UNIVERSIDADE COMUNITÁRIA DA REGIÃO DE CHAPECÓ – UNOCHAPECÓ
Área de Ciências Exatas e Ambientais
Curso de Ciência da Computação

Kassiano José Matteussi

PROTÓTIPO DE INTERFACE WEB COM PHP PARA GERENCIAMENTO DE BANCO DE DADOS COUCHDB

Chapecó – SC, 2010

KASSIANO JOSÉ MATTEUSSI

PROTÓTIPO DE INTERFACE WEB COM PHP PARA GERENCIAMENTO DE BANCO DE DADOS COUCHDB

Projeto de pesquisa apresentado à disciplina de Monografia III, Curso de Ciência da Computação, sob coordenação da professora Monica Tissiani De Toni Pereira e orientação de Elton Luís Minetto.

Chapecó – SC, jul. 2010

PROTÓTIPO DE INTERFACE WEB COM PHP PARA GERENCIAMENTO DE BANCO DE DADOS COUCHDB

KASSIANO JOSÉ MATTEUSSI

Esta Monografia foi julgada para obtenção do título de Bacharel em Ciência da Computação, na área de concentração e aprovada pelo curso de Ciência da Computação

ORIENTADOR(A): Prof. Elton Luís Minetto

COORDENADOR(A) DO CURSO: Prof.º Mônica Tissiani de Toni Pereira

BANCA EXAMINADORA

PRESIDENTE: Prof. Mônica Tissiani de Toni Pereira

Prof. José Carlos Toniazzo

DEDICATÓRIA

Dedico este trabalho a todos que de alguma forma me ajudaram e incentivaram, principalmente aos meus pais, que rezaram por mim e tanto me apoiaram, não posso esquecer-me do restante de minha família com o qual sempre pude contar muito. Obrigado do fundo do coração.

AGRADECIMENTOS

Gostaria de dizer um muito obrigado a todos os meus amigos e colegas de trabalho que certamente tiveram paciência comigo em algum momento da realização deste trabalho. Gostaria de agradecer também ao meu orientador Elton por ter o empenho de ler meus e-mails e de me orientar a distância e ainda ter que vir em alguns domingos para me ajudar.

Não posso esquecer-me de minha família que me deu todo o apoio necessário e estrutura para que um de meus sonhos fosse realizado.

E por fim, agradecer a Deus por ter me iluminado nestes quatro anos e meio de estudos.

“Sem ambição, nada se começa. Sem esforço, nada se completa.”

Ralph Waldo Emerson

SÚMARIO

ÍNDICE DE QUADROS

ÍNDICE DE FIGURAS

LISTA DE ABREVIATURA E SIGLAS

ACID - Atomicidade, Consistência, Isolamento e Durabilidade.

API - Interface de Programação de Aplicativos.

CURL - Cliente para Biblioteca URL.

HTTP - Protocolo de Transferência de Hipertexto.

HTML - Linguagem de Marcação de Hipertexto.

JSON - Objeto de Notação Javascript.

MVC- Modelo Visão e Controlador.

NOSQL - Not Only SQL.

PHP - Pré Processador de Hipertexto.

SGBDR - Sistema Gerenciador de Bancos de Dados Relacionais.

SOAP - Protocolo de Simples Acesso a Objetos.

SQL - Structured Query Language.

URI - Uniform Resource Identifier.

VIEW - Visão.

WEB - World Wide Web.

WSDL - Web Services Description Language.

XML - Extensible Markup Language.

RESUMO

Pode-se observar que o desenvolvimento web vem crescendo exponencialmente em conjunto com a presença de bancos de dados neste ambiente.

O modelo relacional, muito famoso no desenvolvimento desktop, foi adaptado para atuar no ambiente web. No entanto, o rápido crescimento e fluxo de dados na internet trouxeram alguns problemas que não ocorriam no antigo ambiente, como por exemplo, flexibilidade, desempenho, escalabilidade e compatibilidade, fatores estes que são essenciais para aplicações na web.

Recentemente, foi criado um novo modelo para armazenamento de dados com foco em sistemas para internet, que ficou conhecido como modelo orientado a documentos. Este por sua vez, apresenta-se como solução para algumas das principais deficiências apresentadas nos modelos relacionais.

Dentre as várias tecnologias da web abordadas nesta monografia, o PHP e o CouchDB - que pertence à classe de SGBD NoSQL - tiveram um estudo mais detalhado. No caso específico do PHP, foi realizado um estudo de funções e tecnologias disponíveis para uso com a linguagem e no caso dos SGBD NoSQL, optou-se por ressaltar os principais bancos de dados disponíveis. Além disto, este estudo incluiu as vantagens, desvantagens e características destas duas tecnologias. Para uma abordagem prática, foi implementado um protótipo de interface web com PHP para gerenciamento da base de dados CouchDB, apresentando um estudo mais aprofundado deste último.

ABSTRACT

Web development has grown exponentially along with the presence of databases in this environment.

The relational model, very famous in desktop development, has been adapted to operate in a web environment. However, the rapid growth and data flow on the Internet have brought some problems that did not occur in the old environment, including flexibility, performance, scalability and compatibility, factors that are essential for web applications.

Recently, a new model for data storage with focus on systems hosted on the Internet has been created, which became known as document-oriented model (DOM), presenting itself as a solution to some of the major deficiencies presented in relational models.

Among the various web technologies covered in this work, PHP and CouchDB - which belongs to the NoSQL class of DBMS - had a more detailed study. In the specific case of PHP, a study of functions and technologies available for use with this language took place and in the case of NoSQL's DBMS, we chose to highlight the major databases available for use. Moreover, this study included the advantages, disadvantages and features of these two technologies. For a practical approach, a prototype was implemented by the creation of a PHP web interface for managing the database CouchDB, focusing on a detailed study of this technology.

1. INTRODUÇÃO

Pode-se observar que o desenvolvimento Web vem crescendo exponencialmente, devido a sua versatilidade e facilidade de uso. Na mesma linha crescem os bancos de dados neste ambiente. O modelo relacional, muito famoso no desenvolvimento desktop, foi adaptado para atuar no ambiente Web, porém o rápido crescimento e fluxo de dados na internet trouxeram alguns problemas que não ocorriam no antigo ambiente, como por exemplo, flexibilidade, desempenho, escalabilidade e compatibilidade, fatores estes que são essenciais para aplicações na Web modernas.

Somando-se estes dois fatores, podemos observar que o modelo relacional se tornou uma tendência no mercado. Mesmo com avanços como a orientação a objetos, a arquitetura Web continua a usar o modelo de sistema gerenciador de banco de dador relacional como padrão para armazenamento de informações.

Para um número crescente de aplicações alguns destes requisitos (flexibilidade, desempenho, escalabilidade) estão ficando cada vez mais críticos, e mesmo com algumas inovações, grandes problemas aparecem rapidamente. A escalabilidade é um deles. Com um número cada vez maior de aplicações centralizadas em um único servidor com grande carga de trabalho a escalabilidade pode necessitar aumento muito rápido devido a quantia de usuários, podendo levar o hardware ao máximo.

Bancos relacionais possuem boa escalabilidade, mas quando se chega a 100% de uso de um servidor é necessário o balanceamento de carga para vários servidores, assim batendo de frente com a complexidade do modelo com seu potencial de escala. E quando essa necessidade de balanceamento de carga se torna muito grande acaba-se prejudicando a viabilidade nas grandes plataformas de desenvolvimento.

Esta limitação pode ser resolvida no modelo relacional, mas a complexidade da solução é grande. Então foi desenvolvido um novo sistema que trabalha sobre a escalabilidade e leva consigo os outros bons conceitos do modelo relacional, o chamado modelo de banco de dados orientado a documento.

O modelo de banco de dados orientado a documento surgiu para contribuir especialmente com o desenvolvimento de sistemas para web, pois, combina um modelo de armazenamento de documentos de uma forma intuitiva com um poderoso mecanismo de consulta e facilidades para escalar. Este modelo oferece um novo método para armazenar dados, o que é referido como um modelo de banco de dados orientado a documentos livre de esquema. Em vez de o armazenamento de dados altamente estruturado de um modelo relacional, o CouchDB armazena dados de maneira semi-estruturada.

Juntamente com o CouchDB vários outros bancos foram criados na mesma linha com objetivos e funcionalidades semelhantes, podendo-se citar os principais: Apache Cassandra, MongoDB, SimpleDB.

O crescimento do modelo foi tão visado que muitas empresas entraram nesta tendência. Em meados do ano de 2008, o Facebook adotou o modelo para melhorar seu fluxo de dados. Algum tempo depois, em 2010, o Twitter também começou trabalhos com a tecnologia e o Google, por sua vez, usa o projeto "Big Table" em um dos seus principais aplicativos: o Google Earth. O famoso sistema operacional Ubuntu a partir da versão 9.10 introduziu o CouchDB para sincronização de dados de usuário entre diferentes máquinas.

Mas apesar de existirem diversas soluções para o modelo orientado a documentos percebe-se que o mercado carece de uma ferramenta para auxiliar o gerenciamento destes bancos de dados. Desta forma, no presente trabalho será apresentado o desenvolvido um protótipo de uma ferramenta Web com PHP para o gerenciamento de banco de dados CouchDB com o foco em auxiliar na aprendizagem de comunidades acadêmicas e afins, fornecer uma ferramenta para contribuir com o trabalho de DBAs no que se refere a gerenciamento de diversas bases de dados orientadas a documentos.

1.1 Organizações do projeto

Este projeto está dividido em vários capítulos para facilitar o melhor entendimento do conteúdo proposto. Segue abaixo uma pequena descrição do que vai ser abordado em maiores detalhes no decorrer deste projeto.

No capítulo 1 temos a parte introdutória do projeto que serve para explanar ao leitor de uma forma sucinta o que vai ser retratado no decorrer do trabalho.

No capítulo 2 é descrito a tecnologia de SGBD NoSQL, vantagens, desvantagens e características.

Neste capítulo 3 será descrito a tecnologia de banco de dados CouchDB, suas características, vantagens, desvantagens, funcionamento e também outros bancos de dados que seguem o mesmo segmento.

No capítulo 4 temos todo o desenvolvimento do protótipo, as tecnologias utilizadas, a modelagem realizada, os requisitos da aplicação, bem como os recursos necessários para seu funcionamento. Também serão expostas algumas telas da aplicação e descrição detalhada de como foram construídas as mesmas.

O capítulo 5 contém a descrição das conclusões obtidas com o projeto. Bem como encontramos a relação de trabalhos futuros, incluindo alguns temas que poderiam ter sido abordados neste projeto, mas que fugiriam ao escopo inicial do mesmo.

Descreve-se no capítulo 6 toda a referência utilizada para compor este projeto.

2. SISTEMA GERENCIADOR DE BANCO DE DADOS ORIENTADO A DOCUMENTOS NOSQL

Os bancos de dados neste modelo compreendem uma série de documentos, cada um contendo uma série de campos e valores. Cada documento é independente um do outro e não usam nenhuma restrição quanto a sua estrutura durante a criação.

Em vez de um campo de chave primária, cada documento em um banco de dados CouchDB tem um ID único. Esta identificação exclusiva pode ser atribuído pelo usuário ou pelo aplicativo, ou ele pode usar um identificador universalmente exclusivo (UUID), este número aleatório é gerado pelo CouchDB que reduz a chance de duplicar IDs.

Segundo Moraes (2009), as "bases de dados orientados a documento são diferentes". Não existe hierarquia de dados, somente uma coleção de documentos que pode conter virtualmente qualquer espécie de dados. "Os documentos não precisam ser necessariamente do mesmo tamanho, pois alguns podem conter detalhes de campos que outros não precisam armazenar". Ou seja, você não está condicionado a um esquema de banco de dados.

Diferente dos bancos de dados relacionais, o banco orientado a documentos não grava dados em tabelas em linhas com campos uniformes. Neste banco de "dados" são inseridos documentos com determinadas características, a cada domínio são inseridos vários campos e esses campos por sua vez podem possuir vários elementos.

Fonte: MORAES (2009).
Figura 1: Exemplo demonstrando os diversos tamanhos de documentos.

A Figura 1 demonstra um conjunto de documentos de vários tamanhos e formas, ou seja, neste modelo de banco de dados você não ficará preso a linhas e tabelas. Os campos não são obrigatórios ao menos que você queira é claro.

Segundo Moraes (2009), criar uma base orientada a documentos é escrever uma 'view' que seleciona documentos e os mostra de uma forma particular. Você pode navegar pelos seus documentos usando qualquer critério que escolher – tipo de arquivo, tamanho, data de modificação, tags[1] – e então você pode salvar esta busca para usá-la novamente no futuro. Claro que isto pode trazer diferentes resultados da próxima vez que você usar, se os dados tiveram modificação.

Podem ser criados documentos usando XML (Extensible Markup Language) [2], JSON (JavaScript Object Notation)[3] e YAML (Yet Another Markup Language)[4], os documentos criados por estas linguagens contém um registro que possui um valor não padrão de informação.

```
{
   "Nome":"kassiano José Matteussi",
   "Endereço": [
         { "Rua": "Cultura", "Número": "130",  "complemento" : "e"},
         {"Rua": "Uruguai", "Número": "831",  "complemento" : "e"}, {"bairro":
         "Centro"}
    ]
}
```

Quadro 1: Exemplo de estrutura de um documento qualquer.

No Quadro 1 é esboçado um exemplo de como seria a estrutura de um documento em um banco orientado a documento, o exemplo é simples:

- um campo nome, com um valor;

[1] **Tags** são estruturas de linguagem de marcação que consistem em breves instruções, tendo uma marca de início e outra de fim.

[2] **XML** (eXtensible Markup Language) é uma recomendação da W3C, também conhecida como linguagem de marcação especialmente criada para armazenar dados. É muito utilizada para transferir dados entre sistemas distindos.

[3] **JSON** (com a pronuncia djeisón), um acrônimo para "JavaScript Object Notation", é um formato leve para intercâmbio de dados computacionais.

[4] **YAML(**Yet Another Markup Language**)** é um formato de serialização (codificação de dados) de dados legíveis por humanos inspirado em linguagens como XML, C, Python.

- um campo endereço que nada mais é que uma coleção de dados em forma de lista, neste caso contendo os endereços de alguma pessoa.

A seguir, um exemplo ilustrado de um banco de dados orientado a documento seria semelhante à Figura 6.

Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Example of a Typical Key/Value Domain

Fonte: BAIN (2009).
Figura 2: Exemplo de banco de dados com documentos e atributos inseridos no mesmo.

Na Figura 2, podemos observar o banco de dados carro composto por dois registros, cada um com uma chave e diversos atributos.

Segundo Bain (2009), neste banco carro, estão contidos documentos com todos os atributos dos automóveis, ou seja, todos os dados relevantes a cada item estão guardados dentro do mesmo. Um domínio ou 'tabela' neste caso, como pode ser observado, possui redundância nos dados, mas esta pode ser aceita, pois não ocupa tanto espaço, melhorando a escalabilidade do servidor, pois eliminam a necessidade de relacionamentos entre tabelas para se obter informações.

Para Lennon (2009, p.4), caso o programador já esteja acostumado com o modelo relacional, aonde se utilizam muito os conceitos de relacionamentos, chaves primárias, chaves estrangeiras, integridade referencial e assim por diante, não é necessário se preocupar, porque estes conceitos não existem no CouchDB e em outros modelos no mesmo segmento. Alterar um banco de dados SQL pode ser uma experiência devastadora para qualquer de administrador banco de dados, pois uma série de dependências e de problemas de integridade entra em jogo. Este não é o caso com um banco de dados orientados a documentos como o

CouchDB. Cada documento é independente, assim você não precisa armazenar valores nulos e também pode definir novos campos para cada documento.

2.1 Características

Segundo Bain (2009), podem ser descritas algumas características deste modelo de banco de dados. Podem-se criar bancos diferentes com vários esquemas:

- itens são definidos por chaves e a cada item podem ser adicionados vários atributos;
- em algumas implementações os atributos são todos de um tipo em outras assumem inteiros, listas e string entre outros;

Mais algumas características, agora sobre acesso aos dados:

- os dados são criados, atualizados, e recuperados usando métodos de chamada por API[5];
- toda a aplicação e os dados que a integram a lógica estão contidos no código da aplicação.

2.2 Vantagens banco de dados orientados a documento

Algumas vantagens abaixo:

- não grava campos em branco;
- rápido e flexível;
- facilidade de utilização e programação.

[5] **API**, Sigla de Application Program Interface. É o conjunto de recursos que permitem criar uma interface com um sistema operacional

2.3 Desvantagens banco de dados orientados a documento

Algumas desvantagens abaixo:

- pouca documentação;
- replicação desnecessária;
- podem ocorrer frustrações na “modelagem” do projeto;
- o programador pode se confundir facilmente.

O próximo capítulo retrata o histórico, funcionamento, características, vantagens e desvantagens do CouchDB, um dos principais banco de dados que usam o modelo orientado a documentos, o NoSQL.

3. APACHE COUCHDB

CouchDB foi criado em meados de 2005 por Damien Katz, escrito originalmente em C++ e traduzido para Erlang[6] em 2008. Este combina um modelo intuitivo de armazenagem em documentos com um poderoso sistema de pesquisa acessível via RESTful[7] e HTTP/JSON Api[8].

Lennon (*apud* OLIVEIRA, 2009, p.19) observa que o termo 'Couch' é um acrônimo para "Cluster of Unreliable Commodity Hardware" (Conjunto de Hardware Não-Confiáveis), "refletindo o objetivo do banco de dados ser extremamente escalável, oferecendo alta disponibilidade e confiabilidade, mesmo quando executando em um hardware que é tipicamente suscetível a falhas".

O CouchDB combina um modelo intuitivo de armazenagem em documentos com um poderoso sistema de pesquisa acessível via RESTful[9] e (HTTP/JSON Api)[10], dentre sua funcionalidades está a replicação bidirecional com detecção de falhas e resolução de conflitos; sua base de dados é baseada em princípios MapReduce[11], em vez de armazenar dados em linhas e colunas, o CouchDB gerencia uma coleção de documentos, os chamados "JSON documents".

Este ainda utiliza um modelo de views baseado em JavaScript para a geração de agregações e filtros, provendo resultados de consultas a partir destes documentos semi-estruturados. Em vez de armazenar dados em linhas e colunas, o CouchDB gerencia uma

[6] **Erlang** é uma linguagem de programação desenvolvida pela Ericsson para suportar aplicações distribuídas.

[7] **RESTful** é uma técnica de Engenharia de Software para sistemas hipermídia distribuídos para WWW.

[8] **HTTP/JSON Api** são protocolos de transferência/comunicação de dados, unindo funcionalidades do protocolo HTTP com funcionalidades Javascript

[9] **RESTful** é uma técnica de Engenharia de Software para sistemas hipermídia distribuídos para WWW.

[10] **HTTP/JSON Api** são protocolos de transferência/comunicação de dados, unindo funcionalidades do protocolo HTTP com funcionalidades Javascript.

[11] **MapReduce** é Originalmente feito para reduzir funções e reconstruí-las em uma forma menor e funcional, utilizando mapfunctions que são funções Javascript.

coleção de documentos, os chamados “JSON documents”. O mesmo usa um método de visualização (Design documents) com funções definidas que agregam dados e filtros para serem computados em paralelo, mais detalhes sobre este banco serão informados no próximo capítulo.

Dentre suas funcionalidades estão à replicação bidirecional com detecção de falhas e resolução de conflitos. Essa base de dados é baseada em princípios de redução de código o chamado MapReduce[12].

3.1 Características Apache CouchDB

Para Anderson (2009), as principais características do CouchDB são:

- atomicidade;
- consistência;
- isolamento;
- durabilidade;
- confiabilidade;
- tolerante a falhas;
- suporte a replicação.

3.2 Motor de armazenamento

Segundo Anderson (*apud* OLIVEIRA, 2009, p.24), o motor de armazenamento do CouchDB é baseado numa versão modificada de uma árvore B+, uma estrutura de dados que permite buscas, inserções e remoções de nós em complexidade logarítmica. Todos os dados internos, documentos e *views* utilizam esta estrutura de armazenamento. Os dados no banco são representados utilizando pares de chave e valor e armazenados na árvore B+, o que permite uma pesquisa rápida por chaves ou por faixas de chaves, com complexidade O(log N)

[12] **MapReduce**, originalmente feito para reduzir funções e reconstruí-las em uma forma menor e funcional, utilizando mapfunctions que são funções Javascript

e O(log N + K), respectivamente, onde N representa a quantidade de chaves a ser pesquisada e K a faixa das chaves.

No caso de uma alteração ao invés de modificar documentos existentes, uma nova cópia do documento é criada e os dados são adicionados à região relacionada. Em seguida, a árvore B+ é alterada para apontar para a nova versão ou o novo documento. Esta atualização dispara a atualização do nó superior (pai) ao nó que está sendo atualizado, que em seguida realiza a mesma atualização no seu pai, seguindo assim até a raiz da árvore, onde irá modificar o seu cabeçalho para apontar para o novo nó raiz.

3.3 Propriedades transacionais

Segundo Anderson (2009), "o termo ACID, cunhado por Härder e Rothermel, refere-se às propriedades de atomicidade, consistência, isolamento de execução e durabilidade, estas garantem que transações de bancos de dados são processadas com confiabilidade".

- Transação**:** uma transação é um programa em execução que forma uma unidade lógica de processamento no banco de dados. Uma transação inclui uma ou mais operações de acesso ao banco de dados — englobam operações de inserção, exclusão, alteração ou recuperação.
- Atomicidade: a propriedade de atomicidade garante que as transações sejam atômicas (indivisíveis). A transação será executada totalmente ou não será executada.
- Consistência: a propriedade de consistência garante que o banco de dados passará de uma forma consistente para outra forma consistente.
- Isolamento: a propriedade de isolamento garante que a transação não será interferida por nenhuma outra transação concorrente.
- Durabilidade: a propriedade de durabilidade garante que o que foi salvo, não será mais perdido.

Graças a estes princípios, os usuários, conectados ao banco para efetuar leitura ou escrita, nunca são barrados; todas as operações acontecem simultaneamente.

De acordo com Anderson (*apud* OLIVEIRA, 2009, p.22):

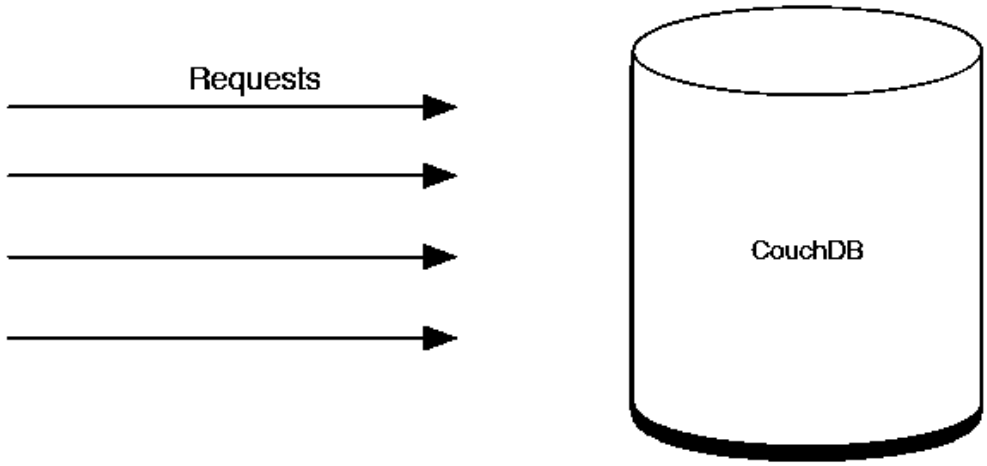
> Cada documento possui um identificador único e global, sendo estes valores do identificador de revisão e do identificador do documento, utilizados para indexação numa árvore B+. A cada atualização no banco de dados, um novo número de revisão é gerado, onde são utilizados posteriormente para encontrar mudanças incrementais no banco de dados. A atualização destas árvores B+ é realizada de forma incremental (atualizações são sempre no final do arquivo) e simultânea quando documentos são salvos ou removidos. Esta árvore é utilizada para encontrar documentos no arquivo de banco de dados.
>
> Quando um documento é atualizado, todos os dados e índices associados são gravados no disco, onde um commit transacional garante que o banco de dados seja deixado em um estado completamente consistente. O commit é realizado em duas fases, onde na primeira fase os documentos e índices são gravados em disco de forma síncrona. A segunda fase realiza a substituição do cabeçalho da árvore B+ que aponta para os documentos, sendo a sua escrita em disco realizada em dois pedaços idênticos e consecutivos, preenchendo os primeiros 4 kilobytes do arquivo de banco de dados. Esta técnica permite que no evento de uma falha do sistema operacional ou queda de energia durante a primeira fase, as atualizações parcialmente gravadas são ignoradas durante o reinício do serviço de banco de dados. Se a queda ocorre durante a segunda fase, uma cópia anterior do cabeçalho do banco de dados já estará gravada, o que garante a coerência de todos os dados já gravados no banco. Fora o cabeçalho do banco de dados, verificações e ajustes não são necessários após eventos de queda de energia ou falha de sistema operacional.

3.4 Aceso aos dados com RESTful/JSON API

REST (Representational State Transfer) é uma sigla que representa a transferência por quatro operações básicas, GET, POST, PUT e DELETE. Estes termos são fundamentais para a comunicação com HTTP.

Se torna muito mais fácil possuir um Api que se comunique e entenda HTTP do que desenvolver um protocolo binário proprietário ou até mesmo um protocolo baseado em XML.

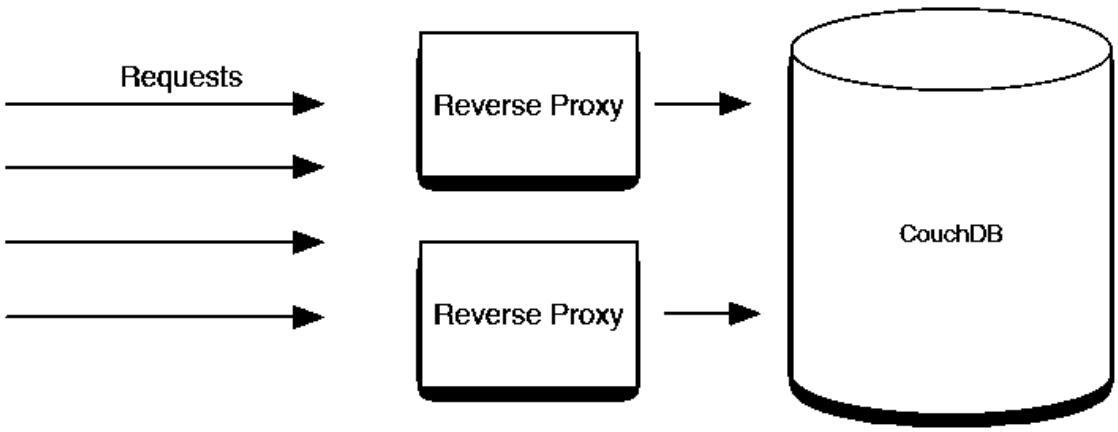
As Figura 3 e 4 demonstram o acesso a dados sem o RESTfull HTTP/JSON Api e com o RESTful HTTP/JSON Api.



Fonte: CHANDLER (2009, p.12).
Figura 3: Sem o RESTfull HTTP/JSON Api.

A Figura 3 está demonstrando o acesso a dados sem o RESTfull. Imagine uma rede social com todo tipo de aplicações requisitando seus dados no SGBD. Iria acabar causando uma total saturação dos recursos do servidor, assim seria necessário melhorar este processo.

Agora na Figura 4, aonde se usa RESTfull, as requisições seriam enviadas a servidores com proxy reverso que tem como objetivo melhorar o acesso a conteúdo WEB por meio do cachê; isso irá diminuir a carga no servidor.



Fonte: CHANDLER (2009, p.13).
Figura 4: Com RESTfull HTTP/JSON Api.

3.5 Compactação

Os bancos de dados gerenciados pelo CouchDB crescem a cada atualização de um documento, mesmo que estas atualizações sejam remoção de documentos. Para recuperar espaço em disco deve-se executar um processo de compactação do banco de dados, que irá expurgar todas as revisões antigas feitas a um documento e reorganizá-los no disco para garantir um acesso seqüencial mais rápido. O processo ocorre de forma seqüencial, nenhuma operação precisa ser adiada ou parada durante a execução do banco de dados.

3.6 Arquitetura de documentos para CouchDB

Um documento é um objeto composto de campos nomeados. Estes campos podem ser cadeias de caracteres, números, datas ou até listas ordenadas e mapas associativos onde não existem limitações de tamanho para campos.

Segundo Lennon (*apud* OLIVEIRA, 2009, p.21), a vantagem de utilizar documentos é que "os dados estão prontos para armazenamento, ao invés de distribuídos entre colunas e linhas ao longo de arquivos de banco de dados. Quando os documentos são salvos no disco, seus campos e metadados são armazenados em buffers, de forma seqüencial, um após o outro".

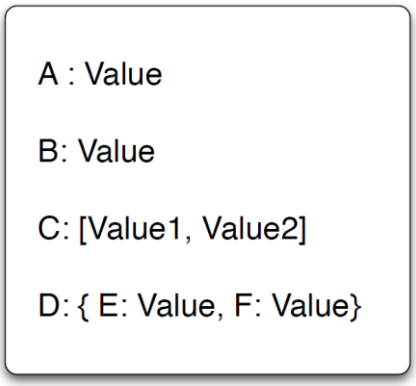
3.7 Documentos JSON

O JSON é projetado para facilitar a leitura e escrita de estruturas. Suas estruturas são baseadas em um subconjunto da especificação da linguagem JavaScript, que permitem a construção de estruturas de dados partindo de dois conceitos: uma coleção de pares chave/valor e uma lista ordenada de valores.

Composta de seis estruturas elementares, o JSON é utilizado pelo CouchDB como forma padrão de armazenamento de dados. A listagem abaixo apresenta como são construídas as estruturas elementares do JSON.

- **Objeto**: sua representação é feita através de chaves.
  Exemplo:{} ou { membros };
- **Pares**: são os pares chave/valor, definidos como uma string e um valor. Exemplo de uma expressão: “nome” : “Kassiano Matteussi”;
- **Membros**: Um ou mais pares de chave/valor;
- **Array**: são listas de elementos, podem ser multidimensionais.
  Exemplo: [{ “chave”: “valor” }, 1234] array de um par e um valor, ambos são elementos;
- **Valor**: são os tipos de dados que o JSON pode assumir: string,
  numérico, objeto, array, true, false ou null.

Podemos verificar estas propriedades na Figura 5 demonstrada abaixo.



Fonte: CHANDLER (2009, p.15).
Figura 5: Arquitetura de um documento.

Em documentos JSON, podem ser representadas complexas estruturas de dados. Seu esquema é livre e não segue os conceitos relacionais conhecidos por muitos.

A seguir mais alguns exemplos.

```
{
"my_key" : "another value",
"different_key" : "another value still!"
}
```

Fonte: CHANDLER (2009, p.16).
Quadro 2: Exemplo de documento com pares de valores.

```
{
 "my_array_key" : ["value 1", "value 2", "etc"]
}
```

Fonte: CHANDLER (2009, p.16).
Quadro 3: Exemplo de documento com lista de valores.

```
{
 "my_hash_key" : { "internal_key" : "internal_value" }
}
```

Fonte:CHANDLER (2009, p.16).
Quadro 4: Exemplo de documento com coleções de valores.

```
{
 "my_array_key" : [
                        {
                            "nested_hash" : "nested hash value",
                            "double_array" : ["scalar"]
                         }
                ]
}
```

Fonte: CHANDLER (2009, p.17).
Quadro 5: Exemplo de documento com arrays de coleções e array com valores.

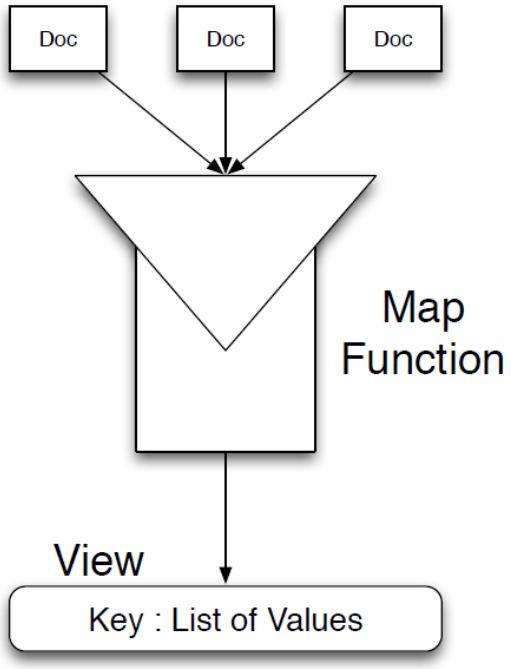
```
{
    "Pessoa" : {
      "Nome" : "Kassiano José Matteussi"
    },
    "Empresa" : {
      "Nome" : "Coor. Inf."
    },
   "Detalhes" : [
      {"Telefone" : "3321-8226"},
      {"Fax" : "49 3321-8226"},
      {"E-mail" : "kassianojm@unochapeco.edu.br"}
    ]
}
```

Quadro 6: Exemplo de documento, cartão de visitas misto.

Estes foram alguns exemplos correspondentes às várias formas que os documentos podem abranger.

3.8 Mapreduce

Este processo buscar diminuir a estrutura dos documentos através de Map's functions, pode ser comparado a um SELECT em SQL. Ele leva para as views somente os dados que você solicitar na função.



Fonte: CHANDLER (2009, p.18).
Figura 6: Estrutura map function.

Agora vamos simular um comando SELECT como sendo uma Map function: SELECT * FROM users WHERE username = "fulano"

Neste caso teria sido selecionando da tabela users todos os valores correspondentes ao username fulano.

Agora vamos montar a estrutura de um JSON document.

```
{
        "type": "user",
        "username" : "fulano",
        "password" : "123567"
}
```

Quadro 7: Conteúdo de um documento qualquer

O Quadro 8 mostra a criação de uma Map function em Javascrip; esta terá ação sobre o conteúdo do documento mostrado no Quadro 7.

```
function(doc){                                  #Esta função Javascript
        if(doc.type = = "users"){               # SELECT * FROM users
                if(doc.usernem= =”fulano”){     #WHERE username
                        emit( doc.username, doc ); #Mostra os dados
                }
        }
}
```

Quadro 8: Exemplo SELECT.

Agora serão demonstrados alguns outros exemplos no mesmo paradigma.

No Quadro 9 pode-se ver a adição de um "campo" ao documento com o tipo definido user usando uma map function.

```
function(doc){
    if(doc.type = = "user"){
        add.doc({"nome":"kassiano", "sobrenome":"jose"});
    }
}
```

Quadro 9: Adicionando campos ao documento.

No Quadro 10 podemos ver o mesmo processo de adição feito no Quadro 17 só que neste momento em SQL.

```
INSERT  INTO user (user, sobrenome) VALUES ('kassiano', 'jose').
```

Quadro 10: Adição em SQL

A seguir o Quadro 11 mostra como é a atualização de um documento usando uma map function.

```
function(doc){
        if(doc.type = = "user"){
                if(doc.nome = = "kassiano"){
                        add.doc({"sobrenome":"matteusi"});
                }
        }
}
```

Quadro 11: Exemplo de UPDATE em map function

A seguir o Quadro 12 mostra como é atualização de um documento usando SQL.

```
UPDATE user SET sobrenome = 'jose' WHERE user = 'kassiano'
```

Quadro 12: Exemplo de update em SQL

Estes métodos apresentados correspondem algumas formas de acesso e alteração dos dados usando map function's.

3. 9 Vantagens Apache CouchDB

As principais vantagens do CouchDB são:

- escalabilidade;
- alto desempenho;
- implementação de sistemas distribuídos e tolerantes a falhas;
- boas comunidades de desenvolvedores;
- desenvolvimento muito rápido.

3. 10 Desvantagens Apache CouchDB

As principais desvantagens, segundo Anderson (2009), são:

- incompatibilidade com bancos de dados orientados a documentos.
- documentação escassa.

3.11 Interface de administração FUTON

Abaixo será descrito a linguagem original em que foi desenvolvida a interface de administração do CouchDB, esta denominada de Futon. Também serão mostradas algumas telas inerentes a funcionalidades da interface.

3.11.1 ERLANG

Para Vinicius (2009), "o CouchDB é feito na plataforma Erlang OTP, uma linguagem de desenvolvimento e programação funcional concorrente". Erlang foi desenvolvida para aplicações em tempo real de telecomunicações com ênfase na extrema confiabilidade e disponibilidade.

Na sintaxe e na semântica, Erlang é muito diferente de linguagens de programação convencionais como C ou Java. Erlang usa "processos" leves e passagem de mensagem por concorrência, não tem compartilhamento de estado de thread e todos os dados são imutáveis. A natureza robusta e concorrente do Erlang é ideal para um servidor de banco de dados.

Dentro da interface do CouchDB, podemos dar destaques a algumas telas que fazem as principais funções no banco de dados, como podemos verificar nas figuras abaixo.

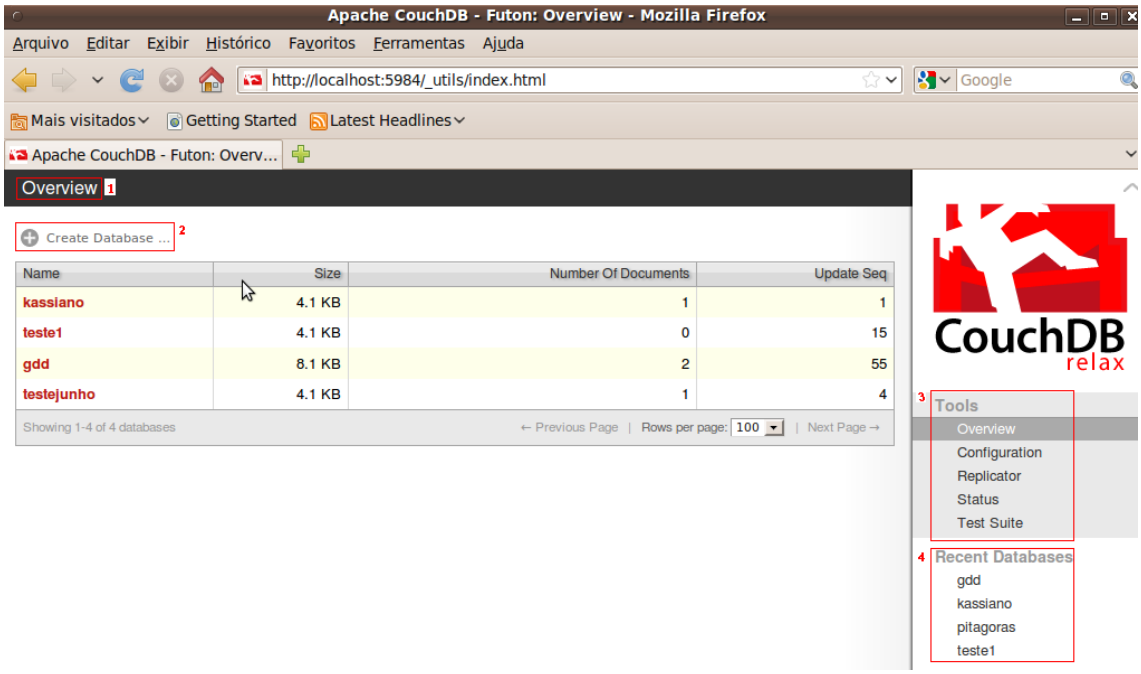


Figura 7: Tela de administração

Na Figura 7, podemos verificar alguns retângulos em vermelho com pequenos números começando pelo número um e terminando no número quatro, abaixo o significado dos mesmos:

- número um: Overview – podemos visualizar todos os bancos de dados já criados no CouchDB;
- número dois: Create Database – criamos um novo banco de dados;
- número três: Tools – leva o usuário a novas telas sendo elas, Overview (lista de banco de dados), Configuration (página com opções para configuração), Replicador (tela para configuração de replicação para outros bancos), Status (leva para uma nova página aonde mostram atividades programadas), Test Suit (lista todos os módulos do CouchDB, desta forma, podendo ser feito um teste geral destes módulos através de um click);
- número quatro – Recent Databases – lista os bancos de dados acessados recentemente.

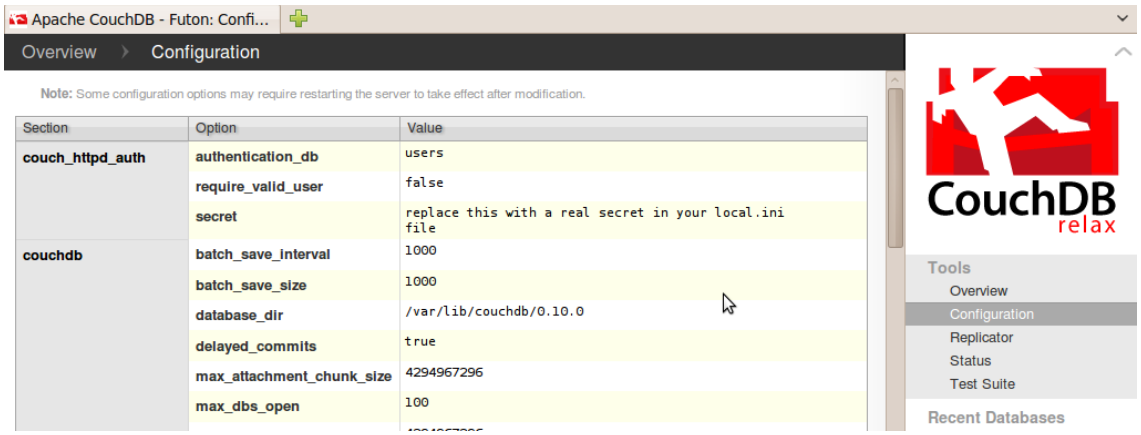


Figura 8: Tela de configuração

Na Figura 8, podemos verificar uma sequência de opções de configuração para o controle da aplicação como pode ser observado.

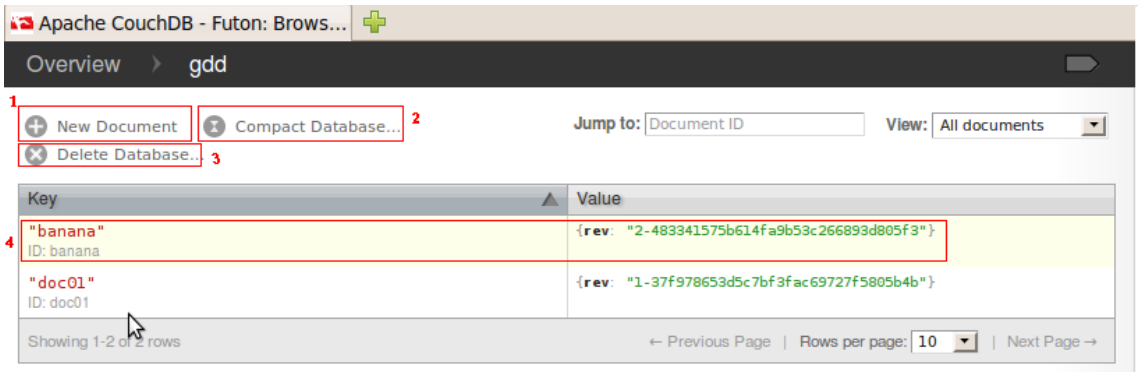


Figura 9: Tela de visualização de documentos

Na Figura 9, podemos verificar alguns retângulos em vermelho com pequenos números começando pelo número um e terminando no número quatro. Segue abaixo o significado dos mesmos:

- número um – New Document: cria um novo documento;
- número dois – Compact Database: Compacta o banco de dados;
- número três – Delete Database: Deleta o banco atual;
- número quatro – lista os documentos pertencentes ao banco.

3.11.2 Considerações finais sobre interface de administração FUTON

Nesta seção serão descritos os pontos positivos e os pontos negativos desta interface.

3.11.3 Pontos positivos

Abaixo alguns pontos positivos da interface:

- navegabilidade é ótima, fácil e rápida;
- visual é confortável e atraente;
- funcionalidades das funções como, criação de bancos, documentos e atributos são dinâmicas e interativas;

3.11.4 Pontos negativos

Abaixo alguns pontos negativos da interface:

- parte de configurações não funciona muito bem, trava e algumas configurações simplesmente não funcionam na interface é necessário editar arquivos no servidor;
- controle de usuários é precário, você somente pode adicionar usuários para administradores ou criar usuários com acesso à visualização;
- gerenciador não funciona no Windows;
- gerenciador não pode controlar mais que um banco, ou seja, cada servidor deve ter sua interface para controle;
- todos os bancos criados são visualizados juntos, não a controle por usuários mesmo que administradores.

Vendo estas deficiências se torna valido o estudo de uma nova interface que venha a contribuir com o banco de dados CouchDB em desempenho, agilidade e segurança.

3.12 Bancos de dados no mesmo segmento

Abaixo serão abordados outros sistemas gerenciadores de banco de dados NoSQL.

3.12.1 Apache MongoDB

MongoDB é um banco de dados orientado a documentos de alta performance, open source e de esquema livre, escrito em C++. Formado por uma mistura entre os repositórios escaláveis e a tradicional riqueza de funcionalidades dos bancos de dados relacionais.

O autor Roberts (2010) descreve que o banco pode ser útil como um repositório de dados simples, rápido e não transacional para aplicações web ou para mecanismos de cache. Você jamais precisará se preocupar com migrações devido a sua natureza esquemas livre.

O mesmo foi projetado para resolver problemas sem grandes exigências de transações e que não são resolvidos facilmente por SGBDR's tradicionais incluindo problemas como escalabilidade e performance.

3.12.1.2 Características Apache MongoDB

A autora Chodorow (2009) aponta as principais características do MongoDB como sendo:

- banco de dados orientado a documentos;
- alta performance;
- open source;
- escrito em c++;
- consultas dinâmicas;
- eficiente para guardar dados como vídeos e fotos;
- suporte a replicação;

- suporte a nuvem (Cloud Computing[13]);
- o MongoDB guarda suas informações em documentos utilizando a sintaxe JSON.

3.12.1.3 Vantagens Apache MongoDB

As principais vantagens do MongoDB são:

- você pode abstrair objetos do mundo real como realmente são: complexos e únicos;
- caso você desejar adicionar um novo campo pode adicionar a um único documento e não a todos os documentos como no caso de uma tabela no modelo relacional.

3.12.1.4 Desvantagens Apache MongoDB

As principais desvantagens do MongoDB são:

- o desenvolvedor precisa tomar cuidado para que suas bases de dados não fiquem muito complexas e confusas.
- instalação complexa e com pouca documentação.

3.12.2 Apache Cassandra

Para Pronschinske (2010), a Apache Cassandra exerce muito bem a função de banco de dados, pois, sendo leve e desenvolvido na plataforma Java não apresenta a sobrecarga de recursos dos SGBD's convencionais. Atualmente o projeto encontra-se incubado pela Fundação Apache.

Segundo o autor Cammyl (2010), Cassandra é um sistema gerenciador de banco de dados distribuído e foi desenvolvido originalmente pelo Facebook, tornando-se um sistema de

[13] A nuvem computacional ou **cloud computing** consiste em compartilhar ferramentas computacionais pela interligação dos sistemas;

código aberto em julho de 2008 como um projeto da Fundação Apache. No final de fevereiro do ano de 2009, ele foi aceito como um Top-Level Project (TLP). Embora ele seja ainda relativamente novo, Cassandra já é utilizado por várias empresas de destaque, como Cisco, Twitter e Digg.

3.12.2.1 Característica Apache Cassandra

As principais características do Apache Cassandra segundo o autor Pronschinske (2010) são:

- descentralizado (não há pontos únicos de falha);
- distribuído;
- tolerante a falhas;
- altamente gerenciável;
- rico modelo de dados;
- replicação;
- usa conceitos de colunas e super colunas;
- open source;
- orientado a documento;
- conceitos de cloud computing.

3.12.2.2 Vantagem Apache Cassandra

Pronschinske (2010) cita as principais vantagens do Apache Cassandra como sendo:

- cada linha é identificada por uma chave única. A chave é uma string e não há limite para o seu tamanho;
- instalação fácil;
- operações de leitura e escrita são muito rápidas;
- manipulação de grande quantidade de dados.

3.12.2.3 Desvantagem Apache Cassandra

As principais desvantagens do Apache Cassandra são:

- documentação escassa;
- pode ocorrer um desconforto para o programador se adaptar ao modelo;
- incompatibilidade com modelo relacional.

O próximo capítulo irá abranger o desenvolvimento do protótipo, as tecnologias utilizadas incluindo mais detalhes sobre o CouchDB, a modelagem realizada, os requisitos da aplicação, bem como os recursos necessários para seu funcionamento; também serão expostas algumas telas da aplicação e descrição detalhada de como foram construídas as mesmas.

4. DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo, serão descritas todas as etapas do processo de desenvolvimento do projeto, bem como a explanação dos assuntos envolvidos.

4.1 Uso do PHP no presente protótipo

Para o desenvolvimento do projeto, a linguagem PHP foi selecionada devido às inúmeras vantagens compostas pela mesma, tais como, sua vasta documentação, facilidade de uso e aprendizagem, bem como sua imensa biblioteca de funções.

4.2 PHP

Segundo o Niederauer (2004, p.19), "PHP significa (um acrônimo[14] recursivo para 'PHP: Hypertext Preprocessor')". É uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico. "Hoje mais de 10 milhões de sites no mundo inteiro utilizam PHP".

O PHP teve inicio por Rasmus Ledorf, que resolveu criar alguns scripts simples para integração com seu site pessoal no ano de 1994. Ledorf queria na verdade somente dar um

[14] *U*m **acrônimo** ou sigla é um agrupamento das iniciais de várias palavras. Acrônimos recursivos são acrônimos onde a expansão inclui o próprio termo, como na definição de funções recursivas. PHP: Hypertext Pre-Processor (Originalmente, PHP significava Personal HomePage).

pouco de vida aos sites estáticos existentes naquele momento, mas ele não imaginava que estava nascendo ali uma das mais poderosas ferramentas para Web, o PHP.

Antigamente o PHP era muito simples, somente se construía um Parser[15] que reconhecia macros e provia de algumas utilidades comuns para Home Pages. Rasmus, Zeev Suraski e Andy Gutmans foram os maiores colaboradores para a criação da primeira versão da linguagem. Dois anos depois, acaba surgindo o PHP 4.0, com muitos recursos adicionais incluindo a orientação a objetos.

Esta linguagem de programação foi desenvolvida originalmente em Perl e após algum tempo foi rescrita em C devido à necessidade de acesso a banco de dados.

No momento do desenvolvimento do presente trabalho, a linguagem está na versão 5.3.2; esta é mantida por várias comunidades de programadores em todo o mundo que contribuem para sua construção e atualização. “O PHP é utilizado principalmente como uma linguagem de script server-side embutida em código HTML” (DALL`OGLIO, 2004, p.26).

A principal diferença do PHP em relação a outras linguagens é a capacidade de interagir com o mundo Web.

Em um Website estático, ao atualizar informações de sua página, sendo por dia ou hora, este deveria fazer as atualização individualmente utilizando o HTML, que posteriormente seriam manualmente alteradas e enviadas a um servidor FTP para que as alterações fossem realizadas e mostradas no Website. Já com o PHP isso tudo é feito automaticamente reduzindo tempo e trabalho, pois é construído um único arquivo tendo que somente alterar o arquivo do banco de dados se for necessário.

4.2.1 Características PHP

A seguir serão descritas algumas características inerentes ao PHP.

[15] Um **Parser** é um programa de computador (ou apenas um componente de um programa) que serve para analisar a estrutura gramatical de uma entrada.

4.2.1.1 Gratuito e aberto

Uma das maiores vantagens do PHP é ser gratuito. O arquivo de instalação pode ser obtido pelo site www.php.net. O PHP 5 apresenta alguns recursos adicionais em relação às versões anteriores, como o suporte à ferramenta SQLite[16], assim funcionando de forma mais eficiente pois não realiza cópias redundante de dados. "Outra característica importante do PHP é que, além de ser gratuito, é um software com código-fonte aberto" (NIEDERAUER, 2004, p.20).

Esta característica possibilita e estimula o desenvolvimento por milhares de programadores espalhados por todo mundo, apresentando uma forma mais rápida e pratica para resolução de problemas. O código-fonte aberto garante flexibilidade, pois possuí suporte às diversas plataformas de hardware, facilmente modificáveis para atender as necessidades específicas da vasta comunidade de programadores.

É recomendável sempre usar uma versão mais recente da linguagem para obter um melhor aproveitamento de novos recursos e suas correções para os defeitos (falhas) encontrados pelos desenvolvedores nas versões anteriores.

4.2.1.2 Embutido ao HTML

"O PHP pode ser mesclado facilmente no HTML, permitindo a união da funcionalidade do HTML com o poder do PHP" (SOARES, 2000, p.6).

As páginas que contém programação PHP geralmente possuem uma extensão *.php. Quando o servidor Web recebe solicitações de páginas que possuem esta extensão, o mesmo irá saber que a página possui linhas de programação e as interpretara. "Porém, você verá que o HTML e o PHP estão misturados, pois começa a escrever em PHP, de repente escreve um trecho em HTML, depois volta para o PHP, e assim por diante" (NIEDERAUER, 2004, p.20).

Desta forma, ao causar um código espaguete[17], a leitura, compreensão e manutenção do código se torna muito difícil.

---

[16] **SQLite** é uma biblioteca C que implementa um banco de dados SQL embutido.

[17] **Código espaguete** Qualifica-se este um trecho de código ou função especifica que não segue as regras da programação estruturada e abusa de desvios, condicionais ou não.

```
<html>
  <head>
    <title>Exemplo</title>
  </head>
 <body>
    <?php
       echo "Olá, isso é um exemplo criado por mim";\\código embutido.
     ?>
 </body>
</html>
```

Quadro 13: Um exemplo básico de código PHP embutido no HTML.

No Quadro 13, pode-se mostra claramente como tags HTML podem ser misturadas a um código PHP, causando assim o chamado código espaguete.

4.2.1.3 Baseado no servidor

O código PHP roda no servidor deixando a parte do cliente mais leve, e assim, proporcionando quase de imediato o processamento de páginas, possibilitando que sejam desenvolvidos sistemas para Web altamente complexos com acesso fácil e rápido para os usuários. “Quando você acessa uma página PHP por meio de seu navegador, todo código PHP é executado no servidor, e os resultados são enviados para o seu navegador” (NIEDERAUER, 2004, p.21).

As linhas de programação PHP não podem ser vistas por ninguém, já que estas são executadas no próprio servidor que apenas retorna o resultado do código executado.

4.2.1.4 Acesso a banco de dados

"Diversos bancos de dados são suportados pelo PHP, ou seja, o PHP possui código que executa funções de cada um" (NIEDERAUER, 2004, p.22).

Para cada um dos bancos de dados acessáveis pelo PHP existe uma série de funções que podem ser aproveitadas nos programas para utilização de todos os recursos. Os bancos de dados utilizados são MySQL, postgreSQL, Oracle, SQL Server e, além destes, pode-se utilizar qualquer outro banco de dados sendo necessário somente uma adaptação dos comandos referentes a ele.

4.2.1.5 Portabilidade

Para Niederauer (2004, p.22), "podemos executar o PHP no Linux, Unix ou Windows NT", pois o PHP é uma linguagem de programação multiplataforma que roda nos mais diversos sistemas operacionais.

4.2.2 Vantagens e desvantagens do PHP

Neste tópico são apresentadas as vantagens e desvantagens mais comuns encontradas na Linguagem de programação PHP.

4.2.2.1 Vantagens PHP

Segundo Souza (*apud* MASETO, 2006, p.22), as principais vantagens do PHP são:

- Licença gratuita;
- Plataforma (SO) gratuita para se rodar ele (GNU/Linux);
- Velocidade de processamento;

- Eficiência de processamento;
- Métodos de segurança eficientes;
- Roda em qualquer tipo de plataforma (SO);
- Código fonte livre;
- Exceptions (para controle de fluxo);
- Orientação a objetos;
- É a linguagem Web mais popular e que mais cresce (em ritmo bem acelerado) no mercado segundo Netcraft18;

Possibilita a utilização dos maiores e mais utilizados Bancos de dados no mercado (Adabas D, InterBase, PostgreSQL, dBase, FrontBase, SQLite);

- Empress, mSQL, Solid, FilePro, Direct MS-SQL Sybase, Hyperwave, MySQL, Velocis, IBM DB2, ODBC, Unix dbm, Informix, Oracle (OCI7 e OCI8), Ingres,Ovrimos, Firebird) sem necessitar de conFiguração externa;
- Está sempre em atualização e tendo corrigidas falhas e adicionados novos recursos;
- É mais estável consumindo menos recursos de hardware do servidor;
- Flexibilidade;
- Componentes nativos, não dependendo de componentes externos para algumas funcionalidades básicas;
- Documentação, controle e reportamento de erros;
- Comunidade de desenvolvimento super participativa e prestativa;
- Planos de hospedagem Web (na grande maioria dos casos) mais baratos e sem nenhum custo extra para a utilização do MySQL em conjunto com o PHP.

Desta forma, pode-se concluir que como o PHP é livre e como possui seu código fonte aberto e rodando em diversos sistemas operacionais, ele se torna muito mais flexível no meio em que é utilizado. Isto ocasiona um processo final mais fácil devido a diversas comunidades de programadores espalhadas pelo mundo, estas por sua vez auxiliam na correção de falhas e ajudam a adicionar novos recursos a esta linguagem.

4.2.2.2 Desvantagens do PHP

De acordo com Maseto (2006, p.23), podemos apresentar as seguintes desvantagens:

- Segundo CANAL Html.(2006), há uma centralização incômoda das variáveis, e propenso a muitas falhas, nos quais um programador desatento pode deixar uma brecha para uma invasão;
- Segundo ARSYS Internet S.L.(2006), a depuração de erros é complexa, ainda que comum em todas as linguagens de script (e mais destacado no próprio Perl);

[18] **Netcraft** é uma companhia de serviços muito conceituada, pode ser visto em mais detalhes em: http://news.netcraft.com/about-netcraft.

- O fato de o PHP ser uma linguagem especificamente concebida para a criação de scripts Web, faz que esteja em desvantagem para realizar outras tarefas, em relação às linguagens de propósito gerais como Perl;

Sem dúvida uma das maiores desvantagens do PHP é ter seu código embutido às tags[19] do HTML, estas sendo separadas somente por blocos de código PHP, formam o chamado código espaguete. Em contrapartida disto podem se utilizar várias técnicas e padrões de projetos como o MVC[20].

4.2.3 Orientação a objetos

Segundo Mattos (2007, p.11), "a orientação a objetos é o paradigma de programação predominante atualmente, e vem aos poucos substituindo a programação procedural, que foi criada no início da década de 1960".

Na OO, um software é composto de objetos com propriedades e operações chamadas de funções (métodos), esses métodos podem ser executados porque sua estrutura de dados é definida juntamente com as mesmas.

4.2.3.1 Objetos

Minetto (2007, P.16) relata que "um objeto como sendo uma representação de um conceito real, possui estados, operações (métodos) e dados (atributos)".

Os objetos representam um conceito real, possuem métodos, estados e atributos. Um objeto pode ser definido da seguinte forma:

[19] **Tags** estruturas de linguagem de marcação que consistem em breves instruções, tendo uma marca de início e outra de fim.
[20] **MVC** é um padrão de arquitetura de Software.

```
<?php
  //Cria um novo Objeto
  $php = new LinguagemProgramacao;
  //Um Objeto possui atributos
  $php->versão;
  //Um Objeto possui um método
  $php->showVersao();
?>
```

Fonte: MINETTO, 2007, p.16.
Quadro 14: Exemplo de um Objeto em PHP

No Quadro 14, podemos abstrair três exemplos com objetos:

- Variável recebendo um novo objeto;
- Variável recebendo um atributo pertencente a um objeto;
- Variável recebendo um método de uma classe.

4.2.3.2 Herança

Segundo Assis (2003, p.2), "esta acontece quando as classes, que são esqueletos para a implementação de características específicas do problema, são implementadas, assim herdando suas características da classe-pai correspondentes".

Desta forma uma classe existente ou criada gera classes filhas com as mesmas características da classe pai com atributos.

```
<?php
  Class LinguagemProgramacaoEstruturada extends LinguagemProgramação{
  function goto($linha) {
          echo "Movendo para linha $linha";
          //implementação
          }
  }
  $cobol new LinguagemProgramacaoEstruturada;
  $cobol->showVersao(); //usando um método herdado da classe-pai
  $cobol->goto(100);       //usando um método da classe-filha
?>
```

Fonte: MINETTO, 2007, p.17.
Quadro 15: Exemplo de herança em PHP.

O Quadro 15 mostra um exemplo do uso de herança em dois casos:

- No uso de uma classe pai no caso ShowVersao() através de um extend;
- No uso de uma classe filha no caso a função goto() que se localiza na classe LinguagemProgramacaoEstruturada.

4.2.3.3 Polimorfismo

Assis (2003, p.2) relata que "o polimorfismo se faz presente no momento que as classes existentes nos projetos são sobrescritas para atender às especificidades da aplicação em desenvolvimento".

O programador não deve preocupar-se com o "como", porque a interface vai reagir a cada evento, sempre sabendo o que se fazer em cada caso.

```
<?php
  class Relogio {
        function descricao() {
                echo "Indico as horas que são… agora!";
        }
  }
  class Livro {
        function descricao() {
                echo "Uma série de páginas com frases escritas";
        }
  }
  $objectos = array(new Relogio(), new Livro());
  foreach($objectos as $objecto);
  echo $objecto->descricao();
?>
```

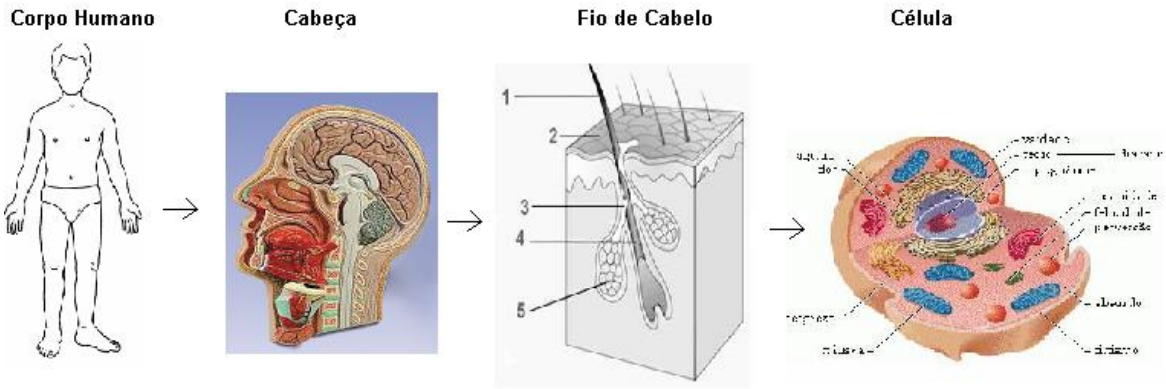
Fonte: Segundo Morais (*apud* MASETO, 2006, p.2).
Quadro 16: Exemplo de Polimorfismo em PHP.

No Quadro 16, podemos ver como funciona o polimorfismo, a variável $objectos recebe um array das classes relogio(), livro() e ao final imprimi um objeto com o método descrição, esse por sua vez trará a descrição das duas classes citadas acima.

4.2.3.4 Abstração de dados

Segundo Assis (2003, p.2), “a abstração acontece em um projeto no momento que se definem quais as funções possíveis de serem implementadas na aplicação específica, ou seja, quais as características que não serão acopladas ao código do sistema que será desenvolvido”.

Esta é a capacidade de olhar apenas uma parte de um todo facilitando o entendimento de códigos mais complexos, pois se divide o código em partes e se analisa cada trecho separadamente.



Fonte: MATTOS (2007, p.19).
Figura 10: Exemplo de Abstração.

Na Figura 10, podemos compreender a abstração de dados da seguinte forma, é mais fácil separar em partes e analisar minuciosamente parte a parte do que analisar tudo de uma só vez.

4.2.4 CURL

O curl ou client URL library foi criado em meados de 1997 por Daniel Stenberg.“O mesmo foi criado com o objetivo de criar um método de comunicação entre diferentes tipos de servidores, utilizando diferentes protocolos, como HTTP, HTTPS, FTP, Gopher, Telnet entre outros” (NIEDERAUER, 2005, p.30).

4.2.4.1 Vantagens CURL

Abaixo algumas das vantagens do Curl:

- funciona em versões do PHP a partir do PHP 4.0;
- vasta biblioteca de funções;
- boa documentação para suas funções;
- facilidade de entendimento e uso de suas funções.

4.2.4.2 Integração com PHP

Neste tópico, será apresentado um exemplo de código PHP semelhante ao que foi usado para o desenvolvimento do protótipo.

```
//$ch faz o papel de identificador da função, o mesmo recebe o curl_init()
//este por sua vez tem o papel de iniciar uma sessão curl.
$ch = curl_init();
//na linha abaixo a variável aux_conect recebe um valor misto composto
//por strings e variáveis com o intuito de formar uma URL.
$aux_conect = "http://".$ip.":".$porta."//".$bde;
//Agora estamos definindo o modo de transferência url e também qual
// será a URL que ira ser alocada para a sessão
curl_setopt($ch, CURLOPT_URL, $aux_conect);
//Agora definimos o método personalizado de solicitação a ser usado, no
//caso atual o PUT
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "PUT");
//Retorna a transferência como uma sequência de caracteres do valor de
//retorno de curl_exec ()
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
//A variavel retorno recebe a sequência de caracteres do curl_exec ()
$retorno = curl_exec($ch);
//finalize a sessão
curl_close($ch);
```

Figura 11: exemplo de integração curl e PHP.

Nesta Figura 11, foi demonstrado um exemplo simples de como criar um banco de dados via solicitação HTTP juntamente com o uso das funcionalidades curl.

4.3 JAVASCRIPT

Para Damiani (2006, p.7), "a linguagem foi criada em 1995 pela Netscape, com o objetivo de permitir a manipulação dinâmica dos elementos de uma página HTML que fosse apresentada no seu navegador: o navegador Netscape. Devido ao seu grande sucesso, a Microsoft decidiu implementar sua versão do JavaScript que seria utilizada no internet Explorer, nascia o JScript".

4.3.1 Características JAVASCRIPT

De acordo com Torque (2010), as principais características do JavaScript são:

- adicionar iteratividade nas páginas HTML;
- linguagem de scripting[21];
- linguagem de programação leve;
- pode ser incorporada diretamente em páginas HTML;
- linguagem interpretada;
- não é necessária licença pra seu uso.

4.3.2 Vantagens JAVASCRIPT

Abaixo algumas vantagens do JavaScript:

- facilidade de manipular o conteúdo HTML;
- suporte cross-browse[22]r;

[21] **scripting** é uma linguagem de programação que permite o controle de uma ou mais aplicações de software.
[22] **Cross-browser** refere-se à habilidade de um site, Aplicação Web, construtor HTML ou script side-client, suportar múltiplos navegadores.

- não é necessário ferramentas extras para trabalhar com o mesmo, pode-se usar o mesmo editor para HTML/PHP/JavaScript;
- pode ser usado para validar os dados;
- grande suporte a expressões regulares.

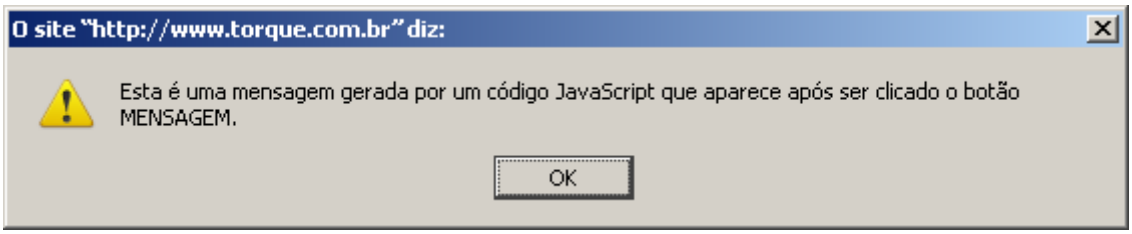
4.3.3 Integração JAVASCRIPT /HTML

Neste tópico será apresentado um exemplo simples em JavaScript.

```
<FORM>
    <INPUT TYPE="button" Value="Mensagem" onClick="alert('Esta &eacute; uma mensagem gerada por um c&oacute;digo JavaScript que aparece ap&oacute;s ser clicado o bot&atilde;o MENSAGEM.')>
</FORM>
```

Fonte: TORQUE (2010).
Quadro 17: Exemplo de formulário com mensagem em JavaScript.

No Quadro 17, podemos observar a integração do JavaScript com HTML, através da criação de uma formulário com o uso de uma função JavaScript, o resultado disso pode ser verificado abaixo na Figura 12.



Fonte: TORQUE (2010).
Figura 12: Exemplo de mensagem em JavaScript.

4.4 AJAX

O autor Niederauer (2007, p. 4) afirma que "a palavra Ajax vem da expressão Asynchronous JavaScript and XML". E é o uso sistemático de JavaScript e XML (entre

outras tecnologias) para tornar o navegador mais interativo com o usuário, utilizando-se solicitações assíncronas de informações. “Isso quer dizer que podemos utilizar o Ajax para fazer uma solicitação ao servidor web sem que seja necessário recarregar a página que estamos acessando”.

4.4.1 Características AJAX

Algumas características inerentes ao Ajax, segundo Ruiz (2008, p.11):

- melhor interatividade usuário e servidor;
- apresentação utilização padrões em XHTML[23] e CSS[24];
- visual dinâmico utilizando DOM[25];
- troca de informações com XML[26] e XSLT[27].

4.5 Usos da Web Service no presente trabalho

No presente trabalho, o uso de Web Service se torna primordial para o funcionamento da base de dados uma vez que todas as manipulações são feitas através de chamadas HTTP, usando RESTful/JSON Api.

[23] **O XHTML**, ou eXtensible Hypertext Markup Language, é uma reformulação da linguagem de marcação HTML, baseada em XML. Combina as tags de marcação HTML com regras da XML;
[24] **CSS** - [Ing. Sigla para Cascading Style Sheets] (Folhas de Estilo Encadeadas). Linguagem padronizada pelo World Wide Web Consortium (W3C), desenvolvida para estabelecer instruções detalhadas à definição de estilos em parágrafos, fontes, bordas, cor ou imagens de fundo, etc;
[25] D**OM** -Document Object Model- São modelos de objetos(com suas propriedades e métodos).
[26] **XML** - eXtensible Markup Language. É uma linguagem baseada em SGML que está sendo desenvolvida pelo W3C para uso em páginas e documentos Web;
[27] **XSL** - Transformations, ou XSLT, é uma linguagem de marcação XML usada para transformar documentos XML.

4.5.1 Web Service

Segundo W3C (2004), com base na definição do Word Wide Web Consortium (W3C), "web services são aplicações autocontidas, que possuem interface baseadas em XML e que descrevem uma coleção de operações acessíveis através de rede, independentemente da tecnologia usada na implementação do serviço".

A Web Service reúne padrões que asseguram a interoperabilidade, estes podem ser acessados via protocolos padrões da Web tais como: HTTP, HTTPS, entre outros; os mesmos podem ser aplicados a diversas plataformas de integração e suportam tanto aplicações ponto a ponto (estruturas mais simples), quanto aplicações distribuídas (estruturas mais complexas).

Figueiredo (2008, p.5) salienta que "os Web Services são por vezes denominados de serviços de aplicações, sendo serviços que foram disponibilizados por servidores Web para utilizadores Web ou para programas ligados à Web, sendo os seus fornecedores denominados, na generalidade, de Application Service Providers".

4.5.2 Características Web Service

Algumas características que podem ser observados em um Web Service:

- uso intenso de XML;
- grande apoio da industria, tais como: IBM, Microsoft;
- baseado em padrões abertos, tais como: XML, HTTP, SOAP, WSDL e UDDI;
- uso de URIs[28] para identificação.

4.5.3 Vantagens Web Service

Abaixo algumas vantagens que são obtidas com o uso de um Web Service;

- interoperabilidade[29] entre sistemas;

[28] **URI** - Uniform Resource Identifier é uma cadeia de caracteres compacta usada para identificar ou denominar um recurso na Internet.

- fácil integração;
- segurança.

4.5.4 Arquitetura Web Service

A arquitetura de um Web Service funciona como um serviço disponibilizado na Internet, o mesmo é descrito via WSDL, registrado via UDDI, acessado utilizando SOAP e com os dados transmitidos sendo representados em XML.

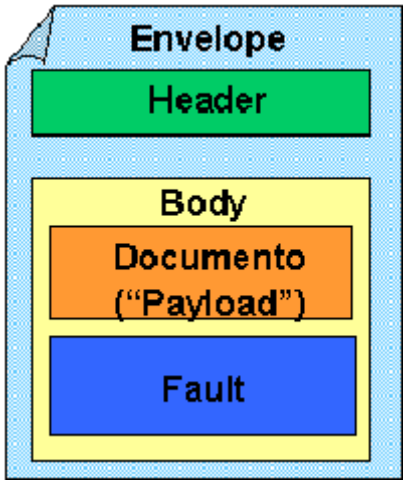
A arquiterura Web service é composta por três tecnologias:

- WSDL (Web Services Description Language) – é a linguagem de descrição dos Web Services, baseada em XML, a mesma é responsável por prover as informações necessárias para a invocação do web service, como sua localização, operações disponíveis e suas assinaturas;
- UDDI (Universal Description, Discovery and Integration) - provê um mecanismo para busca e publicação Web Services;
- SOAP (Simple Object Access Protocol) – protocolo para troca de informações em ambientes distribuídos, composto por XML, é usado para acessar ambientes Web Services e utilizado principalmente com HTTP.

Características SOAP, segundo Devedge (2002):

- definido pelo W3C[30];
- baseado em XML;
- utilizado em Web Services;
- utiliza HTTP como protocolo de transporte;

[29] **Interoperabilidade** - comunicação entre sistemas de forma trasnparente;
[30] **W3C** - World Wide Web Consortium e é a organização oficial para os padrões Web, especialmente HTTP, HTML e XML.



Fonte: DEVEDGE (2002).
Figura 13*:* Estrutura de uma mensagem SOAP

A Figura 13 esboça como seria a estrutura de uma mensagem utilizando SOAP de acordo com Cunha (2002).

- Envelope: Toda mensagem SOAP deve contê-lo. É o elemento raiz do documento XML. O Envelope pode conter declarações de namespaces e também atributos adicionais como o que define o estilo de codificação (encoding style).Um "encoding style" define como os dados são representados no documento XML;
- Header: É um cabeçalho opcional. Ele carrega informações adicionais, como por exemplo, se a mensagem deve ser processada por um determinado nó intermediário (É importante lembrar que, ao trafegar pela rede, a mensagem normalmente passa por diversos pontos intermediários, até alcançar o destino final). Quando utilizado, o Header deve ser o primeiro elemento do Envelope
- Body: Este elemento é obrigatório e contém o payload, ou a informação a ser transportada para o seu destino final. O elemento Body pode conter um elemento opcional Fault, usado para carregar mensagens de status e erros retornadas pelos "nós" ao processarem a mensagem.

4.5.5 RESTful

O termo REST (Representacional State Transfer) vem da dissertação de doutorado de Roy Fielding, publicada em 2000.

Segundo Fielding (*apud* OLIVEIRA, 2009, p.51), REST é "um conjunto de princípios arquiteturais que quando aplicadas como um todo enfatiza a escalabilidade da interação entre

componentes para reduzir a latência de interação, garantir segurança e encapsular sistemas legados".

Fielding também cunhou o termo RESTful e o classificou com os seguintes princípios:

- cliente servidor;
- apoiar sistemas de cache;
- estado nulo[31];
- sistema em camadas, ou seja, suportar escalabilidade;
- stateless[32].
- sistema uniforme composto por 4 diretivas padronizadas, GET, PUT, POST, DELETE.

De acordo com Tilkov (*apud* OLIVEIRA, 2009, p.52), o REST é "um conjunto de princípios que definem como os padrões Web, como o HTTP e URIs devem ser utilizados". O autor destaca que o REST trabalha de uma forma particular, diferente de como se costuma trabalhar com a tecnologia HTTP e o conceito de URI. Tilkov conclui dizendo que "a principal promessa do REST é que se você aderir aos princípios durante o projeto de sua aplicação, você irá acabar com um sistema que explora a arquitetura Web em seu benefício".

Sandoval (2009, p.8) retrata que "um sistema RESTful pode ser implementado em qualquer arquitetura de rede". O mais importante é que não há nenhuma necessidade de criação de novas tecnologias ou protocolos de rede, podendo usar uma rede de infra-estruturas já existentes para criar arquiteturas RESTful. "Consequentemente, a arquitetura RESTful se torna sustentável, renovável, e distribuída" .

4.5.5.1 Características RESTful

Algumas características do RESTful;

- atribuir identificação as unidades de recurso – URI;
- utilização de métodos padronizados, GET, PUT, POST, DELETE;
- apresente recursos com múltiplas representações ou seja como os dados podem ser tratados text/json, text/xml;

[31] **Estado nulo** - cada requisição de um cliente ao servidor deve conter todas as informações necessárias para entender a requisição.
[32] **Stateless -** Comunicação sem controle de estado.

- comunicação sem controle de estado, esta define que não deve haver dados da sessão do cliente armazenados no servidor..

4.5.5.2 URI

Abaixo alguns exemplos de uma URI.

```
http://200.135.202.34/saa
Método: HTTP
Host:  200.135.202.34
Recurso: saa
```

Quadro 18: Exemplo de uma URI - Universal Resource Identifier.

O Quadro 18 demonstra uma URI, que nada mais é que uma URL única.

4.5.5.3 Métodos padronizados

A seguir serão descritos os métodos padronizados, bem com apresentados alguns exemplos:

- GET – operação de Leitura;
- PUT – Relacionada a inserção ou atualização;
- POST - Cria um objeto no servido;
- DELETE – Operações de remoção;

O Quadro 19 representa o uso dos métodos HTTP em uma função curl, idênticas as usadas no presente protótipo.

```
//As funções podem ser usadas de duas formas em php
//Forma 1:
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "GET");
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "DELETE");
//forma 2:
```

```
curl_setopt($ch, CURLOPT_PUT, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_GET, 1);
curl_setopt($ch, CURLOPT_DELETE, 1);
```

Quadro 19: Uso de chamadas HTTP em curl usando PHP.

Na Figura 14 estão demonstrados o que significado o CRUD; também são comparadas as equivalências entre comandos SQL e chamadas HTTP.

Action	SQL	HTTP
Create	Insert	PUT
Read	Select	GET
Update	Update	POST
Delete	Delete	DELETE

Fonte: TYAGI (2006).
Figura 14: CRUD (SQL x HTTP).

4.5.5.4 Recursos com múltiplas representações

No Quadro 20, podemos observar um exemplo do uso de um recurso com múltiplas representações (text); neste caso, idêntico ao utilizado no protótipo, se define o tipo do conteúdo das mensagens a serem criadas, no caso text/json;

```
curl_setopt($curl, CURLOPT_HTTPHEADER, array("Content-Type: text/json"));
```

Quadro 20: Exemplo de recurso com F l ct representações.

4.6 Implementação do protótipo

Nesta seção serão descritos todos os processos que foram utilizados para o desenvolvimento do protótipo.

O protótipo tem por fim gerenciar bancos de dados CouchDB, bem como criar o controle para acesso por usuário sobre estes bancos, além de suprir algumas deficiências encontradas em sua interface de administração original, o Futon.

4.6.1 Preparação do ambiente

Para a preparação do ambiente foram utilizados os seguintes equipamentos:

- 1 Computador Pentium 4, 3.4 ,com sistema operacional Ubuntu 9.10;
- 1 Notebook Acer Core 2 Duo, com sistema operacional Windows Xp Pro;

Também foram usados os seguintes softwares:

- software Geany 0.18, este usado para programação PHP, Java Script e edição de arquivos;
- software Xampp 1.7.3, neste software foram utilizados as ferramentas PhpMyAdmin e o Servidor Apache;
- software DBDesigner, este usado para modelagem relacional do protótipo;
- sofware Apache CouchDB 0.10, este instalado no microcomputador Ubuntu 9.10 com objetivos de testes para o protótipo da interface.

4.6.2 Modelagem do protótipo

A modelagem tem por fim servir a aplicação com registros simples dos usuários, informações dos bancos de dados cadastrados e também guardar a união destes usuários com os bancos.

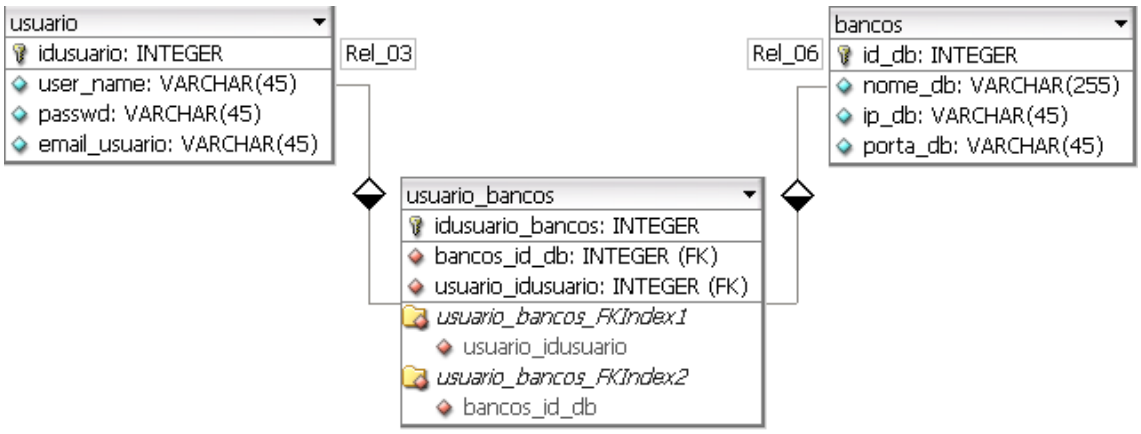


Figura 15: Modelagem

A Figura 15 representa a modelagem do protótipo, esta tem por fim criar uma base de dados aonde pode ser salvos os seguintes registros para:

- usuários: id, nome, senha,e-mail;
- bancos: id, nome, ip, porta;
- usuários: bancos: id, id do banco, id do usuário.

4.6.3 Pré-requisitos para o funcionamento do protótipo

É necessário observar alguns itens para o pleno funcionamento da interface, antes de usá-la. São eles:

- o firewall do servidor deve estar desativado, ou deve conter regras para liberação de leitura e escrita para os clientes;
- o Servidor deve conter o Apache CouchDB 0.10 ou superior instalado para funcionamento, uma vez que o protótipo foi construído para uso nestas versões;
- é necessário que o bind_adress do servidor CouchDB esteja setado com o ip “0.0.0.0”, isso fará com que o banco de dados libere acessos a conexões externas;
- é necessário que as duas pontas estejam conectadas minimamente a uma rede;
- a porta deve estar configurada como 5984/tcp no momento de cadastros de novos bancos;

4.6.4 Apresentação da interface e sua respectiva codificação

Neste tópico serão demonstradas as principais telas da interface, bem como seus principais códigos com respectivas explicações.

A Figura 16 corresponde à tela principal do administrador.

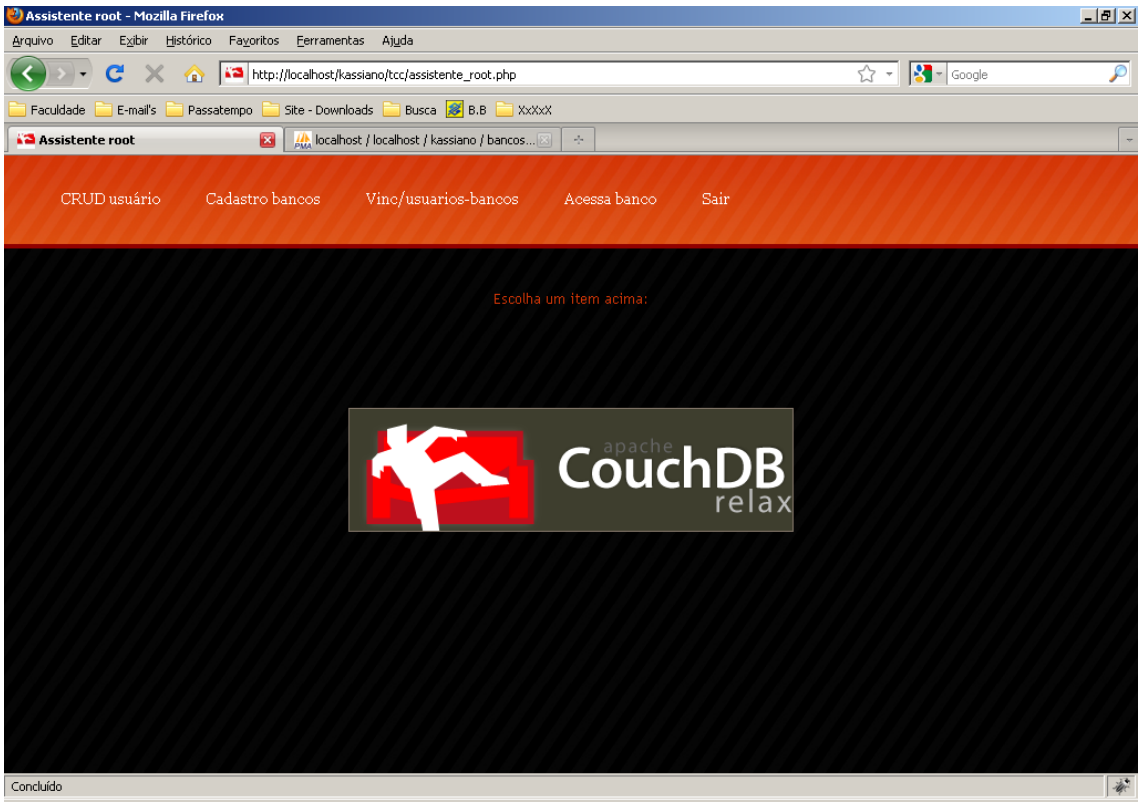


Figura 16: Tela principal do administrador.

Opções encontradas na tela correspondente a Figura 16:

- CRUD usuário – aqui o administrador pode criar, listar, editar e deletar os usuários;
- Cadastro Bancos – aqui o administrador pode criar, listar, editar e deletar os bancos de dados;
- Vinc/usuários-bancos – neste menu, o administrador vincula o usuário aos bancos que o mesmo deve ter acesso;
- Acesso Banco – aqui o sistema trás uma lista detalhada dos bancos de dados criados na máquina, aonde o administrador da mesma pode manipular os bancos com as funções de: inserção, alteração, exclusão, listagem de documentos, consultas ao banco bem como efetuar sua compactação.

A seguir a tela para inserção de usuário.

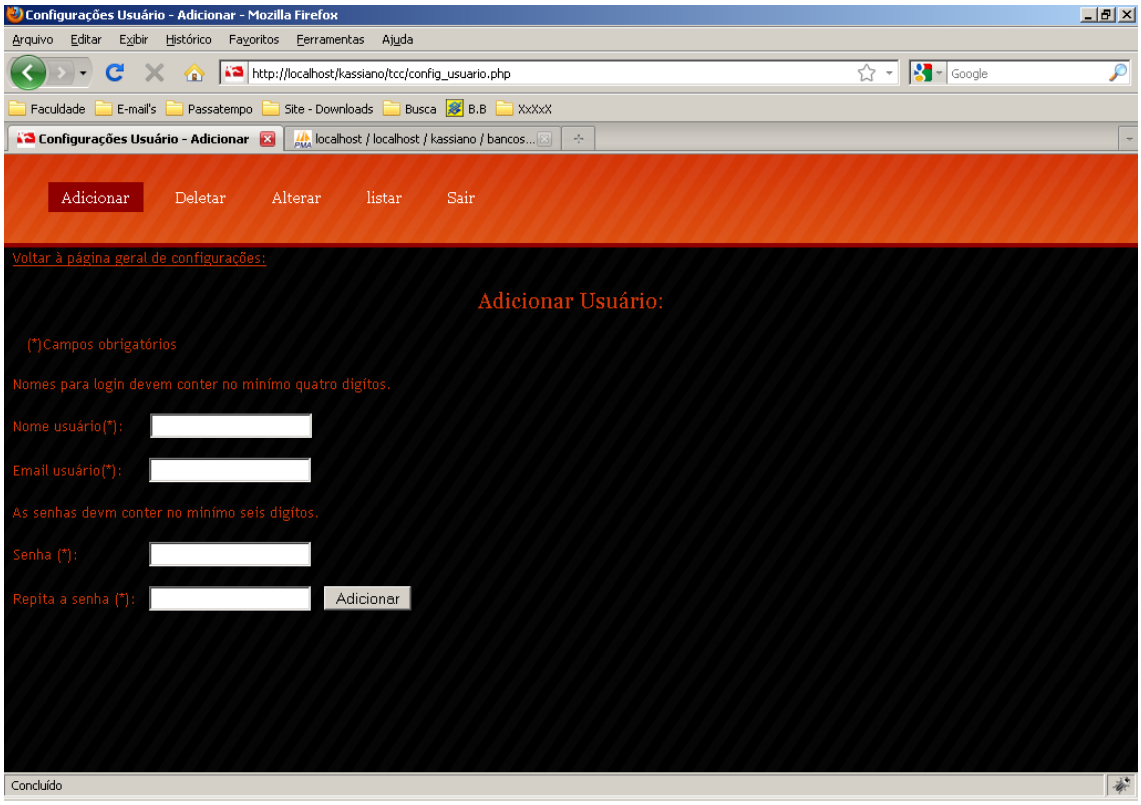


Figura 17: Inserção de usuário.

Na Figura 17, temos a inserção, alteração, exclusão, listagem de usuário. A inserção compreende os seguintes campos mínimos a serem preenchidos, nome usuário, e-mail, senha, confirmação da senha, a validação destes campos é feita com php, javascript.

A seguir tela para inserção de bancos.

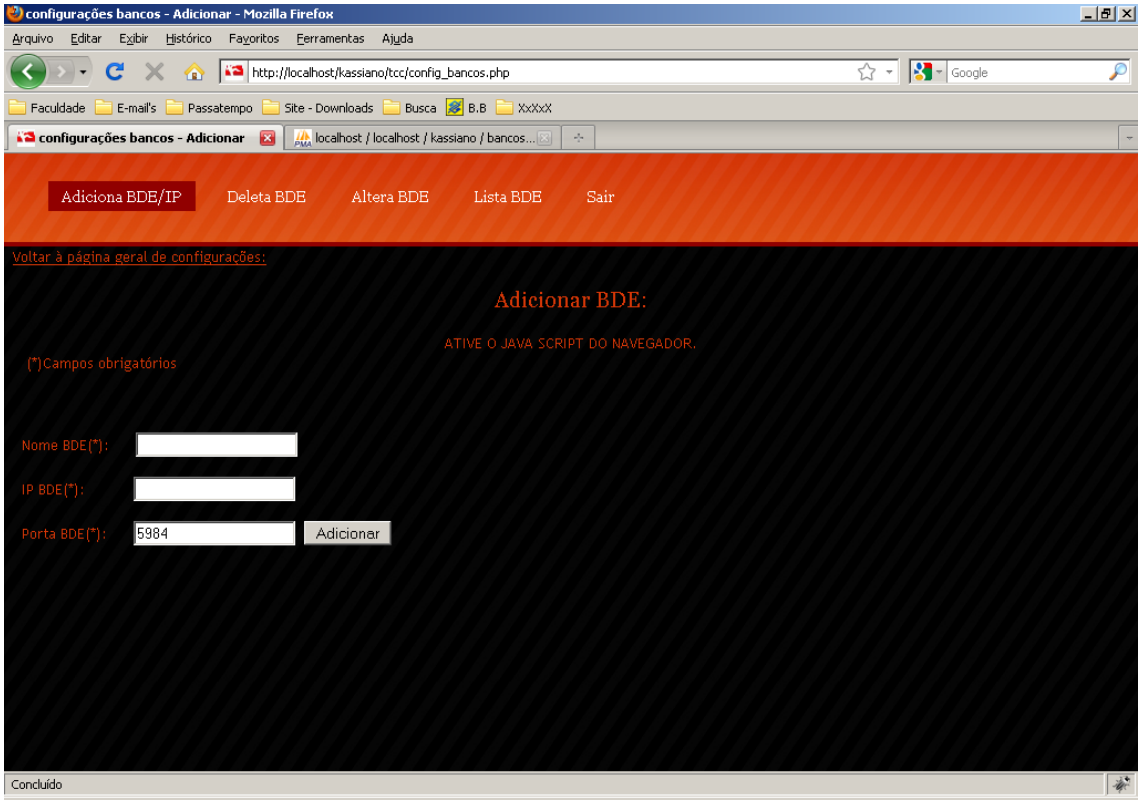


Figura 18: Inserção de bancos.

Nesta Figura 18, encontramos a inserção, alteração, exclusão, listagem de bancos de dados ao servidor; a inserção compreende os seguintes campos mínimos a serem preenchidos: nome, ip e porta (já setada para diminuir a probabilidade de erros no cadastro) do banco, a validação destes campos é feita com php, javascript.

Abaixo está o Quadro 21, que demonstra os códigos para a criação do banco de dados no servidor.

```
//recebendo as variáveis do formulário via post
$bde = $_POST[“bde”];
$ip = $_POST[“ip”];
$porta = $_POST[“porta”];
//inicio  F  l  ct  para controle
$erro = 0;
                                                .
                                                .
//verificando se já existe um banco com o mesmo nome cadastrado
$sql=”SELECT * FROM bancos”;
$c = mysql_query($sql, $con);
while($row = mysql_fetch_array($c)){
        $t = $row[“nome_db”];
        $g = $row[“ip_db”];
```

```
          if( $t == $bde && $g == $ip){
             $erro = 1;
             Exit;
          }
}
                                                    .
                                                    .
//   F se existe porta, ip, nome para banco e verifica se existe erro
   F(($bde && $ip && $porta && $erro != 1)){
   //coloca tudo como minúsculo
   $aux = strtolower($bde);
   // tira os espaço na frente e atrás da string
   $bde = trim($bde);
   //tira todos os espaços e caracteres especiais
    $bde = ereg_replace(“ /* função que trata caracter especiais */ “));
   //inicia a seção do    F   l
    $ch =    F   l_init();
   //criando a url com o banco a ser criado, variável $bde
   $aux_conect = “http://”.$ip.”:”.$porta.”//”.$bde;
   //inserindo na url na seção
      F   l_setopt($ch, CURLOPT_URL, $aux_conect);
   //definindo método HTTP
   curl_setopt($ch, CURLOPT_CUSTOMREQUEST, “PUT”);
   curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
   //executa a operação e retorna a uma variável no caso $retorno
   $retorno =    F   l_exec($ch);
    //fechando seção
      F   l_close($ch);
    //variável $aux666 recebe o conteúdo de retorno da função que descodifica JSON
    $aux666 = json_decode($retorno);
}
 //verifica se existe o banco criado acima
     F ($aux666->ok == 1) {
     //mensagem
  }
```

Quadro 21: Codificação para criação de banco de dados.

A seguir a tela para vincular usuários a bancos de dados.

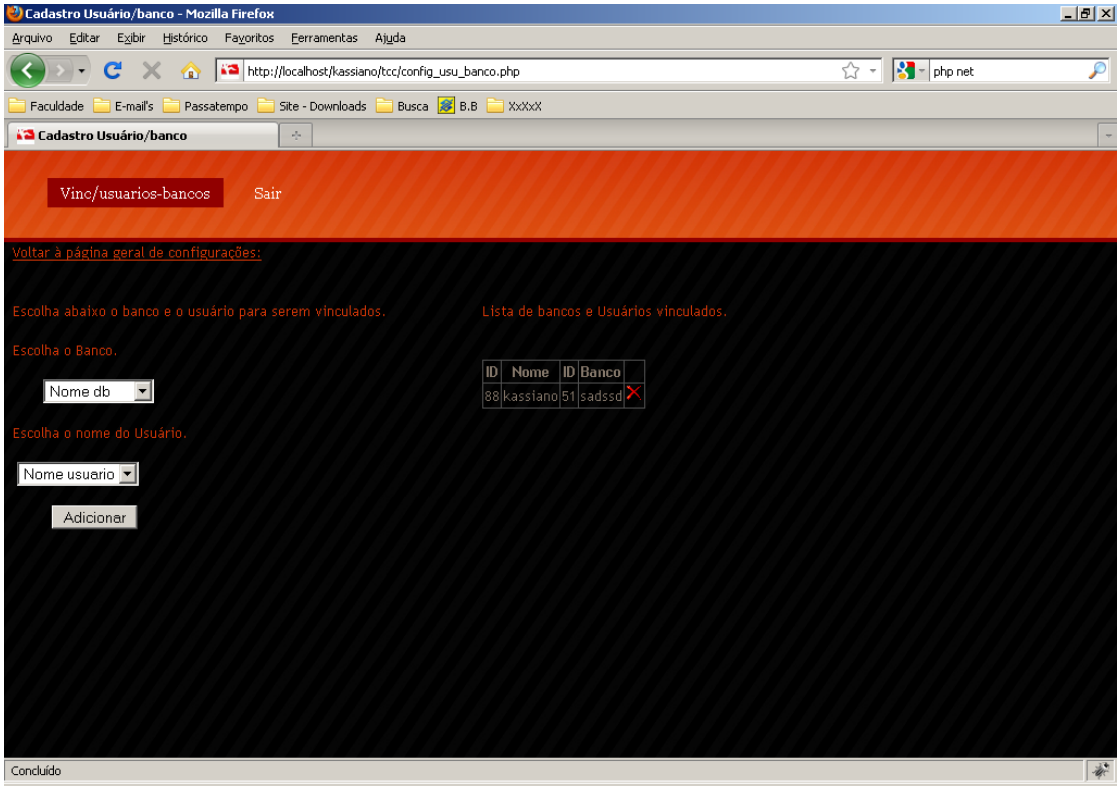


Figura 19: Vinculando banco e usuário.

Na Figura 19, podemos visualizar a tela aonde ocorre à união de um ou mais bancos para os usuários.

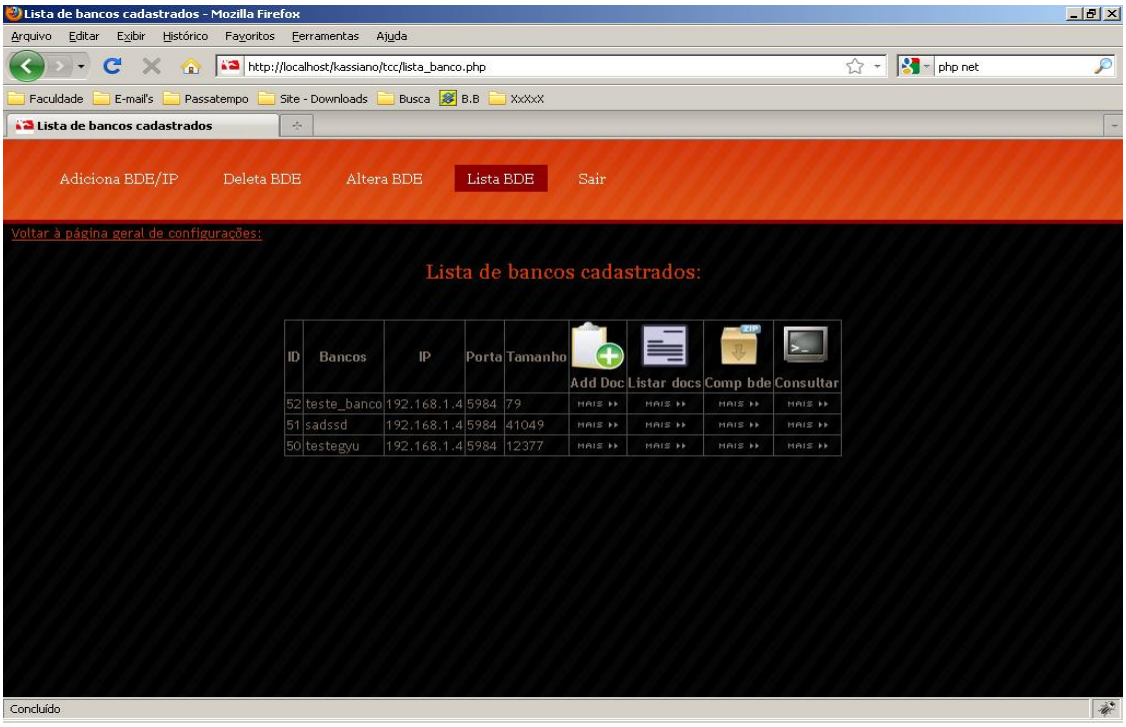


Figura 20: Operações com banco.

Na Figura 20 é possível verificar algumas operações que podem ser realizadas em cada banco; vamos analisar cada uma delas que serão apresentadas a seguir.

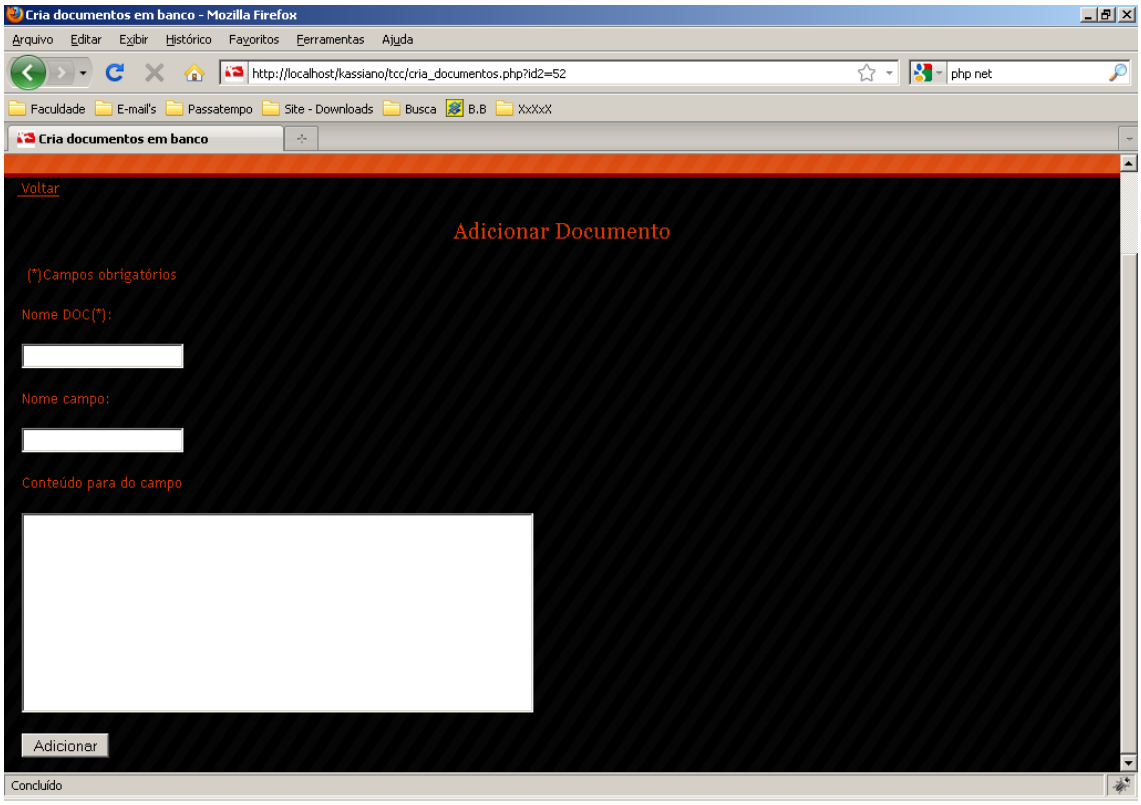


Figura 21: Tela para adicionar documento e atributos.

Nesta Figura 21, o administrador pode cadastrar um documento passando um nome para o documento, um atributo e um valor, as validações são feitas com javascript e a codificação pode ser vista na continuidade.

```
$curl = curl_init();//inicia a seção
$aux_conect = "http://".$ip_v.":".$porta_v."/".$bde_v."/".$documento["_id"];
// remove a chave _id do documento, necessário quando se vai adicionar mais um
//atributo ao documento, caso isso não seja feito o banco retorna um erro de //conflito
$documento["_id"] = false;
$documento = array_filter($documento);
curl_setopt($curl, CURLOPT_URL, $url);
//define o tipo do conteúdo das mensagens a serem criadas, no caso text/json
curl_setopt($curl, CURLOPT_HTTPHEADER, array("Content-Type: text/json"));
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
//definindo a entrada dos dados ao documento
curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($documento));
```

```
$retorno = curl_exec($curl);
curl_close($curl);
$retorno = json_decode($retorno);
//quando um documento é criado com sucesso ele retorno a mensagem {ok ,true}
//sendo assim faço a verificação abaixo
if(isset($retorno->ok) && $retorno->ok == "true") {
   //retorna menssagem
}
```

Quadro 22: Criando documentos e atributos.

O Quadro 22 corresponde à codificação da criação de documentos e atributos.

A seguir será apresentada a tela de alteração de atributos dos documentos.

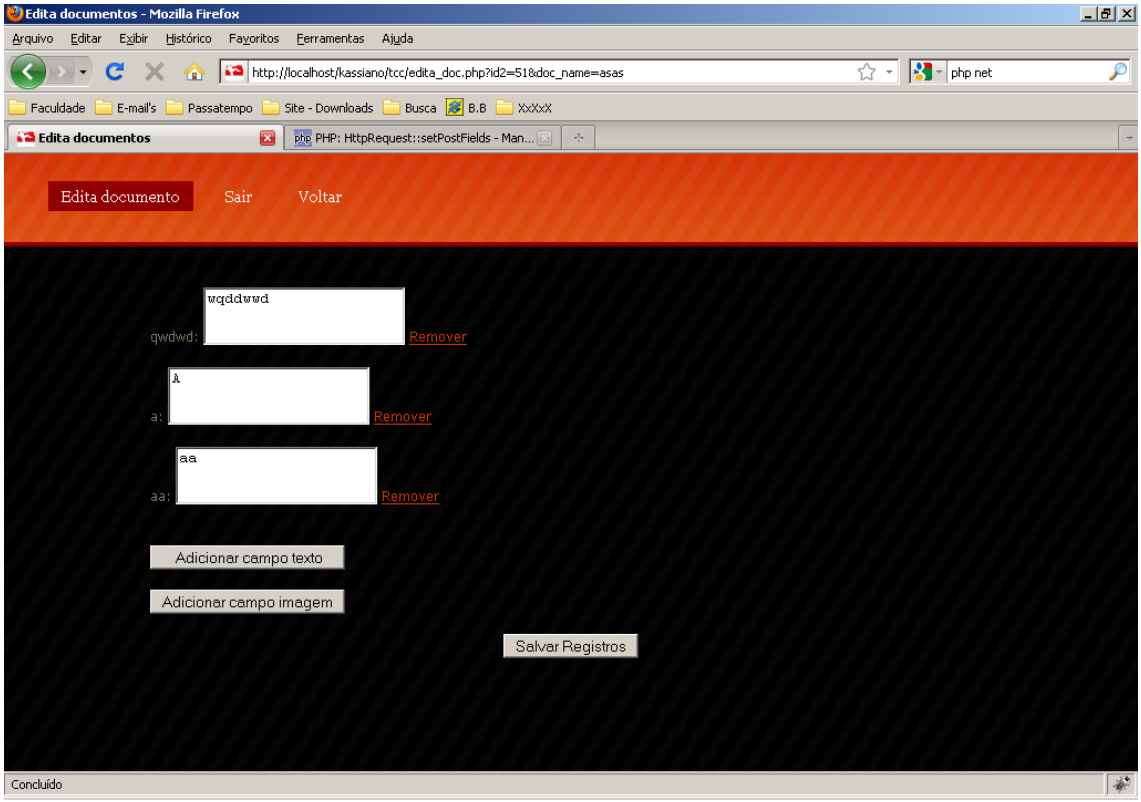


Figura 22: Alteração de atributos de um documento.

A Figura 22 nos mostra a tela de alteração de atributos dentro de um documento. Podemos inserir campos normais, valor, chave ou adicionar documentos tais como: doc, pdf, fotos, entre outros; abaixo segue sua codificação.

```
$curl = curl_init();//inicia a seção
$aux_conect = "http://".$ip_v.":".$porta_v."/".$bde_v."/".$documento;
curl_setopt($curl, CURLOPT_URL, $url);
//definindo método HTTP a ser usado
curl_setopt($curl, CURLOPT_HTTPGET, true);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
$retorno = curl_exec($curl);
curl_close($curl);
//descodifica o documento retornado
$retorno = json_decode($retorno);
//retorna um array com os dados do documento
$documento = get_object_vars($retorno);
                                                .
                                                .
foreach($documento as $campo => $valor){
   if($campo == "_id" || $campo == "_rev"){
   //imprimi todos os dados existentes do documento
   }
}
                                                .
                                                .
//JavaScript
//adiciona campo texto se o usuário desejar, a adição do campo imagem tem //
//codificação semelhante
adicionaCampoTexto = function() {
        var adiciona = criaElemento();
        var adiciona1 = criaElemento();
        var campos = document.getElementById('campos-div');
        var fragmento = document.createDocumentFragment();
        var div = criaElemento("div",{'class':'campo'});
        adiciona.innerHTML = "Campo";
        adiciona1.innerHTML = "Valor";
        var nome = criaElemento("input", {
                                                'class': 'nome',
                                                'name': '',
                                                'type': 'input',
                                                'value': ''
        });
}
//após isso é chamada um função semelhante ao de criar documentos como na
//Figura, mudando somente o método HTTP
```

Quadro 23: Codificação para alteração de um documento.

O Quadro 23 mostra a codificação da alteração de um documento, ou, criação de novos atributos. Na seqüência tela para consultas.

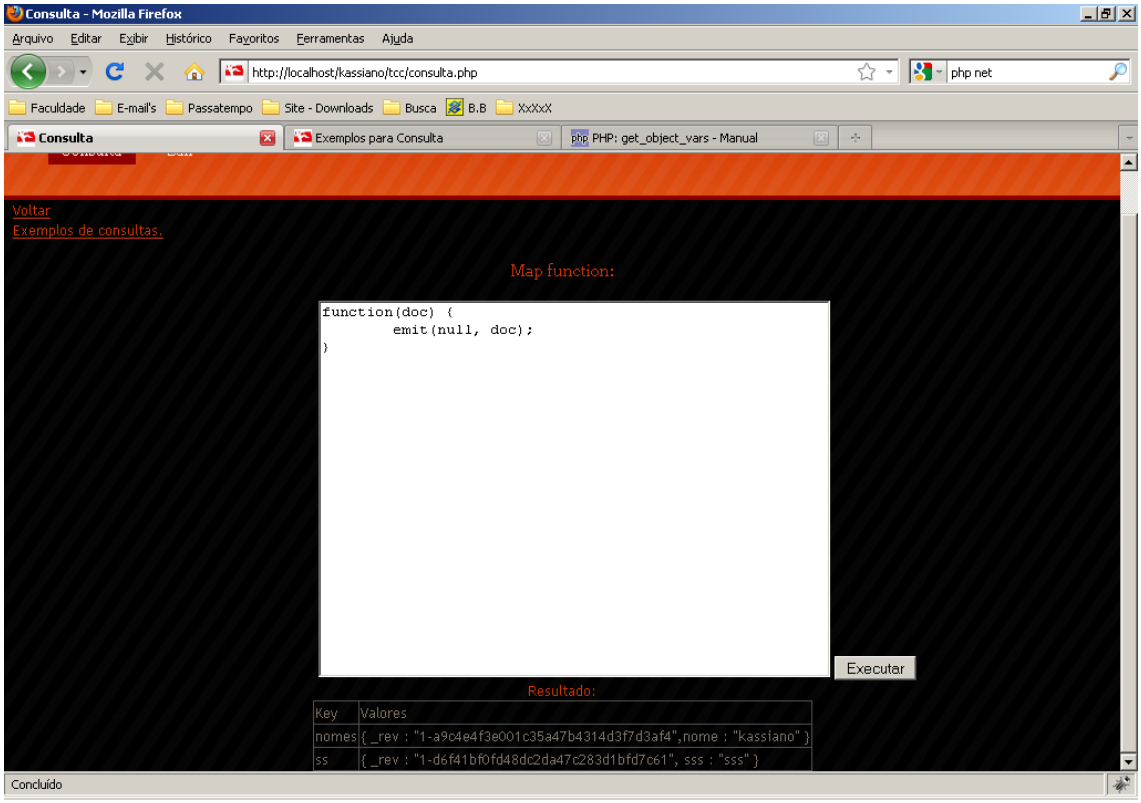


Figura 23: Tela para consultas.

A Figura 23 corresponde a tela para consultas; é possível filtrar dados dos documentos contidos em certo banco de dados através de uma função simples em javascript. Abaixo a Figura que demonstra a codificação desta tela.

```
//vê se tem conteúdo no text área.
if(array_key_exists("query", $_POST)) {
                                        .
                                        .
    $curl = curl_init();
    //cria uma visualização temporária para o documento com o uso do método pré
   //defido _temp_view
    $url = "http://".$ip_v.":".$porta_v."/".$bde_v."/".$documento."/_temp_view";
   //setando os dados do array para map, map tem como objetivo gerar
   //consultas rápidas
    $documento = array("map" => $_POST["query"]);
    curl_setopt($curl, CURLOPT_URL, $url);
    curl_setopt($curl, CURLOPT_HTTPHEADER, array("Content-Type: text/json"));
    curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "POST");
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($documento));
    $retorno = curl_exec($curl);
   curl_close($curl);
   $retorno = json_decode($retorno);
```

```
}
//vê se existe algum retorno do documento, se existir imprimi as informações do
//mesmo
if (($retorno->rows && $documento) != NULL){
   foreach($retorno->rows as $documento){
        echo $documento->id ;
        $pares = array();
        $valores = get_object_vars($documento->value);
        foreach($valores as $campo => $valor) {
            if($campo != "_id") {
               $pares[] = "$campo : \"$valor\"";
            }
        }
        echo implode(",", $pares);
    }
}
```

Quadro 24: Codificação tela consulta.

4.6.5 Vantagens da interface produzida

Abaixo algumas vantagens correspondentes a nova interface produzida:

- gerenciamento multiplataforma;
- gerenciamento distribuído;
- controle de acesso;
- segurança;
- flexibilidade;
- controle de usuários;
- controle sob os dados com maior facilidade;
- controle de níveis de acesso sobre os bancos aos usuários;
- com adaptações no código pode ser usada para gerenciar outros bancos de dados SGBD NoSQL.

5. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Esta seção retrata as conclusões obtidas com a produção deste trabalho bem como os trabalhos futuros

5.1 Conclusão

O presente trabalho teve como objetivo realizar um protótipo de interface Web com PHP para gerenciamento de banco de dados CouchDB. Mesmo com algumas dificuldades quanto à documentação e suas funcionalidades durante o desenvolvimento do protótipo, pode ser dizer que ao final obteve-se êxito.

O CouchDB é um ótimo banco de dados. É um novo paradigma que esta crescendo rápido, graças a sua consistência. Sua documentação melhora gradativamente junto com seus grupos de discussões e sites com conteúdos e informações sobre as suas funcionalidades.

No presente momento em que foi produzido este protótipo, pode-se observar que o esquema de funcionamento RESTful é uma ótima abordagem em comparação com o modelo tradicional Client/Server. Esta abordagem se torna válida devido a grande facilidade de se trabalhar com este modelo de Web Services, no desenvolvimento de aplicações para Web.

O PHP teve papel fundamental no desenrolar do processo, pois combina poderosos mecanismos de programação aliados a sua facilidade de uso e aprendizagem, outro sim se pode observar a evolução obtida pelo acadêmico em técnicas de programação no decorrer do desenvolvimento da interface.

Os estudos apresentados no presente trabalho demonstram que o paradigma NoSQL é muito bom e vem ao mercado para competir com o tão consolidado modelo relacional, em princípio, no domínio da Web e, mais tarde, em programação para o ambiente desktop.

Com este protótipo o controle da aplicação CouchDB fica muito mais fácil, pois combina controle de usuários com gerência sobre os bancos e suas informações, ficando muito semelhante a ferramentas disponíveis para os SGBDR.

Pode-se dizer que trabalhar com PHP e CouchDB é muito confortável e divertido, pois depois que os conceitos estão compreendidos, o desenvolvimento se torna descomplicado e altamente produtivo.

5.2 Trabalhos futuros

Com relação aos trabalhos futuros, pode-se citar:

- refazer o protótipo usando Zend Framework, Cake PHP ou outro;
- utilizar framework JavaScript com jQuery ou Dojo;
- criar uma interface que converta dados do modelo relacional para o modelo orientado a documentos;
- criar uma interface que converta os dados de um gerenciador para outro gerenciador NoSQL.

6. REFERÊNCIAS

AMARAL, Fernando. **Porque não estamos utilizando banco de dados Orientados a Objetos?,** 2009. Disponível em: http://www.fernandoamaral.com.br/Default.aspx?Artigo=17. Acesso em: 01/06/2009.

ANDERSON, Chirs. 2009. **Apache CouchDB: The definitive Guide**. Disponível em: http://couchdb.Apache.org/index.htm Acessado em 05/06/2009.

APACHE. **HTTP**. 2010.Disponivel em: http://wiki.Apache.org/couchdb/HTTP_view_API. Acessado em: 25/06/2010.

ASSIS, Semíramis Ribeiro de. **Frameworks:** Conceitos e Aplicações, 2003. Disponível em: http://www.frb.br/ciente/Textos%20CienteFico%202003.2/INFO/Eng%20Software/Frameworks%20-%20Conceitos%20e%20Aplica%E7%F5es.pdf. Acessado em 06/05/2009.

BAIN, Tony**. Is the Relational Database Doomed?**, 2009. Disponível em: http://www.readwriteWeb.com/archives/is_the_relational_database_doomed.php. Acessado em: 03/06/2009.

BRITO, Ricardo. **Banco de Dados NoSQL x SGBDR**. 2010.

CAMMYL. **Apache Cassandra**. 2010. Disponível em: http://under-linux.org/content/riptano-oferece-suporte-para-o-Apache-cassandra-808/. Acessado em: 17/05/2010.

CASSANDRA. **Apache Cassandra**. 2010. Disponível em: http://cassandra.Apache.org/. Acessado em: 17/05/2010.

CHANDLER, Christopher. **CouchDB in Action:** Greenwich, Manning, 2009.

CHANG, Fay. **Big Table**. 2006.

CHODOROW, Kristina, **Manual MongoDB**. 2009. Disponível em: http://www.mongodb.org/display/DOCS/Manual. Acessado em: 17/05/2010.

CUNHA, Davi **Soap**. 2010. Disponível em: http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html. Acessado em: 11/06/2010.

DALL`OGLIO, Pablo. **PHP-GTK**: Criando aplicações gráficas com PHP. São Paulo: Novatec, 2004.

ESPAKE, Patrick. **MongoDB.** 2010. Disponível em: http://www.slideshare.net/patrickespake/mongodb-3221865. Acessado em: 17/05/2010.

FARIA, Alessandro. **Apache Cassandra**. 2010. Disponível em: http://www.vivaolinux.com.br/artigo/Apache-Cassandra-NoSQL-uma-tecnologia-emergente. Acessado em: 17/05/2010.

_____. **Apache Cassandra**. 2010. Disponível em: http://www.dicas-l.com.br/pdf/20100402.pdf. Acessado em: 17/05/2010.

FIGUEUREDO, Fernando. **XML E Web Service**. 2008. Disponível em: http://www.ticua.net/Apontamentos/TRABALHO1_FINAL_ASES_E_REIS.pdf. Acessado em 11/06/2010.

FROSSARD, Afonso Celso Pagano. **Modelos para Sistema de Informação**: Conceitual, Lógico e Físico, 2007. Disponível em: http://www.flf.edu.br/midias/FLF.EDU/v5mono1.pdf . Acessado em 01/06/2009.

LENNON, Joe. **Begining CouchDB.** USA: Apress, 2009.

MACORATTI, José Carlos. **Banco de dados**. Disponível em: http://www.macoratti.net/banco.htm. Acessado em 01/06/2009.

MASETO, Jhony Maiki. **Análise avaliativa entre Frameworks de PHP**, 2006. Monografia (Conclusão do Curso de Graduação em Ciências da Computação) – Universidade Comunitária Regional de Chapecó, Chapecó, 2006.

MATTOS, Érico Casella Tavares de. **Programação de softwares em JAVA**. São Paulo: Digerati Books, 2007.

MINETTO, Elton. Luís. **Frameworks para Desenvolvimento em PHP**, São Paulo: Érica, 2007.

_____. **Zend Framework**. 2009. Disponível em: http://www.eltonminetto.net/docs/zend_Framework_xxe_II.pdf. Acessado em 06/06/2009.

MORAES, Leonardo. **Introdução ao CouchDB**. Disponível em: http://leonardomoraes.com.br/blog/. Acessado em 06/06/2009.

NEVES, Denise Lemes Fernandes. **PostgreSQL**: Conceitos e Aplicações: Érica, 2002.

NIEDERAUER, Juliano. **Desenvolvimento Websites com PHP**: Aprenda a criar Websites dinâmicos e interativos com PHP e bancos de dados. São Paulo: Novatec, 2004.

_____. **PHP 5**. São Paulo: Novatec Editora LTDA, 2005.

NIEDERAUER, **Ajax.** 2010. Disponível em: http://www.linuxmall.com.br/files/_product/5/4635/cap1_o_que_e_ajax.pdf. Acessado em 11/06/2010.

OLIVEIRA, Leonardo Eloy. **Estado da arte de banco de dados orientados a documento, 2009**. Monografia (Conclusão do Curso de Graduação em Ciências Tecnológicas) – Universidade de fortaleza – UNIFOR, Ceára, 2009.

PRONSCHINSKE, Mitchell. **Apache Cassandra**. 2010. Disponivel em: http://css.dzone.com/articles/cassandra-nosql-database. Acessado em: 17/05/2010.

RAMALHO, José Antônio Alves. **Oracle 8i**: Berkeley Brasil, 1999.

REVERBEL, Francisco. **Web Service**. 2006. Disponível: http://www.ime.usp.br/~reverbel/SOD-06/trabalhos/fachada-ws/node2.html. Acessado em: 11/06/2010.

ROBERTS, Ric, **MongoDB**. 2009. Disponível em: http://www.rubyinside.com.br/iniciando-com-mongodb-e-ruby-1623. Acessado em: 17/05/2010.

ROSA, Everton. **Apache Cassandra**. 2010. Disponível em: http://dotinfo.wordpress.com/2010/04/03/dicas-l-Apache-cassandra-nosql-uma-tecnologia-emergente/. Acessado em: 17/05/2010.

RUIZ, David, **Ajax**. 2010. Disponível em:_http://www.slideshare.net/wupsbr/web-20-e-ajax-parte-1. Acessado em: 11/06/2010.

SOARES, Walace. **Mysql:** Conceitos e aplicações. São Paulo: Érica , 2000.

SOARES, Walace. **Programando em PHP**: Conceitos e aplicações. São Paulo: Érica , 2000.

SOSA, Armando. **CakePHP**. Disponível em: http://www.cakephp.com.br. Acessado em 18/05/2009.

TAPAJÓS, Marcos, **Databases dont scale yet**. Disponível em: http://en.blog.improveit.com.br/articles/2008/11/16/databases-dont-scale-yet. Acessado em 05/06/2009.

THOMPSON, Marco Aurélio. **JAVA 2 & Banco de Dados**. São Paulo: Érica, 2002.

TORQUE, **Java Script**. 2010. Disponível em:http://www.torque.com.br/tutoriais/java_script/janela_alerta/index.htm. Acessado em: 11/06/2010.

TAYT-SON, Leonardo Bogéa, **Web Service**. 2010. Disponível em: http://www.gta.ufrj.br/grad/05_1/webservices/definicao.htm#Um_Pouco_de_Hist%C3%B3ria Acessado em: 11/06/2010.

TYAGI, Sammer, **RESTful**, . 2006. Disponível em: http://java.sun.com/developer/technicalArticles/WebServices/restful/. Acessado em: 11/06/2010.

_____. **Propriedades ACID**. 2009. Disponivel em: http://www.p3m.com.br/blog/propriedades-acid/