



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

Uso de Tecnologia NoSQL para Extração de Informação sobre Unidades de Exploração
e Produção de Petróleo

Augusto Taboransky

Pedro Jardim

Orientador

Prof^ª. Geiza Maria Hamazaki Da Silva

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2015

Uso de Tecnologia NoSQL para Extração de Informação sobre Unidades de Exploração
e Produção de Petróleo

AUGUSTO TABORANSKY

PEDRO JARDIM

Projeto de Graduação apresentado à Escola de
Informática Aplicada da Universidade Federal do
Estado do Rio de Janeiro (UNIRIO) para obtenção do
título de Bacharel em Sistemas de Informação.

Aprovada por:

Prof^ª. Dr^ª. GEIZA M. HAMAZAKI DA SILVA (UNIRIO)

Prof. Dr. ASTERIO KIYOSHI TANAKA (UNIRIO)

Prof. Me. EDUARDO T. HADEU C. LEITE (Instituto Tecgraf/PUC-Rio)

Me. PAULO IVSON (Instituto Tecgraf/PUC-Rio)

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2015

Agradecimentos

Agradecemos a UNIRIO, seu corpo docente, direção e administração por todo o apoio nestes anos de conclusão do curso.

Agradecemos ao Ministério da Ciência, Tecnologia e Inovação, ao Ministério da Educação, ao CNPQ e a CAPES, pela oportunidade a ambos de participar do programa Ciência Sem Fronteiras, que foi fundamental para a confecção deste trabalho, além de nos fornecer uma experiência única, que nos fez crescer como pessoa.

Agradecemos a Prof^a. Geiza pela paciência na orientação e incentivo que tornaram possível a conclusão deste trabalho.

Agradecemos ao Instituto Tecgraf pela parceria na pesquisa dos dados utilizados para o estudo de caso neste trabalho.

Pedro Jardim

Gostaria de agradecer a minha família e amigos próximos que estão sempre por perto para me ajudar nas horas de dificuldades.

A todos os meus colegas da Faculdade, por esses anos de caminhada juntos, foram muitas aulas de revisão em cima da hora, trabalhos em grupo e desesperos por achar uma prova do período passado.

Ao meu colega de trabalho Augusto Taboransky, com o qual iniciei junto meus anos de estudo de TI, cursando primeiramente um período em outra universidade e depois na UNIRIO.

I'd also like to give a very special thanks to Dr. Bernadette Marie Byrne, who was my professor and tutor at Hertfordshire University during my time studying abroad. Who I had the opportunity to work with in a project also about NoSQL.

Augusto Almada

Agradeço primeiramente a minha família por todo o suporte que me deram desde que nasci, em especial meus avós Ary e Aparecida, que foram meus segundos pais, e minha mãe, Maria Cristina, que sempre me apoiou e esteve ao meu lado.

A todos os meus colegas de faculdade, que tornaram essa jornada acadêmica muito mais animada e menos sofrida. Muitas risadas demos juntos.

A minha namorada, Evelina, que sempre me apoiou, tanto em minha jornada acadêmica na UNIRIO e na Halmstad University, quanto em meus desafios pessoais. Pelas risadas e choros, pelo carinho e por me ajudar a crescer como pessoa.

Aos meus amigos de banda, Pedrito, Luiz, Lucas e Filipe, que também sempre estiveram ao meu lado, desde antes de ingressar a UNIRIO, e que certamente tiveram uma grande influência na minha personalidade.

Ao meu amigo e parceiro de trabalho Pedro Jardim. Iniciamos essa jornada juntos, ambos tivemos nosso período no Ciência sem Fronteiras, e agora a terminamos juntos. Foi um prazer trabalhar ao seu lado. Esse não foi o primeiro trabalho que dividimos, e com sorte não será o último.

RESUMO

Este trabalho aborda o uso da tecnologia NoSQL para armazenar e recuperar dados sobre o processo de pintura de plataformas de exploração de petróleo. O objetivo é estudar a viabilidade da utilização de um banco de dados NoSQL para conseguir reunir e analisar dados de diferentes fontes, e reuni-los em um único banco para conseguir inferir conhecimento sobre esse domínio. Os dados escolhidos referem-se a plantas industriais de plataformas de exploração de petróleo e foram obtidas pela parceria com o Instituto Tecgraf. Estes dados estão armazenados em diversas planilhas Excel. Neste trabalho é apresentado o estudo dessa tecnologia e também o desenvolvimento da aplicação em Java que visa transformar os dados dessas planilhas em informações úteis, como custo e tempo estimado para cada projeto.

Palavras-chave: NoSQL, Banco de Dados, Ciência sem Fronteiras, Excel, Java

ABSTRACT

This project discusses the use of the NoSQL technology to store and recover data about the process of painting oil and gas exploration platforms. The goal is to study the possibility of the use of NoSQL to gather and analyse data from different sources, and store them in a single database, to infer knowledge about this domain. The chosen data was about the area of oil and gas exploration platforms, and was obtained through cooperation with Instituto Tecgraf. This data is stored in several Excel spreadsheets. In this work it is presented a study of this technology as for the development of the Java application, which seeks to transform the data from these spreadsheets into applicable information, like the cost and estimated time for each project.

Keywords: NoSQL, Database, Science without Borders, Excel, Java

Índice

1	Introdução.....	10
1.1	Motivação	10
1.2	Objetivos.....	11
1.3	Organização do texto	11
2	NoSQL.....	12
3	Tecnologias aplicadas no projeto	19
4	Desenvolvimento da Solução	26
4.1.	Estudo de caso	26
4.2.	Preparando o ambiente.....	28
4.3.	Etapas de desenvolvimento.....	28
4.3.1.	Leitura, tratamento e gravação	28
4.3.2.	Recuperação dos dados e processamento	33
4.3.3.	Criação dos arquivos .xls com as informações extraídas	40
4.3.4.	Consultas e Gráficos	41
5	Conclusão e Trabalhos futuros	44

Índice de Figuras

Figura 1: Exemplo de um DB <i>collunar</i>	13
Figura 2: Exemplo de <i>Key-value</i>	14
Figura 3: Exemplo da representação de dados em um <i>Graph Store</i>	14
Figura 4: Exemplo de Document stores	15
Figura 5: Diagrama de fluxo para ação de gravar os dados das planilhas para o banco de dados	20
Figura 6: MongoDB shell script	23
Figura 7: Exemplo de representação relacional no MongoDB.....	24
Figura 8: Exemplo de um elemento agregado em MongoDB	25
Figura 9: Visualização da divisão de um módulo.....	27
Figura 10: Exemplo do banco de dados MongoDB rodando no prompt de comando ...	29
Figura 11: Código da função de leitura dos arquivos	29
Figura 12: Trecho do código usado para recuperar e tratar os dados	30
Figura 13: Continuação da função lerArquivo, mostrando o tratamento do nome das zonas	31
Figura 14: Exemplo de um elemento gravado no mongoDB	32
Figura 15: Código da função de busca da área total de cada módulo	36
Figura 16: Resultado da busca do total da área de todos os setores por módulo.....	37
Figura 17: Função que calcula o tempo levado para finalizar um setor	38
Figura 18: Modelo da classe Registro	39
Figura 19: Planilha gerada com os parâmetros: Trabalhadores: 30; Salário: R\$50,00/dia; Duração: 30 dias	40
Figura 20: Gerado com os parâmetros: Trabalhadores: 30; Salário: R\$50,00/dia; Duração: 30 dias	41
Figura 21: Gerado com os parâmetros: Trabalhadores: 50; Salário: R\$50,00/dia; Duração: 18 dias	42
Figura 22: Gerado com os parâmetros: Trabalhadores: 70; Salário: R\$50,00/dia; Duração: 12 dias	43

Introdução

1.1 Motivação

Com o advento da Internet iniciou-se uma nova era onde não só empresas são geradoras de dados, mas sim toda e qualquer pessoa conectada à rede. Estas geram uma grande quantidade de dados, que não estão estruturados, e vindo de diferentes fontes. Muitas empresas usam diferentes planilhas para armazenar e gerar dados, o que pode tornar complicado o cruzamento de dados entre elas, dado que para se obter essas informações é necessário a programação nas mesmas, ou uma pessoa realizar este processamento manualmente. Essas informações são essenciais para empresas, principalmente para levantamento de informações cruciais para o seu negócio¹.

A transformação de dados puros em informação está presente no ramo de *Business Intelligence*² (BI). Isto é, conseguir juntar dados, tratar, gravá-los e processá-los a fim de obter informações que ajudem as tomadas de decisão.

Este trabalho não tem como objetivo criar um BI completo, mas sim abordar uma parte de sua estrutura, o armazenamento de dados e extração de informação. Tendo isto em vista, e se utilizando de dados do mundo real, é proposto mostrar a migração de dados gravados em planilhas para um banco de dados NoSQL, e o processamento e análise dos mesmos a fim de extrair informação, que serão visualizados em planilhas.

¹ <http://searchcio.techtarget.com/podcast/BI-tools-vs-Microsoft-Excel-spreadsheets>, acessado em 14 dezembro 2015

² https://en.wikipedia.org/wiki/Business_intelligence, acessado em 14 janeiro 2016

1.2 Objetivos

Este projeto objetiva apresentar um estudo de caso, com dados sobre áreas e elementos de uma plataforma de exploração de petróleo. Estes foram pesquisados em parceria com Instituto Tecgraf, da PUC-RIO, e sobre o qual deseja-se fazer cálculos e simulações, a fim de se extrair informações úteis para ajudar a análise sobre o custo de manutenção anticorrosiva em plataformas *offshore*.

Os dados e as métricas levantadas são apenas para simulação, pois tratando-se de dados industriais de áreas estratégicas, é muito difícil a obtenção de informações reais, mas é consenso que qualquer resultado positivo, mesmo que mínimo, gera uma grande margem de lucro ou diminuição de custo em situações reais.

1.3 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo II: Descreve a tecnologia NoSQL, sua estrutura, e a compara ao esquema RDBMS.
- Capítulo III: Apresenta as tecnologias usadas no projeto e exhibe a arquitetura do sistema desenvolvido.
- Capítulo IV: Detalha o procedimento realizado pelo sistema, desde a preparação do ambiente até os arquivos de saída.
- Capítulo V: Conclusões – Reúne as considerações finais, assinala as contribuições da pesquisa e sugere possibilidades de aprofundamento posterior.

NoSQL

Facebook³, Youtube⁴ e MercadoLivre⁵ são alguns exemplos de plataformas online que possuem um grande volume de dados, podendo estes serem criados por grandes companhias, a fim de divulgar produtos ou serviço, ou por pessoas comuns que querem, por exemplo, compartilhar fotos de seu gato. São plataformas que possuem grandes quantidades de informações, na forma de texto, vídeo ou áudio. Com o desafio de conseguir armazenar esses dados não estruturados criaram-se os primeiros bancos de dados não relacionais, NoSQL. NoSQL significa “Not Only SQL”, e não “Não SQL”, como muitos acreditam.

Alguns exemplos de bancos de dados NoSQL são: Oracle NoSQL⁶, Apache Cassandra⁷, Amazon DynamoDB⁸, Google Bigtable⁹ e MongoDB¹⁰.

O DynamoDB e Oracle NoSQL são do tipo *key-value*, Cassandra é um híbrido entre *key-value* e *columnar* e BigTable é *columnar*. O MongoDB, o banco escolhido para esse trabalho é do tipo *Document store* (NoSQL for Dummies, 2015). Estes modelos de dados serão apresentados a seguir.

³ <https://www.facebook.com>, acessado em 20 janeiro 2016.

⁴ <http://www.youtube.com>, acessado em 20 janeiro 2016.

⁵ <http://www.mercadolivre.com.br>, acessado em 20 janeiro 2016.

⁶ <http://www.oracle.com/technetwork/database/database-technologies/nosqldb/overview/index.html>, acessado em 3 janeiro 2016

⁷ <http://cassandra.apache.org/>, acessado em 3 janeiro 2016

⁸ <https://aws.amazon.com/pt/dynamodb>, acessado em 3 janeiro 2016

⁹ <https://cloud.google.com/bigtable/>, acessado em 3 janeiro 2016

¹⁰ <https://www.mongodb.org/>, acessado em 3 janeiro 2016

1 - Columnar

Column stores são parecidos com o tradicional modelo relacional RDBMS (*Relational Database Management System*) (NoSQL for Dummies, 2015), onde há os conceitos de linha e coluna. Devem-se definir as colunas primeiro, o que significa que é necessário o conhecimento da estrutura dos dados com antecedência.

Ele é comumente referido como *Big Tables* ou *BigTable clones*, por causa de seu ancestral o *Google Bigtable*.

A diferença é que os dados não são armazenados nas linhas, horizontalmente, e sim nas colunas, verticalmente. Isso permite uma alta eficiência na compressão dos dados e fácil escalabilidade horizontal, tornando-o altamente otimizado para operações como *count*, *sum* e *average*.

EmpID	Salary	Designation
100	10,000	Clerk
200	20,000	Assistant Manager
300	30,000	Manager
400	40,000	Zonal Head

Figura 1: Exemplo de um DB *collunar*¹¹

2 - Key-value

O *Key-value* utiliza o tradicional campo ID, como uma *primary key*, e um conjunto de dados. As funções de busca são muito rápidas devido a bons métodos de indexação presentes nesse modelo. Esse modelo funciona como um mapeamento, de um índice *Key* com um valor *Value*, que pode ser uma lista de outro mapeamento, possibilitando o armazenamento de uma árvore, podendo conter apenas um simples elemento, como números e textos, ou uma lista de elementos.

¹¹ http://www.ijarcse.com/docs/papers/8_August2012/Volume_2_issue_8/V2I800154.pdf, acessado em 19 dezembro 2015

BANK DATABASE	
Key	Value
1	ID:1 Joining Date: 15-July-1985 Designation: Cashier
2	ID:2 Joining Date: 19-March-1982 Designation: Manager
3	ID:3 Joining Date: 4-April-1988 Designation: Front Desk Officer

Figura 2: Exemplo de *Key-value*¹²

3 - Graph stores

Graph stores possuem pares de *key-value* interconectados e seguem a teoria do grafo com nós e arestas. Os nós são as entidades e os vértices mostram as relações entre eles. A forma é parecida com uma relação, mas a diferença para o modelo relacional está no seu custo, pois o modelo de grafos não se utiliza de *joins* e sim de “*Traversals*” *techniques* (Moniruzzaman & Hossain, 2013). Um exemplo é um banco de *TripleStores*, que são bancos de dados que buscam triplas se utilizando de consultas semânticas. A tripla é sempre composta por “sujeito-predicado-objeto” como João conhece Paulo (Graph Databases, 2013).

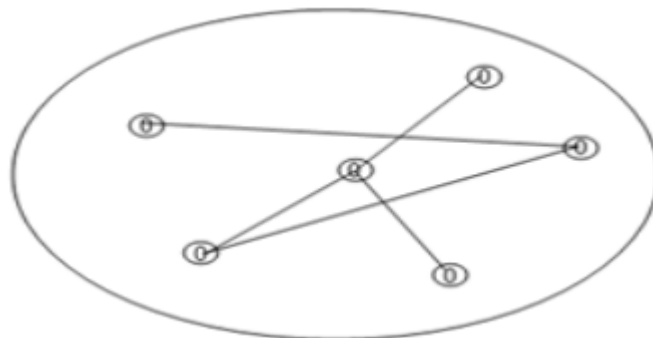


Figura 3: Exemplo da representação de dados em um *Graph Store*¹³

¹² http://www.ijarcse.com/docs/papers/8_August2012/Volume_2_issue_8/V2I800154.pdf , acessado em 19 dezembro 2015

¹³ http://www.ijarcse.com/docs/papers/8_August2012/Volume_2_issue_8/V2I800154.pdf , acessado em 19 dezembro 2015

4 - Document stores

Document stores também são chamados de "*aggregate databases*", pois em um documento pode-se armazenar toda a informação em apenas um registro. Este possui um foco para *Big Data*, grandes quantidades de dados, e consultas eficientes, pois a extração de dados é simples, sem a necessidade de *joins*, sendo necessário apenas uma forma mais lógica para aplicações. O esquema não é previamente definido como nos bancos relacionais, pois a estrutura de dado é maleável, podendo um único elemento ter atributos inseridos ou retirados no decorrer do tempo, sem precisar mudar todos os outros dados.

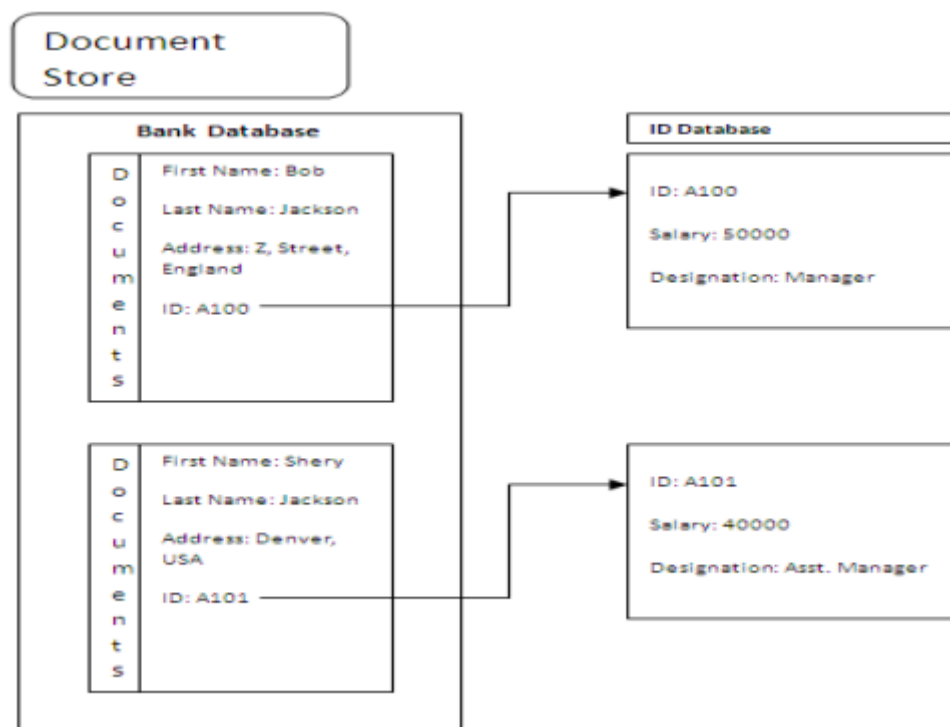


Figura 4: Exemplo de Document stores¹⁴

¹⁴ http://www.ijarcsse.com/docs/papers/8_August2012/Volume_2_issue_8/V2I800154.pdf , acessado em 19 dezembro 2015

Diferença entre NoSQL e RDBMS.

Existem diferenças entre NoSQL e os bancos de dados relacionais, entre elas a considerada mais importante é relacionada ao modelo de dados. Nos modelos relacionais é necessário, previamente, modelar os dados em tabelas e relações, o que pode acarretar em esforço extra para que todos os dados estejam organizados e armazenados de forma a representar duas relações. Com NoSQL, é apresentada uma abordagem *Schemaless*, ou seja, sem esquemas. Isso significa que não há um *schema* pré-determinado dos dados, logo, estes podem ser armazenados de forma desestruturada.

Outra diferença é que os bancos de dados relacionais suportam as funções ACID, enquanto NoSQL segue as propriedades BASE e o teorema CAP. Estas propriedades estão descritas abaixo:

A.C.I.D significa *Atomicity, Consistency, Isolation e Durable*.

A - *Atomicity*. neste contexto, atomicidade significa que uma transação é realizada por completo ou nada é realizado, *All or nothing*. Se uma pequena parte da transação falha, o sistema realiza um *roll back* para o estado inicial da transação.

C - *Consistency*. Com esta propriedade, após uma transação bem sucedida a integridade dos dados estará preservada, ou seja, o banco de dados sempre estará consistente. Se uma transação falha em sua consistência o sistema retorna para o estado inicial, onde os dados estavam guardados de forma consistente.

I - *Isolation*. Um banco de dados relacional pode receber diversas transações ao mesmo tempo, e para impedir que uma transação não interfira nos dados que

outra transação esteja precisando, o banco bloqueia o acesso aos dados até que todo o seu processo seja concluído. Assim, no início de uma transação os dados são isolados para que somente essa transação tenha acesso a eles.

D - *Durable*. Os dados armazenados no banco devem ser duráveis, ou seja, eles devem permanecer guardados mesmo que haja falta de energia ou que o sistema pare.

BASE significa *Basic Available, Soft State e Eventual Consistency*.

Basically Available - Parte dos dados podem não estar sempre disponíveis para o usuário. Os bancos de dados NoSQL podem estar divididos em vários locais, e se um desses locais não estiver disponível o sistema não irá parar por esta causa.

Soft State - O estado dos dados pode mudar a qualquer momento, e não há um impedimento para o acesso a esses dados. Assim, é possível acessar dados que estejam sendo usados por outras transações ao mesmo tempo.

Eventually Consistent - O banco de dados não irá checar se as transações realizadas deixaram o banco em um estado consistente.

Eric Brewer¹⁵ foi o criador do teorema CAP (*Consistency, Availability and Partitioning*) nos anos 2000. Este teorema foi provado em 2002, deixando assim de ser uma teoria¹⁶, e seu enunciado diz que não se pode ter os três atributos simultaneamente. Na maioria do tempo se terá dois atributos dos três.

¹⁵ <http://www.cs.berkeley.edu/~brewer/>, acessado em 13 janeiro 2016

¹⁶ <http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>, acessado em 13 janeiro 2016

Consistency, neste contexto, não tem o mesmo significado que em ACID. De acordo com o teorema CAP, consistência significa que toda vez que os dados são escritos qualquer usuário que lê-los irá ver a última versão dos dados;

Availability significa que o banco de dados sempre responderá quando for acionado; e *Partitioning* quer dizer que ao se ter vários bancos de dados espalhados, os bancos irão continuar a funcionar mesmo que um deles tenha ficado *offline*. Isto é, mesmo que um de seus bancos tenha parado de ser acessado pelos outros, estes não irão parar de gravar e vão continuar com suas rotinas. Assim podendo gerar inconsistência com seus dados, quando o banco que antes estava *offline* voltar a funcionar.

Com o teorema CAP abre-se uma lista de opções em como balancear a consistência, disponibilidade e particionamento de seus dados em um banco de dados no sistema BASE. O que ajudou a melhorar o desenvolvimento de antigos e novos bancos de dados NoSQL.

Tecnologias aplicadas no projeto

Para o desenvolvimento deste projeto foi utilizado como caso de uso um problema pesquisado pelo Instituto Tecgraf, no qual existem diversas planilhas de dados Excel com dados de plataformas petrolíferas, sobre pintura e manutenção das mesmas. Mais informações serão dadas na seção 4.1. Estas planilhas possuem os dados a serem cruzados a fim de retirar informações que ajudem as tomadas de decisões. Desta forma, utilizando a tecnologia NoSQL, os dados dessas planilhas são armazenados em um único Banco de Dados, e a partir dele são retiradas informações pertinentes.

As tecnologias envolvidas nesse processo são: Planilhas Excel, Java e MongoDB. Para tratar as planilhas em que se encontram os dados brutos, será utilizada a linguagem Java, a fim de capturar e tratar esses dados, e então os armazenar no banco de dados MongoDB. A figura 5 representa o diagrama de fluxo de nossa aplicação para o fluxo de migração de dados das planilhas para o banco de dados.

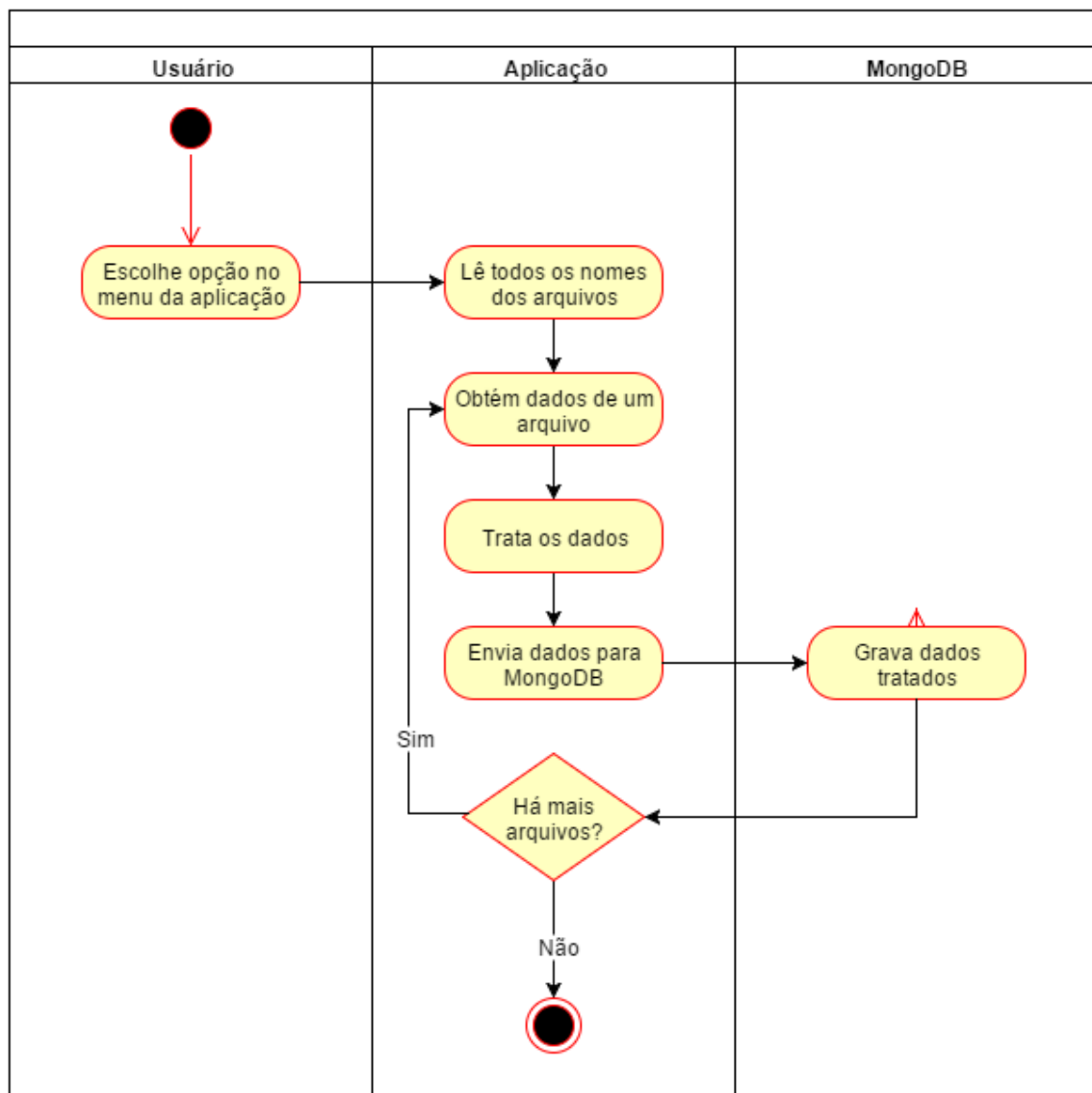


Figura 5: Diagrama de fluxo para ação de gravar os dados das planilhas para o banco de dados

No sistema, primeiramente os dados contidos nos arquivos com a extensão .xls são lidos, transformados e gravados no banco de dados do MongoDB. No passo seguinte é possível, através de *queries* (consultas), extrair os conhecimentos desejados e gerar outros arquivos (.xls) com estas informações.

Abaixo é apresentado um pouco mais sobre o MongoDB, supondo que o leitor possui conhecimento sobre as outras tecnologias.

MongoDB

Dentre todos os tipos de bancos de dados já mencionados no capítulo 2 será utilizado o MongoDB. Esta escolha foi feita principalmente por ela ser uma ferramenta *open source*. Também foi levando em conta a boa documentação encontrada sobre o assunto¹⁷. No momento MongoDB é uma das ferramentas mais populares quando o assunto é banco de dados NoSQL. FourSquare, CartolaFC, NowTV, MTV, FORBES¹⁸, são alguns exemplos desta popularidade .

O tipo de armazenamento de dados é *Document database*. Nativamente o MongoDB usa JSON (*JavaScript Object Notation documents*). JSON é um formato de dados de fácil leitura e compreensão para usuários. Este foi criado como base para a linguagem JavaScript. É apresentado no formato de texto e possui suporte para diversas linguagens de programação como C, C++, C#, Java, Python, Pearl, entre outras.

O objeto JSON pode ser construído de duas formas. Uma com uma simples coleção de chave/valor, sendo muito parecido com um dicionário. Por exemplo:

```
{  
    "nome" : "Pedro",  
    "Sobrenome" : "Jardim"  
}
```

¹⁷ <https://docs.mongodb.org/manual/>, acessado em 17 novembro 2015.

¹⁸ <https://www.mongodb.org/community/deployments>, acessado em 17 novembro 2015.

Outra forma é com uma lista ordenada de valores, como um Array. Por exemplo:

```
{
  usuarios : [
    {
      "nome" : "Pedro",
      "Sobrenome" : "Jardim"
    },
    {
      "nome" : "Augusto",
      "Sobrenome" : "Taboransky"
    }
  ]
}
```

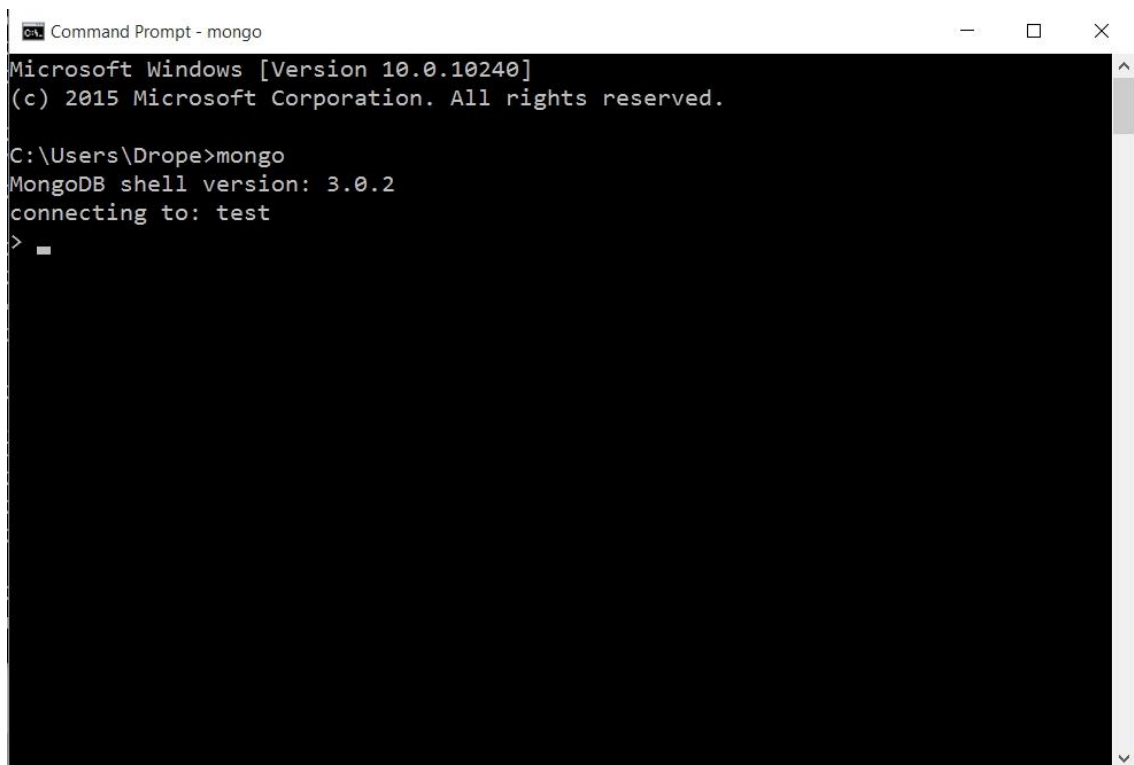
A diferença entre as duas estruturas é sua complexidade para percorrer os dados. A primeira forma é um simples mapa de chave e valor, facilitando o entendimento e a forma de percorrer os dados. A segunda forma é mais complexa, pois nela é possível inserir formas de dados dentro de um simples elemento. Isto é, ter arrays dentro de elementos que já estão dentro de arrays. O que torna complicado percorrer toda a estrutura, pois ela pode ter vários arrays dentro de arrays.

No momento, o MongoDB possui algumas interfaces amigáveis como por exemplo o NoSQL Manager for MongoDB¹⁹. Outra característica é que ele suporta a

¹⁹ <http://www.mongodbmanager.com/>, acessado em 11 janeiro 2016

linguagem JavaScript, sendo assim possível criar scripts para serem processados pelo Banco De Dados.

A figura 6 apresenta uma visão do shell do MongoDB. Através dele é possível visualizar seus dados e inserir comandos. Para o nosso projeto não foi necessário uma interface avançada, visto que a conexão com o banco é feita pela aplicação.



```
Command Prompt - mongo
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Drope>mongo
MongoDB shell version: 3.0.2
connecting to: test
> █
```

Figura 6: MongoDB shell script

Utilizando o MongoDB é possível utilizar duas abordagens para armazenar os seu dados: a relacional e a agregada.

O armazenamento de forma relacional é muito parecido com o que é visto do tradicional banco de dados relacional. Todo objeto no MongoDB possui um

identificador único, muito parecido com uma *primary key*. Assim, é possível associar os elementos da mesma forma relacional de se possuir uma *primary key* e uma *foreign key*. Sendo assim possível associar diversos elementos, como mostrado na figura 7.

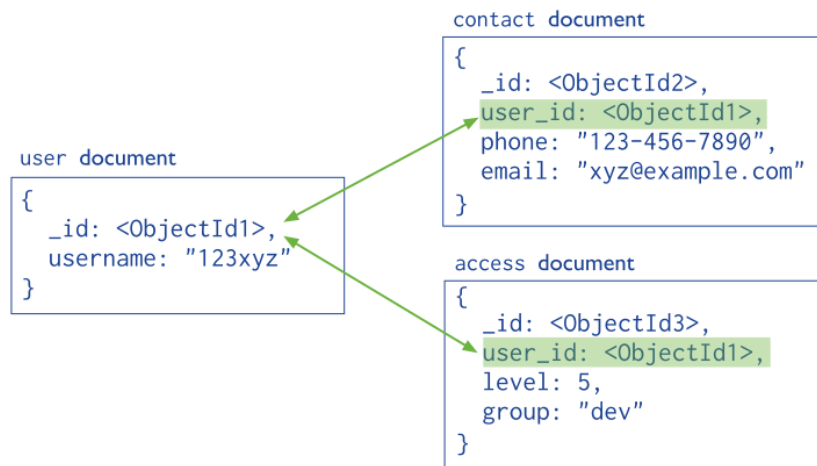


Figura 7: Exemplo de representação relacional no MongoDB

A vantagem dessa abordagem é de se conseguir relacionar os itens de uma forma a criar um conjunto de itens relacionados, muito parecido com o modelo usado pelos Bancos de Dados Relacionais. A desvantagem dessa abordagem é que no MongoDB não existem *Joins* entre os elementos. Sendo assim para se recuperar todos os elementos relacionados é necessário mais de uma consulta ao banco.

Outra forma de se armazenar os dados é de forma agregada, isto é, juntar todos os dados em um único objeto. Por exemplo, ao invés de separar o usuário em um objeto, e seu endereço em outro, junta-se os dois em um único objeto. Isso pode acarretar em informações duplicadas, e por outro lado estará ganhando em performance, já que com apenas uma simples consulta é retornado todos os dados que se procura. A figura 8 é um exemplo desta abordagem.



Figura 8: Exemplo de um elemento agregado em MongoDB

Na figura 8, *Embedded sub-document*: significa sub-documento agregado, isto é, um *document* dentro de outro *document*.

O método escolhido neste projeto foi o segundo, de forma agregada, pois todos os dados utilizados foram agrupados em um documento, e esses não possuem referências entre si.

Desenvolvimento da Solução

4.1. Estudo de caso

Tendo os princípios da tecnologia estudados, é interessante aplicá-los, e para isso era necessário um conjunto de dados de um problema. Como citado anteriormente no capítulo 3, como caso de uso foi utilizado um problema pesquisado pelo Instituto Tecgraf, que aborda o processo de pintura anticorrosiva em plataformas *offshore* de exploração de petróleo.

A divisão desta plataforma pode ser descrita da seguinte forma: Esta é dividida em módulos, que são divididos setores, que são divididos em zonas, que são divididos em sistemas, que por fim contém seus elementos, que possuem diversos valores, como nome, área, diâmetro, etc.

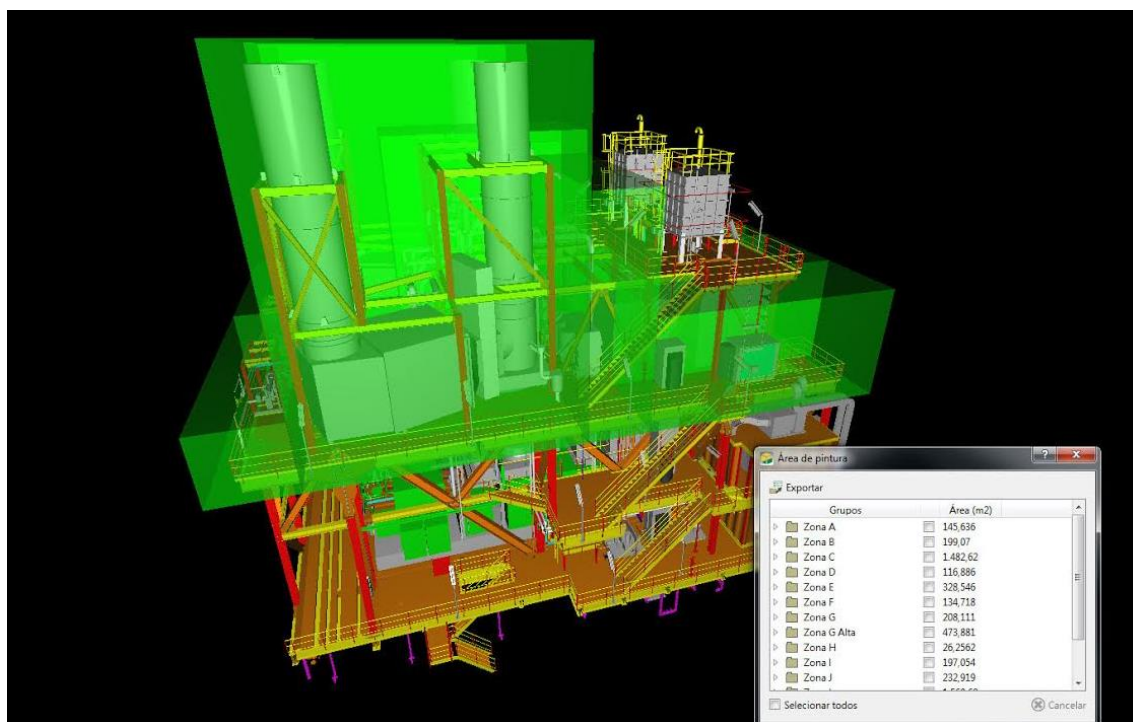


Figura 9: Visualização da divisão de um módulo

A figura 9 auxilia o entendimento da divisão de um módulo. Cada andar desse módulo é denominado um setor, que é definido na figura pela união de todos os retângulos de cor verde. Cada setor possui diferentes zonas. Cada retângulo verde na figura é uma zona. Uma zona é definida pela área que uma pessoa consegue medir em um mês. E por fim, cada zona possui seus diferentes sistemas (e.g. tubulações, teto, suportes, etc).

A pintura anticorrosiva é necessária pois esta plataforma está *offshore*, ou seja, em alto-mar, portanto esta pintura se torna importante, pois evitará ou dificultará futuras corrosões. Entretanto, esse planejamento não é uma tarefa fácil, devido à complexidade da estrutura, por exemplo esta possui uma limitação de pessoas embarcadas. É importante destacar que durante o processo de pintura a plataforma permanece inoperante, o que afeta diretamente o lucro da empresa. Além disso, há o custo do equipamento a ser utilizado e custo de pessoal, referente aos trabalhadores embarcados. Considerando todos esses fatores, torna-se importante a realização *onshore* da simulação e levantamento das operações a serem realizadas, pois é possível obter uma estimativa prévia do custo da operação.

4.2. Preparando o ambiente

Nesta seção são descritos os passos necessários para preparar o ambiente de execução da aplicação.

O primeiro passo é possuir o MongoDB²⁰, versão 3.0, instalado²¹.

Em seguida, é necessário obter o java JDK versão 7 ou superior²².

Como IDE foi utilizado o NetBeans, em conjunto com as seguintes bibliotecas:

Drive para MongoDB no Java²³.

Biblioteca para ler e escrever arquivos Excel para Java²⁴

4.3. Etapas de desenvolvimento

4.3.1. Leitura, tratamento e gravação

Para executar a aplicação, é necessário iniciar o MongoDB. No trabalho ele é iniciado pelo *Command Prompt* (CMD). Ao abrir o *Command Prompt* basta digitar o comando: “mongod”, que inicializará o MongoDB, como pode ser observado na figura 10.

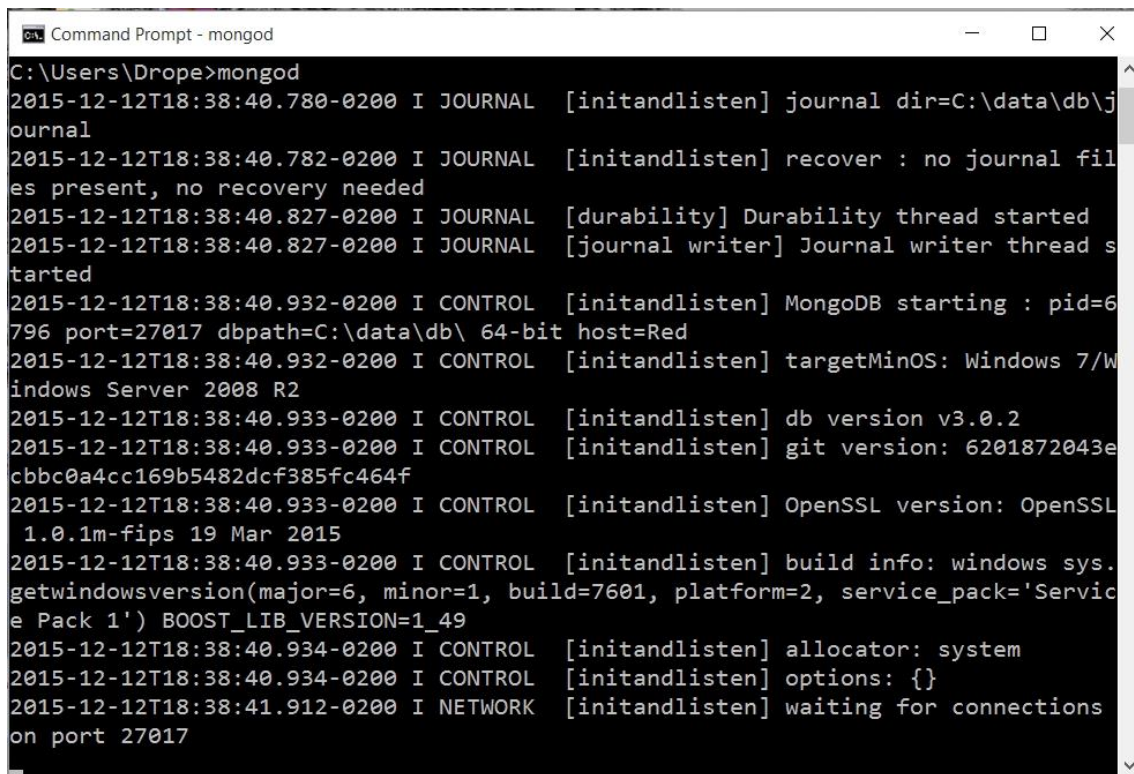
²⁰ <https://www.mongodb.org/>, acessado em 10 novembro 2015

²¹ Instalação para o sistema operacional Windows <https://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/>, acessado em 10 novembro 2015

²² <http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html>, acessado em 10 novembro 2015

²³ <https://docs.mongodb.org/ecosystem/drivers/java/>, acessado em 10 novembro 2015

²⁴ <http://jexcelapi.sourceforge.net/>, acessado em 10 novembro 2015



```
C:\Users\Drope>mongod
2015-12-12T18:38:40.780-0200 I JOURNAL [initandlisten] journal dir=C:\data\db\j
ournal
2015-12-12T18:38:40.782-0200 I JOURNAL [initandlisten] recover : no journal fil
es present, no recovery needed
2015-12-12T18:38:40.827-0200 I JOURNAL [durability] Durability thread started
2015-12-12T18:38:40.827-0200 I JOURNAL [journal writer] Journal writer thread s
tarted
2015-12-12T18:38:40.932-0200 I CONTROL [initandlisten] MongoDB starting : pid=6
796 port=27017 dbpath=C:\data\db\ 64-bit host=Red
2015-12-12T18:38:40.932-0200 I CONTROL [initandlisten] targetMinOS: Windows 7/W
indows Server 2008 R2
2015-12-12T18:38:40.933-0200 I CONTROL [initandlisten] db version v3.0.2
2015-12-12T18:38:40.933-0200 I CONTROL [initandlisten] git version: 6201872043e
cbbc0a4cc169b5482dcf385fc464f
2015-12-12T18:38:40.933-0200 I CONTROL [initandlisten] OpenSSL version: OpenSSL
 1.0.1m-fips 19 Mar 2015
2015-12-12T18:38:40.933-0200 I CONTROL [initandlisten] build info: windows sys.
getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Servic
e Pack 1') BOOST_LIB_VERSION=1_49
2015-12-12T18:38:40.934-0200 I CONTROL [initandlisten] allocator: system
2015-12-12T18:38:40.934-0200 I CONTROL [initandlisten] options: {}
2015-12-12T18:38:41.912-0200 I NETWORK [initandlisten] waiting for connections
on port 27017
```

Figura 10: Exemplo do banco de dados MongoDB rodando no prompt de comando

O primeiro passo da aplicação é o carregamento de dados no MongoDB. Para isso, a aplicação lê os arquivos .xls fornecidos e faz um tratamento nos dados para depois gravá-los no banco de dados.

Na figura 11 é apresentado um trecho do código que obtém e tratam os dados.

```
public static void leituraDeArquivos(MongoDatabase db) throws IOException,
BiffException {
    File folder = new File("src/arquivosPlataforma");
    File[] listOfFiles = folder.listFiles();

    MongoCollection myCollection = db.getCollection("pt");
    myCollection.drop();

    for (File file : listOfFiles) {
        if (file.isFile()) {
            lerArquivo(db, file.getName());
        }
    }
}
```

Figura 11: Código da função de leitura dos arquivos

A aplicação lê todos os arquivos, planilhas usadas na análise, que estão na pasta de origem. Como o código mostrado acima, a função obtém o nome de todos os arquivos que estão na pasta especificada. Depois seleciona-se a *collection*, tratada aqui por “pt”, para que se limpe todos os dados que possam estar gravados nela, esta ação é apenas uma precaução para que a *collection* esteja vazia ao início das gravações.

Tendo os endereços dos arquivos, inicia-se um *loop* percorrendo cada arquivo encontrado, para que seus dados sejam lidos, tratados e gravados no banco de dados.

```
public static void lerArquivo( MongoDBDatabase db, String fileName) throws IOException, BiffException{
    String filePath = "src/arquivosPlataformaP56/" + fileName;
    WorkbookSettings ws = new WorkbookSettings();// resolver o problema de encoding.
    ws.setEncoding("Cp1252");//encoding com utf-8

    Workbook workbook = Workbook.getWorkbook(new File(filePath),ws);
    Sheet sheet = workbook.getSheet(0);

    for(int linha=1;linha<sheet.getRows();linha++){
        Document doc = new Document(); //Mudei para o Tipo Document, pq o mongo pede esse tipo ao invés d

        for(int coluna=0;coluna<sheet.getColumns();coluna++){
            Cell cell=sheet.getCell(coluna,linha);

            if( (!cell.getContents().isEmpty()) && (!sheet.getCell(0, linha).getContents().isEmpty()) ) {

                String fullName = fileName;
                String[] noDot = fullName.split("\\.");
                String[] noUnderline = noDot[0].split("_");
                String modulo = noUnderline[1];
                String setor = noUnderline[2];

                doc.append("#grupo", noDot[0]);
                doc.append("modulo", modulo);
                doc.append("setor", setor);

                if( sheet.getCell(coluna,0).getContents().equals( "Área (m2)" )){ //precisamos converter
                    double value;

                    if(cell.getType() != CellType.NUMBER){
                        String tempCell = "" + cell.getContents();
                        tempCell = tempCell.replace(",", ".");
                        value = Double.parseDouble(tempCell);
                    }
                    else {
                        NumberCell nc = (NumberCell) cell;
                        value = nc.getValue();
                    }
                    doc.append("area", value);
                }
                else if( coluna==2 ) {
                    String cellZone = cell.getContents();
                }
            }
        }
    }
}
```

Figura 12: Trecho do código usado para recuperar e tratar os dados

```

    } else if( coluna==2 ) {
        String cellZone = cell.getContents();
        // Padronizando a coluna de zonas, e tratando de typos (Zona Alta X; Zona X Alt
        if (cellZone.matches("(\\w+) (Alta) (\\w+)")) {
            cellZone = cellZone.replaceAll("(\\w+) (Alta) (\\w+)", "$1 $3 $2");
        } else if (cellZone.matches("(\\w+) (alta) (\\w+)")) {
            cellZone = cellZone.replaceAll("(\\w+) (alta) (\\w+)", "$1 $3 Alta");
        } else if (cellZone.matches("(\\w+) (baixa) (\\s+) (\\w+)")) {
            cellZone = cellZone.replaceAll("(\\w+) (baixa) (\\w+)", "$1 $3");
        } else if (cellZone.matches("(\\w+) (alta) (\\w+)")) {
            cellZone = cellZone.replaceAll("(\\w+) (alta) (\\w+)", "$1 $3 Alta");
        } else if (cellZone.matches("(Zona) (\\w+) (Alta) (\\s+)")) {
            cellZone = cellZone.replaceAll("(Zona) (\\w+) (Alta) (\\s+)", "$1 $2 $3");
        }
        doc.append("subgrupo-zona", cellZone);
    } else {
        doc.append(sheet.getCell(coluna,0).getContents(), cell.getContents());
    }
}
}
db.getCollection("pt").insertOne( doc );//salvar os documentos na collection pt.
}

workbook.close();

```

Figura 13: Continuação da função lerArquivo, mostrando o tratamento do nome das zonas

Após abrir o arquivo, é definido um *loop* para percorrer todas as células do arquivo, como mostram as figura 12 e 13.

Para cada linha é feito um tratamento nos dados para que estes possam ser gravados no MongoDB. Este é realizado com o objetivo de facilitar as futuras *queries*, pois nem todos os nomes e dados estão apresentados de forma homogênea, tendo pequenas diferenças quanto a letras maiúsculas e minúsculas. Por exemplo: zona A e Zona A, sendo os dois a mesma zona.

Após o tratamento os dados são salvos no MongoDB. A figura 14 apresenta um exemplo de como os dados estão armazenados.


```
{
  "_id" : ObjectId("565f78cf6995180384a575a4"),
  "#grupo" : "P-56_M04_S03",
  "modulo" : "M04",
  "setor" : "S03",
  "#nome" : "T-05A 994",
  "subgrupo-zona" : "Zona A",
  "#subgrupo" : "Sistema B - Tubulações, Válvulas e Flanges",
  "area" : 0.012227,
  "*Areapds" : "4",
  "*Cleaning" : "None",
  "*Comp no" : "T-05A",
  "*Construction status" : "New",
  "*LastDate" : "07/12/05",
  "*Length" : "99.825",
  "*Line ID" : "F-B10S-2259",
  "*Line no" : "1\"-F-B10S-2259-",
  "*NPD" : "1\" 1\"",
  "*Nor oper press" : "171 kPa abs",
  "*Nor oper temp" : "25 degC",
  "*Prep" : "PL PL",
  "*SUBSISTEMA" : "5412.01 HIGH PRESSURE",
  "*Sch/thk" : "S-80 S-80",
  "*Status" : "Not approved",
  "*Steam out temp" : "0 degC",
  "*Weight" : "0.322"
}
```

Figura 14: Exemplo de um elemento gravado no mongoDB

Para poder tratar os dados e gravá-los, foi necessário compreender a divisão de uma plataforma de petróleo além dos dados disponíveis nas planilhas. Com base nos que foi explicado anteriormente pode-se entender que:

- Cada plataforma é dividida em Módulos (identificado por “modulo” no objeto mostrado na figura 14);
- Cada Módulo é dividido em Setores (identificado por “setor” no objeto mostrado na figura 14);
- Cada Setor é dividido em Zonas (identificado por “subgrupo-zona” no objeto mostrado na figura 14);
- Cada Zona é dividida em Sistemas (identificado por “#subgrupo” no objeto

mostrado na figura 14).

Tempo de gravação:

O tempo médio que o programa leva para ler todos os arquivos XLS, 27 arquivos com um total de 37.6 MB, tratar e gravar no banco é de : 56 segundos

O teste foi feito em um laptop ASUS i7-4700 2.4GHz - 16.0 GB de ram

4.3.2. Recuperação dos dados e processamento

Com os dados tratados e gravados no MongoDB, o próximo passo é recuperá-los e processá-los de forma a gerar novas informações. O foco, nesta parte, é conseguir cruzar os dados de diferentes setores dos diversos módulos, e inferir métricas a fim de descobrir o gasto total de cada setor em variação ao número de trabalhadores, e o quanto é pago por hora para cada um.

Para os cálculos foram utilizadas as seguintes métricas:

1. Tratamento de superfície

WJ-2 = (onshore: 1 USP/m² ou offshore: 1,25 USP/m²) x 115 x m² x 0,2 (reais)

WJ-3 = (onshore: 0,75 USP/m² ou offshore: 1 USP/m²) x 115 x m² x 0,8 (reais)

O somatório das duas equações acima dá o custo total do tratamento de superfície.

Nota: USP = Unidade de Serviço de Pintura

WJ é uma forma tratamento da superfície utilizando-se um hidrotrato²⁵

2. Aplicação da tinta de alto desempenho

Valor = R\$ 112,00 x m² = total da tinta (reais)

m² = metros quadrados.

3. Equipamento

Máquina de hidrotrato = R\$ 3250,28 x dia (reais)

dia = dias de trabalho

R\$ 3250,28 é o preço do aluguel do equipamento.

4. Custo por trabalhador

Salário = horas trabalhadas por dia x valor da hora x dias

O custo total é dado por: Tratamento de superfície + Aplicação da tinta de alto desempenho + Equipamento + Preço por trabalhador.

É preciso ressaltar que estas métricas são simulações, pois como se tratam de dados industriais de um setor sensível, não foi encontrado as reais métricas e valores de mercado.

Inicialmente precisa-se recuperar os dados do MongoDB. O objetivo é obter os dados referentes à área total de cada setor por cada módulo. A *query* apropriada para obtenção destes dados em comando *shell* no MongoDB é:

²⁵ <http://www.ebah.com.br/content/ABAAABu-oAI/pintura-industrial>, acessado em 10 janeiro 2016

```

db.pt.aggregate([
    $group:{_id:
        { "modulo" : "$modulo" , "setor " : "$setor"},
        total: {$sum : "$area"}
    }
},
{ $sort: { modulo: -1 }}
]);

```

A mesma *query* em SQL agrupará os dados pelos atributos modulo e setor (“GROUP BY”), somando o atributo area (“SUM”), e ordenando pelo atributo modulo (“ORDER BY”).

Na figura 15 segue a implementação em Java.

```

private static List<Object> retornaResultadoQueryComDoisParametros(String
nomeModulo, String nomeSetor){

    MongoClient<Document> ptCollection = initiateMongoCollection();

    AggregateIterable<Document> agg = ptCollection.aggregate(asList(

        new Document("$match",new
Document("modulo",java.util.regex.Pattern.compile("^.*$")),

        new Document("$group",new Document("_id",new
Document(nomeModulo,"$"+nomeModulo).append(nomeSetor,"$"+nomeSetor))

            .append("Total", new Document("$sum","$area"))),

        new Document("$sort", new Document("Total",1)) ));

    List<Object> list0 = new ArrayList<>();
    agg.forEach(new Block<Document>() {

        @Override

        public void apply(final Document document) {

            int control = 1;

            for(Object o : document.values()) {

                if(control==1){

                    Document aux = (Document) document.get("_id");

                    String modAux = aux.getString(nomeModulo);

                    String setAux = aux.getString(nomeSetor);

                    list0.add(modAux);

                    list0.add(setAux);

                    control = 2;

                } else {

                    list0.add(document.getDouble("Total"));

                    control = 1;

                }

            }

        }

    });

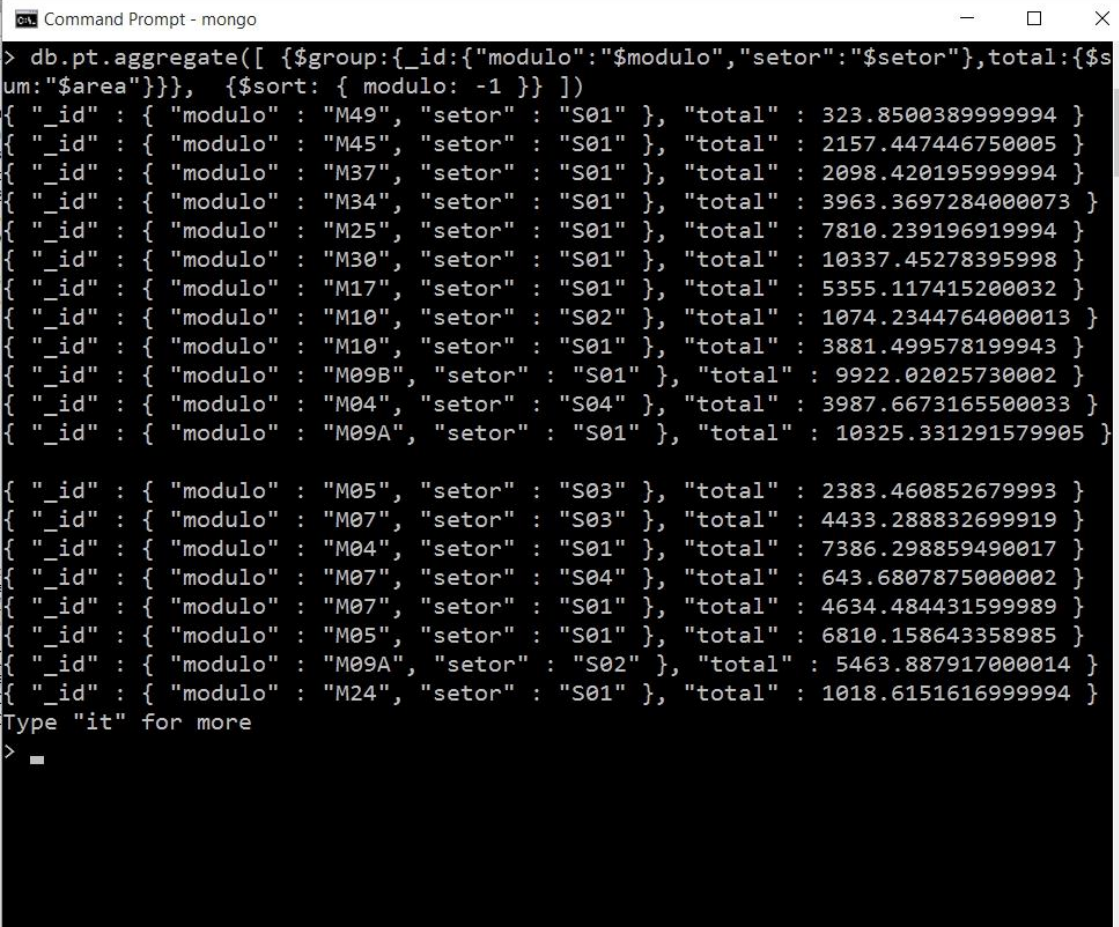
    return list0;

}

```

Figura 15: Código da função de busca da área total de cada módulo

As consultas escritas no *shell* do MongoDB e em Java têm o mesmo objetivo: agregar todos os setores por módulos e somar suas áreas. O retorno para tal busca, ilustrado na figura 16, é o mesmo nos dois formatos.



```
> db.pt.aggregate([{$group:{$_id:{$modulo: '$modulo', 'setor': '$setor'}, total: {$sum: '$area'}}}, {$sort: { modulo: -1 }}])
{ "_id" : { "modulo" : "M49", "setor" : "S01" }, "total" : 323.8500389999994 }
{ "_id" : { "modulo" : "M45", "setor" : "S01" }, "total" : 2157.447446750005 }
{ "_id" : { "modulo" : "M37", "setor" : "S01" }, "total" : 2098.420195999994 }
{ "_id" : { "modulo" : "M34", "setor" : "S01" }, "total" : 3963.3697284000073 }
{ "_id" : { "modulo" : "M25", "setor" : "S01" }, "total" : 7810.239196919994 }
{ "_id" : { "modulo" : "M30", "setor" : "S01" }, "total" : 10337.45278395998 }
{ "_id" : { "modulo" : "M17", "setor" : "S01" }, "total" : 5355.117415200032 }
{ "_id" : { "modulo" : "M10", "setor" : "S02" }, "total" : 1074.2344764000013 }
{ "_id" : { "modulo" : "M10", "setor" : "S01" }, "total" : 3881.499578199943 }
{ "_id" : { "modulo" : "M09B", "setor" : "S01" }, "total" : 9922.02025730002 }
{ "_id" : { "modulo" : "M04", "setor" : "S04" }, "total" : 3987.6673165500033 }
{ "_id" : { "modulo" : "M09A", "setor" : "S01" }, "total" : 10325.331291579905 }
{ "_id" : { "modulo" : "M05", "setor" : "S03" }, "total" : 2383.460852679993 }
{ "_id" : { "modulo" : "M07", "setor" : "S03" }, "total" : 4433.288832699919 }
{ "_id" : { "modulo" : "M04", "setor" : "S01" }, "total" : 7386.298859490017 }
{ "_id" : { "modulo" : "M07", "setor" : "S04" }, "total" : 643.6807875000002 }
{ "_id" : { "modulo" : "M07", "setor" : "S01" }, "total" : 4634.484433599989 }
{ "_id" : { "modulo" : "M05", "setor" : "S01" }, "total" : 6810.158643358985 }
{ "_id" : { "modulo" : "M09A", "setor" : "S02" }, "total" : 5463.887917000014 }
{ "_id" : { "modulo" : "M24", "setor" : "S01" }, "total" : 1018.6151616999994 }
Type "it" for more
>
```

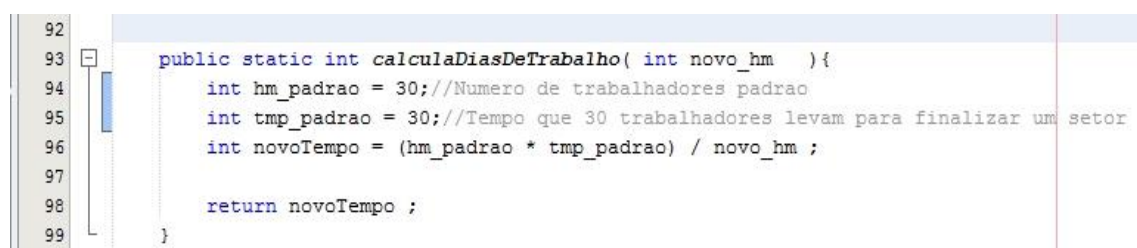
Figura 16: Resultado da busca do total da área de todos os setores por módulo

Tendo a lista de cada setor por módulo e a área total de cada um, é inicializado o processo de cálculo, utilizando as métricas citadas anteriormente. Foi calculado o tratamento de superfície, quantidade de tinta de alto desempenho usada, aluguel do equipamento e salário de todos os funcionários no espaço de tempo necessário para concluir a obra.

Além disso, para se ter o custo total, é necessário saber o quanto será gasto com

o custo de pessoal, que é dado pelo tempo de trabalho dos funcionários. Na métrica utilizada para a simulação foi estipulado o valor de 30 trabalhadores, que levam 30 dias para concluir a pintura de um setor. Utilizando essa informação como média, foi calculado quanto tempo N trabalhadores levariam para pintar cada módulo, sendo N um número inteiro positivo qualquer. Assim será obtido o tempo de pintura, que será necessário para calcular o preço total de cada setor. O número de trabalhadores pode ser inserido pelo usuário, a fim de descobrir quanto a mais será gasto na pintura, e quanto tempo levará para terminar o setor.

Vale ressaltar que essa métrica é limitada, pois está sujeita a limitações físicas e espaciais do ambiente.



```
92  
93 public static int calculaDiasDeTrabalho( int novo_hm ){  
94     int hm_padrao = 30; //Numero de trabalhadores padrao  
95     int tmp_padrao = 30; //Tempo que 30 trabalhadores levam para finalizar um setor  
96     int novoTempo = (hm_padrao * tmp_padrao) / novo_hm ;  
97  
98     return novoTempo ;  
99 }
```

Figura 17: Função que calcula o tempo levado para finalizar um setor

Se tratando de uma simples aplicação, não foi necessário criar várias classes para armazenar os dados tratados. Para guardar as informações tratadas do total da área de cada setor por módulo foi criado uma classe de domínio chamada Registro, que armazena o nome do setor, o nome do modulo, a área total e os preços de cada item da métrica utilizada. A figura 18 apresenta representação desta classe, seguindo o modelo UML para diagrama de classe.



Figura 18: Modelo da classe Registro

4.3.3. Criação dos arquivos .xls com as informações extraídas

Após obter a informação de cada setor, é formada uma lista de Registros, com os dados obtidos, a partir da qual gera-se o arquivo .xls apresentando todos os valores referentes ao cálculo do preço total de um setor, utilizando as variáveis “quantidade de trabalhadores” e “salário do trabalhador por hora” além da métrica para cálculo de pintura *offshore*.

	A	B	C	D	E	F	G	H	I	J	K	L
	Módulo/Setor	Área Total	Wj2	Wj3	Preço Tratamento de Superfície	Preço Aplicação de Tinta de Alto Desempenho	Preço Equipamento	Custo Pessoal	Dias Trabalhandos	Preço Total	Homem M2	Preço total de cada módulo
1	M02-S01	1031.948328	29.668.51	94.939.25	124.607.76	115.578.21	97.208.40	270.000.00	30	607.394.37	34.40	607.394.37
3	M04-S01	7386.298859	212.356.09	679.539.50	891.895.59	827.265.47	97.208.40	270.000.00	30	2.086.369.46	246.21	
4	M04-S02	5850.453311	168.200.53	538.241.70	706.442.24	655.250.77	97.208.40	270.000.00	30	1.728.901.41	195.02	
5	M04-S03	3301.337341	94.913.45	303.723.04	398.636.48	369.749.78	97.208.40	270.000.00	30	1.135.594.67	110.04	
6	M04-S04	3987.667317	114.645.44	366.865.39	481.510.83	446.618.74	97.208.40	270.000.00	30	1.295.337.97	132.92	6.246.203.51
7	M05-S01	6810.158643	195.792.06	626.534.60	822.326.66	762.737.77	97.208.40	270.000.00	30	1.952.272.82	227.01	
8	M05-S02	4712.783497	135.492.53	433.576.08	569.068.61	527.831.75	97.208.40	270.000.00	30	1.464.108.76	157.09	
9	M05-S03	2383.460853	68.524.50	219.278.40	287.802.90	266.947.62	97.208.40	270.000.00	30	921.958.91	79.45	4.338.340.49
10	M07-S01	4634.484432	133.241.43	426.372.57	559.614.00	519.062.26	97.208.40	270.000.00	30	1.445.884.65	154.48	
11	M07-S02	3710.531906	106.677.79	341.368.94	448.046.73	415.579.57	97.208.40	270.000.00	30	1.230.834.70	123.68	
12	M07-S03	4433.288833	127.457.05	407.862.57	535.319.63	496.528.35	97.208.40	270.000.00	30	1.399.056.38	147.78	
13	M07-S04	643.6807875	18.505.82	59.218.63	77.724.46	72.092.25	97.208.40	270.000.00	30	517.025.10	21.46	4.592.800.83
14	M09A-S01	10325.33129	296.853.27	949.930.48	1.246.783.75	1.156.437.10	97.208.40	270.000.00	30	2.770.429.26	344.18	
15	M09A-S02	5463.887917	157.086.78	502.677.69	659.764.47	611.955.45	97.208.40	270.000.00	30	1.638.928.31	182.13	4.409.357.57
16	M09B-S01	9922.020257	285.258.08	912.825.86	1.198.083.95	1.111.266.27	97.208.40	270.000.00	30	2.676.558.61	330.73	
17	M09B-S02	5397.184729	155.169.06	496.541.00	651.710.06	604.484.69	97.208.40	270.000.00	30	1.623.403.15	179.91	4.299.961.76
18	M10-S01	3881.499578	111.593.11	357.097.96	468.691.07	434.727.95	97.208.40	270.000.00	30	1.270.627.43	129.38	
19	M10-S02	1074.234476	30.884.24	98.829.57	129.713.81	120.314.26	97.208.40	270.000.00	30	617.236.47	35.81	1.887.863.90
20	M17-S01	5355.117415	153.959.63	492.670.80	646.630.43	599.773.15	97.208.40	270.000.00	30	1.613.611.98	178.50	1.613.611.98
21	M24-S01	1018.615162	29.285.19	93.712.59	122.997.78	114.084.90	97.208.40	270.000.00	30	604.291.08	33.95	604.291.08
22	M25-S01	7810.239197	224.544.38	718.542.01	943.086.38	874.746.79	97.208.40	270.000.00	30	2.185.041.57	260.34	2.185.041.57
23	M30-S01	10337.45278	297.201.77	951.045.66	1.248.247.42	1.157.794.71	97.208.40	270.000.00	30	2.773.250.54	344.58	2.773.250.54
24	M34-S01	3963.369728	113.946.88	364.630.02	478.576.89	443.897.41	97.208.40	270.000.00	30	1.289.682.70	132.11	1.289.682.70
25	M35-S01	2655.208666	76.337.25	244.279.20	320.616.45	297.383.37	97.208.40	270.000.00	30	985.208.22	88.51	985.208.22
26	M37-S01	2098.420196	60.329.58	193.054.66	253.384.24	235.023.06	97.208.40	270.000.00	30	855.615.70	69.95	855.615.70
27	M45-S01	2157.447447	62.026.61	198.485.17	260.511.78	241.634.11	97.208.40	270.000.00	30	869.354.29	71.91	869.354.29
28	M49-S01	323.850039	9.310.69	29.794.20	39.104.89	36.271.20	97.208.40	270.000.00	30	442.584.50	10.80	442.584.50

Figura 19: Planilha gerada com os parâmetros: Trabalhadores: 30; Salário: R\$50,00/dia; Duração: 30 dias (Anexo B: ResultadoModuloSetor-30h-50s.xls)

4.3.4. Consultas e Gráficos

Após a implementação da aplicação, o passo seguinte são as simulações com diferentes valores para a quantidade de trabalhadores e valor pago a esses por hora, a fim de calcular o custo total da pintura e a quantidade de dias que tal obra poderá levar.

Os gráficos, presentes na figuras 20, 21 e 22, foram gerados com os valores encontrados nas planilhas no Anexo B:Arquivos de saída.

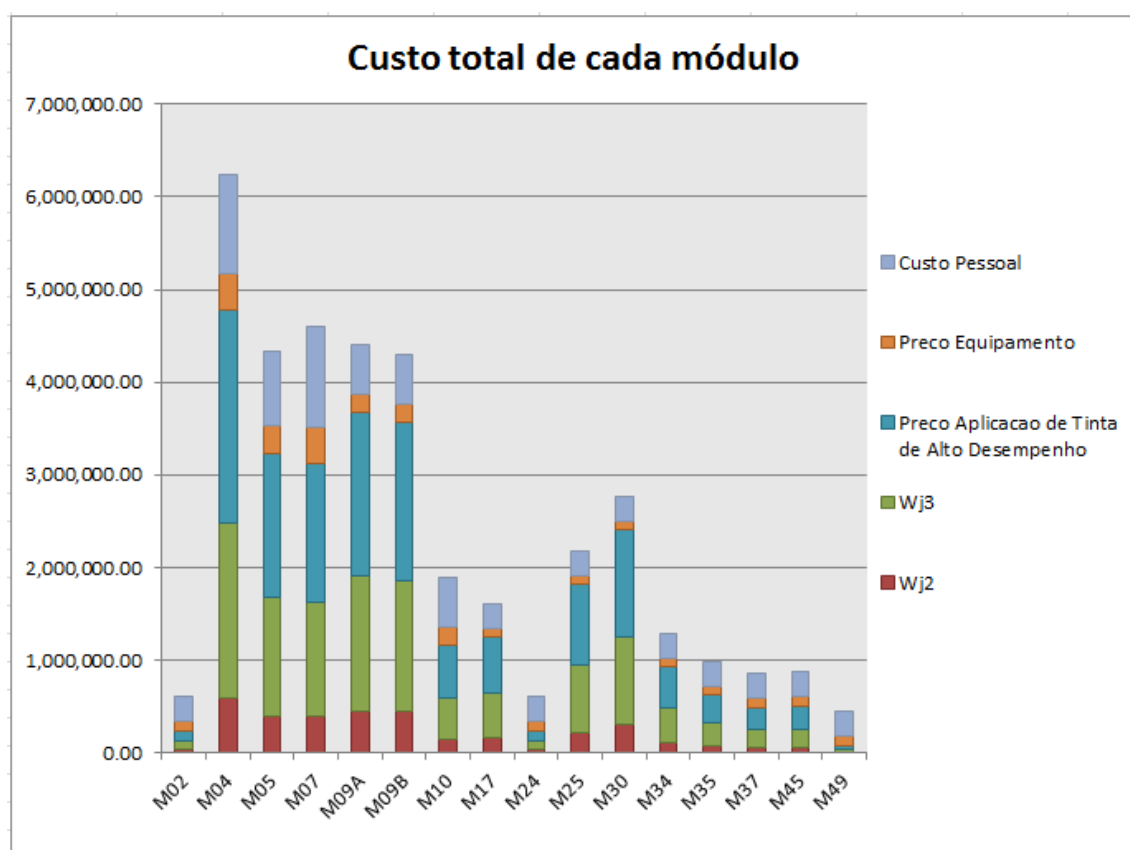


Figura 20: Gerado com os parâmetros: Trabalhadores: 30; Salário: R\$50,00/dia; Duração: 30 dias; (Anexo B: ResultadoModuloSetor-30h-50s.xls)

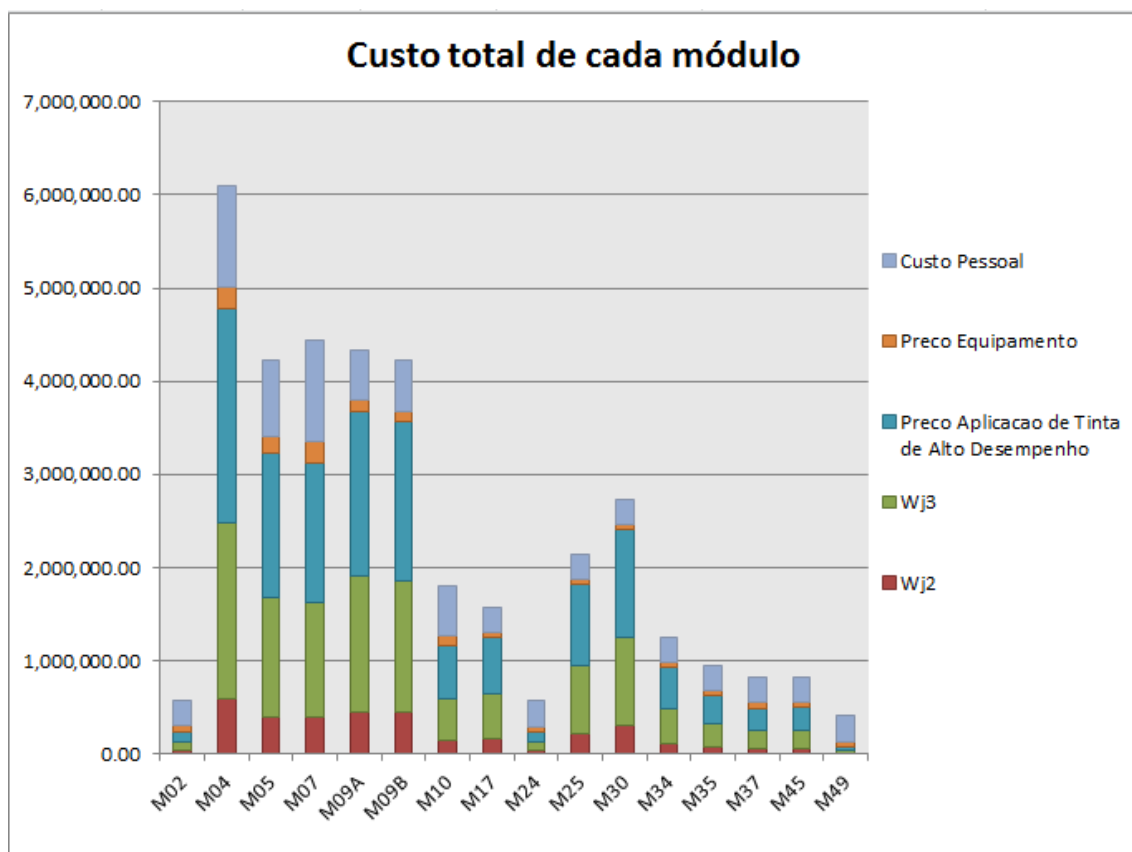


Figura 21: Gerado com os parâmetros: Trabalhadores: 50; Salário: R\$50,00/dia; Duração: 18 dias; (Anexo B: ResultadoModuloSetor-50h-50s.xls)

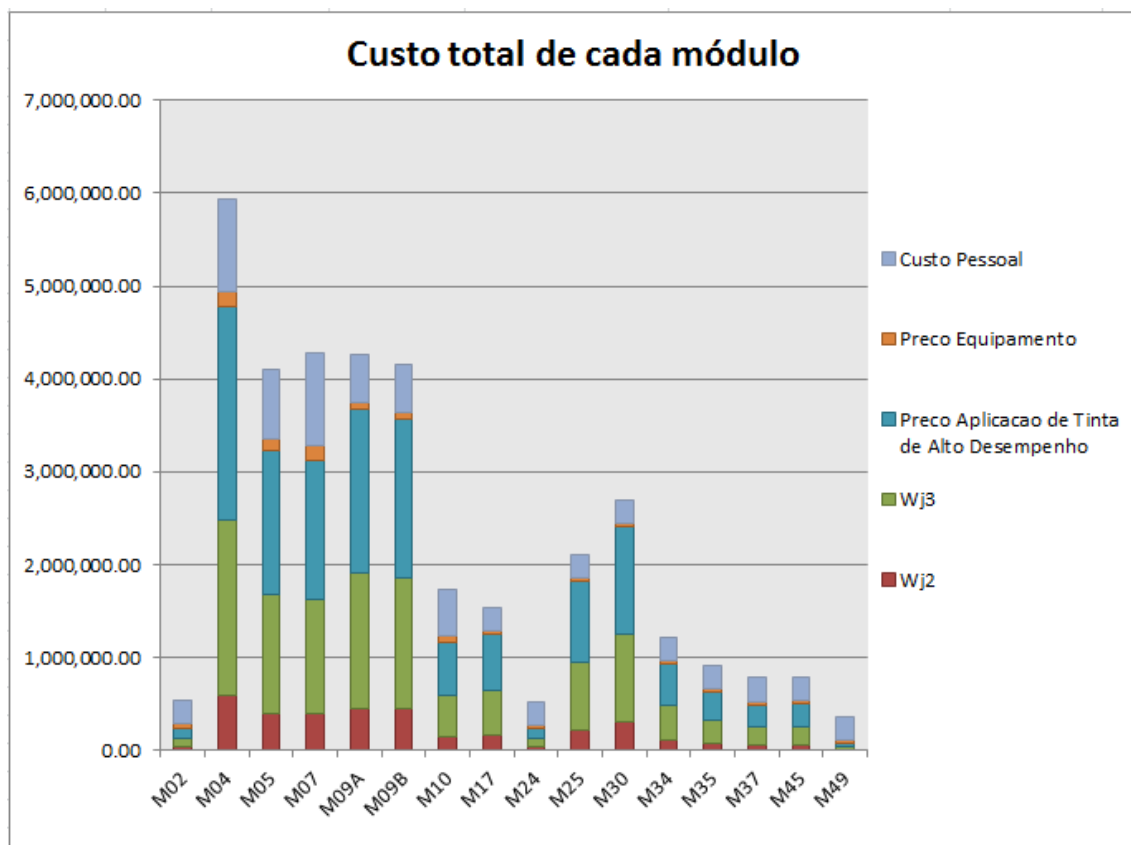


Figura 22: Gerado com os parâmetros: Trabalhadores: 70; Salário: R\$50,00/dia; Duração: 12 dias; (Anexo B: ResultadoModuloSetor-70h-50s.xls)

Comparando os gráficos das figuras 20 e 21, é perceptível que o custo total é praticamente o mesmo, mas a quantidade de dias necessários é menor, ou seja, o trabalho seria finalizado com antecedência. A contratação de mais trabalhadores agiliza a pintura, diminuindo assim o tempo de aluguel das máquinas. O preço da tinta, neste caso, não se altera. Essa é uma informação importante para a empresa, pois torna possível prever gastos, entre outras métricas. Sendo que para realizar a manutenção e pintura anticorrosiva da plataforma, esta fica parada, não realizando seu trabalho. Assim, descobrir que há uma forma de diminuir esse tempo de ócio é de grande relevância.

Conclusão e Trabalhos futuros

Após a realização do trabalho foi percebida a possibilidade da utilização da tecnologia NoSQL para migrar dados de planilhas Excel para um banco de dados não relacional, e conseguir extrair informações destes, para que se possa inferir conhecimento sobre os dados obtidos.

A não necessidade de se ter um esquema de dados definido foi fundamental para a rápida implementação da aplicação. Essa flexibilidade em relação aos bancos de dados relacionais fez com que o início da execução do trabalho realizada com maior rapidez, pois o banco NoSQL consegue se adaptar facilmente aos dados fornecidos. Para modelar os dados apresentados em um banco de dados relacional, seriam necessárias no mínimo 5 tabelas: Modulo, Setor, Zona, Sistema e atributos. Logo, seriam 5 tabelas para manter, além de que, para retirar informações como a área de setores por módulo, a operação de *join* ocorreria entre 3 tabelas. Outro fator importante é com relação aos atributos, pois nas planilhas fornecidas, muitos elementos não possuem certos atributos, pois não fazem parte de suas características. No modelo relacional a tabela atributos teria muitos campos *null*, enquanto no MongoDB o campo apenas não é registrado.

Como mencionado anteriormente, este trabalho de levantamento de dados e análise era realizado manualmente. Com a aplicação desenvolvida neste projeto, o que uma pessoa levaria dias para concluir, pode ser realizado em minutos. Além disso, é

eliminada a ocorrência de erro humano, que pode acontecer no processamento dos dados, quando realizado manualmente. Utilizando a aplicação, o usuário fica responsável apenas pela interpretação dos resultados.

Como sugestão para trabalhos futuros, tendo os dados completos sobre as plataformas e as métricas reais usadas, há a possibilidade de se estender essa ferramenta para responder outras consultas, ou para cruzar dados entre tabelas distintas, isto é, o programa ler duas tabelas distintas e através dos tipos encontrados nas colunas retornar os possíveis cruzamentos entre as mesmas.

Vale observar que os dados aqui usados foram extraídos apenas de uma das plataformas de petróleo. Um possível trabalho futuro poderia comparar diferentes plataformas de petróleo existentes.

Como mencionado anteriormente, este é apenas um protótipo, com apenas algumas consultas e sem uma interface amigável, sendo assim aperfeiçoamento da ferramenta com o desenvolvimento de uma interface para o usuário com a possibilidade de criar outras consultas que abrangem outros atributos seria interessante. Realizando assim um *Business Intelligence* (BI), um sistema que combina junção de dados, armazenamento e processamento dos dados e aplicação de conhecimento sobre ele para a extração de informações importantes para a tomada de decisão²⁶.

Vale ressaltar que é interessante uma análise de desempenho comparando o banco de dados relacional e o banco NoSQL .

²⁶ <http://aisel.aisnet.org/cgi/viewcontent.cgi?article=3234&context=cais>, acessado em 25 janeiro 2016

Referências Bibliográficas

Atzeni, P. (2013) “The relational model is dead, SQL is dead, and I don’t feel so good myself.” http://www.orsigiorgio.net/wp-content/papercite-data/pdf/ajo*13.pdf, acessado em Dezembro 2015.

Bartholomew, D. (2010) “SQL vs NoSQL” <http://www.linuxjournal.com/article/10770>, acessado em Dezembro 2015.

Fowler, Adam (2015), NoSQL for dummies, Wiley Brand, 1th edition.

Nance, C. (2013) “NoSQL vs RDBMS, why there is a room for both.” <http://sais.aisnet.org/2013/Nance.pdf>, acessado em Dezembro 2015.

Robinson, Ian and Webber, Jim and Eifrem, Emil (2013). Graph Databases, O’Reilly’s, 2nd edition.

Sharma V. and Dave M. (2012). SQL and NoSQL Databases. International Journal of Advanced Research in Computer Science and Software Engineering, pages 20-27. http://www.ijarcse.com/docs/papers/8_August2012/Volume_2_issue_8/V2I800154.pdf, acessado em Dezembro 2015.

Sherman, R. (2009). “BI tools vs. Microsoft Excel spreadsheets” <http://searchcio.techtarget.com/podcast/BI-tools-vs-Microsoft-Excel-spreadsheets>, acessado em Dezembro 2015.

Anexos

Anexo A: Archivos de entrada

P_M02_S01.xls

P_M04_S01.xls

P_M04_S02.xls

P_M04_S03.xls

P_M04_S04.xls

P_M05_S01.xls

P_M05_S02.xls

P_M05_S03.xls

P_M07_S01.xls

P_M07_S02.xls

P_M07_S03.xls

P_M07_S04.xls

P_M09A_S01.xls

P_M09A_S02.xls

P_M09B_S01.xls

P_M09B_S02.xls

P_M10_S01.xls

P_M10_S02.xls

P_M17_S01.xls

P_M24_S01.xls

P_M25_S01.xls

P_M30_S01.xls

P_M34_S01.xls

P_M35_S01.xls

P_M37_S01.xls

P_M45_S01.xls

P_M49_S01.xls

Anexo B: Arquivos de saída

Resultado-Modulo-Setor-Zona.xls

ResultadoModuloSetor-30h-50s.xls

ResultadoModuloSetor-50h-50s.xls

ResultadoModuloSetor-70h-50s.xls