Aplikasi Web Menggunakan Framework CakePHP (Studi Kasus di Perpustakaan STTII-UKRIM Yogyakarta)

Yosafat E. H. Rissy*[1], Febe Maedjaja[2]
[1,2]Universitas Kristen Immanuel; Jl. Solo Km. 11,1, Kalasan, DIY, (0274)496256
e-mail: *[1]fariz_ulen@yahoo.co.id, [2]febe@ukrimuniversity.ac.id

Abstrak

Framework sangat membantu developer membangun aplikasi berbasis web dengan cepat dan lebih terstruktur dalam menulis kode program serta mudah dalam pengembangan program, namun framework juga menimbulkan keterbatasan bagi developer yang perlu diteliti. CakePHP adalah sebuah framework open source yang digunakan untuk mengembangkan aplikasi web dengan dasar kerja CRUD (Create, Read, Update, Delete), dan memungkinkan developer membuat sebuah aplikasi berbasis web dengan pola pengembangan RAD (Rapid Application Developments). Dengan menggunakan konsep MVC (Model View Controller), suatu konsep pemrograman yang memisahkan pemrograman logika aplikasi dengan presentasinya, sistem yang dibangun dengan framework CakePHP memungkinkan untuk pemeliharaan dan pengembangan. Kami menguji CakePHP dalam studi kasus pembuatan system informasi pada perpustakaan STTII-UKRIM di Yogyakarta. Kami menyimpulkan bahwa framework CakePHP memberikan kemudahan dan juga menimbulkan keterbatasan bagi seorang developer. Namun tetap saja dengan menggunakan fitur-fitur yang telah disediakan, framework CakePHP dapat menjadi pertimbangan bagi developer dalam membangun aplikasi berbasis web.

Kata kunci*— CakePHP, MVC, framework*

Abstract

Frameworks greatly assist a developer in building web base applications quickly and more sttuctured in progrom code writing, as well as giving ease in program development. However, frameworks also cause limits to the developer which need to be researched. CakePHP is an open source framework used to develop web application based on the CRUD (Create, Read, Update, Delete) functions, and it allows developers to make a web based application with a RAD (Rapid Application Development) pattern. With the MVC (Model View Controller) concept, a programming gconcept which separates the application logic programming from its presentation, the system that is built by CakePHP framework allows for maintenance and later development. We tested CakePHP in a case study of creating an information system at the library of STTII-UKRIM in Yogyakarta. We concluded that CakePHP framework facilitates and at the same time causes restrictions to a developer. Still, by using the available features, CakePHP framework is worth to be considered by developers in building web based applications.

Keywords – CakePHP, MVC, framework

1. PENDAHULUAN

Dalam membangun aplikasi berbasis *web*, *framework* merupakan salah satu hal penting yang perlu diperhatikan *programmer*. *Framework* sangat membantu dalam membangun suatu aplikasi *web* yang tergolong besar, dimana aplikasi yang di buat akan lebih terstruktur dan lebih mudah dalam pengembangannya serta memudahkan anggota tim untuk bekerja bersama dengan satu cara pandang. *Framework* bisa berupa *Content Management System* (CMS), dimana *programmer* lebih sedikit melakukan *coding* program, juga bisa melakukan *coding* program secara keseluruhan dengan fitur-fitur yang tertentu dari setiap *framework* yang berbeda-beda.[1]

CakePHP adalah sebuah *framework open source* yang digunakan untuk mengembangkan aplikasi *web* dengan dasar kerja CRUD (*Create, Read, Update, Delete*). CakePHP juga menjadi salah satu *framework* pilihan yang memungkinkan *developer* membuat sebuah aplikasi *web* dengan pola pengembangan RAD (*Rapid Application Developments*).[1]

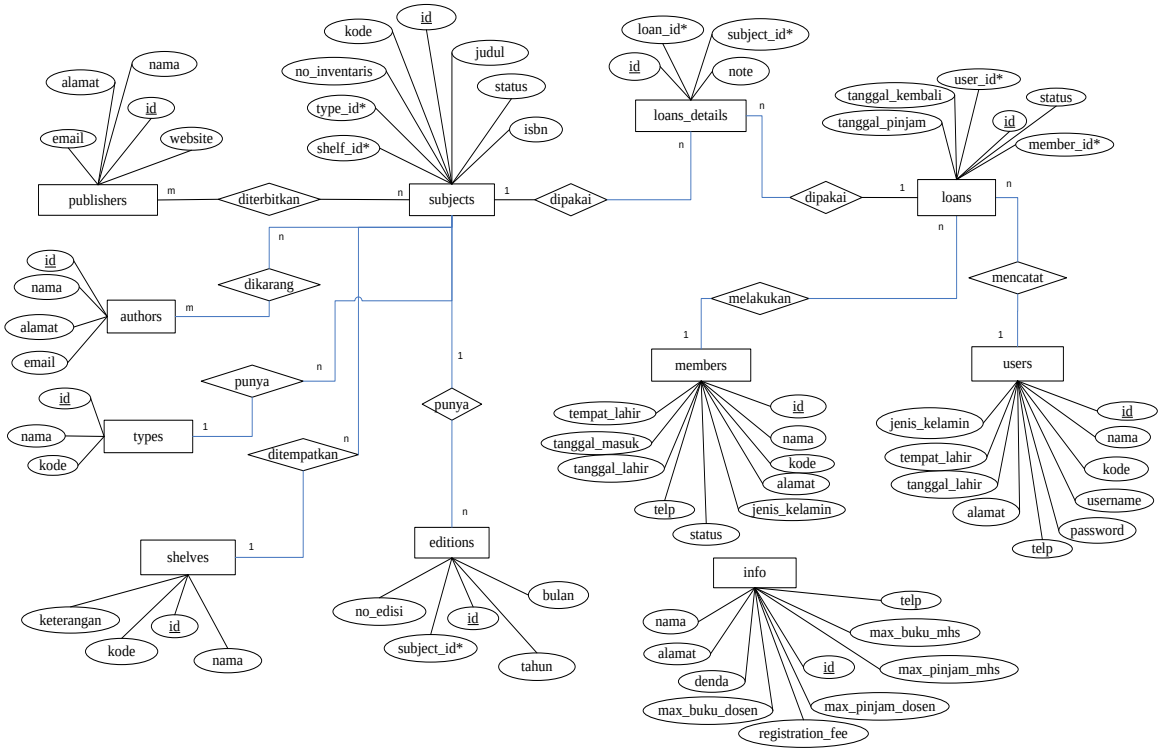
Kami akan melakukan penelitian tentang konsep kerja dan fitur-fitur *framework* CakePHP dan mengimplementasikan hasil penelitian tersebut dengan membangun sistem informasi pada perpustakaan STTII-UKRIM Yogyakarta. Hasil penelitian ini juga bisa menjadi tinjauan / referensi bagi para *developer* dalam membangun aplikasi berbasis *web*.

Dalam perancangan aplikasi kami membatasi masalah dengan penggunaan PHP versi 5.3.1, Apache Tomcat 2.2.14 sebagai *server*, MYSQL 5.1.41 sebagai *database management system*, dan CakePHP sebagai *framework* yang menjadi obyek penelitian ini. *Software* pembantu lain mencakup Windows 7 Service Pack 1 sebagai system operasi, Mozilla Firefox 19.0 dan Google Chrome 25.0.1 sebagai *web browser*, Netbeans IDE 7.2.1 sebagai *web editor*, dan Notepad++ sebagai *editor*.

2. RANCANGAN PENELITIAN

Untuk meneliti *framework* CakePHP kami akan menggunakannya dalam membangun sebuah sistem informasi untuk perpustakaan di kampus STTII-UKRIM Yogyakarta. Kami merancang *Model*, *View*, dan *Controlle*r dalam arsitektur MVC dengan menggunakan referensi-referensi.[2][3][4][5]

Bagian *Model* (M) dalam arsitektur MVC untuk sistem perpustakaan STTII-UKRIM membutuhkan rancangan *Relational Database* dengan diagram entitas yang dapat dilihat dalam Gambar 1. Dalam implementasinya dibutuhkan 12 buah tabel. Sebagai contoh, struktur tabel untuk Buku (*subjects*) dapat dilihat pada Tabel 1 dan Penerbit (*publishers*) dapat dilihat pada Tabel 2. Tabel-tabel lain adalah tabel Pengarang (*authors*), Pengarang_buku (*author_subjects*), Edisi (*editions*), Biaya (*fees*), Anggota (*members*), Petugas (*users*), Peminjaman (*loans*), Rak (*shelves*), Kategori Buku (*types*), dan Edisi (*editions*).



Gambar 1 Diagram Entitas Relational Database Perpustakaan

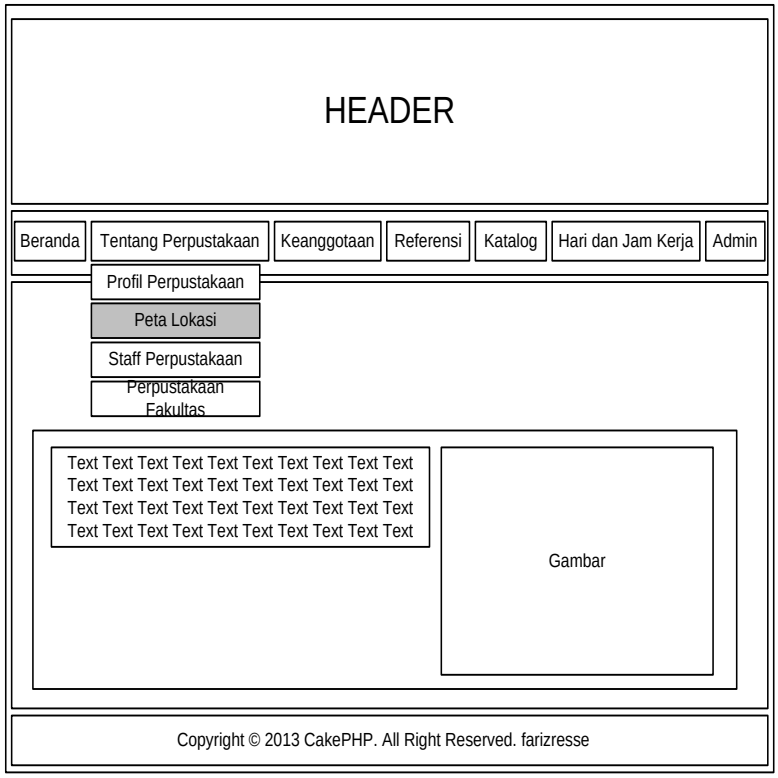
Tabel 1 Struktur Tabel Buku

Subjects		
Field	**Type**	**Keterangan**
id	int(11)	Primary key
shelf_id	int(11)	Foreign key
type_id	int(11)	Foreign key
no_inventaris	char(11)	
kode	char(11)	
judul	char(200)	
status	char(1)	
isbn	char(20)	

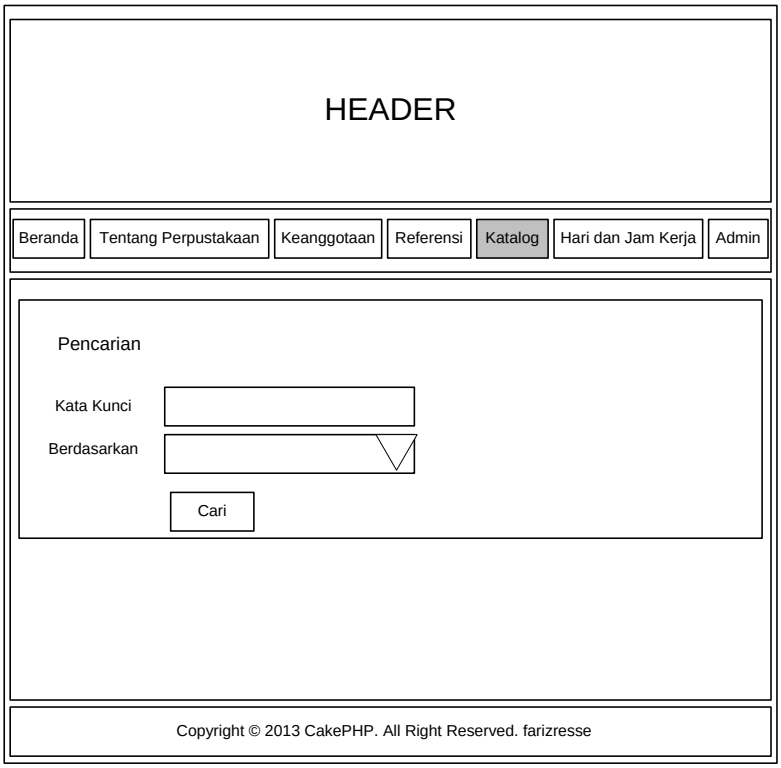
Tabel 2 Struktur Tabel Penerbit

publishers		
Field	**Type**	**Keterangan**
id	int(11)	Primary key
nama	char(50)	
alamat	char(200)	
email	char(50)	
website	char(50)	

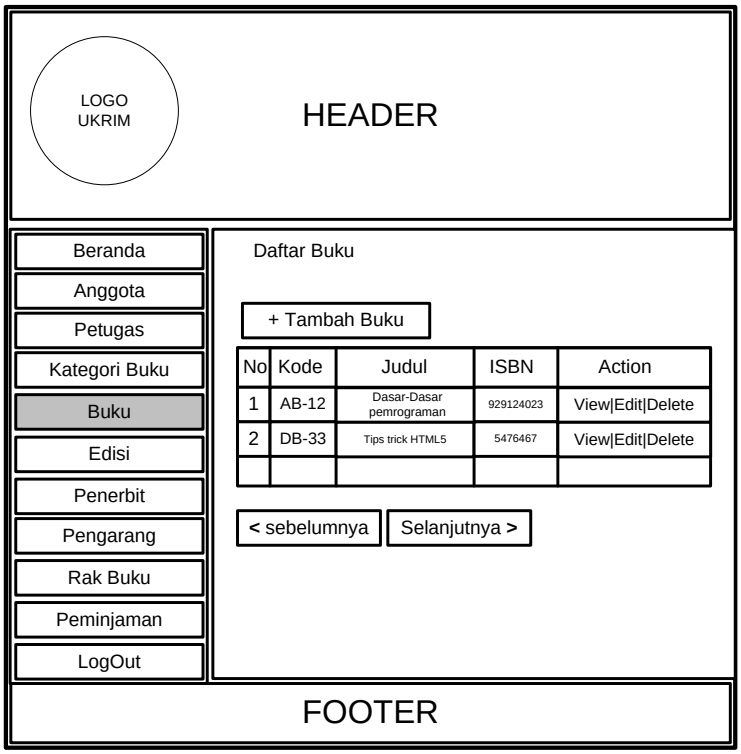
Untuk Bagian View (V) dalam arsitektur MVC aplikasi perpustakaan ini, halaman-halaman *interface* perlu dirancang dua bagian utama, yaitu bagian *front end* dan *back end*. Bagian *front end* terdiri dari halaman menu utama agar pengunjung dapat melakukan navigasi ke halaman-halaman berikut untuk mendapatkan informasi Tentang Perpustakaan, Keanggotaan, Referensi, Katalog, Hari dan Jam Kerja, serta *link* khusus untuk administrator. Sebagai contoh *design* untuk *interface* halaman Peta Lokasi dapat dilihat pada Gambar 2, sedangkan *design* untuk *interface* halaman katalog untuk mencari buku dapat dilihat pada Gambar 3. Bagian *back end* terdiri dari halaman Login Admin dan halaman-halaman administrator untuk melakukan fungsi-fungsi *Create, Read, Update and Delete (CRUD)* Angota, Petugas, Kategori Buku, Edisi, Penerbit, Pengarang, Rak Buku, dan Peminjaman. Sebagai contoh *design* untuk *interface* halaman Buku dapat dilihat pada Gambar 4, sedangkan *design* untuk *interface* halaman Tambah Buku dapat dilihat pada Gambar 5.



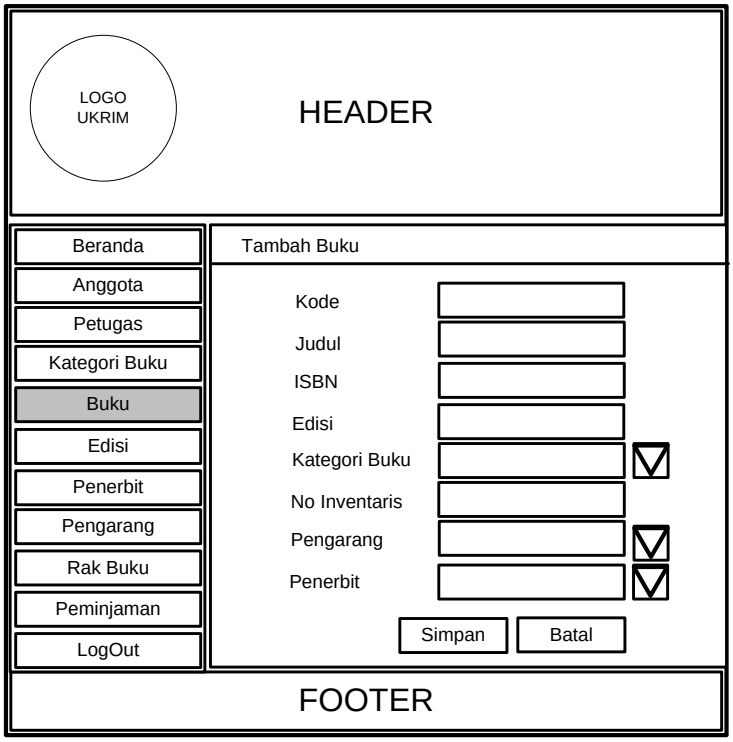
Gambar 2 Rancangan *Interface* Halaman Peta Lokasi



Gambar 3 Rancangan *Interface* Halaman Katalog



Gambar 4 Rancangan *Interface* Halaman Buku



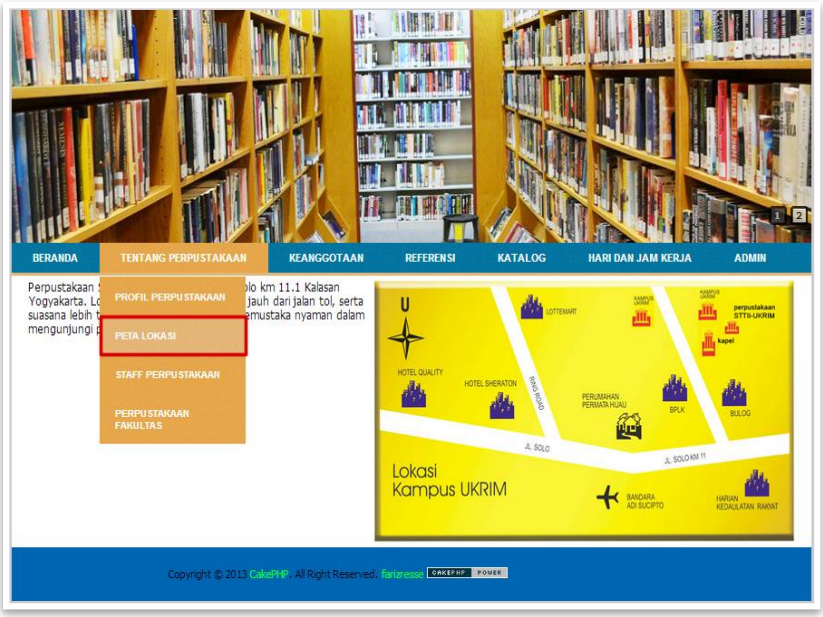
Gambar 5 Rancangan *Interface* Halaman Tambah Buku

Bagian C (Controller) akan menerima *request* dari *user*, apakah itu melalui *front end* mau pun *back end*, mengakses *database* yang telah didefinisikan, dan menampilkan *response* berdasarkan *view* yang telah didefinisikan. [2][3]

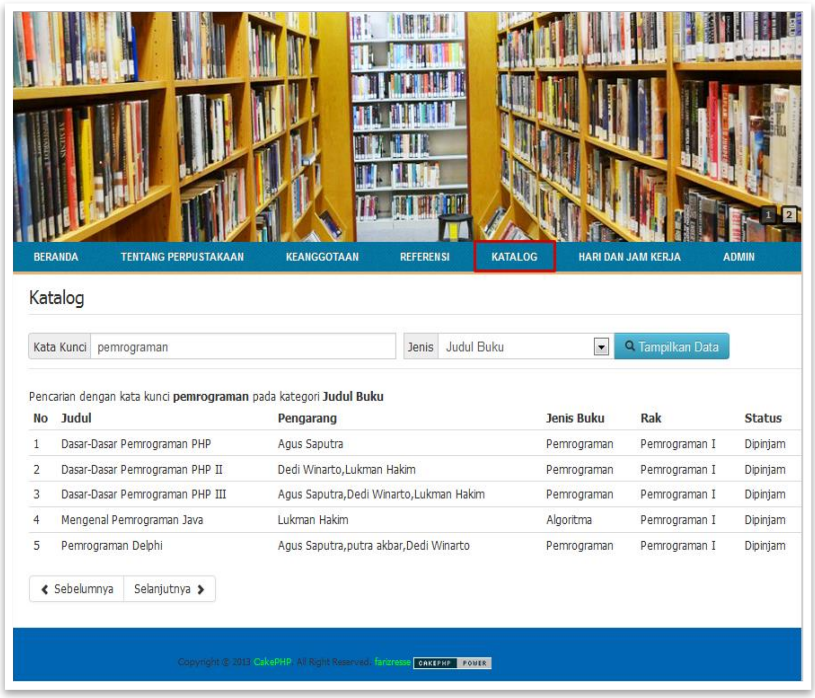
3. HASIL DAN PEMBAHASAN

Bagian *front end* menampilkan pertama-tama halaman Beranda. Dari halaman Beranda pengguna dapat bernavigasi untuk mendapatkan informasi tentang Profil Perpustakaan, Keanggotaan, Referensi, Katalog, Hari Jam Kerja, dan *administrator* dapat mengakses *back end*. Pengguna dapat melakukan navigasi ke halaman-halaman Profil Perpustakaan, Peta Lokasi, Staff Perpustakaan, dan Fasilitas Profil untuk mendapatkan informasi Profil Perpustakaan. Pengguna juga dapa bernavigasi ke halaman-halaman Syarat Keanggotaan dan Peminat Luar Kampus. Halaman Referensi menampilkan informasi tentang buku-buku rujukan (referensi). Dari halaman Katalog pengguna dapat mencari buku dengan memasukkan kata kunci atau judul buku, dan memperoleh informasi tentang buku, pengarang, jenisnya, rak penyimpanan, serta status apakah buku ada atau sedang dipinjam. Sebagai contoh implementasi dari rancangan *interface* halaman Peta Lokasi pada Gambar 2 dapat dilihat pada Gambar 6, dan implementasi rancangan halaman Katalog pada Gambar 3 terdapat pada Gambar 7.

Bagian *back end* diakses oleh *administrator* dari menu utama. Dari halaman Beranda Administrator ia dapat melakukan fungsi-fungsi CRUD pada Anggota, Petugas, Penerbit, Pengarang, Kategori Buku, Buku, Peminjaman dan Pengembalian, serta mendapatkan Laporan. Sebagai contoh, implementasi dari rancangan *interface* halaman Buku pada Gambar 4 dapat dilihat pada Gambar 8, sedangkan implementasi rancangan *interface* halaman Tambah Buku pada Gambar 5 dapat dilihat pada Gambar 9.

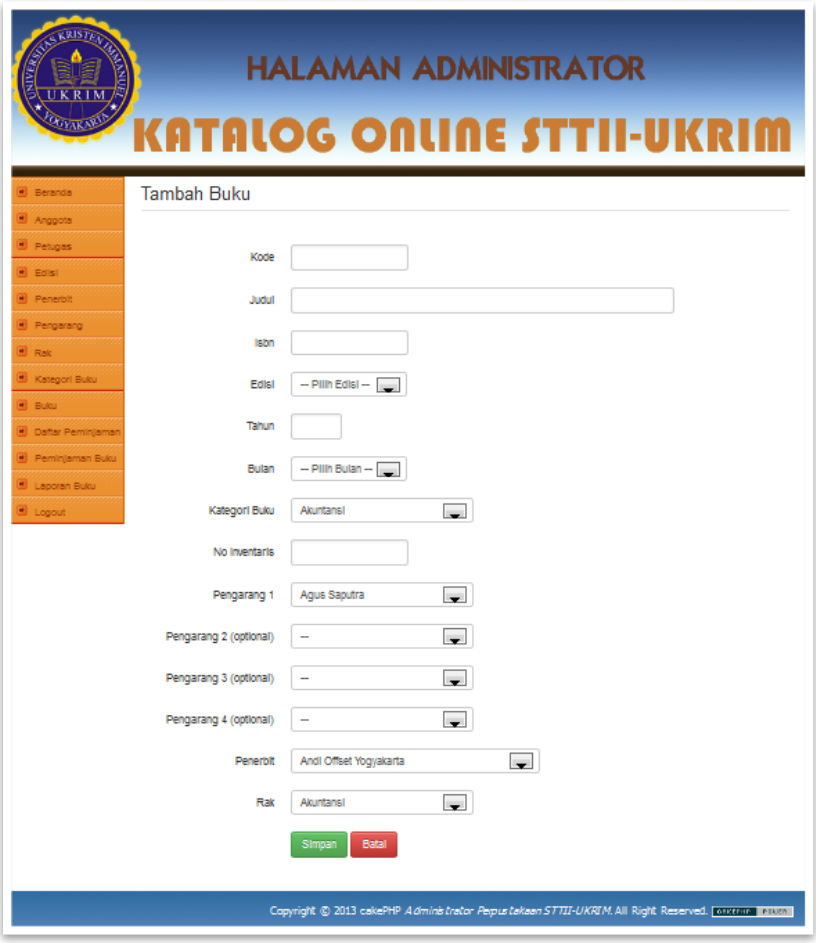


Gambar 6 Halaman Peta Lokasi



Gambar 7 Halaman Katalog

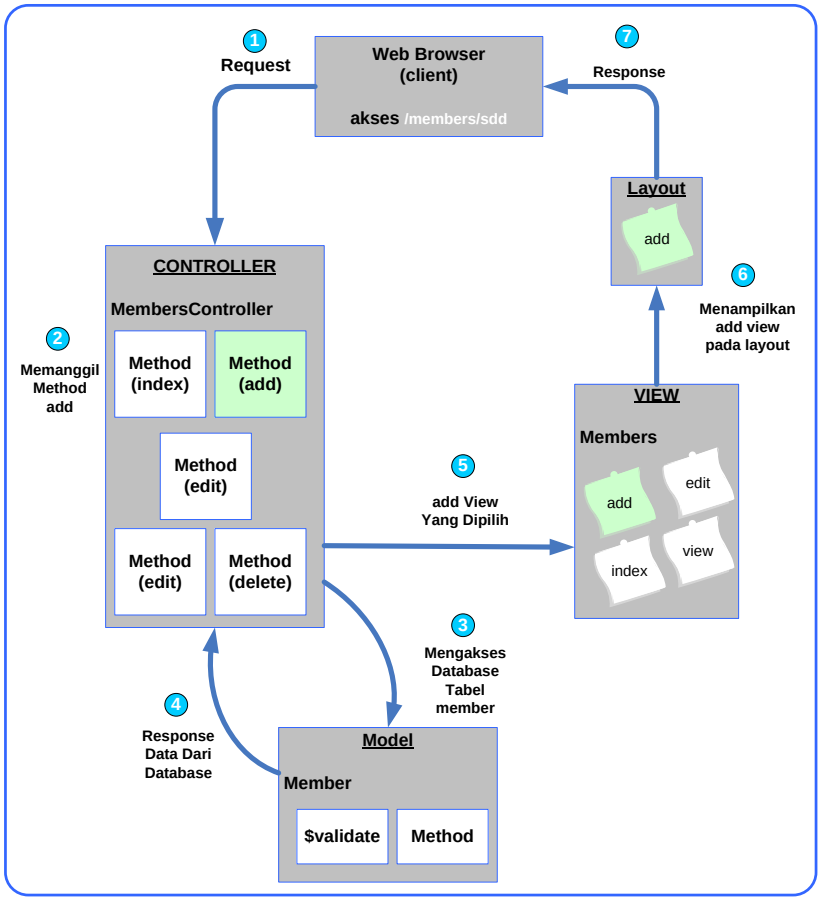
Gambar 8 Halaman Buku



Gambar 9 Halaman Tambah Buku

Pembahasan Cara kerja *Framework* CakePHP

Berdasarkan penelitian yang telah dilakukan, secara umum cara kerja *framework* CakePHP tidak terlepas dari konsep MVC pada CakePHP. Sebagai contoh, proses seorang administrator membuat *request* melalui halaman Administrator untuk melakukan CRUD pada tabel Anggota (members) dapat dilihat pada Gambar 10 berikut.



Gambar 10 Cara kerja *framework* CakePHP untuk CRUD pada "member"

Pertama-tama, melalui *web browser* client melakukan *request*, yang akan ditangani oleh *controller*. *Controller* memiliki banyak *method* (fungsi-fungsi) yang akan menangani *request* dan akan mengakses *method* sesuai dengan *request* dari *client*. *Controller* juga akan mengakses *database* melalui *model*. *Model* memberikan data yang relevan kepada *controller*. *Controller* akan mengolah data dan memberikannya pada *view* (tampilan) yang dipilih. Tampilan inilah yang akan di kirim ke *layout* sehingga dapat dilihat oleh *client* pada *browser*-nya.

Controller

Setiap *request* yang masuk dari *user*/*client* akan ditangani oleh *controller*. Didalam *controller* terdapat *function* / *method* yang berisi logika kerja aplikasi. *Controller* juga dapat berinteraksi dengan *Model* untuk mengkases *database.* Terlihat pada Gambar 10 *controller* menggunakan contoh *MembersController* dan mengakses *method add*. *Listing 1* di bawah merupakan potongan *source code method add* pada *MembersController*.

```
51      public function add() {
52          if ($this->request->is('post')) {
53              $this->Member->create();
54              if ($this->Member->save($this->request->data)) {
55                  $this->Session->setFlash(__('Sukses Tambah Data!'),'success_Message');
56                  $this->redirect(array('action' => 'index'));
57              } else {
58                  $this->Session->setFlash(__('Gagal Tambah Data! Silahkan Coba Lagi! '),
                    'error_Message');
59              }
60          }
61          $title_for_layout = 'Anggota';
62          $statusAnggota = $this->statusAnggota;
63          $this->set( compact('title_for_layout', 'statusAnggota'));
64      }
```

Listing 1 Function add pada MenbersController

- Pada baris 52, **if($this->request->is('post'))** menangkap *method post* dari view add.ctp.
- Pada baris 54, **$this->Member->save($this->request->data)** menyimpan data dalam *database* pada tabel *member*, serta dilakukan validasi ketika akan mengakses *database*.
- Pada baris 55, **$this->Session->setFlash(__('Sukses Tambah Data!'),'success_Message');** menampilkan pemberitahuan bahwa data telah berhasil disimpan.
- Pada baris 56, **$this->redirect(array('action'=> 'index'));** controller akan mengembalikan ke posisi *index*, dimana semua data ditampilkan kepada *client*.
- Pada baris 63,

$this->set(compact('title_for_layout','statusAnggota')); Mengirim parameter *title_for_layout* dan statusAnggota pada *view*.

Dalam *controller* juga terdapat *function-function* yang telah didefenisikan oleh CakePHP yang dapat di jalankan selain *function-function* yang di defenisikan sendiri. Beberapa *function* cakePHP yang dipakai dalam penelitian ini yaitu :

- ***beforeFilter(),*** *script* dalam *function* ini akan dijalankan terlebih dahulu sebelum mengakses logika pada *controller*.
- ***beforeRender(),*** *script* dalam *function* ini akan dijalankan setelah logika di *controller* dijalankan tetapi sebelum data dikirim ke *view*.
- ***afterFilter(),*** akan dijalankan setelah mengakses semua logika pada *controller* dan didalamnya telah ada data yang akan dikirim ke *view*.

```
8   class MembersController extends AppController {
9       public function beforeFilter(){
10          parent::beforeFilter();
11          $this->layout = 'admin';
12      }
```

Listing 2 Function *beforeFilter()* pada *MenbersController.php*

Listing 2 memuat potongan *source code* function *beforeFilter()* pada *MenbersController.php* di mana:

- baris 8, adalah *script* untuk membuat *class* MembersController.
- Baris 9, **function beforeFilter(){**, *function* ini akan dijalankan sebelum logika pada *controller* dijalankan.
- Baris 10, **parent::beforeFilter();** , adalah *script* dimana mengikuti defenisi dari *function beforeFilter()* yang ada dalam *AppController*.

- baris 11, **$this->layout = 'admin';,** mengatur layout admin ketika mengakses *function* ini.

Model

Model diakses ketika logika-logika di c*ontroller* akan mengakses *database*. Dalam *model* terdapat validasi juga *method-method* yang telah didefenisikan dalam CakePHP diantaranya *beforeValidate(), beforeFilter(), beforeFine(), afterFine(), beforeSafe(), afterSafe(), beforeDelete(), afterDelete()* dan lain sebagainya.

```
7  class Member extends AppModel {
8      public $validate = array(
9          'nama' => array(
10             'notempty' => array(
11                 'rule' => array('notempty'),
12                 'message' => 'Nama Tidak boleh kosong!',
13                 'allowEmpty' => false,
14                 'required' => true,
15             ),
16         ),
17         'kode' => array(
18             'role1' => array(
19                     'rule' => 'numeric',
20                     'message' => 'NIM Tidak Boleh Kosong! NIM Harus Numerik!',
21                     'allowEmpty' => false,
22                     'required' => false,
23             ),
24             'role2' => array(
25                     'rule' => 'isUnique',
26                     'message' => 'Nim Sudah Ada!!!',
27             )
28         ),
```

Listing 3 *Class Member* pada Member.php

Listing 3 memuat potongan *source code Class Member* pada Member.php di mana :

- Pada baris 7, **class public Member extends AppModel {** adalah *script* untuk membuat *model* baru dengan nama *class Member*.
- Pada Baris 8, **public $validate = array(** adalah *script* untuk membangkitkan validasi data.
- Pada Baris 9, **'nama'=> array(** merupakan nama *field* dari tabel members yang akan divalidasikan dalam form input maupun edit.
- Baris 10 – 14 , **'notempty' => array('rule' => array('notempty'), 'message' => 'Nama Tidak boleh kosong!', 'allowEmpty' => false, 'required' => true,**

adalah *syntax* yang sudah didefensikan oleh CakePHP dan fungsinya masing-masing.

Beberapa pengertian *modifier* yang sering dipakai dalam *model* :

- **Rule,** mendefinisikan metode validasi baik untuk nilai tunggal maupun *array*. Beberapa aturan validasi yang dipakai misalnya *alphaNumeric, boolean, date, fileSize* dan lain-lain.
- **Message,** menampilkan pesan error ketika terjadi kesalahan validasi.
- **Required,** operator ini menerima nilai boolean dimana jika di set *true* maka selalu dibutuhkan.
- **allowEmpty,** validasi untuk *field* pada tabel dalam *database*, boleh null atau tidak. Nilainya bersifat boolean, yaitu *true* dan *false.*

selain validasi data, berikut merupakan contoh dimana sistem akan menjalankan method *beforeSafe* sebelum *insert data* dalam *database.*

```
95      public function beforeSave($options = array()) {
96          if (!empty($this->data['Member']['tanggal_masuk']) &&
97          !empty($this->data['Member']['tanggal_lahir'])) {
98              $this->data['Member']['tanggal_masuk'] =
99              $this->dateFormatBeforeSave($this->data['Member']['tanggal_masuk']);
100             $this->data['Member']['tanggal_lahir'] =
101             $this->dateFormatBeforeSave($this->data['Member']['tanggal_lahir']);
102         }
103         return true;
104     }
105
106     public function dateFormatBeforeSave($dateString) {
107         return date('Y-m-d', strtotime($dateString));
108 }
```

Listing 4 *Function beforeSafe* pada Member.php

Listing 4 memuat potongan *source code Function beforeSafe* pada Member.php di mana:

- *function beforeSafe* akan dijalankan ketika akan *insert* data kedalam *database* atau *update* data dalam *database* pada tabel *members*.
- Baris 95, **public function beforeSave($options = array()){** adalah mendefinisikan *function beforeSafe* dengan parameter *$options.*
- Baris 96-101, adalah *script* untuk pengecekan *field* tanggal_masuk dan tanggal_lahir, jika tidak kosong maka tanggal diubah ke *format Y-m-d*.
- Baris 106, adalah *function dateFormatBeforeSafe* yang di panggil dalam *function beforeSafe*.

Selain validasi dan menggunakan *method-method* yang telah didefenisikan oleh cakePHP, dalam *model* juga didefenisikan relasi antara tabel dari *database*. Dengan mendefenisikan relasi antara tabel dalam *model*, maka sistem akan mengenal bahwa *model* yang terkait berelasi dengan *model* apa saja. Terdapat 4 jenis hubungan antara tabel yang di defenisikan dalam cakePHP, yaitu :

- *hasOne*, adalah relasi *one to one* antara tabel,
- *hasMany*, adalah relasi *one to many* antara tabel,
- *belongsTo*, adalah relasi *many to one* antara tabel, dan
- *hasAndBelongsToMany*. Adalah relasi *many to many* antara tabel.

```
10  class Loan extends AppModel {
11      public $belongsTo = array(
12          'User' => array(
13              'className' => 'User',
14              'foreignKey' => 'user_id',
15              'conditions' => '',
16              'fields' => '',
17              'order' => ''
18          ),
19          'Member' => array(
20              'className' => 'Member',
21              'foreignKey' => 'member_id',
22              'conditions' => '',
23              'fields' => '',
24              'order' => ''
25          ),
26      );
27      public $hasMany = array(
28          'LoanDetail' => array(
29              'className' => 'LoanDetail',
30              'foreignKey' => 'loan_id',
31              'conditions' => '',
32              'fields' => '',
33              'order' => ''
34          )
35      );
```

Listing 5 Contoh relasi antar tabel dalam class Loan.php

Listing 5 memuat potongan *source code* untuk contoh relasi antar tabel dalam class Loan.php di mana:

- baris 10, adalah mendefenisikan *class Loan* dengan *extends* dari *class AppModel*.
- Baris 11-26, **public $belongsTo = array(,** mendefenisikan relasi *many to one* tabel *loans* pada tabel *users* dan *members*.
- Baris 27-35, **public $hasMany = array(,**mendefenisikan relasi one to many tabel loans pada tabel loanDetails.
- **className,** adalah nama *class* yang terkait dengan *model* tersebut.
- **foreignKey,** adalah kunci asing yang ditemukan dalam *model* terkait.
- **Conditions,** adalah kondisi yang kompatibel atau string SQL.
- **Fields,** adalah daftar *field* yang akan diambil ketika data model yang terkait di akses.
- **Order,** adalah urutan klausa yang kompatibel atau string SQL untuk pengurutan data seperti *array* (Member.nama' => 'ASC').

View

View bertanggung jawab untuk menghasilkan *output* khusus yang diperlukan, yang diatur dari *controller*. Dalam penelitian ini *view* dapat berupa HTML dan JSON. Tipe file untuk *view* adalah .ctp (*cakePHP template*). *View* berisi *tag-tag* HTML yang mengikuti penulisan pada sistem *framework* cakePHP.

Berikut penulis akan membahas beberapa penjelasan tentang *view* di cakePHP berdasarkan penelitian ini.

```
1  <?php echo $this->Form->create('Member'); ?>
2      <?php
3          echo $this->Form->input('nama');
4          echo $this->Form->input('kode',array(
5              'class' => 'span2',
6              'label' => array(
7              'class' => 'control-label',
8              'text' => 'NIM / NIP')
9              ));
10     ?>
```

Listing 6 *Form* tambah data anggota di add.ctp

Listing 6 memuat potongan *source code form* tambah data anggota di add.ctp di mana:

- Baris 1, **<?php $this->Form->create('Member',array(,** adalah *script* untuk membuat *form* buka dan digunakan untuk mengirim data kedalam proses penginputan.
- Baris 3, berfungsi untuk membuat *textbox* dengan label nama.
- Baris 4-9, berfungsi untuk membuat *textbox* dengan beberapa *costum* tampilan dan dengan label NIM / NIP.
-

```
7      <table class="table  table-striped table-condensed">
8      <tr>
9          <th>No</th>
10         <th><?php echo $this->Paginator->sort('nama'); ?></th>
11         <th><?php echo $this->Paginator->sort('kode'); ?></th>
12         <th><?php echo $this->Paginator->sort('status'); ?></th>
13         <th class="actions"><?php echo __('Actions'); ?></th>
14     </tr>
15     <?php
16         $no = $this->Paginator->counter('{:start}');
17         foreach ($members as $member):
18     ?>
19     <tr>
20         <td><?php echo $no;?></td>
21         <td><?php echo h($member['Member']['nama']); ?> </td>
22         <td><?php echo h($member['Member']['kode']); ?> </td>
```

Listing 7 Tampilan data *member* pada index.ctp

Listing 7 memuat potongan *source code* untuk Tampilan data *member* pada index.ctp di mana:

- Baris 7, membuat tabel dengan *class* sesuai dengan yang didefinisikan di CSS.
- Baris 9-13, membuat tabel *header*, sesuai dengan nama *field* dari *database* dan dapat ditampilkan dengan pengurutan sederhana yaitu *ascending* dan *descending* menggunakan teknik *paginator*.
- Baris 16, mendefinisikan variabel $no, untuk nomor urut data.
- Baris 17, **foreach ($members as $member):,** adalah *script* untuk membuat seluruh *record* diubah menjadi pengulangan *record* secara berurutan satu per satu, dan akan di simpan dalam variabel **$member**. Hal ini sama dengan kode sql :

While ($member = mysql_fetch_array($members))
data-data;
}

- Baris 21 – 22, adalah *script* untuk menampilkan data berdasarkan *field* yang ditentukan.

Selain beberapa potongan *script* diatas, didalam *view* juga terdapat beberapa penanganan HTML yang mengikuti penamaan dari cakePHP seperti css, *style*, *link* dan sebagainya. Berikut beberapa contoh-contoh penggunaan kode HTML dalam cakePHP berdasarkan penelitian ini.

- **echo $this->Html->css(array('menu_admin','main'));** , adalah *script* yang digunakan untuk memanggil *file* css kedalam suatu *file view framework* cakePHP.
- **echo $this->html->link('Enter', '/pages/home', array('class'=>''button, 'target'=>'_blank')),** maka dari *script* tersebut akan menghasilkan kode html sebagai berikut : ** Enter .**
- **echo $this->form->create('post');** maka akan menghasilkan kode html sebagai berikut :

<form id="PostAddForm" method="post" action="/cakephp/ posts/add">. Selain itu, element method pada form mempunyai dua variabel yaitu *post* dan *get*. Teknik penulisan diatas merupakan *method post*, dan apabila menggunakan method *get* maka penulisannya sebagai berikut :

echo $form->create('post', array('type' => 'get'));

- **echo $this->form->input('nama');** maka akan menghasilkan kode html :

<input type="text" name="data[Post}[nama]">nama</input>

- **echo $this->form-submit();** maka akan menghasilkan kode html sebagai berikut :

<input type="submit" value="submit"></input>

Layout

Berdasarkan penelitian ini, *layout* adalah tempat dimana *view* yang telah dipilih oleh *controller* di tampilkan, atau *layout* merupakam *template* untuk *view*. Penelitian ini menggunakan dua *layout* yaitu *layout* untuk halaman pengunjung dan *layout* untuk halaman admin.

Isi dari *file layout* sama dengan *file view* dimana berisi kode-kode HTML yang berinteraksi dengan *file-file* lain seperti CSS, javaScript, *Element*, dan sebagainya.

Fitur-Fitur dalam *Framework* CakePHP

Berdasarkan hasil penelitian ini, dua buah fitur dari *framework* CakePHP yang digunakan dalam membangun sistem informasi di perpustakaan STTII-UKRIM Yogyakarta perlu kami tonjolkan di sini.

1. Fasilitas *Scaffolding*

Fitur ini di pakai penulis untuk membuat simulasi aplikasi CRUD, dimana hal ini berfungsi untuk menganalisa konten pada saat pertama kali, sebelum memulai membuat program. Dengan fitur ini juga penulis dapat melihat lagi apakah tabel-tabel di *database* sudah sesuai dengan yang diinginkan atau belum. Untuk menggunakan fasilitas ini, misalnya dengan menggunakan tabel anggota/*members* maka langkah-langkahnya adalah :

- Tabel anggota/*members* sudah harus tersedia.
- Membuat *controller* dengan nama *MembersController.php* seperti pada Listing 8. :

```php
<?php
    class MembersController extends AppController {
    var $scaffold;
    }
?>
```

Listing 8 *Scaffolding* dengan MembersController.php

- Setelah itu bisa langsung dicoba dengan membuka aplikasi http://localhost/nama_aplikasi/members.

2. Fasilitas *Bake*

adalah fitur CakePHP yang mampu meng-*generate model*,*view*, dan *controller* secara otomatis sesuai dengan *database* yang ada pada konfigurasi. Fasilitas ini cocok digunakan sebagai *template*, karena *bake* akan

otomatis menghasilkan kode - kode umum yang sering digunakan (kode CRUD) yang bisa diubah ataupun ditambahkan sesuai kebutuhan.

Keuntungan dalam menggunakan fasilitas ini adalah *developer* tidak harus memulai membuat *coding* dari awal, tetapi bisa melanjutkan dengan mengedit kode yang sudah di *generate* sehingga lebih efisien dalam penggunaan waktu pembuatan aplikasi.

Langkah-langkah menggunakan fasilitas *bake* dengan *bake console :*

- *Setting path* dengan cara *Klick Start* → *Control Panel* → *System and Security* → *System* kemudikan di sebelah kiri *klick Advanced System Setting*.
- Pada jendela *System Setting, klick Enviroments Variables*.
- Pada bagian *System Variables klick Path* dan *Edit*.
- Pada *Variable Value* Ditambahkan C:\xampplite\php dan C:\xampplite\htdocs\perpus\lib\Cake\Console, setelah itu klick OK sampai menutup semua jendela.
- Setelah itu, masuk kedalam kedalam jendela *Command Prompt* kemudian mengetikan '*cake'* dan apabila *setting path* berhasil maka akan menampilkan info *path* yang sedang di akses.
- Untuk meng-*generate* kode, maka harus masuk ke direktori aplikasi yaitu C:\xampplite\htdocs\perpus.
- Dengan perintah **Cake Bake All,** maka kode CRUD akan dibuatkan otomatis setelah memilih tabel-tabel apa saja yang mau di *generate* kode.

4. KESIMPULAN

Kami menyimpulkan beberapa kelebihan dan kekurangan *framework* CakePHP sebagai berikut.

1. Membangun sistem informasi berbasis *website* dengan menggunakan *framework* CakePHP dapat membantu *developer* karena CakePHP memiliki beberapa kelebihan, yaitu :

a. Membangun aplikasi dengan cepat menggunakan fasilitas *Bake.* Dengan fasilitas ini *developer* dapat memulai membuat aplikasi berbasis *website* tidak harus dari awal, tetapi cukup dengan membuat *database* dan mengkoneksikan ke *framework* CakePHP, maka fungsi CRUD dari setiap tabel dalam *database* dapat di-*generate* dengan menggunakan fasilitas *Bake*.

b. Mudah dalam mendefenisikan relasi antara tabel-tabel dalam *database,* yaitu hanya dengan mendefenisikan ke dalam *class model,* maka *framework* CakePHP akan langsung mengenal sebuah tabel berelasi dengan tabel apa saja.

c. Validasi data yang mudah dan terintegrasi, yaitu dengan pendefinisian pada setiap *field* tabel di *database* pada setiap *class* di *model.*

d. Mendukung AJAX dan ORM, *library* AJAX di dalam *framework* CakePHP sudah disediakan, dan *developer* tinggal memakai dan mudah dalam penggunaannya.

2. *Framework* CakePHP juga memiliki beberapa kekurangan, yaitu:

a. Dokumentasi yang belum terpadu. CakePHP adalah *framework* yang selalu berkembang, dan setiap versi memiliki dokumentasi masing-masing, serta *syntax-syntax* yang berbeda dengan versi sebelumnya.

b. Hingga saat ini, *release* terbaru dari *framework* CakePHP belum men-*support* i18n, atau internasionalisasi bahasa-bahasa yang ada di dunia, sehingga mengharuskan *developer* yang berasal dari berbagai negara untuk mempelajari *framework* ini hanya dengan bahasa inggris.

c. *Framework* CakePHP memiliki banyak aturan yang harus di terapkan dalam aplikasi, seperti penamaan untuk setiap tabel dalam *database* yang harus plural, nama *controller* harus sama dengan nama *folder* untuk *view*, nam*a function* di *controller* harus sama dengan nama *file* di *view*.

d. Butuh waktu yang lebih lama untuk menguasai *framework* CakePHP, karena *framework* ini memiliki *syntax* tersendiri yang telah didefinisikan.

e. Fasilitas Bake hanya membantu developer pada tahap awal, yaitu pembuatan fungsi CRUD dari setiap tabel di *database*. Sedangkan untuk tahap selanjutnya seperti melakukan validasi, menentukan layout, mengatur CSS dan hal-hal lain yang berhubungan dengan sistem yang dibangun, harus dilakukan secara manual / *coding* manual.

5. SARAN

Karena implementasi *framework* CakePHP dalam penelitian ini masih tergolong dalam pemgimplementasian pada aplikasi yang berskala keci, perlu dilakukan pengujian *framework* CakePHP dengan membangun aplikasi dengan skala yang lebih besar.

DAFTAR PUSTAKA

[1] Anderson, J.;& Master, L.E. (ed), 2006a, *CakePHP Programmer's Reference Guide.*USA : CakePHP Software Foundation, Inc. 141 p.

[2] Anderson, J.;& Master, L.E. (ed), 2006b, *CakePHP-API Documentation version 1.1.8.3544.* USA : CakePHP Software Foundation, Inc.

[3] Saputra, Agus, 2011, *Teknik Cepat Membangun Aplikasi Web dengan Framework CakePHP,* Yogyakarta : Lokomedia.

[4] Saputra, Agus, 2012, *Proyek Membuat Web Profesional dengan Framework CakePHP,* Yogyakarta : Lokomedia.

[5] Sunarfrihartono, Bimo, 2006, *Makalah Kuliah Analisis dan Perancangan Sistem Informasi : Pengembangan Sistem Informasi,* Yogyakarta : Teknik Elektro UGM. 22h.