

# Relatório desenvolvimento do sistema votaMinas

Leandro G. da Silva<sup>1</sup>, Lucas C. de Lima<sup>1</sup>, Pablo A. Silva<sup>1</sup>

<sup>1</sup>Instituto de Ciências Exatas e Aplicadas – Universidade Federal de Ouro Preto (UFOP)  
Departamento de Engenharia de Computação e Sistemas de Informação  
João Monlevade, MG – Brasil

leandrogs99@gmail.com, lucaslima2004@gmail.com, phsil@gmail.com

**Resumo.** Neste documento será apresentado a documentação do trabalho prático da disciplina Sistemas WEB I, onde foi construído um site para consulta de relação de votos de candidatos (dados fictícios) assim como o cadastro de usuários para disponibilizar a opção de votar em algum dos candidatos listados. Para a criação do mesmo foi utilizado o framework CakePHP.

## 1. Introdução

Pesquisa de intenção de votos são frequentemente realizadas para se obter estimativas sobre qual candidato possivelmente irá ser eleito, estas pesquisas muitas vezes são realizadas nas ruas perguntando a cidadãos normais sobre sua opinião. Em 2014 ocorreu mais uma vez as votações para presidente assim como os demais cargos e, aproveitando deste contexto, foi proposto aos alunos que criassem um site onde fosse possível à um usuário comum votar e consultar os resultados atuais de uma pesquisa online. A utilização de sistemas web para este fim facilita a realização de tal pesquisa, uma vez que está seria muito melhor difundida na internet, além da praticidade da consulta em tempo real dos resultados parciais.

Este projeto utiliza de linguagens bastante conhecidas no meio, tais como: HTML/CSS, PHP e JavaScript. Para o desenvolvimento do website foi utilizado também a ferramenta CakePHP, um framework robusto, gratuito e de código aberto em PHP para desenvolvimento ágil. Tem como objetivo permitir que o usuário trabalhe em uma estrutura que possa programar de forma rápida e sem perda de flexibilidade.

A ferramenta CakePHP (ou apenas Cake) utiliza da arquitetura MVC (model-view-controller) que separa a representação da informação da interação do usuário com ele através das três partições: (i) Model, que consiste nos dados da aplicação, regras de negócios, lógica e funções; (ii) View, que pode ser qualquer saída de representação dos dados, como uma tabela ou um diagrama e (iii) Controller, que é quem faz a mediação da entrada, convertendo-a em comandos para o modelo ou visão.

## 2. Implementação

Antes de iniciar a implementação, foi discutido, analisado e estruturado os requisitos da página. Foi criado um documento de levantamento de requisitos (enviado junto com este documento) especificando quais funcionalidades estariam disponíveis no sistema, e a classificação de suas prioridades em: essencial, importante e desejável. Em resumo, para usuários o site contém operações como:

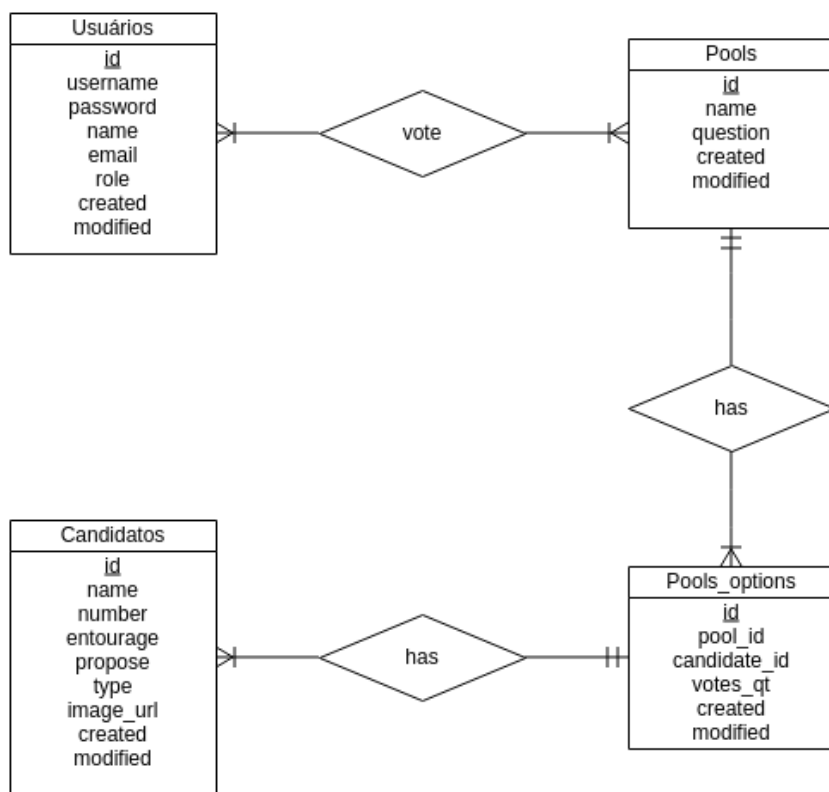
- Cadastro: Nome, Email, Senha;

- Autenticação: controlado via login e senha;
- Cadastro de votos: registro de votos por usuários autenticados;
- Visualização de resultado parcial: visualizar os resultados;

O sistema também possui o modo administrador, onde é possível ser feito:

- Cadastro de candidato;
- Edição e remoção de candidato;
- Cadastro de pesquisa eleitoral;

Visto a necessidade clara de armazenamento de tais dados foi criado um banco de dados MySQL hospedado em um servidor local, que pode ser conferido através do diagrama entidade-relacionamento da figura 1.



**Figura 1. Diagrama ER do banco de dados.**

Para que o cakePHP consiga ler o banco é necessária a criação do seguinte usuário no mysql:

Usuário: 'VMG\_user'

O arquivo de configuração do banco no cakephp possui a seguinte senha:

Senha: 'Fxen567Q'

Portanto é necessário que o usuário no banco também possua essa senha ou que ela seja alterada em ambos lugares. Além disso, antes de importar o banco de dados é preciso criar uma tabela obrigatoriamente com o nome:

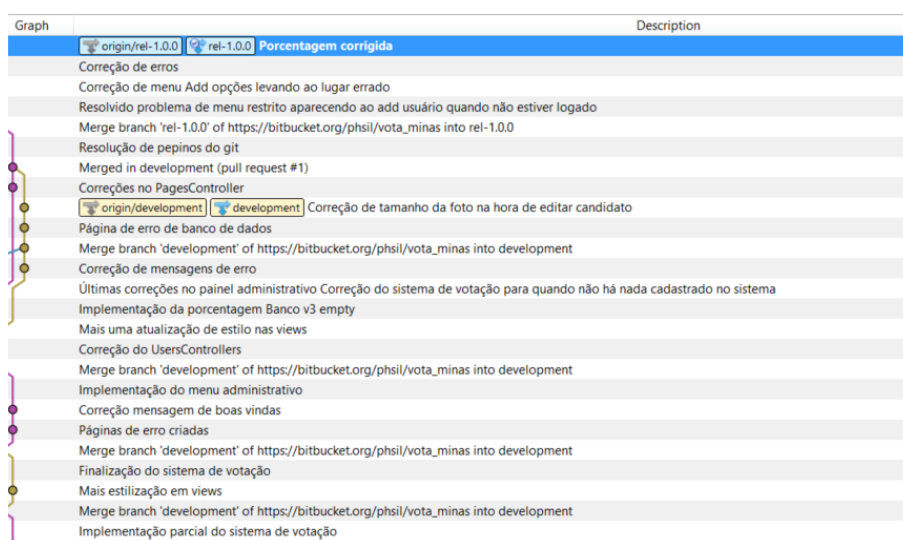
## vota\_minas

para que a importação do banco seja bem sucedida.

Após a documentação das idéias foi necessário aprender a utilizar a nova ferramenta CakePHP. Através do site oficial do cake foi possível encontrar a teoria e tutoriais para desenvolvimento. Em primeiro momento foi realizado o simples exemplo do blog fornecido pelo site, e assim, com algumas dificuldades, foi realizado a adaptação para o contexto do nosso problema. Alterações e criações de novos arquivos foram necessários para que fosse possível esta adaptação.

Outro recurso fornecido pelo CakePHP, é o BakePHP. Com este é possível automaticamente gerar formulários utilizando como base algum banco de dados que esteja dentro das normas estabelecidas pelo Cake. Este recurso foi utilizado no projeto para geração do modelo básico do formulário de cadastro de usuário.

Durante todo o desenvolvimento foi utilizado o sistema de controle versão Git (Bitbucket), um software livre distribuído sob os termos da versão 2 da GNU (General Public License). Parte das atividades realizadas podem ser vistas na figura 2



**Figura 2. Linha do tempo de ações no Bitbucket.**

A interface da página é simples e de fácil utilização, possui poucos links com uma organização hierárquica. Para o design da página foi escolhido o tema da bandeira brasileira, apresentando fortes cores em amarelo e verde, simples, porém agradável. A página também oferece um design responsivo (figura 3), responsável pelo ajuste dos elementos presentes à medida em que o tamanho da tela se modifica, não sendo necessário implementar um novo site para cada tipo de dispositivo.

## 2.1. Implementação técnica

### 2.1.1. Mas por que usar um framework?

Simples, para não ficar reinventando a roda. Não precisamos ficar escrevendo aquelas velhas “queries” de “insert”, “update”, “select” e todas aquelas funções básicas que todo



**Figura 3. Layout responsivo.**

sistema CRUD tem. Usamos um framework para acelerar nosso desenvolvimento deixando de lado todo aquele código chato que fica se repetindo em quase todo sistema e passamos a nos preocupar apenas com a regra de negócio do projeto. CakePHP tem uma licença livre, é de código aberto, o que é ideal para qualquer tipo de projeto seja um freelance ou um projeto da sua empresa. Tem um objetivo claro de tornar o desenvolvimento organizado e rápido sem tirar flexibilidade.

Alguns motivos para usar o CakePHP, de acordo com o manual do mesmo. • Comunidade ativa e amigável • Licença flexível • Compatível com o PHP 5.2.6 e superior • CRUD integrado para interação com o banco de dados • Scaffolding para criar protótipos • Geração de código • Arquitetura MVC • Requisições feitas com clareza, URLs e rotas customizáveis • Validações embutidas • Templates rápidos e flexíveis (Sintaxe PHP, com helpers) • Helpers para AJAX, JavaScript, formulários HTML e outros • Componentes de Email, Cookie, Segurança, Sessão, e Tratamento de Requisições • Controle de Acessos flexível • Limpeza dos dados • Sistema de Cache flexível • Funciona a partir de qualquer diretório do website, com pouca ou nenhuma configuração do Apache.

### **2.1.2. CakeBake**

Todo bom framework deve oferecer, de alguma forma, a possibilidade de gerar código, e o CakePHP como bom framework que é, nos fornece o Cake Bake, que é uma aplicação console que faz uma boa parte do trabalho para nós. Gerando um código muito organizado e limpo, inclusive com comentários nas funções, ele constrói as ações do CRUD e suas views, como também mensagens de erros e sucessos. Embora essas mensagens sejam criadas em inglês, traduzi-las será pouco a fazer comparado ao tempo que economizamos. O CakeBake é o nosso “padeiro”, ou seja, é uma aplicação console para gerar código

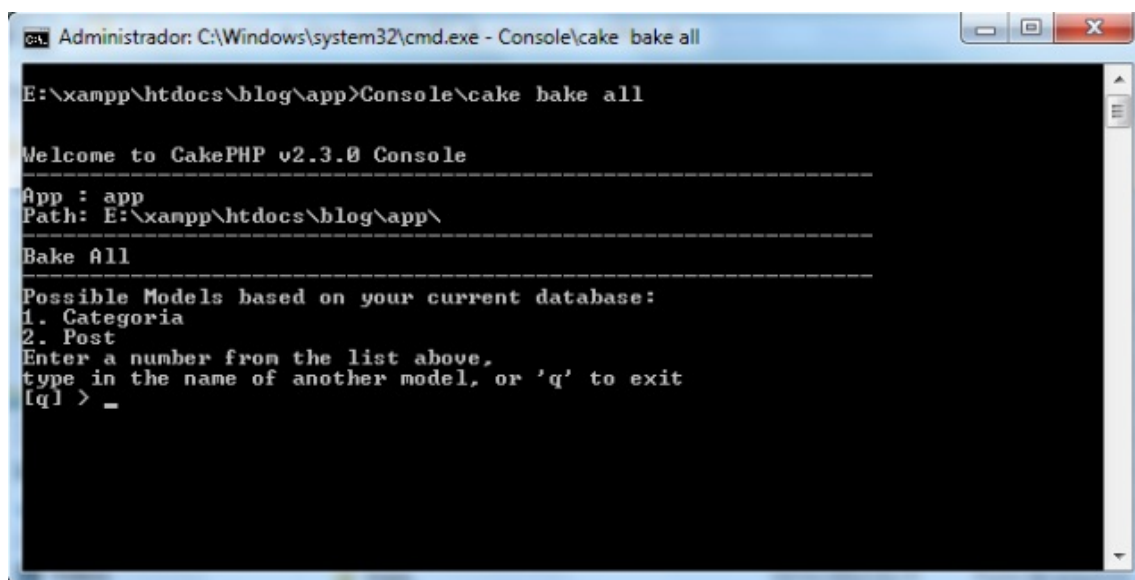
PHP no framework CakePHP, que pode ser muito útil quando estamos começando uma aplicação do zero, já que precisamos que nossas tabelas sigam as normas do framework. Para poder utilizar o Bake basta seguir as convenções de criação das tabelas no banco de dados, que incluem: nomes em inglês e no plural. Para autenticação, a tabela de usuário deve ter as colunas de username, password e role. As tabelas created e modified são desejáveis, mas não obrigatórias. Tendo configurado o banco desta maneira, só precisamos abrir o terminal na pasta Console do projeto que fica dentro de app e digitar o seguinte comando

---

```
Cake bake
```

---

e em seguida aparecerá um menu com as opções de criação, algo como mostrado na figura 4.



```
Administrador: C:\Windows\system32\cmd.exe - Console\cake bake all

E:\xampp\htdocs\blog\app>Console\cake bake all

Welcome to CakePHP v2.3.0 Console
-----
App : app
Path: E:\xampp\htdocs\blog\app\
-----
Bake All
-----
Possible Models based on your current database:
1. Categoria
2. Post
Enter a number from the list above,
type in the name of another model, or 'q' to exit
[q] > _
```

**Figura 4. CakeBake.**

Após realizar esse processo, o CRUD básico do seu processo terá sido gerado automaticamente.

### **2.1.3. Detalhes técnicos da implementação – Funcionalidades avançadas**

O Bake é um mecanismo ótimo, mas bem limitado. Não há automaticidade para o caso por exemplo se queremos criar um formulário que aceita upload de arquivos ou fotos, a não ser utilizando o helper Form para indicar o tipo deste formulário na parte da view, que iremos falar mais adiante, mas ainda caímos na parte do back end que deve ser programada manualmente. A seguir serão discutidas as principais funcionalidades consideradas de nível mais avançado de um sistema e que são necessárias para o funcionamento correto do mesmo.

#### 2.1.4. Componentes do cakephp

Componentes são pacotes de dados, tais como dados do usuário, que são compartilhados entre controllers. O cake já vem com uma grande quantidade de componentes implementados. Foram utilizados componentes para Session e Autenticação do sistema.

#### 2.1.5. Sessions

Sessão é um recurso do PHP que permite que você salve valores (variáveis) para serem usados ao longo da visita do usuário. Valores salvos na sessão podem ser usados em qualquer parte do script, mesmo em outras páginas do site. São variáveis que permanecem setadas até o visitante fechar o browser ou a sessão ser destruída. Você precisa iniciar a sessão antes de poder setar ou pegar valores dela. Não há limite de valores salvos na sessão. A sessão é pessoal de cada visitante. Quando um visitante acessa o site, é gerado um cookie no computador dele informando um id único de sessão e o PHP usa esse identificador pra ‘organizar’ as sessões entre os visitantes do seu site. Mas esse cookie tem validade apenas enquanto o browser estiver aberto. Para o caso do nosso sistema, a session ainda possibilitou que restrições de uso fossem implementadas baseadas no nível que o usuário na session corrente tem, uma vez que essas informações são armazenadas somente quando o usuário está logado no sistema, ou seja, em uma session. A implementação de uma session no CakePHP é extremamente simplificada utilizando-se o componente Session. O componente Session do CakePHP oferece uma maneira de persistir dados do cliente entre requisições de páginas. Este componente funciona como um encapsulador para a variável `$_SESSION` além de oferecer métodos de conveniência para diversas funções relacionadas ao `$_SESSION`. O componente precisa ser adicionado manualmente no seu AppController para ser então herdado por todos os outros controllers. A implementação do referido componente pode ser visualizada abaixo:

---

```
public $components = array(
    'Session',
    'Auth' => array(
        'loginRedirect' => array('controller' =>
            'pages', 'action' => 'index'),
        'logoutRedirect' => array('controller' =>
            'pages', 'action' => 'index'),
        'authError' => 'You cant access that page',
        'authorize' => array('Controller')
    )
);
```

---

Como pode ser visto, somente incluir a palavra ‘Session’ como um parâmetro para o componente já irá habilitar nossa session automaticamente e só precisaremos configurar o campo de login e logout para que tudo já esteja funcionando. A implementação do login e logout são actions no UserController:

---

```

public function login(){

    if($this->request->is('post')){

        if($this->Auth->login()){
            $this->redirect($this->Auth->redirect());
        }
        else{
            $this->Session->setFlash('Nome de usuario ou
            senha incorretos!');
        }
    }

    public function logout(){

        $this->redirect($this->Auth->logout());

    }

}

```

---

### 2.1.6. Autenticação

O sistema de autenticação usa os dados da session para controlar quem pode logar no sistema e qual parte do sistema está reservada a que nível de usuário. Utilizando-se dessa lógica todos os controler possui o seguinte código:

---

```

public function isAuthorized($user){
    if($user['role'] == 'admin'){
        return true;
    }
    return false;
}

```

---

em que o código demonstrado pertence a classe AppController e é dele que vem toda a lógica de autenticação. Como pode ser visto, o método só retorna true se o usuário tiver o nível de administrador, do contrário retorna false. Isso faz com que todas as actions estejam bloqueadas para usuários de nível regular e que todas as actions estejam liberadas para usuários com nível de administrador. Dessa maneira fechamos o sistema e só precisamos liberar para os usuários aquelas actions que são realmente necessárias para o usuário comum e que não tem funcionalidades de um administrador do sistema. Por exemplo, para liberarmos a action de logout para o usuário, somente precisamos colocar o seguinte código no UsersController:

---

```

public function isAuthorized($user){

```

```
if (!parent::isAuthorized($user)) {  
  
    if ($this->action === 'logout') {  
        return true;  
    }  
  
    return $this->redirect(array('controller' =>  
        'pages', 'action' =>  
        'not_authorized_page'));  
}  
else{  
    return true;  
}  
}
```

---

Que verifica se o usuário está tentando acessar a action logout e se sim, dá autorização para o mesmo. Note que se tiver tentando acessar uma action não autorizada, o usuário será enviado para uma página de erro. Para liberar algumas actions antes que o usuário faça login, como por exemplo a action login e a action add (que é utilizada para que o usuário se cadastre no sistema), utilizamos o método beforeFilter da seguinte maneira:

---

```
public function beforeFilter(){  
  
    parent::beforeFilter();  
  
    if(!$this->Auth->loggedIn()){  
        $this->Auth->allow('login', 'add');  
    }  
}
```

---

e esse método pode ser utilizado em todos os controllers para liberar actions para o usuário que não está logado. Útil para mostrar o conteúdo do site.

### 2.1.7. Upload de imagens

Um mecanismo complicado encontrado foi o de upload de imagens. Para fazer upload de imagens é preciso utilizar o Helper Form da seguinte maneira:

---

```
<?php echo $this->Form->create('Candidate', array('type' =>  
    'file')); ?>
```

---

que indica que o formulário a ser criado tem que ser preparado para receber um arquivo. Em seguida a opção que vai receber a imagem também deve conter essa marcação:

---



```
echo \$this->Form->input('image\_url', array('label' => 'Foto',
      'type' => 'file'));
```

---

Nesse caso, o que vamos armazenar no banco de dados será a url da imagem, que ficará salva na pasta UPLOADS dentro do nosso WEB ROOT. A parte do backend é mais complicada e utiliza uma extensa etapa de validações para garantir a consistência dos dados, veja a seguir:

---

```
/**
 * add method
 *
 * @return void
 */
public function add() {
    if ($this->request->is('post')) {
        if($this->request->data['Candidate']){
            $image = $this->request->data['Candidate']['image_url'];
            $imageTypes = array("image/gif", "image/jpeg", "image/png");
            $uploadFolder = "upload";
            $uploadPath = WWW_ROOT . $uploadFolder;

            foreach ($imageTypes as $type) {
                if($type == $image['type']){
                    if($image['error'] == 0){
                        $imageName = $image['name'];

                        if(file_exists($uploadPath . '\\' . $imageName)){
                            $imageName = date('His') . $imageName;
                        }

                        $full_image_path = $uploadPath . '\\' . $imageName;

                        if(move_uploaded_file($image['tmp_name'], $full_image_path)){
                            $this->request->data['Candidate']['image_url'] = $uploadFolder
                                . '\\' . $imageName;
                            // debug($this->request->data) or die();
                            $this->Candidate->create();
                            if ($this->Candidate->save($this->request->data)) {
                                $this->Session->setFlash(__('The candidate has been saved.'));
                                return $this->redirect(array('action' => 'index'));
                            }
                        }
                        else{
                            $this->Session->setFlash(__('The candidate could not be saved.
                                Please, try again.'));
                        }
                    }
                }
            }
            else{
                $this->Session->setFlash('There was a problem uploading file.
                    Please try again.');
```

```

}
}
else{
$this->Session->setFlash('Error uploading file.');
```

---

```

}
}
else{
$this->Session->setFlash('Unacceptable file type');
```

---

```

}
}
}
}
}
}
}

```

Note que o path para inserção da imagem na pasta upload utiliza uma constante do cakePHP que é a WWW\_ROOT. Essa constante tem o retorno do caminho do web root dentro da nossa aplicação. Além disso, temos a etapa de verificação para o tipo do arquivo, ou seja, se ele é realmente uma imagem e por último utilizamos a validação padrão para saber se os dados foram salvos no banco de dados. Ainda é importante notar que depois de fazer upload da imagem, trocamos o parâmetro image\_url pela url onde a imagem foi salva e não mais utilizamos o nome da imagem como parâmetro.

### 2.1.8. Helpers

Aqui está uma definição do próprio manual do cake: “Helpers são componentes para uma apresentação em camadas de sua aplicação. Eles contém apresentações lógicas que podem ser compartilhadas entre views, elements, ou layouts. Neste capítulo mostraremos para você como criar seu próprio helper e o básico do core helpers do CakePHP” Em resumo, os Helpers nos ajudam, na camada de View, a criar componentes HTML usando funções em PHP. Isso torna o desenvolvimento mais rápido e organizado. O CakePHP já traz em seu core Helpers muito importantes, como, por exemplo, O FormHelper e o HtmlHelper, esses já vêm habilitados para uso. Ele traz, entre outros, o TimeHelper e o NumberHelper que são utilizados para formatar data e números respectivamente. Esses não vêm habilitados. Para habilitar um helper podemos fazer de três formas, são elas: • No AppController, assim o helper fica disponível para toda a aplicação; • Em um controlador específico, dessa forma o helper fica disponível para todas as ações do controller; • Em uma função para que fique visível apenas para a view respectiva. Para a nossa aplicação, utilizamos os helpers Html, Form, Javascript e Votation. Html e Form são os mais básicos do cake e extensamente utilizados. Uma funcionalidade que demandou a utilização do helper Javascript foi quando queríamos adicionar candidatos as enquetes e como os candidatos estão todos juntos na tabela candidatos, ficaria inviável mostrar todos os candidatos para o administrador se ele quisesse por exemplo criar uma enquete do tipo presidente. Por isso foi implementado um mecanismo do tipo Estado/Cidade muito visto em sites quando no momento em que clicamos em um estado, o select de cidades é preenchido com cidades daquele estado selecionado. Para o nosso caso, quando o usuário seleciona um tipo de enquete, o select de candidatos mostra somente candidatos daquele

tipo. O helper utilizado pode ser visto a seguir:

---

```
<?php
    $this->Js->get('#PoolsOptionPoolId')->event(
        'change',
        $this->Js->request(
            array(
                'controller'=>'candidates',
                'action'=>'getByType'
            ),
            array(
                'update'=>'#PoolsOptionCandidateId',
                'async' => true,
                'method' => 'post',
                'dataExpression'=>true,
                'data'=> $this->Js->serializeForm(
                    array(
                        'isForm' => true,
                        'inline' => true
                    )
                )
            )
        )
    );
?>
```

---

O evento requerido é o “change” e o requeste é feito através de um http request do tipo post que será processado pelo CandidateController e enviará uma resposta com os candidatos.

### 2.1.9. Posso criar helpers?

Se não há um helper para atender a sua necessidade específica, então você pode criar um. Para isso basta criar uma classe na pasta Helper como a que segue:

---

```
<?php

class VotationHelper extends AppHelper{

    public $helpers = array('Html');

    public function makeCustomRadios($candidates){

        $customRadioButtons = ''; //<div class="contentList">';
        $customRadioButtons .= '<input type="hidden"
            name="data[Vote][candidate_id]" id="VoteCandidateName_"
            value="">';
    }
}
```

```

foreach ($candidates as $key => $value) {
$customRadioButtons .= '<div class="contentList">';
$customRadioButtons .= '<input type="radio"
    name="data[Vote][candidate_id]"
    id="VoteCandidateName' . $value['Candidate']['id'] . '" required
    value="" . $value['Candidate']['id'] . '">';
// $customRadioButtons .= '';
$customRadioButtons .= '<label for="VoteCandidateName' . $key .
    '"> ' .
' ' .
'<p class="radioName">Candidato: ' .
    $value['Candidate']['name'] . '</p> ' .
'<p class="radioEntourage">Partido: ' .
    $value['Candidate']['entourage'] . '</p> ' .
'<p class="radioNumber">Numero: ' .
    $value['Candidate']['number'] . '</p> ' .
'</label>';
$customRadioButtons .= '</div>';
}

// $customRadioButtons .= '</div>';

return $customRadioButtons;
}
}

```

---

Foi preciso criar uma helper porque o helper Form não permite colocar mais de um label entre radio buttons. Sendo assim a função do nosso helper e fazer isso. Para utilizar o helper criado basta chamá-lo no local desejado da seguinte maneira:

---

```

echo $this->Votation->makeCustomRadios($candidates);

```

---

### 2.1.10. O Sistema de votação

O Sistema de votação utiliza mais de um modelo dentro de seu controler. O sistema de votação segue o seguinte fluxo: - Verifica se tem enquetes cadastradas para aquele tipo de candidato - Se tem verifica se tem candidatos cadastrados para aquele tipo de enquete - Se sim, identifica se o usuário já votou naquela enquete. - Se o usuário já votou, bloqueia a tela e mostra um link para ir diretamente aos resultados parciais. - Se não mostra os candidatos disponíveis para a votação. - Permite que o usuário escolha somente um candidato. - Depois que o voto é enviado redireciona o usuário para a mesma página e avisa que ele não pode mais votar naquela enquete. - Em nenhum nível de usuário é possível alterar nenhum dos resultados das enquetes. - O sistema é blindado a fraudes e o

administrador somente conseguiria fazer isso acessando o banco de dados diretamente.

### 2.1.11. Validações

As validações são feitas pelos modelos e de simples implementação. Para criar uma validação é preciso somente alterar o código do modelo. Por exemplo, para que um candidato não possa ser adicionado sem nome, somente precisamos definir a seguinte regra no Candidate Model:

---

```
public $validate = array(
    'name' => array(
        'notEmpty' => array(
            'rule' => array('notEmpty'),
            //'message' => 'Your custom message
            here',
            //'allowEmpty' => false,
            //'required' => false,
            //'last' => false, // Stop validation
            after this rule
            //'on' => 'create', // Limit validation
            to 'create' or 'update' operations
        ),
    ),
);
```

---

A regra notEmpty não permite que sejam cadastrados candidatos sem nome. O Helper Form utiliza essas regras para geração do formulário HTML e já implementa tais validações automaticamente.

### 2.1.12. Proteções contra erros

O sistema possui proteções contra os seguintes erros:

404 Page Not Found 401 Not Authorized 400 Bad Request 500 Internal Server Error Erro de banco de dados

Foram implementadas views específicas para cada erro que mostram uma mensagem para o usuário avisando-o sobre tais erros.

### 2.1.13. O painel administrativo

A implementação do painel administrativo se aproveita das sessions e autenticação que permite somente que usuários com nível de administrador acessem determinadas páginas. O menu geral do painel administrador está em todas as views de nível de administrador e foi implementado utilizando o helper HTML como se segue:

---

```
<div class="actions">
    <ul>
```

```
<li><?php echo $this->Html->link(__('Listar
    enquetes'), array('controller' => 'pools',
    'action' => 'index')); ?> </li>
<li><?php echo $this->Html->link(__('Listar opcoes
    em enquete'), array('controller' =>
    'poolsoptions', 'action' => 'index')); ?> </li>
<li><?php echo $this->Html->link(__('Listar
    usuarios'), array('controller' => 'users',
    'action' => 'index')); ?> </li>
<li><?php echo $this->Html->link(__('Listar
    candidatos'), array('controller' =>
    'candidates', 'action' => 'index')); ?> </li>
<li><?php echo $this->Html->link(__('Listar
    votos'), array('controller' => 'votes', 'action'
    => 'index')); ?> </li>
</ul>
</div>
```

---

É basicamente um menu com as principais actions que permitem realizar operações de CRUD no sistema. Ao se clicar em uma das opções disponibilizadas um submenu específico para cada tipo de action é mostrado, como por exemplo, na view Candidates, um submenu para adicionar candidatos irá aparecer. O painel administrativo pode ser acessado, após um usuário com nível de administrador fazer login. Nesse momento uma opção irá aparecer uma opção na página inicial que permite que usuário vá para o menu de administrador.

O restante da implementação pode ser encontrado no código fonte, que está todo comentado com as principais funções.

### 3. Testes de integração

#### 3.1. Funcionalidades com resultados esperados

Nesta etapa, todo o sistema passou por testes para verificar a presença de erros ou a correta funcionalidade do sistema como um todo. Nem todas as etapas estão demonstradas com imagens, mas os testes puderam garantir o correto funcionamento das funcionalidades implementadas.

##### 3.1.1. CRUD para candidatos

A figura 5 mostra o formulário apresentado ao administrador para cadastrar um novo candidato. A resposta desta operação após o pressionamento do botão “Cadastrar” se encontra na figura 6.

Também foram testadas a exclusão e atualização do candidato e o sistema se comportou da forma esperada.

# votaMINAS

**INÍCIO** **CANDIDATOS** **VOTE**

Bem vindo [admin](#) **Painel Administrativo** [Logout](#)

[Listar enquetes](#) [Listar opções em enquete](#) [Listar usuários](#) [Listar candidatos](#) [Listar votos](#)

**Adicionar candidato**

Nome

Dilma

Número

13

Partido

PT

Propostas

Proposta #1  
Proposta #2  
Proposta #3

Posição

Presidente

Foto

Browse... dilma.jpg

Cadastrar

Figura 5. Primeira etapa no cadastro de candidato.

# votaMINAS

**INÍCIO** **CANDIDATOS** **VOTE**

Bem vindo [admin](#) **Painel Administrativo** [Logout](#)

[Listar enquetes](#) [Listar opções em enquete](#) [Listar usuários](#) [Listar candidatos](#) [Listar votos](#)

[Novo candidato](#) [Nova enquete](#)

**Candidatos**

The candidate has been saved.

<b>Id</b>	<b>Name</b>	<b>Number</b>	<b>Entourage</b>	<b>Propose</b>	<b>Type</b>	
1	Dilma	13	PT	Proposta #1 Proposta #2 Proposta #3	Presidente	<a href="#">Visualizar</a> <a href="#">Editar</a> <a href="#">Remover</a>

Página 1 de 1, mostrando 1 de 1 candidatos, começando no candidato 1 e terminando no 1

< voltar avançar >

Figura 6. Segunda etapa no cadastro de candidato.

### **3.1.2. CRUD para usuários**

A figura 7 mostra o estado da lista de usuário antes do cadastro deste. Já a figura 8, é tela exibida ao usuário e, uma vez com o formulário preenchido, os dados são enviados ao servidor.

Também foram testadas as operações de edição e remoção de usuários, comprovando o funcionamento do sistema como planejado.

### **3.1.3. CRUD para enquetes**

Assim como nos demais testes citados, as enquetes também podem ser criadas, editadas e removidas. Vale ressaltar que uma enquete não pode ter a quantidade de votos modificada através do sistema.

## **3.2. Erros encontrados**

Realizando diversos testes no sistema, foi possível identificar um erro de compatibilidade do navegador Firefox com o sistema responsivo desenvolvido. Este erro não pôde ser resolvido a tempo e foi classificado como melhorias futuras.

## **4. Conclusão**

A Web cresceu e cresce muito a cada dia, e com isso, o desenvolvimento voltado a websites. Aprender e desenvolver conhecimentos nessa área é de suma importância para nós programadores, assim como a utilização de boas práticas para a criação de páginas bem estruturadas e apresentáveis aos navegantes.

Com este trabalho prático foi possível aprender e se aprofundar nas linguagens HTML/CSS, assim como scripts, design responsivo, CakePHP e mais. Um projeto pequeno porém trabalhoso, que exigiu muito estudo e tempo; erros e bugs foram inevitáveis. Entretanto, os mesmos foram corrigidos e o site cumpre quase 100% do planejado, deixando de fora alguns detalhes que não foram implementados devido a falta de tempo e às complicações que o mesmo gerou durante seu desenvolvimento.

## **5. Referências**

<https://developer.mozilla.org/pt-BR/docs/Web/HTML>  
<http://pt-br.html.net/tutorials/css/lesson1.php>  
[http://php.net/manual/pt\\_BR/faq.general.php](http://php.net/manual/pt_BR/faq.general.php)  
<http://book.cakephp.org>  
<http://pt.wikipedia.org/wiki/MVC>  
<http://pt.wikipedia.org/wiki/Git>  
<http://blog.popupdesign.com.br/>



# votaMINAS

INÍCIOCANDIDATOSVOTE

Bem vindo [admin](#)**Painel Administrativo**[Logout](#)

[Listar enquetes](#)[Listar opções em enquete](#)[Listar usuários](#)[Listar candidatos](#)[Listar votos](#)

Users

Id	Username	Name	Email	Role		
1	admin	admin	admin@votaminas.com	admin	<a href="#">Editar</a>	<a href="#">Excluir</a>
3	teste1	teste1	teste1@teste1.com	regular	<a href="#">Editar</a>	<a href="#">Excluir</a>
5	teste2	teste2	teste2@teste2.com	regular	<a href="#">Editar</a>	<a href="#">Excluir</a>
6	teste3	teste3	teste3@teste3.com	regular	<a href="#">Editar</a>	<a href="#">Excluir</a>
7	teste4	teste4	teste4@teste4.com	regular	<a href="#">Editar</a>	<a href="#">Excluir</a>
8	teste5	teste5	teste5@teste5.com	regular	<a href="#">Editar</a>	<a href="#">Excluir</a>
9	teste6	teste6	teste6@teste6.com	regular	<a href="#">Editar</a>	<a href="#">Excluir</a>

Página 1 de 1, mostrando 7 de 7 candidatos, começando no candidato 1 e terminando no 7

< Anterior Próximo >

Figura 7. Primeira etapa no cadastro de candidato.

# votaMINAS

INÍCIOCANDIDATOSVOTE

**CADASTRE-SE**[Login](#)

[Listar enquetes](#)[Listar opções em enquete](#)[Listar usuários](#)[Listar candidatos](#)[Listar votos](#)

Cadastro

Username

joaosilva

Password

\*\*\*\*

Name

João Exemplo da Silva

Email

joao@exemplo.com

Cadastrar

Figura 8. Segunda etapa no cadastro de candidato.