UFMT

UNIVERSIDADE FEDERAL DE MATO GROSSO

INSTITUTO DE COMPUTAÇÃO

COORDENAÇÃO DE ENSINO DE ESPECIALIZAÇÃO EM

ENGENHARIA DE SISTEMAS WEB

GERENCIAMENTO DE DOCUMENTOS OFICIAIS: UMA ABORDAGEM UTILIZANDO NOSQL E ZEND FRAMEWORK PARA ARMAZENAMENTO E ORGANIZAÇÃO DOS DADOS

LUIZ CEZAR BRANDÃO JUNIOR

Cuiabá - MT

2012

UNIVERSIDADE FEDERAL DE MATO GROSSO

INSTITUTO DE COMPUTAÇÃO

COORDENAÇÃO DE ENSINO DE ESPECIALIZAÇÃO EM

ENGENHARIA DE SISTEMAS WEB

GERENCIAMENTO DE DOCUMENTOS OFICIAIS: UMA ABORDAGEM UTILIZANDO NOSQL E ZEND FRAMEWORK PARA ARMAZENAMENTO E ORGANIZAÇÃO DOS DADOS

LUIZ CEZAR BRANDÃO JUNIOR

Orientador: Prof. Msc Nielsen Simões

Monografia apresentada ao Curso de Especialização em Engenharia de Sistemas Web, do Instituto de Computação da Universidade Federal de Mato Grosso, como requisito para obtenção do título de Especialista em Engenharia de Sistemas Web.

Cuiabá - MT

2012

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

AGRADECIMENTOS

Agradeço a Deus pela oportunidade dada, aos meus familiares por me amparem em todos os meu momentos e a minha namorada, pela companhia de cada dia.

DEDICATÓRIA

Dedico este meu trabalho aos meus familiares que sempre me apoiaram, a minha namorada que sempre esteve ao meu lado e acima de tudo a Deus pela oportunidade dada.

RESUMO

Empresas públicas e privadas manipulam uma grande quantidade de informações diariamente, arquivos esses que são de fundamental importância para a manutenção e comunicação da empresa. Diante desse cenário, este trabalho propõe uma nova abordagem de edição, armazenamento e gerenciamento desses documentos, universalizando o acesso às informações contidas dentro desses documentos e agilizando o processo de recuperação dessas informações utilizando software *open-source* e uma base de dados não relacional. Alem disso iremos desenvolver uma aplicação web para edição e manipulação desses documentos automatizando assim os processos de criação desses documentos.

CAPÍTULO 1

INTRODUÇÃO

Organizações públicas e privadas lidam com uma grande quantidade de documentos, essenciais para o seu correto funcionamento e que nem sempre recebem a devida atenção quanto a sua criação, armazenamento e organização. Causando confusão, desorganização e lentidão na recuperação dos dados, além de atrasos nas tomadas de decisões. Esses problemas são causados devido ao modelo de gerenciamento corrente, que visa o armazenamento em pastas e armários, modelo esse herdado das décadas passadas onde a máquina de escrever era a principal forma de edição de documentos e o papel era o centro das atenções. Atualmente, o computador é a principal ferramenta de edição de documentos, porém o papel ainda é utilizado em larga escala para a impressão dos documentos editados no computador.

Com a grande popularização da internet e o consequente aumento na busca por novas informações, além da popularização das tecnologias móveis, a quantidade de informações geradas e armazenadas chegou a níveis nunca vistos anteriormente. Para dar conta desse novo paradigma, novas tecnologias foram surgindo para suprir a demanda de armazenamento e recuperação de dados em tempo real. O avanço da Web 2.0 e seu ritmo avassalador de produção de conteúdo proporcionaram o surgimento dos primeiros modelos de banco não relacionais. Uma vez que o paradigma relacional é custoso, lento e de grande complexidade arquitetural, as principais empresas da Web 2.0 procuraram, então, por alternativas de armazenamento que suprissem essa necessidade (FINLEY, 2012).

O paradigma não relacional tem como lema o não relacionamento dos dados, a desconstrução das associações e uma maior flexibilidade na hora de projetar as bases de dados, flexibilidade essa que permite que alguns registros tenham campos que outros não tenham, sem comprometer a integridade das informações. A desconstrução das associações não significa que no paradigma não-relacional não exista esse tipo de comportamento. Na prática, não existe relacionamento dos dados da forma convencional como conhecemos.Na verdade, o que existe é o referenciamento das informações, essa determinada informação pode existir ou não na coleção referenciada.

A escolha de um banco não relacional, orientado a documento, vem do fato que iremos armazenar documentos, não documentos convencionais como documentos do word ou um documento em pdf (FRANCIA, 2012), mas sim a sua estrutura e o seu conteúdo será armazenado dentro da base de dados criada para esse propósito. Chamamos aqui de estrutura os campos contidos dentro do documento em questão, também conhecido como dados estruturados.

Além disso, seguindo esse modelo, podemos facilmente estender a ferramenta para trabalhar com outros tipos de documentos, poupando tempo e esforços, aumentando assim a segurança e o controle sobre os arquivos armazenados.

Dessa forma, a aplicação visa manter os documentos armazenados de maneira segura, diminuindo o risco de perda e aumentando a velocidade na recuperação das informações contidas nesses documentos, tendo em vista que eles estarão armazenados digitalmente, não mais correndo riscos de perda e extravios humanos, e sem sofrer a ação do tempo, sendo armazenado em armários e prateleiras.

Diante desse cenário, este trabalho tem como objetivo principal construir uma aplicação que será capaz de manipular grandes volumes de dados, mostrando as vantagens e desvantagens da utilização desse novo paradigma de armazenamento de dados. Além disso, objetivamos mostrar as vantagens de se utilizar um framework de desenvolvimento web para a edição e manipulação dos documentos.

A aplicação tem como principal função, automatizar a edição e confecção de documentos oficiais utilizando tecnologias *open-source*, melhorando assim a manipulação e o armazenamento de documentos oficiais dentro das organizações.

1.1 METODOLOGIA

Diante do problema citado, o trabalho será desenvolvido levando em consideração uma pesquisa bibliográfica feita sobre os principais aspectos da aplicação a ser desenvolvida. A pesquisa abrangerá livros técnicos, artigos, textos e reportagens abordando assuntos como NoSQL e metodologias de desenvolvimento ágil. Os dados coletados servirão de base para o desenvolvimento da aplicação que inicialmente denominados de GeraAta. Após esse levantamento teórico os dados analisados serão aplicados no estudo de caso que consiste no desenvolvimento de uma aplicação que tem como objetivo gerenciar documentos de maneira não relacional.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Diante do cenário apresentado, a expectativa é que a ferramenta irá gerar uma grande quantidade de dados diariamente para suportar essa demanda, optamos pela utilização de banco de dados não relacional. A escolha se justifica pelo fato de que o objetivo dos bancos não relacionais é atender os requisitos de gerenciamento de grandes volumes de dados, semi-estruturados ou não estruturados (LÓSCIO, 2011). Tendo em vista que uma das grandes vantagens da utilização da GeraAta, é que ela pode se adaptar rapidamente as especificações do documento em questão, tornando assim, a GeraAta uma ferramenta com alta capacidade de adaptação ao meio onde a ferramenta for implantada. Essa característica se dá pela escolha das ferramentas para o seu desenvolvimento que são o MongoDB, para o armazenamento não relacional e o Zend Framework para a construção da aplicação. A seguir contextualizaremos as ferramentas em questão, sendo a primeira delas o MongoDB e em seguida o Zend Framework.

2.1 MONGODB

MongoDB é um banco não relacional, orientado a documentos, que tem como principais características a sua leveza, fácil manipulação dos dados, escalabilidade e uma estrutura de busca e consulta como o JSON (JavaScript Object Notation). Diferentemente dos bancos de dados relacionais, os bancos não relacionais não possuem uma estrutura rígida de armazenamento, na realidade, eles armazenam os dados em documentos vaga-

mente definidos. Isso garante que ao ser adicionado novos campos em um documento, os documentos anteriores não possuiriam aquele campo vazio como nos modelos relacionais. A Figura 1 ilustra um exemplo de documento armazenado no MongoDB.

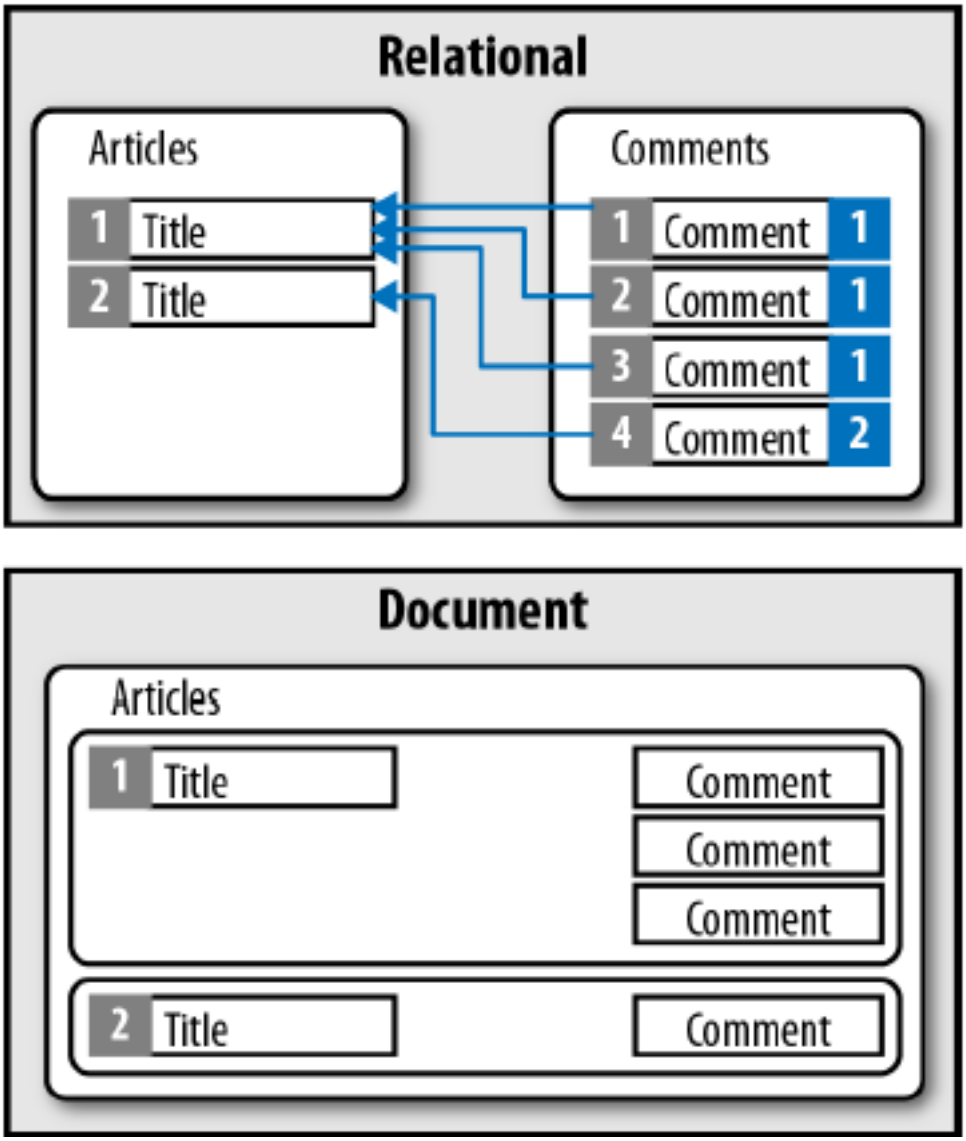

Figura 1: Comparação entre o Modelo Relacional e o Não Relacional
(FRANCIA, 2012)

Na Figura 1 ilustra a comparação do modelo relacional com o modelo não relacional orientado a documento utilizado pelo MongoDB. Na ilustração do primeiro quadro temos para cara valor da tabela *comments* uma referencia da tabela *articles*. Já no segundo quando para todos os comentários relacionados aos *articles* é armazenado juntamente como ele dessa forma otimizando a recuperação das informações.

Essas características tornam o MongoDB uma ótima opção para armazenamento de dados, para aplicações que tenham um grande volume de acesso, escrita e consulta. Outra grande diferença em relação aos bancos relacionais é a forma de trabalho do MongoDB, nele temos banco de dados, coleções e documentos (FRANCIA, 2012). Em uma comparação direta as coleções seriam as tabelas do banco relacional e os documentos as linhas daquela tabela. Cada um desses conceitos será detalhado nas próximas seções.

2.1.1 BASE DE DADOS

MongoDB agrupa os dados em base de dados de forma semelhante aos banco relacionais, mas diferentemente do paradigma relacional uma base de dados em MongoDB é uma série de coleções (FRANCIA, 2012).

2.1.2 COLEÇÕES

No MongoDB uma coleção é um agrupamento de documentos e índices (FRANCIA, 2012). Coleções seriam as tabelas no paradigma relacional, para fins de comparação podemos chamá-las dessa maneira.

2.1.3 DOCUMENTOS

Documento é a estrutura onde são armazenados os dados no MongoDB. Não tem uma comparação direta com o paradigma relacional, mas podemos compará-lo às linhas da tabela. Diferentemente da linhas, um documento não tem uma estrutura prédefinida, as colunas do paradigma relacional, no MongoDB cada documento pode conter uma estrutura diferente da outra isso tudo dentro da mesma coleção (FRANCIA, 2012). Por exemplo podemos ter o campo endereço em um determinado documento de um usuário, e em outro documento de usuário, não termos o campo endereço.

A melhor forma de se pensar em um documento do MongoDB é um array multidimensional, onde se pode mapear os valores de cada campo, facilitando na conversão de documentos para objetos (FRANCIA, 2012). Outra característica em comparação entre o modelo relacional e o modelo não relacional são os índices, a qual chamamos de ObjectsID. Cada documento no MongoDB é identificado através de um chave única gerada apartir do timestamp do momento em que é criado o documento.

Dessa maneira o ObjectID do MongoDB provê um valor equivalente a chave primária, não seqüencial, garantindo assim a identificação única de cada documento, garantindo assim sua integridade.

2.1.4 REFERÊNCIAS DE DADOS

Assim como os bancos relacionais fazem referência a dados de outras tabelas via uma chave estrangeira, o MongoDB fornece um mecanismo chamado DBRef, para referenciar dados que estão em outras coleções. Diferentemente dos modelos relacional, o DBRef não provê uma chave estrangeira, ele cria um link entre os dados referenciados (FRANCIA, 2012).

2.1.5 DATAS E HORÁRIO

Assim como o id do documento é um objeto no MongoDB, data e horários são instancias da classe MongoDate. O MongoDB armazena as datas de forma similar ao Unix timestamp, ignorando as zonas temporais e trabalhando com os milisegundos. Para tratar essas datas é preciso utilizar as bibliotecas padrões de cada linguagem de programação, permitindo obter um resultado mais compreensível ao ser humano.

2.2 OUTROS TIPOS DE BANCOS NÃO RELACIONAIS

O movimento não relacional possui uma série de projetos que armazenam dados de forma não relacional, além do armazenamento em documentos. Outras formas de armazenamento não relacional são os bancos orientados a objeto, chave-valor, tabular e grafos ilustrados na Figura 2.

Os bancos orientados a objetos armazenam os dados com a mesma estrutura das classes de modelos para acesso aos dados. Os dados são armazenados em um arquivo que possui as estruturas passados pela aplicação para armazenamento dos dados. No quesito desempenho os dados são acessados muito rapidamente, devido a natureza dos objetos armazenados, sendo o acesso feito de forma nativa. Um exemplo de banco não relacional orientado a objetos é o db4o que é mantido pela comunidade.

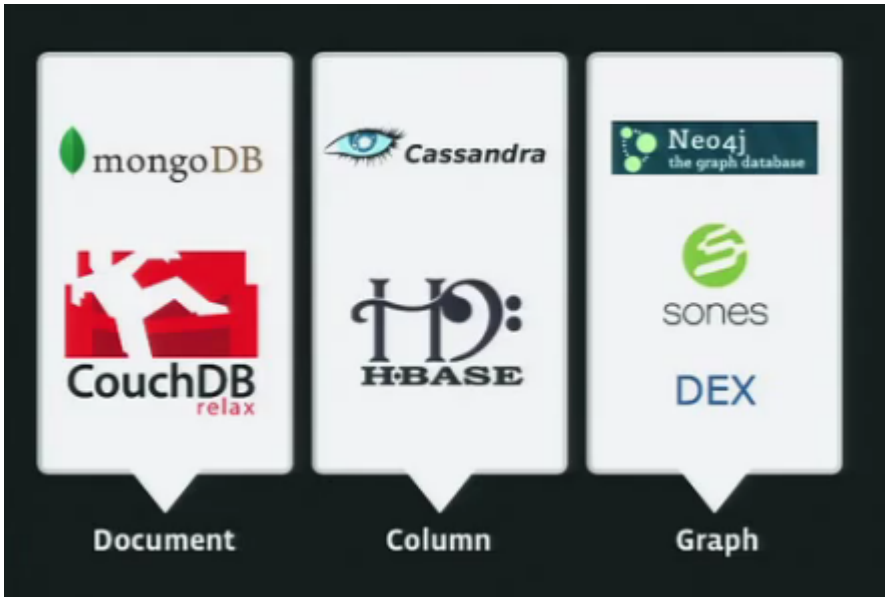

Figura 2: Segmentação não-relacional
(VENKATESH, 2012)

Uma forma interessante de armazenamento não relacional, são as bases de dados chamados de BigTable ou Column, os dados são armazenados em uma estrutura semelhante a uma tabela que armazena todas as informções possiveis sobre o objetos a ser analisado. Destaques desse tipo de estrutura são o Cassandra, hoje também chamado de Hadoop e H-Base ambos ilustrados na Figura 2.

Outra forma de armazenar dados não relacionais é utilizando bancos de chave-valor, nessa forma de armazenamento os dados são gravados em arrays de pares chave-valor, onde a definição do campo a ser armazenado vai do lado esquerdo e o valor a ser gravado vai do lado direito. Exemplos de bancos que armazenam os dados nessa forma são o Redis, mantido pela comunidade e o SimpleDB da amazon.

Além dessas formas existem também os bancos que são chamados de grafos, nessa forma de armazenamento os dados são armazenados em grafos, esses grafos são compostos por nodes, relacionamentos e propriedades. Um grafo pode ter uma série de nodes, relacionamentos e propriedades, em que cada node tem seu próprio índice. Uma base de dados destaques nesse tipo de armazenamento é o Neo4j ilustrado na Figura 2 (ANGLES; GUTIERREZ, 2008).

2.3 DESVANTAGENS DA UTILIZAÇÃO DE UM BANCO NÃO-RELACIONAL

Dentre as vantagens citadas anteriormente, descreveremos aqui algumas desvantagens da utilização de bancos não-relacionais. Uma das desvantagens da utilização de banco não-relacional é a perda da capacidade de validar determinada informação tendo em vista que a responsabilidade de fazer a validação dos dados fica a cargo da aplicação, a responsabilidade do banco não-relacional é somente armazenar os dados passados pela aplicação (DIANA; GEROSA, 2010). Outra desvantagem do modelo não-relacional é a falta de uma linguagem padrão para consulta, recuperação e inserção de dados, tendo em vista que cada banco não relacional possui a sua linguagem de manipulação de dados. Isso é um obstáculo para novos adeptos da tecnologia, tendo em vista que para cada banco não-relacional deve-se aprender a sua linguagem de manipulação de dados daquele determinado banco. Apesar de cada mecanismo de banco lidar com o armazenamento dos dados de maneira diferente, mas no modelo relacional eles possuem um linguagem padrão que de certa forma uniformiza o conhecimento na hora de se realizar mudanças de SGBDs.

2.4 ZEND FRAMEWORK

Zend Framework é um framework open-source para construção de aplicações web, todo construído em PHP na versão 5 da linguagem, orientado a objetos e com suporte às arquiteturas MVC e HMVC, disponibilizando aos programadores uma coleção de ferramentas para a construção de aplicações web robustas e de qualidade. A versão 1 do framework já superou a marca de 15 milhões de downloads (ZEND, 2012).

A Zend é a principal mantenedora do PHP, juntamente com a comunidade e com outras empresas. Atualmente o Zend é um dos principais frameworks do mercado. As próximas subseções detalham a arquitetura, características, vantagens e desvantagens da sua utilização.

2.4.1 ARQUITETURA

O Zend Framework é todo escrito com código orientado a objetos, o que garante uma maior extensibilidade e também um maior reuso de código da aplicação, alem de facilitar o acesso às melhores práticas de programação usando PHP. A seguir teremos uma breve introdução das principais tecnologias por trás do Zend Framework.

MVC

MVC é uma arquitetura da engenharia de software que consiste em separar a aplicação em três camadas distintas: Model(Modelo), View(Visão), Controller(Controlador ou Controle). A arquitetura foi proposta para otimizar e delegar as responsabilidades de cada camada, possibilitando assim membros de uma mesma equipe trabalhar em camadas diferentes ao mesmo tempo, sem correr o risco de um atrapalhar o trabalho do outro. Além disso com a separação em camadas aumentamos o nível de organização dos códigos dentro da aplicação. Cada camada possui suas devidas atribuições, a camada de Modelo é responsável pelo acesso aos dados e pela modelagem de dados na aplicação, a camada de visão é como os dados serão apresentados na aplicação e a camada de controle é a responsável pelas requisições entre as camadas de visão e modelo, conforme ilustra Figura 3.

Como é ilustrado na figura acima, a arquitetura MVC nos permite separar a aplicação em camadas, sendo que cada camada possui suas atribuições, permitindo assim uma melhor estruturação das classes responsáveis pela aplicação. O MVC é um dos padrões mais utilizando hoje no mercado de desenvolvimento de aplicações.

HMVC

HMVC(*Hierarchical Model View Controller*) é uma extensão do MVC e tem como função modularizar a aplicação de forma a se permitir que a aplicação se torne o mais escalável possível. A representação do HMVC está ilustrada na Figura 4.

Dessa forma podemos notar que a função de transportar a requisição entre as hierarquias MVC é do *controller* do contexto da aplicação. Dessa maneira podemos modularizar as aplicações no Zend Framework em várias outras pequenas aplicações que

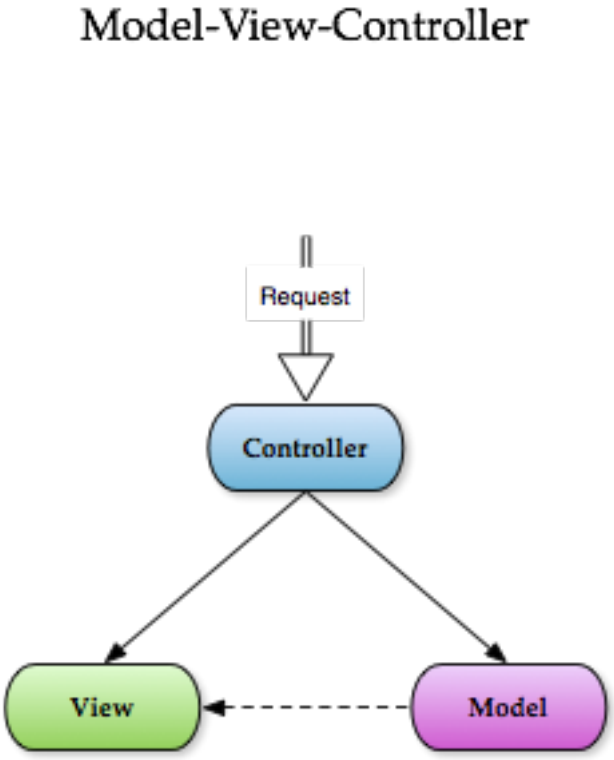

Figura 3: Exemplo da Arquitetura MVC
(FREYSSINET, 2012a)

podem estar sendo desenvolvidas e mantidas por equipes ou pessoas diferentes ao mesmo tempo, podendo assim reutilizar código em vários pontos da aplicação.

2.4.2 CARACTERÍSTICAS

GUIADO A TESTE

O Zend Framework é fortemente integrado com o PHPUnit, um framework php para testes unitários. Isso confere ao Zend um grande poder de manutenabilidade de código, dessa forma podemos saber onde as alterações feitas irão impactar na aplicação como um todo.

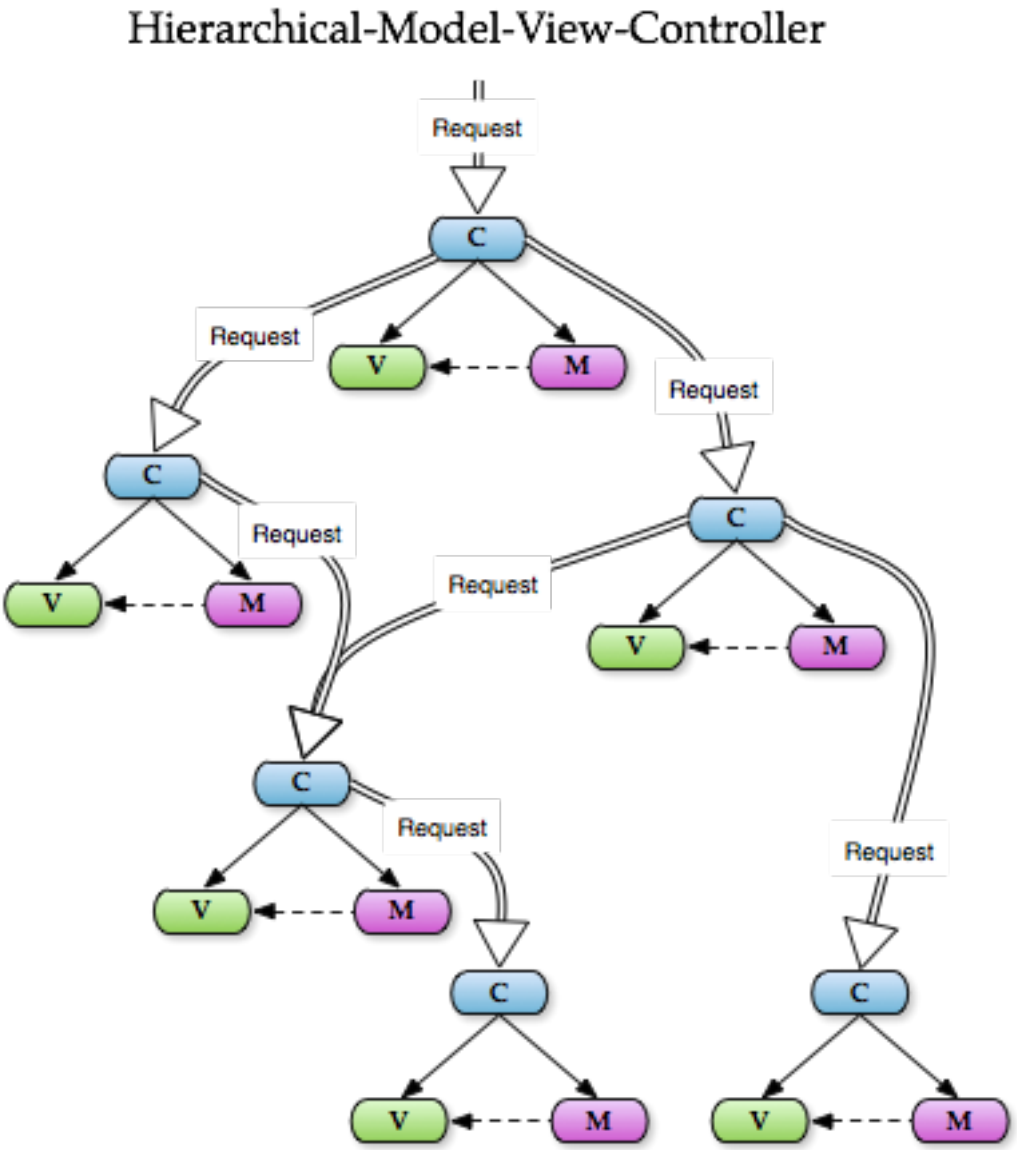

Figura 4: Exemplo da Arquitetura HMVC
(FREYSSINET, 2012b)

DOCUMENTAÇÃO

O Zend Framework possui grande documentação isso facilita ainda mais o aprendizado sobre as bibliotecas disponibilizadas pelos framework, alem de acelerar o desenvolvimento de aplicações utilizando o framework.

COLEÇÃO DE BIBLIOTECAS

O Zend Framework possui uma coleção de bibliotecas que podem ser usadas de maneira independente e o torna ainda mais poderoso. O Zend disponibiliza bibliotecas para trabalhar com várias tecnologias como ACL(Listas de Controle de Acesso), JSON, Abstração de Dados, entre vários outras. A lista completa pode ser encontra na página do framework na internet, disponível em: framework.zend.com. (ZEND, 2012)

2.4.3 VANTAGENS DE UTILIZAÇÃO DO FRAMEWORK

A grande vantagem da utilização de um framework de desenvolvimento orientado a objetos como o Zend, é ter acesso a uma gama de bibliotecas que permite aos desenvolvedores construírem aplicações web de larga escala, com qualidade e segurança, garantindo assim que a aplicação possa ser mantida de forma mais natural e com um ciclo de vida maior.

CAPÍTULO 3

APLICAÇÃO

Aplicação que demos o nome de GeraAta terá a função de armazenar os dados das Atas inseridas. Para ter acesso a área de edição de Atas e Gerenciamento de Usuários do GeraAta é preciso realizar o processo de Login. O Login é composto de usuário e de uma senha previamente inseridos no GeraAta, conforme ilustra a Figura 5.

Figura 5: Tela de Login do GeraAta

Após a verificação da existência e da validação dos dados do usuário ele é redirecionado para a tela inicial da Aplicação. A tela inicial da aplicação será apresentada conforme a Figura 6.

Figura 6: Index do GeraAta

Dentro da página de index do GeraAta, ele pode escolher quais ações ele irá tomar, entre gerenciar usuário e gerenciar atas. A tela de gerenciamento de atas é mostrada conforme a Figura 7.

Figura 7: Tela inicial do processo de adição de ata dentro do GeraAta

Dentro da tela de Gerência de atas o usuário poderá visualizar as atas já inseridas e poderá também inserir novas atas, alem de buscar atas já homologadas no sistema. A seguir temos a tela inicial do processo de adição de uma nova Ata, que começa com a adição dos principais dados da ata que são o dia em que ela está sendo redigida, o assunto, os pontos de pauta, as pessoas presentes e uma pequena descrição da Ata, conforme ilustra a Figura 8.

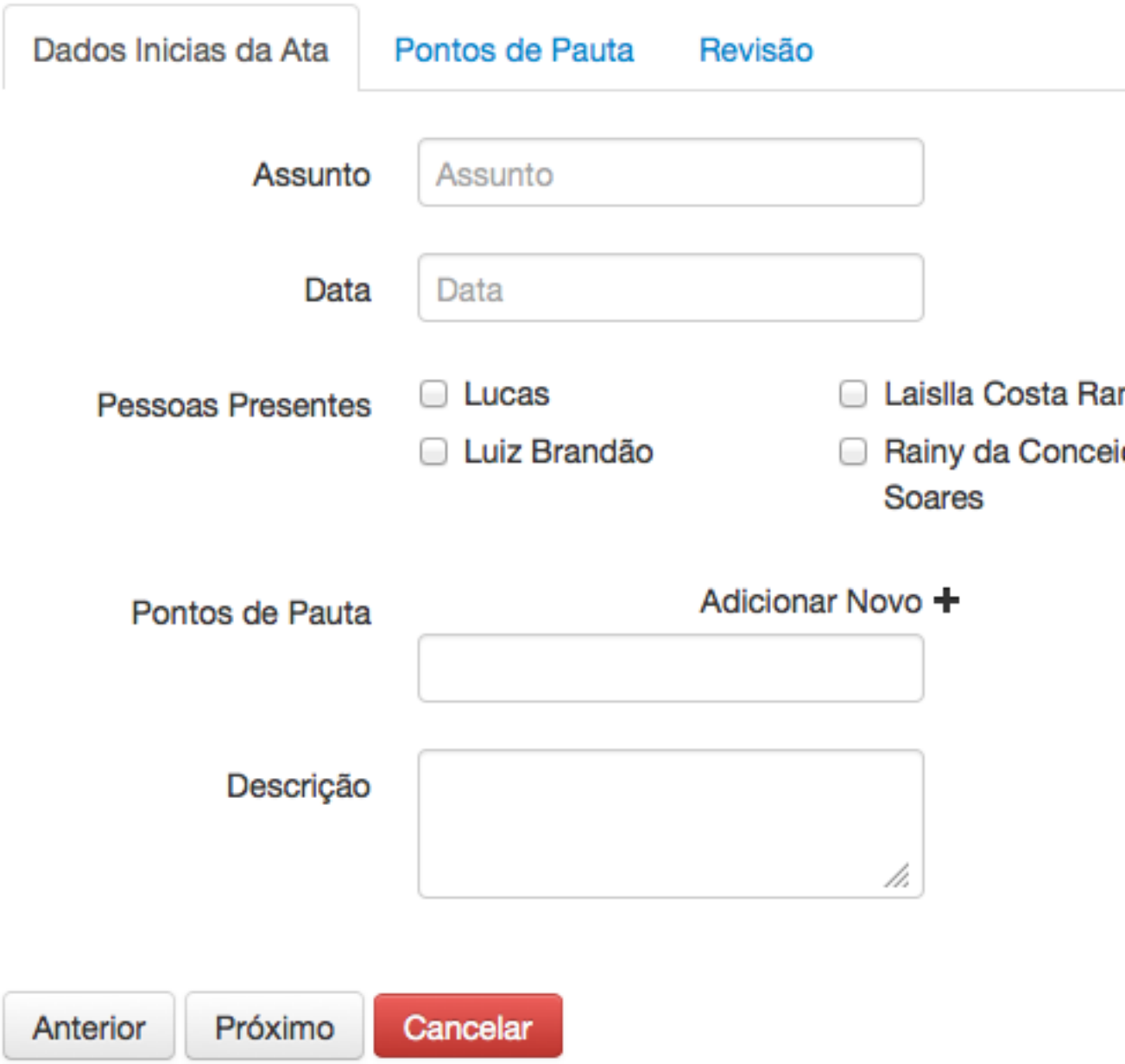

Figura 8: Tela inicial do processo de adição de ata dentro do GeraAta

Na figura 9 temos os pontos de pauta tratados na reunião, onde o usuário deverá inserir o que foi tratado sobre cada ponto de pauta na reunião.

Após todos os dados serem preenchidos o usuário seguirá para a próxima tela, onde existem duas opções para o usuário a de salvar a ata e a de homologar a ata. A diferença entre esses dois métodos é que caso o usuário escolha por salvar os dados eles poderão ser editados no futuro, o que não será possível no caso de homologação,

Figura 9: Lista dos pontos de pauta inseridas na Ata

após a homologação os dados apenas estarão disponíveis para visualização. A tela que representa a página de revisão da Ata é ilustrada conforme a Figura 10

Além das funções ditas anteriormente o sistema possui as funcionalidades de Gerenciamento de usuários. A tela de Gerência de Usuários é ilustrada conforme a Figura 11.

O GeraAta permite que o administrador gerencie novos usuários que possam realizar as operações no sistema. A tela de cadastro de um novo usuário é exemplificada conforme a Figura 12.

Além do cadastro é possível atualizar os dados de um usuário já cadastrado no sistema, a tela de atualização dos dados é conforme a Figura 13.

Figura 10: Revisão Geral da Ata a ser inserida no GeraAta

Figura 11: Tela de Gerência de Usuarios do GeraAta

Figura 12: Tela de Adição de Usuário no GeraAta

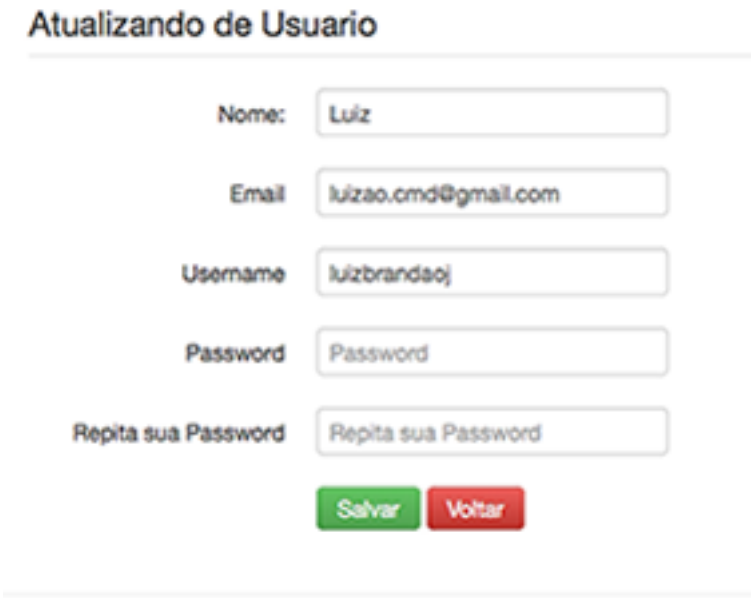

Figura 13: Tela de Atualização de Usuário no GeraAta

3.1 ARQUITETURA DA APLICAÇÃO

A vantagem de se desenvolver o GeraAta da maneira como a aplicação foi desenvolvida é fato de que ela pode ser estendida para outros documentos de for que para isso, basta criar novos módulos para atender a esse novo documento. Com bases nos conceitos citados durante o trabalho, contextualizaremos a arquitetura utilizada na construção do GeraAta.

3.1.1 MODELAGEM

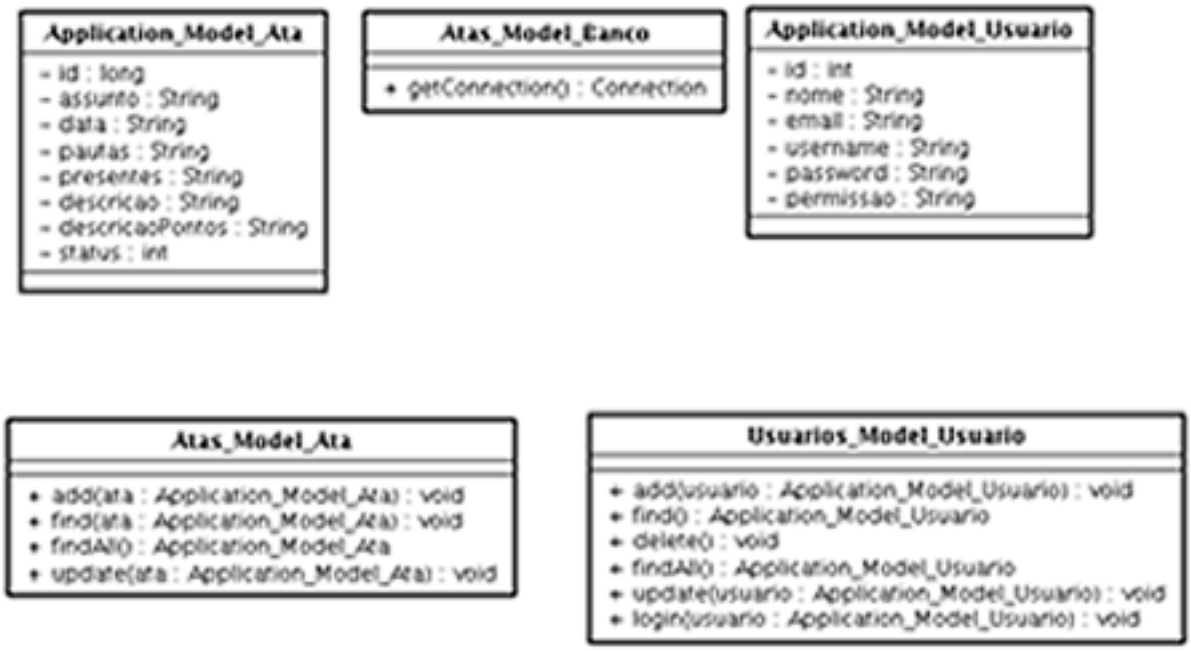

Figura 14: Modelagem do GeraAta

Na Figura 14 visualizamos o Diagrama de Classe criado a partir do software de Modelagem Jude Community. Nela temos as classes *Application Model Ata* e *Application Model Usuario*, elas seria as classes VO do GeraAta, nelas estão contidos somente os atributos necessários para o correto funcionamento da aplicação.

Já a classe *Atas Model Banco* é responsável pela conexão com o MongoDB, nela temos uma instancia da classe *Mongo* responsável pela conexão com uma instancia da base de dados instalada localmente no servidor. O trecho responsável pela conexão com a base de dados está representada na Figura 15.

```php
public function getConnection(){
    $mongo = new Mongo();
    $db = $mongo->docs;
    return $db;
}
```

Figura 15: Método de conexão com a base de dados.

A estrutura responsável pela definição de quais campos serão armazenados é ilustrado na Figura 16

```php
$ata = array(
    'assunto' => $ata->assunto,
    'data' => $ata->data,
    'pautas' => $ata->pautas,
    'presentes' => $ata->presentes,
    'descricao' => $ata->descricao,
    'descricaoPontos' => $ata->descricaoPontos,
    'status' => $ata->status,
);
```

Figura 16: Estrutura que será armazenada na Base de dados.

Dessa forma é possível aumentar ou diminuir o número de campos existentes em cada documento, para atender as novas especificidade de cada estrutura. Esse é um exemplo da facilidade de adição e remoção de campos em uma estrutura não relacional isso torna a aplicação mas extensível, diminuindo assim a complexidade da ferramenta. Alem disso essas características transformam o GeraAta em uma aplicação que tem um longo tempo de vida, pode ser mantida por vários outros desenvolvedores ao longo do tempo.

CAPÍTULO 4

CONSIDERAÇÕES FINAIS

O estudo de caso, mostra que a solução proposta atende as necessidades básicas, tendo em vista que o estudo ainda não está na sua plenitude. Graças as características das tecnologias escolhidas durante o estudo, a aplicação pode ser estendida para uma gama maior de documentos e atender a uma série de departamentos ao mesmo tempo. Sendo assim o GeraAta é uma ferramenta que tem grande potencial de utilização e atualização para outras necessidades onde o objetivo seja editar e manter documentos na internet, de forma rápida e acessível. Essas características se devem ao fato da nossa tecnologia de armazenamento de dados ser flexível o suficiente para comportar mudanças ao longo do tempo, graças ao armazenamento não relacional e a flexibilidade do framework escolhido para desenvolvimento da aplicação. O sistema proposto vem para sanar uma necessidade existente em vários setores da nossa sociedade e automatizar o máximo possível a manipulação de documentos oficiais, tentando assim diminuir o número de falhas possíveis.

REFERÊNCIAS BIBLIOGRÁFICAS

ANGLES, R.; GUTIERREZ, C. Survey of graph database models. *ACM Computing Survey*, v. 40, p. 39, 2008.

DIANA, M. d.; GEROSA, M. A. Nosql na web 2.0: Um estudo comparativo de bancos nosql na web 2.0: Um estudo comparativo de bancos não-relacionais para armazenamento de dados na web 2.0. *WTDBD*, p. 8, 2010.

FINLEY, K. *How Twitter uses NoSQL*. 03 2012. Disponível em: <http://www.readwriteweb.com/cloud/2011/01/how-twitter-uses-nosql.php>.

FRANCIA, S. *MongoDB and PHP*. [S.l.: s.n.], 2012.

FREYSSINET, S. de. *Scalling Web Applications wtih HMVC*. 12 2012. Disponível em: <http://techportal.inviqa.com/2010/02/22/scaling-web-applications-with-hmvc/>.

FREYSSINET, S. de. *Scalling Web Applications wtih HMVC*. 12 2012. Disponível em: <http://techportal.inviqa.com/2010/02/22/scaling-web-applications-with-hmvc/>.

LÓSCIO, B. F. Nosql no desenvolvimento de aplicações web colaborativas. *SBC2011*, p. 17, 2011.

VENKATESH. *Its time to use NoSQL*. 12 2012. Disponível em: <http://enterprise-now.com/architecture/2012/06/26/its-time-to-use-no-sql/>.

ZEND. 12 2012. Disponível em: <http://framework.zend.com/>.