Desenvolvimento de uma aplicação móvel para manipulação de documentos utilizando o banco de dados MongoDB

Luiz Cláudio Afonso dos Santos, Evaldo de Oliveira da Silva

Curso de Bacharelado em Sistemas de Informação
Centro de Ensino Superior de Juiz de Fora (CES/JF) – Campus Academia
36016-000 – Juiz de Fora – MG– Brasil

luiz.santos89@yahoo.com.br, evaldo.oliveira@gmail.com

Abstract. *This work shows a prototype developed as manner to apply concepts and techniques approached, in order to verify the usage of a NoSQL Database named MongoDB. It was developed a mobile application based on MongoDB database during the execution of the research project entitled "Services Web for data and document maintenance" what has happened in CESJF. The mobile application was developed using the Android webview and MongoDB for the purpose of handle documents and data non structured. Finally, is showed the results obtained through the implementation of the tool that treats the data for storage in the database and that analyzes them*.

Resumo. Este trabalho mostra um protótipo desenvolvido como forma de aplicar conceitos e técnicas abordadas, a fim de verificar o uso de um banco de dados NoSQL chamado MongoDB. Foi desenvolvido um aplicativo mobile baseado em banco de dados MongoDB durante a execução do projeto de pesquisa intitulado "Web Services para manutenção de dados e documentos", o que tem acontecido no CESJF. O aplicativo móvel foi desenvolvido usando o webview Android e o MongoDB com o objetivo de manipular documentos e dados não estruturados. Por fim, são apresentados os resultados obtidos através da implementação da ferramenta que trata os dados para armazenamento no banco de dados e os analisa.

1. Introdução

O constante avanço no número de usuários da web alavancado pelas facilidades oferecidas por dispositivos e redes de dados móveis, seja para acessar redes sociais, para atividades profissionais e/ou pessoais, para comunicação, compartilhamento ou também buscas de informações, tem como consequência o aumento no volume de conteúdo disponível na internet e pelas organizações.

Atualmente, existe a necessidade de controlar ou manter dados de diversos formatos, sendo estes binários ou texto. Os dados produzidos pelas organizações estão presentes em diferentes fontes, por exemplo: banco de dados relacionais, documentos controlados, procedimentos operacionais, e redes sociais, tais como, *Facebook* ou *Twitter*. Diante deste cenário, é possível observar uma diversidade de dados estruturados ou não (LOSCIO et al., 2011), os quais necessitam ser armazenados, extraídos e analisados. Neste contexto, os bancos de dados chamados NoSQL (*Not Only SQL*) estão sendo amplamente sendo utilizados devido a facilidade de manipulação dos

tipos de dados citados (MONGODB, 2018).

A partir da utilização de tecnologias de banco de dados NoSQL, este projeto tem como objetivo desenvolver o compartilhamento de dados extraídos de documentos armazenados a partir do banco de dados MongoDB. Além disso, tem como plataforma de desenvolvimento para manipulação dos tipos de dados citados, as tecnologias PHP e *webview* (CHARLAND, 2011; PHP, 2018).

A aplicação produzida neste trabalho também é resultado dos estudos desenvolvidos pelo Projeto de Iniciação Científica chamado "Serviços Web para manutenção da dados e documentos" registrado no Centro de Extensão do Centro de Ensino Superior de Juiz de Fora.

O restante do artigo segue organizado em seções a serem definidas. A Seção 2 descreve os conceitos que definem demais conceitos sobre as tecnologias utilizadas para o desenvolvimento da proposta deste trabalho. A Seção 3 discute características sobre o a proposta da aplicação descrita nesta Introdução. A Seção 4 apresenta o protótipo desenvolvido como proposta de implementação da manipulação de arquivos. Finalmente, a Seção 5 apresenta as considerações finais e trabalhos futuros.

2. Referencial Teórico

Nesta seção serão descritos os conceitos que fundamentam este trabalho, as técnicas, procedimentos e implementação da solução proposta.

2.1. Banco de dados NoSQL

O grande volume de dados gerado por aplicações web juntamente com base em requisitos indispensáveis para manipulação de grandes de volumes de dados, tais como a escalabilidade e o elevado grau de disponibilidade, têm contribuído para o surgimento e a implementação cada vez maior de banco de dados não-relacionais. As redes sociais e a necessidade de armazenamento e extração de informações oriundas de documentos, por exemplo, requerem o gerenciamento de grandes quantidades de dados não estruturados que são produzidos e processados diariamente por milhões de usuários (LOSCIO et al., 2011 p. 1), no conceito de *Big Data*.

O termo *Big Data* pode ser, resumidamente, definido como uma coleção de bases de dados tão complexa e volumosa que se torna muito difícil (ou impossível) fazer algumas operações simples (por exemplo, remoção, ordenação, sumarização) de forma eficiente utilizando Sistemas Gerenciadores de Bases de Dados (SGBD) tradicionais. (VIEIRA et al., 2012 p. 2).

Neste contexto, surgiu uma nova categoria de Banco de Dados, chamada NoSQL (*Not Only SQL*), que foi proposta com o objetivo de atender aos requisitos de gerenciamento de grandes volumes de dados, semi-estruturados ou não estruturados, que necessitam de alta disponibilidade e escalabilidade. A necessidade de uma nova tecnologia de BD surgiu como consequência da ineficiência dos bancos de dados relacionais em lidar com esta tarefa. (LOSCIO et al., 2011 p. 1).

De acordo Santos e Silva (2017), dados não estruturados são gerenciados pela

forma padronizada oferecida pelos padrões estabelecidos pelos sistemas gerenciadores de banco de dados relacionais.

Ramos e Nascimento (2018) descreve as características de um banco de dados NoSQL. São elas: a) escalabilidade horizontal, onde ocorre um aumento de máquinas disponíveis para armazenamento de dados e processamento visando melhorar o desempenho; b) ausência de esquema, ou um esquema mais flexível facilitando a escalabilidade e aumentando a disponibilidade, porém não garante a integridade dos dados. A Figura 1 apresenta as diferenças entre os bancos de dados relacionais e banco de dados NoSQL.

Quanto a	Modelo Relacional	NoSQL
Organização do armazenamento	Basicamente em Tabelas e Relacionamentos entre elas	Existem várias implementações: Wide Column Store/Column Families, Document Store, Key Value/Tuple Store, Eventually Consistent Key Value Store, Graph Databases, Object Databases, Grid Database Solutions e outras
Aumento de dados manipulados e/ou de acessos	Necessidade de Escalabilidade Vertical (ver quadro)	Necessidade de Escalabilidade Horizontal (ver quadro)
Redundância de dados	Evita-se ao máximo a existência de redundância com intenso uso de Tabelas e Chaves Estrangeiras, ou seja, de Relacionamentos	Não há preocupação com redundância. O objetivo é evitar custo computacional com joins entre várias tabelas que podem conter muitas linhas
Linguagem de acesso a dados	SQL e extensões de SGBDs	Várias linguagens, inclusive o próprio SQL, conforme o SGBD
Mudanças estruturais	Criação/remoção/alteração de campos e/ou chaves e/ou restrições	Mudanças apenas nos registros pertinentes, mantendo-se uma estrutura herogênea.
Transações	Devem ser sempre ACID. Consistência a todo custo	ACID é opcional ou até inexistente. O importante é velocidade e não consistência perfeita

Figura 1. Principais diferenças entre o Modelo Relacional e o NoSQL. (RAMOS e NASCIMENTO, 2018)

Assim, como objeto de estudo no Projeto de Pesquisa "Manutenção de documentos usando Banco de Dados NoSQL" em desenvolvimento no Centro de Ensino Superior de Juiz de Fora, foi proposto o desenvolvimento de uma aplicação móvel que manipularia documentos e os persistiria em um Banco de Dados NoSQL, o que permitiria a análise do conteúdo de tais documentos.

Na próxima seção fica detalhado as características de Banco de Dados NoSQL orientados a documentos que foram fundamentais para a escolha deste tipo de NoSQL para o desenvolvimento do protótipo.

2.2. Banco de Dados NoSQL orientado a documentos

Os Bancos de Dados orientados a documentos, ou documentais, são bancos NoSQL que persistem os dados em documentos. A estrutura que armazena os documentos é chamada de coleção (*collection*) (SOUZA, 2015). Ao contrário de um banco relacional, onde qualquer alteração em uma tabela refletirá em alterações na estrutura já pronta, no modelo documental é possível acrescentar, editar ou excluir atributos de forma isolada a

cada registro, visto que cada um deles é um documento independente. Essa característica permite, também, que elementos com características diferentes sejam armazenados na mesma coleção.

Suponha, como exemplo, uma loja que comercialize produtos com características diversas. Em um modelo relacional seria bastante trabalhoso representar esses dados - vide Tabela 1, porém em um banco de dados NoSQL documental seria simples, conforme visto na Figura 2.

Tabela 1. Tabelas utilizando modelo relacional. (Do autor, 2018)

Produtos			
	Peso	Potência	Diâmetro
Tubo			10 cm
Cimento	25 Kg		
Lâmpada		7 W	

```
produtos: [
    {nome: "Tubo", diametro: "10 cm"},
    {nome: "Cimento", peso: "25 Kg"},
    {nome: "Lâmpada", potencia: "7 W"}
]
```

Figura 2. Utilizando o modelo documental. (Do autor, 2018)

O protótipo, resultado desse artigo, vem para tratar dados que não têm padrões de estruturação, os chamados não-estruturados. Para esse tipo de dados é fundamental que o protótipo utilize para a persistência dos dados um banco de dados NoSQL visto a dificuldade técnica por parte de SGBDs relacionais.

2.3. O Banco de Dados MongoDB

O MongoDB é um sistema gerenciador de banco de dados documental escrito em C++, sob licença GNU AGPL (*Affero General Public License*) versão 3.0, que armazena dados dentro de documentos no formato *BSON* (Binary JSON), uma versão "binária" do JSON (*JavaScript Object Notation*). Este banco de dados foi criado por Dwight Merriman (ex-Fundador do DoubleClick e CTO) e Eliot Horowitz (CTO 10gen e cofundador) baseados em suas experiências de dados em grande escala, alta disponibilidade e sistemas robustos. (MONGODB, 2018).

De acordo com Cross et. al. (2018) a capacidade de armazenar objetos no formato JavaScript nativamente do MongoDB, permite reduzir o tempo de processamento tornando a manipulação de dados mais rápida. Em vez de uma linguagem específica de domínio como o SQL, o MongoDB utiliza uma interface

JavaScript simples para realizar consultas. Consultar um documento é tão simples como passar um objeto JavaScript que descreve parcialmente o destino da pesquisa.

Assim, o MongoDB tornou-se o banco NoSQL mais indicado para o desenvolvimento do protótipo pela preocupação no desempenho, processando grande quantidade de informação em questão de segundos, além de ser um software livre, que juntamente com a plataforma de desenvolvimento Android, teve nenhum custo – relativo a software – para o desenvolvimento do protótipo.

2.4. O formato JSON e o *BSON-Document*

JSON (*JavaScript Object Notation* - Notação de Objetos em JavaScript) é uma formatação considerada "leve" para troca de dados. Deste modo, a interpretação e geração dos dados possui grande performance. O JSON é baseado em um subconjunto da linguagem de programação JavaScript, a qual está especificada de acordo com o padrão Standard ECMA-262, 3a Edição, publicada em Dezembro de 1999 (JSON, 2018).

JSON é em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados (JSON, 2018). Além disso, é um formato que pode estar constituído a partir de duas estruturas: uma coleção de pares nome/valor ou uma lista ordenada de valores. As Figuras 3 e 4, permitem apresentar a representação destas estruturas.

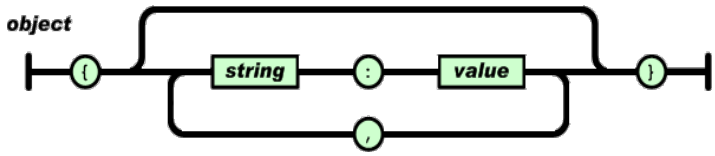

Figura 3. Representação dos dados em JSON. (JSON, 2018)

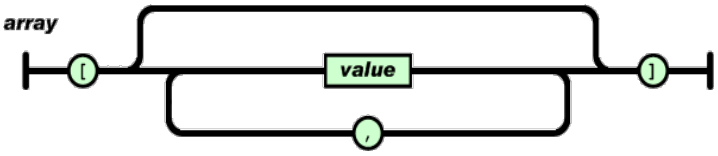

Figura 4. Representação de um *array* de objetos. (JSON, 2018)

Santos e Silva (2017) ressalta que o JSON não implementa uma padronização para o formato de data nem como trabalhar com dados binários. Por este motivo o MongoDB introduz o BSON (Binary JSON) que é uma extensão ao JSON que além dos formatos de dados suportados por este também comporta:

- MinKey, MaxKey , TimeStamp - Tipos utilizados internamente no MongoDB;
- BinData - Array de bytes para dados binários;
- ObjectId - Identificador único de registros do MongoDB;
- Date - Representação de data;

Com esses novos dados, implementados pelo BSON, o MongoDB oferece outra vantagem ao protótipo desenvolvido, pela questão da sumarização de grandes quantidades de informação, usando os tipos *MaxKey* e *MinKey*, além da identificação única de cada documento com o *ObjectId*, tornando possível, assim, a análise dos dados armazenados de forma mais eficiente.

Na próxima seção o sistema operacional móvel utilizado é estudado, e posteriormente mostrado como foi feita a escolha do mesmo para a implementação do protótipo em si.

2.5. O sistema operacional Android

A plataforma Android desfruta hoje de um papel de destaque no mercado, tanto pela quantidade significativa de dispositivos produzidos, como também por oferecer uma API rica, disponibilizando acesso a vários recursos de hardware [...] (MONTEIRO, 2012 p. 1)

De acordo com estudos da empresa StatCounter, de meados de 2017 até setembro de 2018 figurou como o sistema operacional mais utilizado do mundo – posto retomado pelo Windows em outubro de 2018. A Figura 5 os números a respeito do mercado de sistemas operacionais (STATCOUNTER, 2018).

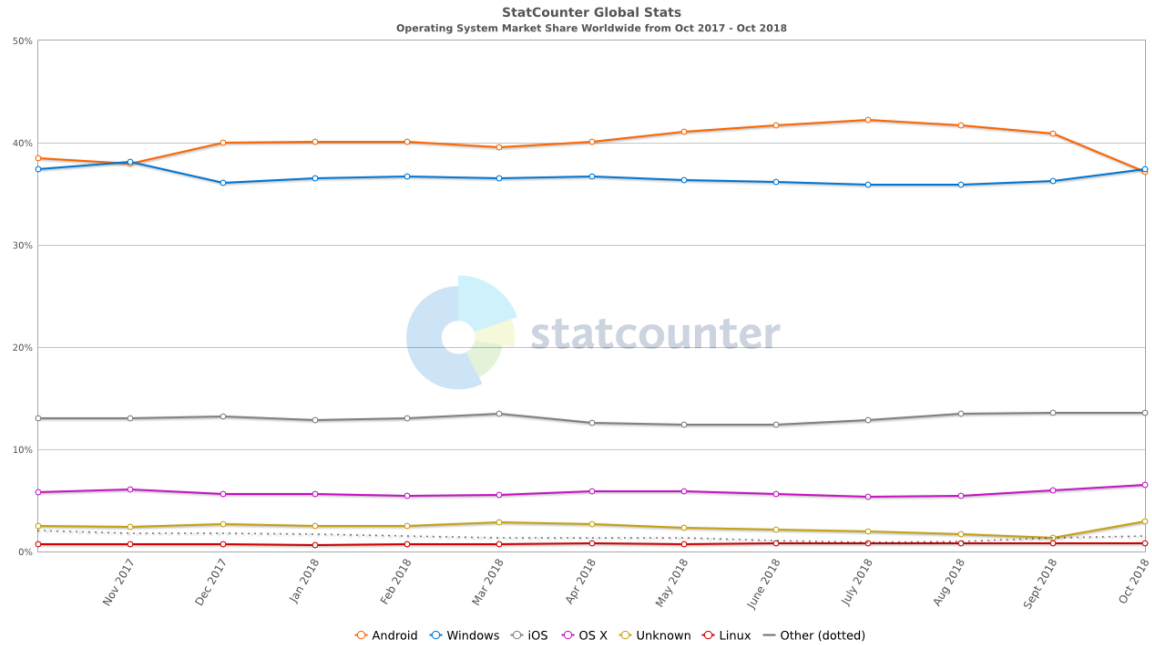

Figura 5. Operating System Market Share Worldwide – Oct 2017 - Oct 2018. Fonte: Statcounter (2018).

Além dos índices apresentados pela StatCounter, o ambiente para criação de aplicações em Android também possui grande facilidade para o desenvolvimento de aplicações, visto que sua IDE oficial – o AndroidStudio – e todo o ambiente de desenvolvimento são gratuitos e têm versões para os principais sistemas operacionais existentes, não limitando o desenvolvimento para plataformas distintas, como no caso do iOS e do Windows Phone.

Sendo um ambiente rico de recursos, com toda a documentação e desenvolvimento gratuitos, além da familiaridade por parte do desenvolvedor e grande gama de dispositivos para acesso, tornou-se o ambiente ideal para o desenvolvimento do protótipo.

Na próxima seção é levantada as tecnologias de desenvolvimento para sistemas móveis e qual a mesma foi utilizada para desenvolvimento do protótipo Android.

2.6. Tecnologias utilizadas para desenvolvimento

O desenvolvimento nativo é o modo tradicional para desenvolvimento de aplicações *mobile*, utilizando assim, recursos próprios e ambientes de desenvolvimento criados pela empresa mantedora do sistema operacional, por exemplo, o desenvolvimento nativo para Android utiliza como padrão o IDE (*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado*)* Android Studio e as linguagens de programação Java ou Kotlin, porque, de acordo com SILVA et.al (2014) aplicativos nativos são desenvolvidos de acordo com um conjunto de especificações fornecidas pelo fabricante do sistema operacional.

Para desenvolvimento de uma aplicação nativa possa ser executada em diferentes plataformas, é necessária a criação de uma plataforma, o que exige tempo, investimento e uma equipe com conhecimento nas variadas tecnologias usadas pelas plataformas alvo. (MATOS e SILVA 2016).

Além do desenvolvimento de aplicações utilizando o ambiente nativo, também é o desenvolvimento para multiplataforma, adotando um processo de desenvolvimento onde o resultado final são aplicações em plataformas diferentes. Isto é possível através da existência de várias ferramentas e *frameworks* para desenvolvimento, tais como Apache Cordova, PhoneGap e React Native, que se utilizam de linguagens Web, principalmente HTML 5, CSS 3 e JavaScript (ou EcmaScript - ES), tendo em vista que os principais sistemas operacionais *mobile* suportam essas linguagens e suas APIs (*Application Programming Interface* - Conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web).

Com base nesta discussão, não é necessária a existência de equipes de desenvolvimento específicas para cada sistema e plataforma, reduzindo, assim, tempo e custo dos desenvolvimentos. Porém, como desvantagem, há a questão de experiência de usuário, visto que não há como extrair todo o potencial de cada plataforma, sendo a aplicação gerada por essas técnicas bastante genérica a recursos comuns às plataformas.

De acordo com Bezerra e Schimiguel (2015) a construção de uma aplicação móvel utilizando determinadas tecnologias, tais como, Javascript, HTML, CSS para

diversos dispositivos, ao invés de criar uma aplicação nativa para cada dispositivo, significa utilizar menos código. A utilização de menos código para desenvolvimento, significa menos manutenções e manutenções mais rápidas.

Dessa forma, com todo o conjunto de especificações sobre o Android e com prévio conhecimento do desenvolvedor, o protótipo foi desenvolvimento de forma nativa, utilizando linguagem de programação Java e ambiente de desenvolvimento Android Studio.

3. Desenvolvimento de uma aplicação móvel para manipulação de documentos utilizando o banco de dados MongoDB

Os métodos e técnicas descritas a seguir foram usados para a construção do protótipo de aplicação, com o objetivo de orientar o desenvolvimento da aplicação proposta neste trabalho é discutida na Seção 1 (Introdução).

Esta seção também caracteriza a proposta do trabalho por meio da implementação de uma aplicação móvel, e que também pode ser acessada via plataforma web, para manipulação de documentos utilizando o banco de dados MongoDB.

Como foi descrito no referencial teórico, é possível a criação de aplicações que tenham como propósito a manipulação de grandes volumes de dados. Estes dados podem ser estruturados ou não-estruturados. Também devido a necessidade de armazenamento de grandes volumes de dados, optou-se pelo banco de dados MongoDB como plataforma para armazenamento no formato BSON, uma vez que a proposta é a apresentação de uma solução que possa manipular dados no formato binário.

É importante ressaltar que a manipulação implementada no protótipo visa o upload e download de arquivos binários ou no formato texto. Também visando flexibilizar o uso da aplicação, o desenvolvimento foi realizado em uma plataforma que permita o acesso das funcionalidades da aplicação tanto em aplicativos móveis, quanto em ambiente web desktop, seguindo assim a necessidade de acesso em multiplataforma. As próximas seções descrevem as tecnologias utilizadas e que permitiram a geração da aplicação em plataforma distintas.

Devido a necessidade de explorar o compartilhamento e integração de dados em documentos, o projeto deste protótipo iniciou-se a partir de estudos para criação de serviços web em PHP visando a manipulação de documentos. O PHP (*Hypertext Preprocessor*) é uma linguagem de programação para web de script *open-source*, trabalha mesclado ao HTML (*Hypertext Markup Language*) e é executado no lado servidor, o que possibilita que o site seja dinâmico (PHP, 2018). Neste projeto, o serviço web para manipulação de dados de documentos foi desenvolvido em PHP pelo conhecimento prévio do autor e, também, por todo o processamento ser do lado servidor o que não sobrecarregaria o sistema móvel consumindo recursos do dispositivo utilizado.

Para a aplicação da proposta deste trabalho utilizou-se a técnica de *webview* para construção de interface, a fim de facilitar o acesso tanto em smartphones quanto em ambiente desktop por meio de browsers (CHARLAND, 2018).

O nome dado ao protótipo citado neste trabalho é *NoSQLMobile*. As funcionalidades básicas utilizadas para implementação do *NoSQLMobile* são vistas abaixo:

- InserirCSV. Esta funcionalidade permite inserir um arquivo no formato texto, e que visa oferecer coleções de dados armazenados no banco de dados MongoDB para extração, transformação e carga de dados para geração de gráficos. Esta funcionalidade é um exemplo da manipulação de grandes volumes de dados, porém através de arquivos no formato texto.
- Inserir GridFS. Esta funcionalidade permite inserir documentos de qualquer tipo de formato.
- Collections. Serve para listar as coleções criadas para armazenar os dados de documentos armazenados no formato BSON.
- GridFS. Esta opção permite a consulta dos documentos manipulados, tanto pela funcionalidade de "InserirCSV" ou "Inserir GridFS".
- Análise. Permite consultar os dados dos arquivos inseridos na funcionalidade "InserirCSV", a fim de gerar o gráfico para análise dos dados armazenado no arquivo selecionado.

4. Características de implementação do *NoSQLMobile*, ferramenta de manipulação de documentos utilizando o banco de dados MongoDB

4.1. *Android Webview*

Como, para desenvolvimento do protótipo, seria necessário realizar *upload* e *download* de arquivos diretamente em dispositivos quaisquer, inclusive móveis, foi necessário implementar um *webview* ao protótipo móvel, visto a impossibilidade de outra forma de realizar operações com arquivos no desenvolvimento para o sistema operacional *Android*.

```
webView = (WebView) findViewById(R.id.Webview1);
assert webView != null;
WebSettings webSettings = webView.getSettings();
webSettings.setJavaScriptEnabled(true);
webSettings.setAllowFileAccess(true);

webView.loadUrl("http://localhost/Projeto");
```

Figura 6. Definição do webview do protótipo. Fonte: Do Autor.

O tipo de dados *webview* propõe alguns métodos indispensáveis tais como `loadUrl` onde é definido qual o endereço que será exibido dentro daquele *Webview*, `setJavaScriptEnable` que define se o *webview* será habilitado a executar *Javascript* e `setAllowFileAccess` que define se o *webview* poderá acessar – ou não – arquivos pelo dispositivo.

Em contrapartida, o desenvolvimento do serviço *web* que faria toda a comunicação da interface móvel com o banco de dados MongoDB foi desenvolvido em PHP pela experiência prévia do autor com essa linguagem e a conexão simplificada através do *driver* MongoDB para PHP, que será descrito na próxima seção.

4.2. Driver de conexão do MongoDB

A conexão é feita através de um *driver* disponibilizado pela MongoLab para a maioria das linguagens de programação atuais, tais como Java, C, PHP, C#, entre outras. Especificamente para o PHP, o driver é um arquivo *DLL* (biblioteca de vínculo dinâmico) que é disponibilizada como extensão, devendo ser adicionada ao diretório de extensões (EXT) da instalação PHP no servidor Web.

Para o desenvolvimento deste protótipo foi utilizado o ambiente XAMPP versão 3.1.0 para Windows, que contém a versão 2.4 do servidor web Apache e a versão 5.4.7 do PHP. Para a instalação do MongoDB foi escolhida a versão mais recente até o momento, a 4.0.2.

O driver é o responsável por "traduzir" os métodos das linguagens para comandos específicos do MongoDB, estabelecendo a conexão, as operações de inclusão, edição, atualização e deleção de registros.

4.3. O serviço Web para manipulação de arquivos

A aplicação possui algumas funcionalidades, tais como:

- *Upload* de arquivos no formato CSV – que possibilita análise de dados e inserção linha-a-linha no banco de dados;
- *Upload* de arquivos binários utilizando GridFS;
- *Download* dos arquivos do banco;
- Análise de dados com a geração de relatórios.

Figura 7. Formulário para upload do CSV. Fonte: Do Autor.

Primeiramente há um formulário, conforme visto na Figura 7, que é responsável pela localização do arquivo CSV e pelo encaminhamento do mesmo ao serviço que fará sua leitura linha a linha e gerará um *array* com todos os seus dados.

```php
<?php
    if(isset($_FILES['arquivo'])) {
        $lines = file($_FILES['arquivo']['tmp_name'], FILE_BINARY);
        foreach ($lines as $index => $line) {
            if ($index === 0) {
                # this is the header line
                $headers = str_getcsv($line);
            } else {
                $data = str_getcsv($line);
                $obj = new stdClass();
                foreach ($headers as $index => $header) {
                    $obj->$header = $data[$index];
                }
                $dataObjects[] = $obj;
            }
        }

        //String de Conexão
        $conn = new MongoClient('mongodb://127.0.0.1:27017');

        // Acessando o Banco
        $db = $conn->Data;

        $nomeArquivoSemExt = strstr($_FILES['arquivo']['name'], '.', true);

        // Criando e acessando a coleção com o nome do arquivo sem extensão
        $collection = $db->$nomeArquivoSemExt;

        foreach ($dataObjects as $id => $item) {
            $collection->insert($item);
        }

        echo 'Inserido no MongoDB com sucesso';
    }
?>
```

Figura 8. Serviço PHP que recebe o arquivo CSV. Fonte: Do Autor.

Na *string* de conexão é informado o endereço do servidor, que pode ser IP ou DNS, e a porta utilizada pelo serviço do MongoDB. Caso seja configurada autenticação, devem ser passados, também, os dados de acesso – usuário e senha – para aquele serviço. Após a conexão e acesso da coleção devida, é realizada a leitura de cada item do *array* e sua inserção no banco.

A recuperação desses dados é realizada da mesma forma e pode ser observada na figura N, onde utilizamos o método `selectDB`() para selecionar a base de dados e o `listCollections`() para listar todas as coleções dentro daquela base.

Em cada coleção é utilizado o método `jsonDecoded`() para criar um *array JSON* com os dados daquela coleção e, posteriormente com o método `fputcsv`(), montar o arquivo CSV linha a linha.

```php
<?php
    //String de Conexão
    $m = new MongoClient('mongodb://localhost:27017');

    //Lista as Bases de Dados
    $bases = $m->listDBs();

    //Seleciona a Base de Dados Data
    $db = $m->selectDB('Data');

    //Lista as Collections;
    $colecoes = $db->listCollections();

    foreach ($colecoes as $colecao) {
        //Define DATA/HORA
        $data = new DateTime('now', new DateTimeZone('America/Sao_Paulo'));
        $dataAtual = $data->format('Y-m-d-H-i-s');

        //Define nome do arquivo CSV
        $csvFileName = 'file-'.$dataAtual.'.csv';

        //Abre o arquivo - w significa que está aberto para escrita
        $fp = fopen($csvFileName, 'w');

        $jsonDecoded = $colecao->find();

        //Lê cada item no array.
        foreach($jsonDecoded as $row){
            //Escreve uma linha no arquivo CSV.
            fputcsv($fp, $row);
        }

        //Fecha o arquivo.
        fclose($fp);
    }
?>
```

Figura 9. Recuperação dos dados do MongoDB para um arquivo CSV. Fonte: Do Autor.

4.3.1. Inserção de binários com GridFS

O GridFS é uma especificação para o armazenamento e a recuperação de documentos que excedem o limite de tamanho *BSON-document*, que é de 16 MB. [...] Para contornar esse limite, o GridFS, ao invés de armazenar um arquivo em um único documento, divide o arquivo em partes ou pedaços, chamados *chunks*, e armazena cada um desses pedaços como um documento separado. Por padrão, o GridFS limita o tamanho dos *chunks* a 255 KB. (B. SOUZA, 2015 p. 149)

Ao adicionar um arquivo por GridFS, o MongoDB cria, automaticamente, duas coleções para tal, a coleção *fs.files* e a coleção *fs.chunks*. Na primeira serão armazenados os metadados do arquivo, tais como tamanho total do arquivo em bytes (*length*), tamanho de cada *chunk* (*chunkSize*), data que o arquivo foi armazenado (*uploadDate*), o nome do arquivo (*filename*) e o tipo de arquivo (*contentType*). Já a coleção que armazena as partes do arquivo em si, a fs.chunks, contém apenas o id do arquivo (na coleção *fs.files*), um número sequencial dos *chunks* desse arquivo e o binário do "pedaço" (*data*).

```php
<?php
    if(isset($_FILES['arquivo'])) {
        print_r($_FILES['arquivo']);

        echo '<br><br><br>';

        //Define timezone padrão
        date_default_timezone_set("Brazil/East");

        //String de Conexão
        $m = new MongoClient('mongodb://localhost:27017');

        $gridfs = $m->selectDB('test')->getGridFS();

        $id = $gridfs->storeFile($_FILES['arquivo']['tmp_name'],
                array('contentType' => $_FILES['arquivo']['type']));

        $gridfsFile = $gridfs->get($id);

        var_dump($gridfsFile->file);

        echo '<br><br><br>Inserido com sucesso';
    }
?>
```

Figura 10. Inserção de arquivo utilizando GridFS. Fonte: Do Autor.

O método `getGridFS` cria a estrutura do GridFS na base de dados informada em `selectDB`. O método `storeFile` adiciona o arquivo carregado via formulário PHP para o servidor à estrutura GridFS no MongoDB.

O comando `var_dump` imprime o conteúdo de uma variável de forma explanativa, muito comum para realizar *debug* (DALL'OGLIO, 2007), permitindo, assim, verificar se o arquivo foi inserido no MongoDB com sucesso. Após iterar em todas os registros de cada coleção, o comando `fclose` encerra o arquivo.

4.3.2. Análise dos dados com Google *Charts*

No desenvolvimento do protótipo foi, inicialmente, levantada a possibilidade de, através dos dados recolhidos nos arquivos CSV, gerar análise dos mesmos através de gráficos diversos. Para tal, foi utilizada a API do Google Chart.

Na Figura 11 é mostrado o script que gera o array de dados para o gráfico ($jsonTable). Em seguida, a Figura 12 é mostrado o Javascript do Google Charts que gera o gráfico em tela.

```php
<?php
    if(isset($_FILES['arquivo'])) {
        $arquivo = $_FILES['arquivo']['name'];
        $indiceRotulo = $_REQUEST['rotulo'];
        $indiceDados = $_REQUEST['dados'];
        $tituloRotulo = $_REQUEST['Trotulo'];
        $tituloDados = $_REQUEST['Tdados'];

        $lines = file($_FILES['arquivo']['tmp_name'], FILE_BINARY);
        $headers = array();
        $dataObjects = array();
        $rows = array();
        $table = array();

        $table['cols'] =
        array(
            array(
             'label' => "$tituloRotulo",
             'type' => 'string'
            ),
            array(
             'label' => "$tituloDados",
             'type' => 'number'
            )
        );

        foreach ($lines as $index => $line) {
            if ($index === 0) {
                $headers = str_getcsv($line);
            } else {
                $data = str_getcsv($line);
                $sub_array = array();
                $sub_array[] =  array(
                     "v" => $data[$indiceRotulo]
                    );
                $sub_array[] =  array(
                     "v" => $data[$indiceDados]
                    );
                $rows[] =  array(
                    "c" => $sub_array
                   );
            }
        }

        $table['rows'] = $rows;
        $jsonTable = json_encode($table);
    }
?>
```

Figura 11. Script PHP para alimentar o array com dados para o gráfico de análise. Fonte: Do Autor.

```php
<script type="text/javascript">
   google.charts.load('current', {'packages':['corechart']});
   google.charts.setOnLoadCallback(drawChart);
   function drawChart() {
    var data = new google.visualization.DataTable(<?php echo $jsonTable; ?>);

    var options = {
     title:'<?php echo $arquivo?>',
     legend:{position:'bottom'},
     chartArea:{width:'65%', height:'95%'}
    };

    var chart = new google.visualization.ColumnChart(document.getElementById('line_chart'));

    chart.draw(data, options);
  }
</script>

[...]

<div id="container">
    <h2 align="center">Análise dos Dados [<?php echo $arquivo; ?>]</h2>
    <div id="line_chart" style="width: 100%; height: 100%;"></div>
</div>
```

Figura 12. *Script Google Charts* com o array de dados criado anteriormente ($jsonTable). Fonte: Do autor.

Ao fazer upload de um arquivo CSV para a análise de dados, é fundamental informar o índice dos campos que serão rótulo e dados para o gráfico Google Chart, conforme visto nas figuras 13 e 14, respectivamente.

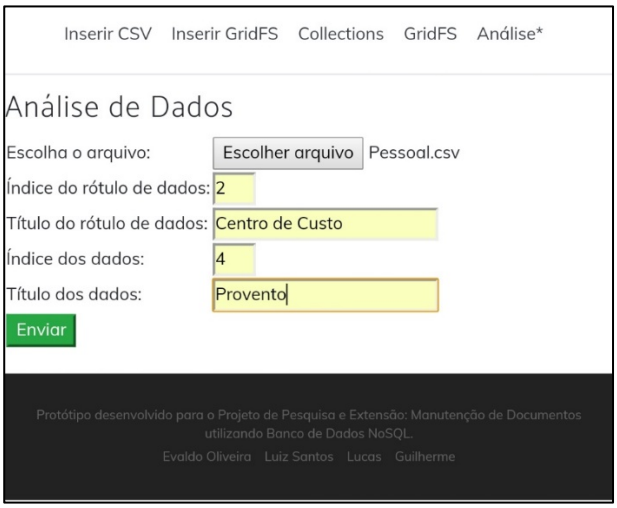

Figura 13. Formulário de upload do arquivo CSV para análise de dados. Fonte: Do autor.

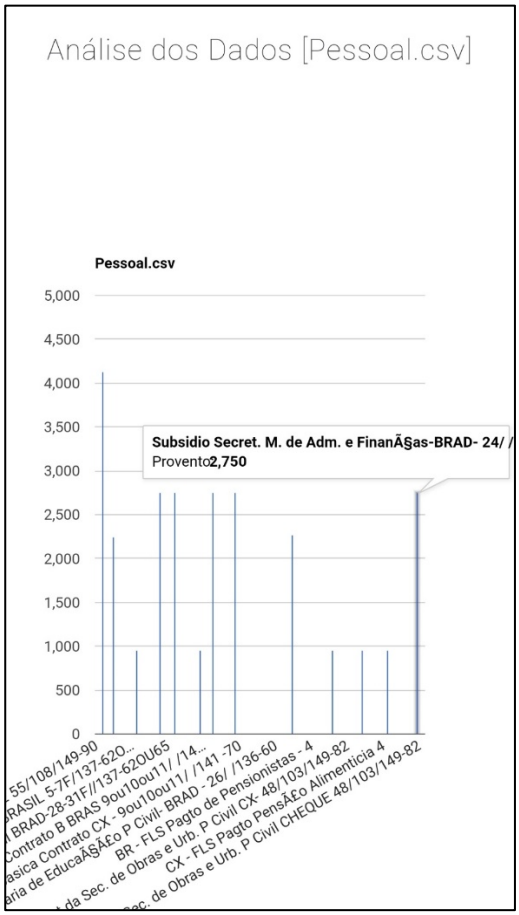

Figura 14. Análise de dados do arquivo [Pessoal.csv] usando o *Google Charts*. Fonte: Do autor.

4.3.3. Contribuição do Protótipo

A principal contribuição de tal protótipo é a possibilidade de análise de dados de arquivos em geral. Suponhamos, como exemplo de uso, um ERP de uma empresa que gere relatórios em CSV. Esses relatórios serão, possivelmente, impressos e guardados para posterior consulta e análise, o que pode acarretar a perda de tal, gerando todo o retrabalho de se gerar novamente o mesmo.

Com tal protótipo, pode-se extrair o relatório do ERP e exportá-lo para o MongoDB e, além, gerar relatórios a partir da análise dos dados do MongoDB, dos documentos BSON.

Além disso, pode-se criar repositórios de dados mais volumosos. Por exemplo, uma empresa que trabalhe com streaming de vídeos e áudios, pode utilizar o GridFS para persistir tais arquivos e obter maior desempenho no processamento desses arquivos.

5. Considerações Finais

Devido ao contexto de *Big Data* e a dados cada vez menos estruturados, o conceito de NoSQL ganha cada vez mais popularidade no desenvolvimento de soluções de tratamento desses dados. As redes sociais com milhões, até bilhões de usuários, geram imenso volume de dados diariamente que são processados e analisados por grandes soluções de Bancos de Dados NoSQL.

O protótipo desenvolvido vem demonstrar a utilização de um banco de dados não-relacional, o MongoDB, no processamento de grandes quantidades de dados e arquivos e sua utilização a partir de dispositivos móveis, conectados a serviços Web, que garantem disponibilidade e flexibilidade de uso.

Apesar das limitações que o protótipo tem atualmente, o trabalho teve sua contribuição ao apresentar o processamento de grandes quantidades de dados em pequenos intervalos de tempo, visando a análise de dados semiestruturados ou não-estruturados provenientes de qualquer fonte. Assim, por exemplo, um relatório CSV gerado por uma ferramenta ERP de uma empresa qualquer não precisa ser impresso pra depois analisado de forma manual, a ferramenta pode importar esse relatório e processar de forma automaticamente gerando gráficos para facilitar a análise.

Como trabalho futuro, há o desejo de se implementar a análise de outros tipos de arquivos, como planilhas em Excel, arquivos texto puro ou até mesmo arquivos PDF.

Agradecimentos

Agradecemos ao Centro de Ensino Superior de Juiz de Fora pela oportunidade de realizar este projeto.

6. Referências Bibliográficas:

BEZERRA, Peterson Tubini. SCHIMIGUEL, Juliano. **Desenvolvimento de Aplicações Mobile Cross-Platform utilizando Phonegap**. Disponível em: [http://www.eumed.net/cursecon/ecolat/br/16/phonegap.zip]. Acesso em: 05/11/2018.

CHARLAND, Andre; LEROUX, Brian. **Mobile application development: web vs. native**. Queue, v. 9, n. 4, p. 20, 2011.

CROSS, Zach. POCHE, Aga. SANTIAGO, Daniel. SINGH, Divit. **Desenvolvendo aplicativos móveis com Node.js e MongoDB, parte 1: Os métodos e resultados de uma equipe. Acelerando o tempo de retorno dos sistemas de engajamento.** Disponível em: [https://www.ibm.com/developerworks/br/library/mo-nodejs-1/mo-nodejs-1-pdf.pdf]. Acesso em: 15/10/2018.

DALL'OGLIO, Pablo. **PHP Programando com Orientação a Objetos**. São Paulo: Novatec Editora, 1.ª ed. 2007.

JSON. **Introducing JSON**. Disponível em: [http://json.org/]. Acesso em: 05/10/2018.

LOSCIO, Bernadette Farias. OLIVEIRA, Hélio Rodrigues de. PONTES, Jonas César de Sousa. **NoSQL no desenvolvimento de aplicações Web colaborativas**. Disponível em: [https://www.researchgate.net/profile/Bernadette_Loscio/publication/268201466_NoSQL_no_desenvolvimento_de_aplicacoes_Web_colaborativas/links/576aa72008aef2a864d1ef8c/NoSQL-no-desenvolvimento-de-aplicacoes-Web-colaborativas.pdf]. Acesso em: 05/11/2018.

MATOS, Beatriz Renezer Dourado. SILVA, João Gabriel de Britto e. **Estudo comparativo entre o desenvolvimento de aplicativos móveis utilizando plataformas nativas e multiplataforma**. Disponível em: [https://fga.unb.br/articles/0001/5113/Beatriz_Joao_TCC_Aplicativos_M_veis.pdf]. Acesso em: 04/11/2018.

MONGODB. Manual. Disponível em: [https://docs.mongodb.com/manual/]. Acesso em: 10/10/2018.

MONTEIRO, João Bosco. **Google Android: Crie aplicações para celulares e tablets**. São Paulo: Casa do Código - 1.ª ed. 2012.

PHP. **Manual do PHP**. Disponível em :http://php.net/manual/pt_BR/. Acesso em 22 de out de 2018.

RAMOS, José Yoshiriro Ajisaka. NASCIMENTO, Adriana de Farias. **NoSQL. Conceitos e Evolução**. Disponível em: [http://www.univale.com.br/unisite/mundo-j/artigos/51Nosql.pdf]. Acesso em: 23/09/2018.

SANTOS, R. ; SILVA, Evaldo de Oliveira . Armazenamento e Manipulação da Representação Semântica de Dados utilizando as tecnologias Java e MongoDB. CADERNO DE ESTUDOS EM SISTEMAS DE INFORMAÇÃO, v. 1, p. 1-18, 2017.

SILVA, Marcelo Moro da. SANTOS, Marilde Terezinha Prado. **Os Paradigmas de Desenvolvimento de Aplicativos para Aparelhos Celulares**. Disponível em:

[http://revistatis.dc.ufscar.br/index.php/revista/article/download/86/80]. Acesso em: 06/12/2018.

SOUZA, Marcio Ballem de. **Desvendando o mongoDB. Do Mongo Shell ao Java Driver**. Rio de Janeiro: Ciência Moderna - 1.ª ed. 2015.

STATCOUNTER. **Operating System Market Share Worldwide**. Disponível em: [http://gs.statcounter.com/os-market-share]. Acesso em: 22/09/2019.

VIEIRA, Marcos Rodrigues. FIGUEIREDO, Josiel Maimone. LIBERATTI, Gustavo. VIEBRANTZ, Alvaro Fellipe. **Banco de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Caso no Contexto de Big Data. Simpósio Brasileiro de Banco de Dados**. Disponível em: [http://data.ime.usp.br/sbbd2012/artigos/pdfs/sbbd_min_01.pdf]. Acesso em: 09/08/2018.