData* yang ada pada *file services/FormulirKontakService.php* :

```
function addData($email, $komentar)
{
$user_log = Zend_Auth::getInstance()->getIdentity()->pengguna;
$tanggal_log = date('Y-m-d H:i:s');
$params = array(
        'email' => $email,
        'komentar' => $komentar,
        'user_input' => $user_log,
        'tanggal_input' => $tanggal_log,
        'user_update' => $user_log,
        'tanggal_update' => $tanggal_log
        );
$this->formulir_kontak->insert($params);
lastId = $this->formulir_kontak->getAdapter()->lastInsertId();
return $lastId;
}
```

3.3.1.2. Kendala yang ditemukan

Dalam pembuatan sistem ini terjadi kendala yang menghambat kelancaran dalam tahap perancangan aplikasi. Berikut adalah kendala yang penulis rasakan :

- Penulis baru mengetahui tentang *zend framework* saat diberi tugas oleh pembimbing lapangan untuk membuat aplikasi *e-*

commerce berbasis *web* dengan menggunakan *zend framework* yang mengakibatkan waktu yang dibutuhkan untuk penyelesaian proyek akan lebih lama dari biasanya.

- Masih banyak terdapat *bugs* dan *error* pada aplikasi dikarenakan waktu pengerjaan aplikasi yang sangat terbatas.

3.3.1.4. Solusi

Berikut adalah solusi yang penulis sarankan untuk menghadapi permasalahan tersebut adalah :

- Mempelajari *zend framework* dari forum - forum atau komunitas *programmer* yang membahas tentang *zend framework.*
- Melakukan *testing* terhadap aplikasi, menganalisisnya dan memperbaikinya.