UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE LUIS LOPES

ELASTICSEARCH: ESTUDO E DEMONSTRAÇÃO ATRAVÉS DE UMA APLICAÇÃO DE RECEITAS

BAURU - SP

2015

FELIPE LUIS LOPES

ELASTICSEARCH: ESTUDO E DEMONSTRAÇÃO ATRAVÉS DE UMA APLICAÇÃO DE RECEITAS

Trabalho de Conclusão de Curso de Curso de graduação apresentado à disciplina Projeto e Implementação de Sistemas do curso de Bacharelado em Ciência da Computação da Faculdade de Ciências da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como requisito parcial para obtenção do título de bacharel.

**Orientador**: Prof. Dr. Luiz Francisco da Cruz

BAURU - SP

2015FELIPE LUIS LOPES

ELASTICSEARCH: ESTUDO E DEMONSTRAÇÃO ATRAVÉS DE UMA APLICAÇÃO DE RECEITAS

Trabalho de Conclusão de Curso de Curso de graduação apresentado à disciplina Projeto e Implementação de Sistemas do curso de Bacharelado em Ciência da Computação da Faculdade de Ciências da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como requisito parcial para obtenção do título de bacharel.

Aprovado em: \_\_\_/\_\_\_/\_\_\_

**BANCA EXAMINADORA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Prof. Dr. Luiz Francisco da Cruz**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Prof. Dr. Aparecido Nilceu Marana**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Profa. Dra. Roberta Spolon**

**AGRADECIMENTOS**

A Deus pоr tеr mе dado saúde е força pаrа superar аs dificuldades.

Ao mеυ orientador, pelo emprenho dedicado à elaboração deste trabalho e ao Marcos Fabricio que me orientou na elaboração do trabalho e pelo suporte.

A todos qυе direta оυ indiretamente fizeram parte dа minha formação, о mеυ muito obrigado.

*“Information is power.*

*But like all power, there are those*

*Who want to keep it for themselves.”*

*(Aaron Swartz)*

**RESUMO**

O uso de ferramentas de busca, traz muitos benefícios e ajuda a otimizar o tempo quando se deseja encontrar informações em uma base de dados. Uma dessas ferramentas se chama Elasticsearch. Esta ferramenta foi concebida com o objetivo de otimizar o tempo de busca (consulta) em banco de dados, seja grande ou não. Mas como essas ferramentas funcionam? Este trabalho apresenta como implementar uma busca utilizando o Elasticsearch e quais os conceitos por traz do seu núcleo. Afim de mostrar Elasticsearch em funcionamento, é desenvolvido um aplicativo de receitas, onde será realizada a busca com base nos ingredientes e não o nome da receita. Também são apresentados conceitos de API REST, plataforma móvel e banco de dados não relacional.

Ao final deste trabalho, como resultado das uniões das tecnologias mencionadas, conseguiu-se obter informações quanto a implementação e buscas utilizando o Elasticsearch, juntamente com subsídios que garantam essa implementação tanto em plataformas Web como Móveis.

**Palavras-Chaves**: elasticsearch; busca; rest;

**ABSTRACT**

The use of search engines, brings many benefits and helps optimize the time when you want to find information in a database. One of these tools is called ElasticSearch. This tool has been designed in order to optimize the search time (query) in the database, whether big or not. But how these tools work? This paper presents how to implement search by ElasticSearch and the concepts behind it's core. In order to show ElasticSearch running, a cooking recipe application was developed, where the search is based on the ingredients and not the name of the recipe. It is also presented concepts of REST API, mobile platform and non-relational database.

At the end of this work, as a result of the union of the mentioned technologies, we were able to obtain information on the implementation and searches using the ElasticSearch along with subsidies to ensure this implementation on both Web and Mobile platforms.

**Keywords**: elasticsearch; search; rest;

**LISTA DE ILUSTRAÇÕES**

[Figura 1 - Arquitetura ElasticSearch. 19](file:///C:\Users\felul\Downloads\Monografia%20Revisada_v3%20(1).docx#_Toc441789514)

[Figura 2 - Exemplo de estrutura de Mapping 25](file:///C:\Users\felul\Downloads\Monografia%20Revisada_v3%20(1).docx#_Toc441789515)

[Figura 3 - Exemplo de estrutura de Leaf Query 26](file:///C:\Users\felul\Downloads\Monografia%20Revisada_v3%20(1).docx#_Toc441789516)

[Figura 4 - Arquitetura do Sistema 34](file:///C:\Users\felul\Downloads\Monografia%20Revisada_v3%20(1).docx#_Toc441789517)

[Figura 5 - Leaf Query para pesquisar por nome de receita 35](#_Toc441789518)

[Figura 6 - Compound Query para realizar a busca por ingredientes 36](#_Toc441789519)

[Figura 7 - Fluxo de telas da Aplicação 37](#_Toc441789520)

[Figura 8 - Tela de Busca por nome 38](#_Toc441789521)

[Figura 9 - Tela da receita 38](#_Toc441789522)

[Figura 10 - Tela de busca por ingredientes 38](#_Toc441789523)

**LISTA DE TABELAS**

[Tabela 1 - Lista de dispositivos e navegadores 39](#_Toc441789535)

**LISTA DE ABREVIATURAS**

URL - *Uniform Resource Locator*

ANSI *- American National Standards Institute*

API - *Application Programming Interface*

REST - *Representational State Transfer*

JSON - *JavaScript Object Notation*

noSQL - *Not Only SQL*

HTTP - *Hypertext Transfer Protocol*

RSS *– Really Simple Sindication*

FTP *– File Transfer Protocol*

**SUMÁRIO**

[1 INTRODUÇÃO 13](#_Toc441789567)

[1.1 JUSTIFICATIVA 15](#_Toc441789568)

[1.2 OBJETIVOS 15](#_Toc441789569)

[1.2.1 Objetivos Específicos 15](#_Toc441789570)

[2 FUNDAMENTAÇÃO TEÓRICA 17](#_Toc441789571)

[2.1 BREVE INTRODUÇÃO A BUSCA DE INFORMAÇÃO 17](#_Toc441789572)

[2.2 APACHE LUCENE 17](#_Toc441789573)

[2.3 ELASTICSEARCH 18](#_Toc441789574)

[2.3.1 Pesquisa Relevante 20](#_Toc441789575)

[2.3.2 Mappings 24](#_Toc441789576)

[2.3.3 Query DSL 25](#_Toc441789577)

[2.4 REST 27](#_Toc441789578)

[2.5 MONGODB 28](#_Toc441789579)

[2.6 NODE.JS E EXPRESS.JS 29](#_Toc441789580)

[2.7 HTML, JAVASCRIPT E CORDOVA 30](#_Toc441789581)

[3 METODOLOGIA 33](#_Toc441789582)

[3.1 MODELAGEM DO DADO 33](#_Toc441789583)

[3.2 ARQUITETURA DO SISTEMA 33](#_Toc441789584)

[3.3 MÓDULO API REST BANCO DE DADOS 34](#_Toc441789585)

[3.4 ELASTICSEARCH 35](#_Toc441789586)

[3.5 MÓDULO APLICAÇÃO MÓVEL 36](#_Toc441789587)

[3.6 EXPERIMENTOS 39](#_Toc441789588)

[4 RESULTADOS 40](#_Toc441789589)

[4.1 BUSCA POR NOME DA RECEITA 40](#_Toc441789590)

[4.2 BUSCA POR INGREDIENTES 40](#_Toc441789591)

[5 CONCLUSÃO 43](#_Toc441789592)

[REFERÊNCIAS 45](#_Toc441789593)

[ANEXOS 47](#_Toc441789594)

# INTRODUÇÃO

Com o avanço da tecnologia, observou-se o aumento exponencial da informação de várias maneiras (SCHONS, 2007). A forma como a internet evoluiu, proporcionou caminhos diferentes de acesso a informação e a geração destas informações, como por exemplo, o surgimento do RSS, forma de assinatura de conteúdo dinâmico (KING, 2001), e as redes sociais, redes na qual cada usuário se interliga com outro usuário gerando conteúdo (RECUERO, 2007). Transformar o conglomerado dessas informações em conhecimento útil para a necessidade humana, desde discussões sociais a conhecimento científicos, se torna um grande benefício para a sociedade.

Entretanto, a transformação das informações traz alguns dos maiores problemas: Como realizar uma busca em toda a informação adquirida? Como resgatar essa informação e transformá-la em relevante para o usuário?

O conglomerado de informações que os usuários criam diariamente, torna essas informações desorganizadas e esparsas (SCHONS, 2007).

Um dos problemas do aumento de informações é citado por SCHONS

Tal fenômeno a princípio é positivo, mas no decorrer do tempo esse processo originou um grande problema na rede pelo fato de conduzir e fomentar o processo de explosão informacional gerando grandes conflitos, uma vez que o excesso de informações parece ser pior que a sua falta. Um exemplo pode ser colocado da seguinte maneira: um usuário, ao efetuar uma pesquisa na Internet, tende a ficar mais frustrado pelo excesso de “respostas” que pela falta delas. Isto porque o excesso de informações não representa uma solução, mas um problema: a desinformação. (SCHONS, 2007, p. 4).

Juntamente com os problemas do aumento da informação, surgem as soluções. Entre as soluções atuais, encontra-se o mecanismo disponibilizado pelo Google[[1]](#footnote-1). Muito difundida, a ferramenta de busca do Google tornou-se um centralizador de busca para documentos, realizando buscas através de palavras-chaves digitadas pelo usuário (PEREIRA, 2004).

Dentre as soluções, como Yahoo Search, Bing.com e Google Search, possuem o modelo de funcionamento composto por dois processos: a indexação dos documentos e a busca dos documentos (PEREIRA, 2004).

Como requisito para o resgate dos documentos, é necessário a indexação dos documentos na base. A indexação é o processo de leitura do documento, a análise para obter as informações importantes e o armazenamento na base (FUJITA, 2003). E por fim, a busca dos documentos através de palavras-chave. O resgate das informações vem classificado pela sua relação com as palavras-chave (PEREIRA, 2004).

Entretanto como realizar a mesma solução para busca de informações que estão restringidas em um domínio específico, como um sistema de arquivos do Windows[[2]](#footnote-2) ou sistema administrativo SAP[[3]](#footnote-3)? A resposta é usando o mesmo modelo de conceito implementado para o maior sistema observado, a internet.

Nesse aspecto, surgem softwares que tornam possíveis onde é possível as buscas de informações em ambientes restritos. Dentre as soluções disponíveis, uma se chama Elasticsearch[[4]](#footnote-4).

De acordo com GORMLEY e TONG (2014)

Elasticsearch é um mecanismo distribuído de busca e analítico em tempo real. Ele permite que você explore seus dados a uma velocidade e em uma escala nunca antes possível. [...] mas Elasticsearch não é apenas para megacorporações. Ele permite que muitas startups [...] criem protótipos de ideias e transformá-las em soluções escaláveis. (GORMLEY, TONG, 2014, p. 1)

Observando empresas que utilizam o Elasticsearch como Wikipedia[[5]](#footnote-5) para buscas de conteúdo, The Guardian[[6]](#footnote-6) para análise de relatórios de visitas por notícia e no GitHub[[7]](#footnote-7) para busca de palavras em linhas de código fonte (GORMLEY e TONG, 2014), os mecanismos de organização e busca de informações faz presentes no nosso cotidiano.

O crescimento de mecanismos para a organização e busca das informações é um grande sinal de que mesmo com o aumento da informação, sempre é possível planejar um modelo de organização ou busca e executá-lo.

## JUSTIFICATIVA

O problema das informações desorganizadas é real e muito grave (SCHONS, 2007). Para realizar buscas sobre as informações, é necessário ter organização e análise. Sem a proposta de novas soluções, não há meios de diversificação para a resolução do problema.

A fim de buscar o aperfeiçoamento dos modelos e softwares existentes ou como forma de buscar outras alternativas, o Elasticsearch é uma ferramenta poderosa tanto para pequenos negócios como grandes negócios e deve ter seu funcionamento analisado para seu uso.

## OBJETIVOS

Demonstrar e implementar o funcionamento do mecanismo de busca Elasticsearch através de um aplicativo de receitas.

### Objetivos Específicos

* Estudo do mecanismo de busca do Elasticsearch;
* Modelagem dos dados que serão armazenados;
* Desenvolver um aplicativo de receitas para dispositivos móveis, implementando a comunicação com o Elasticsearch;

# FUNDAMENTAÇÃO TEÓRICA

## BREVE INTRODUÇÃO A BUSCA DE INFORMAÇÃO

CENDÓN (2001, p.1) diz: “Desde os primórdios da Internet, houve a preocupação de se criarem ferramentas para localização de seus recursos informacionais”. Observando essa necessidade, houve-se então a criação de mecanismos para realizar a busca nos documentos.

No início da internet surgiram mecanismos de buscas. Podemos citar o Archie, um buscador de arquivos em repositórios FTP muito utilizado no início da década de 90 (CENDÓN, 2001). Atualmente, o mais famoso é o mecanismo de busca do Google, que surgiu em 1998 na Universidade de Stanford (PEREIRA, 2004).

Observando fora da internet, encontram-se vários mecanismos de busca onde se pode realizar busca, seja ele um sistema de arquivos ou uma biblioteca de músicas.

A seguir, será apresentado um projeto onde iniciou-se um dos softwares mais utilizado no mercado computacional para mecanismo de busca em sistemas (GORMLEY e TONG, 2014).

## APACHE LUCENE

O Apache LUCENE é uma biblioteca de alto desempenho para a obtenção de informações (HATCHER e GOSPODNETIC, 2004), de acesso livre ao público podendo ser utilizado educacionalmente ou comercialmente.

Foi escrito e batizado por Doug Cutting e inicialmente disponibilizado no *website* SourceForge[[8]](#footnote-8) no ano 2000. Em 2001, foi incorporado pela família de *softwares* *open source* Apache Software Foundation’s[[9]](#footnote-9) Jakarta e em 2005 foi publicada como projeto da família Apache em seu *website*[[10]](#footnote-10) (HATCHER e GOSPODNETIC, 2004).

O funcionamento do mecanismo de busca é composto por algoritmos e modelos matemáticos usando o Vector Space Model (Modelo de Espaço Vetorial) (HATCHER e GOSPODNETIC, 2004). O Modelo de Espaço Vetorial é a representação de documentos e consultas como vetores de termos (CARDOSO, 2000).

O Apache LUCENE foi desenvolvido na linguagem JAVA, obtendo várias portabilidades para outras linguagens de programação como .NET, Python e Perl. Atualmente a biblioteca é utilizada por empresas como FedEx, Netflix, Linked In, Hewlett-Packard, Atlassian, Akamai's EdgeComputing. (HATCHER e GOSPODNETIC, 2004).

Após ser publicado como projeto da família Apache, outro projeto começou a surgir e usando como suporte o Apache LUCENE. Esse projeto será abordado no próximo tópico.

## ELASTICSEARCH

Com o mecanismo de busca do Apache LUCENE por base, o Elasticsearch incorporou várias outras características que o impulsionou adiante (GORMLEY e TONG, 2014).

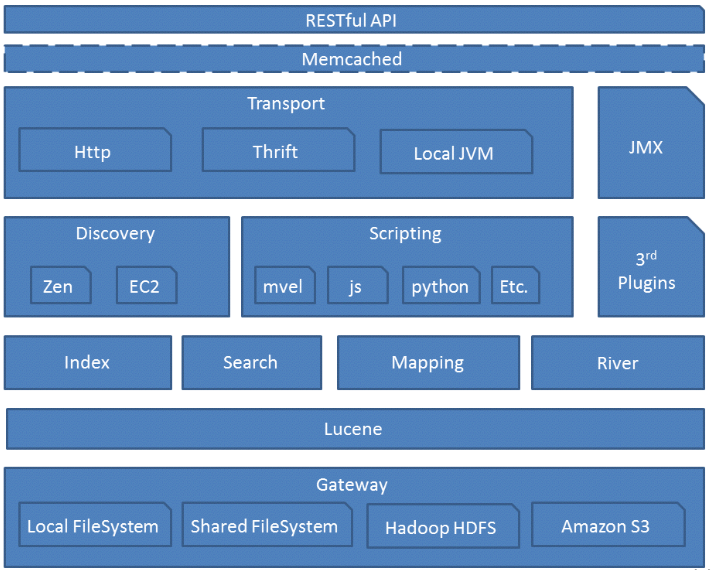
Em 2010, foi disponibilizado no GitHub e a partir disso obteve muitos colaboradores para a ferramenta aumentando as funcionalidades e sua popularidade.

A seguir, as principais características do Elasticsearch (GORMLEY e TONG, 2014):

* Armazenamento de dados em tempo real distribuídos, onde cada campo é indexado e pesquisável;
* Um motor de busca distribuído com análises em tempo real;
* Capaz de escalar para centenas de servidores e *petabytes* de dados estruturados e não estruturados;

Fonte: http://log.medcl.net/wp-content/uploads/2011/08/es-architecture.png

Figura 1 - Arquitetura ElasticSearch.



A Figura 1 contém a arquitetura do Elasticsearch. A explicação será na orientação da camada inferior para superior. A primeira camada chamada Gateway tem o propósito de armazenar os dados podendo ser dados localmente, em rede ou na internet (GORMLEY e TONG, 2014). Na segunda camada, o Apache LUCENE, já que o mecanismo é base para o Elasticsearch (GORMLEY e TONG, 2014). A terceira camada é composta pelos módulos mais importantes depois do mecanismo de busca. Cabe a cada módulo:

* Index: módulo de indexação dos documentos. O processo é composto pela análise e armazenamento;
* Search: módulo de busca dos documentos. Composto pela interface de busca;
* Mapping: módulo de mapeamento do documente. Análise do documento verificando o que significa semanticamente cada campo;
* River: módulo de leitura de dados em tempo real de outras bases de dados;

A quarta camada é composta por três módulos: Discovery, Scripting e 3rd Plugins. A camada é conhecida como “desenvolvedora”, pois os três módulos são direcionados a desenvolvedores que queiram desenvolver extensões para o Elasticsearch (GORMLEY e TONG, 2014).

A quinta camada é chamada de Transport. Nessa camada é realização do tráfego dos dados e por quais protocolos, sendo o mais importante, o protocolo HTTP (GORMLEY e TONG, 2014).

A sexta camada é chamada de API RESTful. Aqui é a interface de comunicação com o Elasticsearch como um todo (GORMLEY e TONG, 2014).

Embora a arquitetura do Elasticsearch seja composta de vários elementos, para o desenvolvimento deste trabalho, somente serão analisadas algumas camadas e módulos como o Gateway, usando o módulo de Local FileSystem para armazenamento local, os módulos de Index, Search e Mapping localizados na terceira camada. A última camada superior para a comunicação com o Elasticsearch e por fim o módulo mais importante, o LUCENE. O número de camadas foi restringido, pois elas estão diretamente envolvidas com o que será proposto por esse trabalho.

A seguir, é abordado o funcionamento do mecanismo de busca do Apache LUCENE incluso no Elasticsearch.

### Pesquisa Relevante

GORMLEY e TONG (2014) afirmam que o Elasticsearch usa o conceito chamado “Boolean Model” para encontrar os documentos em sua base e depois usa a equação de “Pratical Scoring Function” onde calcula a relevância.

O conceito de “Boolean Model” baseia-se na procura de termos dentro dos documentos que batem com a consulta realizada. Usando de lógica booleana para fazer as comparações e a procura de termos dentro dos documentos e trazendo uma lista com os resultados.

Após isso, realiza-se o cálculo da relevância usando a equação de “Pratical Scoring Function” usando de conceitos como “Term Frequency/Inverse document Frequency” “Field Length Normalization”, “Term Boosting” e “Vector Space Model” (GORMLEY e TONG, 2014).

A seguir, serão abordados os indicadores da equação de Pratical Scoring Function.

O conceito de “Term Frequency/Inverse document Frequency” é dividido em “Term Frequency” e “Inverse Document Frequency”.

Term Frequency (Frequência de Termos) é a frequência em que surge o termo no documento, ou seja, quando mais vezes o termo surgir no documento, mais importância o termo tem naquele documento (GORMLEY e TONG, 2014).

Na eq. 1 de Term Frequency (), destaca-se a variável onde representa a quantidade de aparições do termo no documento. O resultado tem como a quantidade de vezes *(f)* por documento *(d)*:

|  |  |
| --- | --- |
|  | (1) |

Próximo indicador é o Inverse Document Frequency (Frequência Inversa do Documento). Inverse Document Frequency é a frequência do termo em todos os documentos na lista de resultados. É necessário para poder separar a relevância entre termos comuns e termos incomuns. Quanto maior a frequência, menor é o indicador (GORMLEY e TONG, 2014).

Na eq. 2 de Inverse Document Frequency (), a variável representa a quantidade de documentos indexados e a variável representa a quantidade de documentos indexados que contém o termo *(t)*:

|  |  |
| --- | --- |
|  | (2) |

Field Length Normalization é o indicador de tamanho do campo onde realizou-se a pesquisa. Quanto menor o campo, maior é o indicador. Um exemplo é a busca do termo no campo título que normalmente é menor do que o campo de descrição, significando que no campo título, o termo encontrado tem mais relevância do que na descrição (GORMLEY e TONG, 2014).

Na eq. 3 de Field Length Normalization (), a variável representa a quantidade de termos no campo a ser indexado:

|  |  |
| --- | --- |
|  | (3) |

Todos esses indicadores são calculados no momento da indexação dos documentos e armazenados na base do *ElasticSearch*.

Na eq. 4 de Pratical Scoring Function (), representa a junção dos três indicadores multiplicando-os (GORMLEY e TONG, 2014).

|  |  |
| --- | --- |
|  | (4) |

Ao final, armazena-se os indicadores para uso futuro em consultas no Elasticsearch.

Nesse caso, o *Elasticsearch* trabalha com o modelo chamado “Vector Space Model” que proporciona uma maneira de comparação de múltiplos termos em uma consulta e documento. O resultado da comparação é um único valor que mostra a relevância do documento em relação aos termos na consulta (GORMLEY e TONG, 2014).

Antes da realização da comparação dos vetores, ele transforma a consulta e o documento em vetores para serem comparados. O vetor da consulta é constituído de coordenadas onde cada uma é o cálculo da relevância do termo em comparação a todos os termos. O vetor do documento é também constituído de coordenadas com o cálculo da relevância do termo, mas com relação ao documento em questão (GORMLEY e TONG, 2014).

Ao construir os vetores, é realizada a comparação através da manipulação da fórmula do produto escalar para conseguir o ângulo entre os vetores. A relevância do documento em relação a consulta é o ângulo entre os vetores (GORMLEY e TONG, 2014).

O produto escalar entre vetores é um produto interno definido no espaço euclidiano e o resultado é um número real chamado escalar (DA CRUZ, 2012). Como o Elasticsearch está à procura do ângulo entre os vetores, há a manipulação da fórmula para se conseguir o ângulo.

Na eq. 5 é a equação original do produto escalar vetorial entre dois vetores e , na qual e são os módulos dos vetores e a variável é o ângulo entre os vetores:

|  |  |
| --- | --- |
|  | (5) |

A eq. 6 representa o resultado da manipulação da eq. 5 para obter o ângulo entre os vetores:

|  |  |
| --- | --- |
|  | (6) |

Tendo a relevância do documento e a consulta, o *Elasticsearch* consegue ordenar do maior para o menor, trazendo a lista de resultados ordenados. A seguir, será expliar-se-á o processo de Mapping do Elasticsearch.

### Mappings

Existe um processo no Elasticsearch chamado Mapping que interpreta como realiza-se a indexação em quais campos e como ele irá realizar a indexação.

Por padrão, se não for especificado alguma estrutura de Mapping para o Elasticsearch, ele irá analisar a estrutura a ser indexada na primeira inserção no banco de dados de busca. Irá verificar cada campo e qual é o seu tipo e indexará por todos os campos usando um analisador chamado “Standard Analyzer” (GORMLEY e TONG, 2014).

Standard Analyzer é um dos principais analisadores de conteúdo encontrado no Elasticsearch. Funciona da seguinte forma: ele analisa o conteúdo em forma de texto quebrando em partes o conteúdo, analisando palavra por palavra e indexando no banco de dados usando as fórmulas acima de “Term Frequency/Inverse document Frequency” e “Field Length Normalization” (GORMLEY e TONG, 2014).

Podemos visualizar na Figura 2 como deve ser uma estrutura de Mapping para ser adicionada caso não queira seguir o comportamento padrão do Elasticsearch. Temos indicações como os números 2 e 8 indicam o nome do documento indexado. O número 3 indica se todos os campos serão de busca por nome do documento. Os números 4 e 9 indicam os campos inclusos no documento indexado. Os números 5, 6, 7, 10, 11, 12 e 13 indicam o tipo de representação do campo.



Fonte: https://www.elastic.co/guide/en/elasticsearch/reference/2.1/mapping.html

Figura - Exemplo de estrutura de Mapping

### Query DSL

Para a realização das buscas no Elasticsearch, existe uma interface chamada Query DSL. Essa interface é uma API REST onde podemos passar uma estrutura de query e ele faz o processo de busca e nos retorna uma lista com os resultados.

A Query DSL é baseada no formato JSON e possui dois tipos diferentes de cláusulas: Leaf Query e Compound Query (GORMLEY e TONG, 2014).

Leaf Query é mais utilizada quando queremos realizar uma consulta em campos específicos e com um valor específico. Já o Compound Query é utilizado quando se quer realizar várias consultas.

Todas as cláusulas se comportam diferentemente baseadas em dois tipos de contexto: Query Context e Filter Context.

Query Context realiza a seguinte pergunta: “Até que ponto essa cláusula corresponde ao documento indexado?”. Ela calcula uma pontuação de quanto que à consulta corresponde com o documento indexado (GORMLEY e TONG, 2014).

Já o Filter Context realiza a seguinte pergunta: “Será que esse documento corresponde a cláusula?”. Nesse tipo de contexto, não é calculado nenhuma pontuação e a resposta a ser esperada é bem simples, um Sim ou Não. Esse tipo de contexto é bastante utilizado para realizar filtros em Query Context (GORMLEY e TONG, 2014).

Na Figura 3 visualiza-se um exemplo de uma Leaf Query. Temos indicadores como o número 1, o qual representa o início de uma Leaf Query. O número 2 representa o tipo da Leaf Query. Os números 3 e 4 representam os campos a serem pesquisados pelo Search Context. O número 5 representa a seção Filter Context. Os números 6 e 7 representam os campos a serem filtrados.



Fonte: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-filter-context.html

Figura - Exemplo de estrutura de Leaf Query

## REST

De acordo com FIELDING

[…] é uma abstração da arquitetura de elementos dentro de um sistema hipermídia distribuídos. REST ignora os detalhes do componente implementação e sintaxe do protocolo, a fim de concentrar-se nos papéis de componentes, as restrições sobre a sua interação com outros componentes, e sua interpretação de elementos de dados significativos. Ela abrange as restrições fundamentais sobre os componentes, conectores, e os dados que definem a base da arquitetura da Web, e, assim, a essência de seu comportamento como um aplicativo baseado na rede. FIELDING (2002, p. 104)

Foi escrito inicialmente para o protocolo HTTP, entretanto pode ser utilizado para outros protocolos que contenham um vocabulário uniforme e rico para as transferências de estado, mas há certas restrições que devem ser seguidas para maximizar a abstração e manter o nível alto de desacoplamento.

As condições são (FIELDING, 2002):

1. De uma interface uniforme ajuda a simplificar a interação e desacopla a arquitetura, podendo ter a independência na comunicação cliente e servidor;
2. Ter a arquitetura cliente/servidor, pois são intercambiáveis e independentes, podendo ser trocados a qualquer momento sem prejuízo para o funcionamento da aplicação;
3. Ser Stateless, significando que o estado é gerenciado pelo cliente, sendo de total responsabilidade do cliente enviar o estado atual da aplicação a cada requisição. Logo se o servidor não mantém estado então permite que o cliente faça cache de recursos;
4. Uma aplicação que usa a arquitetura REST deve permitir a separação por camadas sem que o cliente possua dependência do endereço de origem do servidor, pois isso impede a introdução de camadas entre o servidor e o cliente, para aumentar a escalabilidade, segurança, performance e cache;
5. A possibilidade de permitir realizar cache é fundamental numa arquitetura REST, pois elimina *roundtrips[[11]](#footnote-11)* entre cliente e servidor. As respostas nessa arquitetura devem definir se o cliente pode fazer cache ou não de seus recursos, aumentando assim a escalabilidade e a performance;

Com todas essas condições, existe uma chave fundamental que se chama Recursos. Cada recurso é identificado pela interface uniforme e nisso o servidor e cliente negocia o formato que vai recebê-lo podendo manipular seus valores e consumi-los. Tudo isso feito com a transferência de estado do sistema.

## MONGODB

Atualmente as aplicações estão recebendo um volume muito grande de dados e uma quantidade de acessos maior do que antigamente pelo fato de que a internet tornou universal o meio de acessarmos as aplicações (VIEIRA, 2012).

Com esses fatores em jogo, os sistemas de gerenciamento de banco de dados relacionais ficaram sobrecarregados, pois teriam que retornar os dados rapidamente sendo que muitos acessos estavam ocorrendo simultaneamente. Isso acontece, pois os bancos relacionais se baseiam no conceito.

O conceito ACID se divide em 4 princípios ACID (ABREU, 2013):

* Atomicidade – Na transação ou se faz tudo, ou nada, sem meio termo;
* Consistência - Tem por objetivo garantir que o banco de dados antes da transação esteja consistente e, que após a transação o banco permaneça consistente, sem problemas de integridade;
* Isolamento - Objetiva garantir que nenhuma transação seja interferida por outra até que ela seja completada;
* Durabilidade - Garante que a informação gravada no banco de dados dure de forma imutável até que alguma outra transação de atualização, ou exclusão afete-a;

Para garantir o ACID, acarreta em um custo alto de processamento, aumentando o tempo de resposta.

Nesse ambiente, foi criado um novo tipo de banco de dados, o noSQL, para o trabalho com um grande volume de dados.

Os bancos de dados noSQL, trabalham com um modelo relacional não normalizado (MRNN). A maior diferença é que a modelagem não é normalizada, por conta disso, muitas das vezes as tabelas ficam aglutinadas uma com as outras, não necessitando de funções de junções (VIEIRA, 2012).

Entre os sistemas de gerenciamento de bancos não relacionais, surgiram vários tipos de sistemas e um deles é o MongoDB, um sistema de gerenciamento banco de dados orientado a documentos. A explicação de ser orientado a documentos, porque cada linha da tabela é um documento JSON. A estrutura de um JSON é um objeto composto por pares e valores, podendo esses valores serem compostos de listas e objetos.

## NODE.JS E EXPRESS.JS

No início da década de 2000, muitos servidores de internet como PHP, JAVA, ASP.NET estavam no auge e muitas pessoas o utilizavam. No final da década de 2000, surgiram os bancos de dados não relacionais para poderem trabalhar com um volume muito grande de dados e requisições e esses servidores não estavam adaptados para esse tipo de abordagem, pois utilizavam de conexões bloqueantes de entrada/saída (CANTELON e HARTER e HOLOWAYCHUK, 2013).

Esse tipo de conexão é do seguinte tipo: uma conexão é realizada no servidor e se a operação for de escrita e leitura na memória, ele para todo o processamento do servidor para realizar a operação. Isso dificultava muito para esses tipos de banco de dados que precisavam que fossem rápidos, mas os servidores não suportavam.

A partir de dessa dificuldade, Ryan Dahl criou em 2009, um servidor de internet onde o tipo de conexão de entrada/saída não era bloqueante. O servidor foi criado em cima de uma plataforma chamada V8 utilizado no navegador Google Chrome para interpretar a linguagem Javascript (CANTELON e HARTER e HOLOWAYCHUK, 2013).

Ao surgir o servidor Node.js, apareceram muitas bibliotecas para trabalhar com Node.js seja como servidor de API REST, servidor de web ou como servidor de Websocket[[12]](#footnote-12). O Express.js surgiu para facilitar a criação de aplicações web e aplicações API REST, tendo muitas estruturas já construídas para auxiliar no desenvolvimento (CANTELON e HARTER e HOLOWAYCHUK, 2013).

O desenvolvimento da API REST se torna muito fácil, pois ele tem métodos auxiliares que constrói um servidor HTTP e atribui escutadores em todos os métodos para realizar a lógica programada.

Para o desenvolvimento de aplicações web, além de construir um servidor HTTP para poder resgatar as páginas, utiliza um renderizador de HTML chamado JADE onde se pode construir documentos HTML dinâmicos ou estáticos.

## HTML, JAVASCRIPT E CORDOVA

A linguagem foi baseada em outra linguagem existente na época chamada SGML. O propósito do SGML era estruturar documentos a partir de marcações chamadas *tags*. *Tags* como <ul>, <h1> e <p> eram usadas para descrever que aquele pedaço de conteúdo era uma lista, um título ou parágrafo (PILGRIM, 2010).

O principal motivo da linguagem HTML ser criada em cima do SGML era o aproveitamento dessas *tags* para a estruturação do documento com a inclusão de uma *tag* que iria mudar todo o funcionamento da internet. A inclusão da *tag* <a> que em conjunto com o atributo href podia interligar documentos por toda internet o que mantém o funcionamento da internet até os dias atuais.

Mas sua concepção veio na década de 1980, com Tim Benners-Lee e seu projeto chamado ENQUIRE no qual foi construído na linguagem PASCAL (LONGMAN, 1998).

Em 1989, com a ajuda de um estudante chamado Robert Cailliau, conseguiu implantar a primeira comunicação entre cliente HTTP e servidor tendo assim criado a *World Wide Web* ou para os dias atuais, a internet.

A chave que Tim Benners-Lee encontrou era a utilização do protocolo de comunicação para o transporte dos documentos em linguagens *HTML* realizando a exibição em clientes (LONGMAN, 1998).

Após a criação da linguagem de marcação HTML, surgiu o problema de tornar o documento HTML mais dinâmico com a interação do usuário. Foi necessária a criação de uma nova linguagem que solucionasse esse tipo de problema e então surgiu o Javascript.

A linguagem Javascript foi criada por Brendan Eich em 1995 com o nome de Mocha, mas foi lançado com o nome LiveScript como característica do navegador Netscape 2.0 Beta. Com a inclusão da tecnologia da linguagem JAVA chamada Applet, foi renomeado para Javascript (LONGMAN, 1998).

A linguagem foi criada para poder manipular os documentos dinamicamente no cliente dando maior interatividade para o usuário. Atualmente é uma das linguagens mais utilizadas pelo seu poder garantido por navegadores como Google Chrome e Mozilla Firefox e também com o advento do AJAX, tecnologia que permite realizar requisições assíncronas na página.

Como essas linguagens são utilizadas na plataforma WEB, era possível a exibição do documento HTML em dispositivo móvel somente através do navegador do dispositivo. O surgimento de uma nova plataforma chamada Cordova tornou possível a criação de aplicações a partir da linguagem HTML, CSS e Javascript (WARGO, 2015).

Ele proporciona uma interface para cada sistema operacional móvel que consequentemente disponibiliza métodos em comum para a linguagem Javascript poder utilizá-lo.

A plataforma é composta pelos seguintes componentes: *Command-Line* e *Interfaces*;

*Command-Line* é um programa que recebe comandos e executa de acordo com o que foi pedido. Dentre os comandos estão a criação de projetos, a inclusão de plataformas como Android e iOS e a compilação do projeto transformando em um executável que irá rodar nos sistemas operacionais.

*Interfaces* são bibliotecas programadas para realizar a comunicação entre o Javascript e a linguagem nativa de cada sistema operacional possibilitando-o chamar funcionalidades que só poderiam ser chamadas utilizando a linguagem nativa como tirar uma foto com a câmera do dispositivo móvel ou resgatar a lista de contatos salva no aparelho.

# METODOLOGIA

## MODELAGEM DO DADO

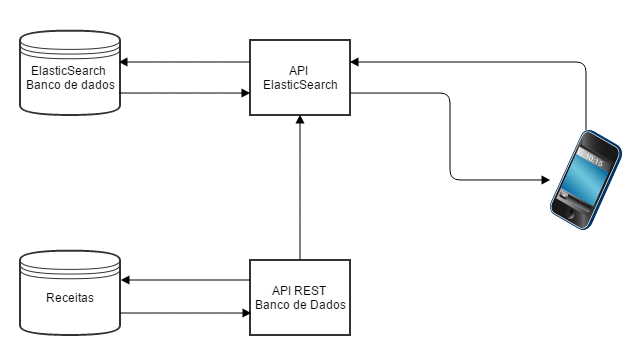
Inicialmente, foi realizado a concepção do modelo de dados a ser armazenado no banco de dados, modelo este que seria composto por informações que caracterizasse uma receita. Após a análise da maioria das receitas em sites do gênero, foi proposto um modelo satisfatório para ser utilizado em testes e que abrangesse as principais informações.

O modelo de dados foi planejado de maneira não relacional, já que optou-se por um banco de dados não relacional para o armazenamento dos dados. Essa restrição ocorre porque o Elasticsearch utiliza banco de dados não relacional para o armazenamento (GORMLEY e TONG, 2014). O resultado da modelagem do dado que irá ser armazenado pode ser visualizado no Anexo A.

## ARQUITETURA DO SISTEMA

Após a modelagem do dado, iniciou-se a etapa de construção da arquitetura do sistema entre o aplicativo, banco de dados e o Elasticsearch. A princípio, foi proposto uma comunicação via API REST, já que o Elasticsearch utiliza desse método para comunicação. Portanto, o único módulo do sistema que está faltando, é a comunicação com o banco de dados.

Pensando nisso, foi elaborado a estrutura na Figura 4. Ela é composta por um fluxo de dados entre os módulos do sistema. O módulo chave é a API REST do Elasticsearch, onde se conecta todos os módulos. O módulo de API REST Banco de Dados será necessário para indexar os documentos no Elasticsearch e propiciar o ambiente para teste. E o módulo final que é a aplicação móvel que terá como contexto a busca de receitas e que consultará o módulo do Elasticsearch para resgatar a lista conforme os critérios a serem informados. Os critérios são pelo nome da receita e por ingredientes.



Fonte: Elaborado pelo autor

Figura - Arquitetura do Sistema

## MÓDULO API REST BANCO DE DADOS

Após o planejamento da arquitetura, observou-se que seria melhor iniciar pela API REST que irá se comunicar com o banco de dados para realizar as seguintes operações: incluir, resgatar, alterar e excluir.

Optou-se por usar o Node.js para ser o servidor API REST para a comunicação com o banco de dados porque tem a facilidade de trabalhar com bancos de dados não relacionais e utilizamos a biblioteca Express.js para o desenvolvimento da API REST que realize as operações citadas. As chamadas criadas podem ser visualizadas no Anexo B.

Finalizado o desenvolvimento da API REST, iniciou-se a construção de uma interface web onde se poderia adicionar as receitas no banco de dados. Observou-se que seria mais trabalhoso a construção dessa interface, optamos pelo modo manual, onde ocorreu a inserção de 50 receitas no banco de dados.

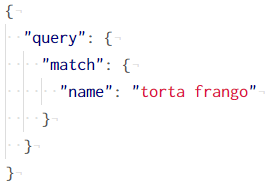
## ELASTICSEARCH

Realizado o módulo anterior, foi realizado o download da ferramenta localmente para iniciar os testes com a API REST de busca. Foi necessário a desenvolvimento de um método na API REST que inicializava a indexação no Elasticsearch. Foi adotado por um período, a abordagem padrão do Elasticsearch para Mapping, mas conforme o andamento, observou-se que era necessário ter algumas alterações para que a busca fosse efetiva. As mudanças podem ser visualizadas no Anexo C.

Na etapa de realizar consultas no ElasticSearch através da comunicação via API REST, foi utilizado Compound Query para realizar mais de uma Leaf Query por causa dos vários ingredientes existentes na pesquisa e Leaf Query para consultas por nome da receita.

A Figura 5 mostra a Leaf Query utilizada para busca por nome da receita. Foi planejado no Query Context para que utilize o campo nome do dado armazenado para busca.

Figura - Leaf Query para pesquisar por nome de receita



Fonte: Elaborado pelo autor

A Figura 6 mostra a Compound Query utilizada para busca por ingredientes. A Compound Query foi planejada para ter uma quantidade variável de Leaf Query com o Query Context para que realizasse a busca pelo nome do ingrediente utilizando no Filter Context, filtros como quantidade e tipo de unidade.

Figura - Compound Query para realizar a busca por ingredientes



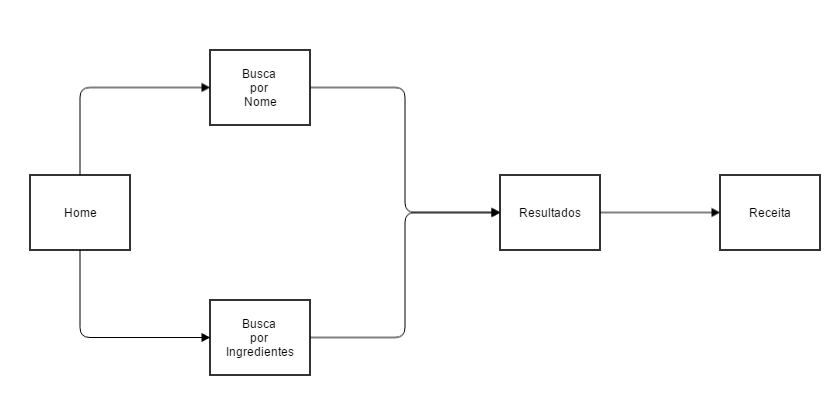
Fonte: Elaborado pelo autor

## MÓDULO APLICAÇÃO MÓVEL

Após a elaboração das cláusulas, começou-se a desenvolver o aplicativo em HTML, CSS e Javascript com a ajuda da biblioteca Framework7 que disponibiliza vários componentes para poder construir as telas o mais rápido possível e totalmente funcional, seja ele no dispositivo móvel ou navegador.

Na Figura 7 é proposto dois fluxos de telas. O primeiro fluxo era o caminho para a busca pelo nome da receita, caminho simples onde se digita parte do nome da receita e o Elasticsearch retorna uma lista com os resultados. O outro fluxo era a inclusão de ingredientes e sua quantidade e o Elasticsearch retornaria também uma lista com as receitas que atendiam os ingredientes.

Figura - Fluxo de telas da Aplicação



Fonte: Elaborado pelo autor

O primeiro bloco é a Home que será a entrada da aplicação. O bloco Busca por Nome é a tela onde se digita o nome da receita e a listagem da busca. Ao tocar na tela na região da listagem irá ser direcionado para a tela de Receita onde poderá ser observado o detalhe da receita.

O bloco Busca por ingredientes é a tela onde pode ser adicionado os ingredientes a partir de uma janela na posição inferior do dispositivo e será composto pelas informações de quantidade, tipo da unidade e o nome do ingrediente.

Após o planejamento do fluxo da aplicação, foi realizado a construção das telas da aplicação para cada bloco.

A Figura 8 foi planejada para ser composta com um campo de inclusão de texto a ser procurado e a listagem logo a seguir. A Figura 9 foi planejada para representar as informações do nome da receita, ingredientes, modo de preparo, tempo e a quantidade de porções. A Figura 10 foi planejada para ser composta por uma listagem dos ingredientes a serem procurados. Dois botões, um para adicionar o ingrediente e outro para iniciar a procura. O formulário na posição inferior tem o propósito de inserir as informações do ingrediente.

|  |  |
| --- | --- |
| Figura - Tela de Busca por nome    Fonte: Elaborado pelo autor | Figura - Tela da receita    Fonte: Elaborado pelo autor |
| Fonte: Elaborado pelo autor  Figura - Tela de busca por ingredientes | |

## EXPERIMENTOS

A Tabela 1 consta os dispositivos usados para teste.

Tabela - Lista de dispositivos e navegadores

|  |  |
| --- | --- |
| **Dispositivo** | **Sistema Operacional** |
| *Browser* | *Google Chrome* 47.0.2526.106m (*Windows*) |
| *Browser* | *Google Chrome* 46.0.2490 (*Linux*) |
| *Moto X Play* | *Android* 5.1.1 |

Nos experimentos foram realizados dois tipos de testes. O primeiro visava a busca de receitas pelo próprio nome. Foi utilizado no campo nome da receita, os seguintes conteúdos:

* “Torta frango”;
* “Bolo”;

O segundo caminho visava a busca de receitas pelos ingredientes e sua quantidade. Foi inserido na listagem os seguintes ingredientes:

* Primeiro teste
  + 5 xícaras de farinha;
  + 3 xícaras de leite;
* Segundo teste
  + 3 unidades de ovos;
  + 8 xícaras de farinha;
  + 4 xícaras de açúcar;
* Terceiro teste
  + 2 unidades de bacalhau;
  + 3 unidades de cebola;

# RESULTADOS

## BUSCA POR NOME DA RECEITA

O primeiro teste foi realizado e o resgate da lista foi composta com os cinco primeiros itens:

* Torta de Frango;
* Fricassé de Frango;
* Estrogonofe de Frango;
* Salpicão de Frango;
* Torta salgada de liquidificador;

O segundo teste foi realizado e o resgate da lista foi composta com os cinco primeiros itens:

* Bolo simples;
* Bolo de milho cremoso;
* Bolo de fubá;
* Bolo de chocolate molhadinho;
* Bolo de banana rápido de liquidificador;

## BUSCA POR INGREDIENTES

O primeiro teste foi realizado e o resgate da lista foi composta pelos itens:

* Bolo de banana rápido de liquidificador;
* Torta salgada de liquidificador;
* Rabanada tradicional;
* Batata gratinada fácil;
* Bolo simples;
* Massa de panqueca simples;
* A melhor receita de bolo de chocolate;
* Bolo de chocolate molhadinho;
* Nega maluca;
* Torta de frango;
* Torta de morango fantástica;
* Torta de liquidificador;

O segundo teste foi realizado e o resgate da lista foi composta pelos itens:

* Bolinha de chuva;
* Torta salgada de liquidificador;
* Bolo de banana rápido de liquidificador;
* Sonho de creme;
* Pudim de leite condensado;
* Nega maluca;
* Bolo simples;
* Torta de frango;
* Bolo de fubá;
* A melhor receita de bolo de chocolate;
* Bolo de chocolate molhadinho;
* Farofa fácil de natal;
* Bacalhoada;
* Costelinha com molho *barbecue* (Outback);

O terceiro teste foi realizado e o resgate da lista foi composta pelos itens:

* Torta salgada de liquidificador;
* Salpicão de frango;
* Camarão na moranga;
* Bacalhau de forno;
* Torta de liquidificador;
* Estrogonofe de frango;
* Torta de frango;
* Pernil assado com batata;
* Chester assado;
* Peru de natal;
* Tabule;
* Chester com farofa de frutas;
* Molho branco para macarrão;
* Farofa fácil de natal;
* Frango assado inteiro e descomplicado;

# CONCLUSÃO

Na primeira seção de experimentos feitos pela busca do nome da receita, os dois testes realizados, trouxeram itens relacionados com os termos informados. Nesse quesito, o Elasticsearch se mostra confiável em realizar buscas por termos e expressões.

Na segunda seção de experimentos feitos pela busca por ingredientes, obteve alguns resultados conforme o previsto. Entretanto, outros resultados não obtiveram muita satisfação como o terceiro teste. Nesse teste era esperado qualquer receita de bacalhau no topo da listagem, entretanto, apareceu na quarta colocação da listagem, não atendendo ao objetivo.

Foi realizado o terceiro teste com a quantidade menor de ingredientes e o resgate foi a mesma listagem, salvo receitas que não se enquadravam na restrição de quantidade de ingredientes.

Um dos fatores relevantes na busca por ingredientes é a quantidade do ingrediente. Por adicionar esse fator, a listagem é mais criteriosa com os documentos armazenados e, por esse motivo, não retorna uma receita com bacalhau no topo da listagem.

Uma das soluções possíveis é a inclusão de um indicador no ingrediente representando o que é principal. Adicionando na cláusula de busca de ingrediente esse fator, para que possam ser retornadas receitas com os ingredientes principais no topo da listagem.

A modelagem atendeu as necessidades para a inclusão das receitas no banco de dados e também para ser indexado no Elasticsearch. O único detalhe seria a inclusão do campo para informar que o ingrediente é importante na receita. Com a inclusão esse campo, a busca por ingredientes poderia ter sido mais satisfatória.

Uma das necessidades desse trabalho era a contingência de informações no banco de dados. A inserção dos dados no banco de dados foi realizada de forma manual resultando em 50 receitas cadastradas. O ideal seria realizar esse estudo com maior número de receitas, mas não foi possível por pouco tempo hábil. Uma das alternativas a ser estuda é a inserção de receitas automaticamente no banco de dados. Pode-se usar um programa que navegaria na internet em websites de receitas e resgataria o conteúdo do site, formatando-o para a inserção no banco de dados.

Em relação a aplicação, executou-se satisfatoriamente comunicando-se com a API REST do Elasticsearch conforme a arquitetura. Foi possível a amostra dos resultados em todos os dispositivos testados. Todos os testes foram realizados no dispositivo móvel.

REFERÊNCIAS

ABREU, F. Papo SQL. **Os conceitos da propriedade ACID**, 2013. Disponível em: <http://paposql.blogspot.com.br/2013/05/os-conceitos-da-propriedade-acid.html>. Acesso em: 23 Maio 2015.

CANTELON, M.; HARTER, M.; HOLOWAYCHUK, T. J. **Node.js in Action**. Manning Publications, 2013. ISBN 9781617290572.

CENDÓN, B. V. **Ferramentas de busca na Web**. Ci Inf., Brasília, v. 30, n. 1, p. 39-59. Jan./Abr. 2001.

DA CRUZ, L. F. **Cálculo vetorial e geometria analítica**. 2012. Disponível em: < http://wwwp.fc.unesp.br/~lfcruz/GA\_CAP\_04.pdf>. Acesso em: 08 Janeiro 2016.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. University of California. Irvine, p. 180. 2002.

FUGITA, M. S. L. **A identificação de conceitos no processo de análise de assunto para indexação**. [Editorial]. *Revista Digital de Biblioteconomia e Ciência da Informação*, Campinas. v.1,  n.1,  p. 60-90, jul./dez., 2003.

GORMLEY, C.; TONG, Z. **Elasticsearch: The Definitive Guide**. 1ª. [S.l.]: O'Reilly Media, 2014. ISBN 9781449358549.

HATCHER, E.; GOSPODNETIC, O. **Lucene in Action. 2ª**. Shelter Island: Manning Publications, 2004. ISBN 1933988177.

KING, A. **The Evolution of RSS**, 2001. Disponível em: < http://www.webreference.com/authoring/languages/xml/rss/1/index.html>. Acesso em: 23 Janeiro 2016.

LONGMAN, A. W. **A history of HTML**, 1998. Disponível em: < http://www.w3.org/People/Raggett/book4/ch02.html>. Acesso em: 17 Dezembro 2015.

PEREIRA, V. N. S. S. **Arquitetura de um motor de busca: exemplo do Google**. Universidade de Coimbra. Departamento de Engenharia Informática. 2004.

PILGRIM, M. **HTML5: Up and Running - Dive into the Future of Web Development**. O'Reilly Media, 2010. ISBN 9780596806026.

RECUERO, R. **Considerações sobre a Difusão de Informações em Redes Sociais na Internet**. In: CONGRESSO BRASILEIRO DE CIÊNCIAS DA COMUNICAÇÃO DA REGIÃO SUL, 8., 2007, Passo Fundo.

SCHONS, C. H. **O volume de informações na internet e sua desorganização: reflexões e perspectivas**. Inf. Inf., Londrina, v. 12, n. 1, 2007.

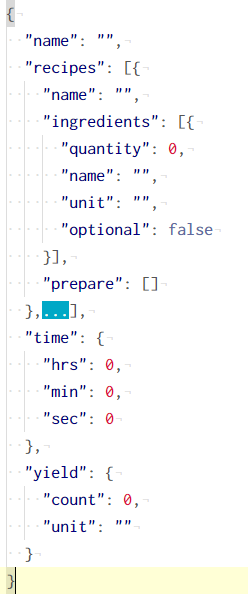
VIEIRA, M. R. et al. **Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data**. Universidade Federal de Mato Grosso. [S.l.]. 2012.

WARGO, J. M. **Apache Cordova 4 Programming (Mobile Programming)**. 1ª. Addison-Wesley Professional, 2015. ISBN 9780134048192.

ANEXOS

ANEXO A

Estrutura da receita armazenada no banco de dados.



ANEXO B

*API REST* que faz a comunicação com o banco de dados não relacional.

**GetAllRecipes**:

**URL**: https://tcc-recipes-admin-server.herokuapp.com/recipes/

**Method**: GET

**Retorno**: Array de Receitas

**GetRecipe**:

**URL**: https://tcc-recipes-admin-server.herokuapp.com/recipes/<id>

**Method**: GET

**Retorno**: Receita ou Status Code 404

**CreateRecipe**:

**URL**: https://tcc-recipes-admin-server.herokuapp.com/recipes

**Method**: POST

**Body Object**: Receita

**Retorno**: Id da Receita

**EditRecipe**:

**URL**: https://tcc-recipes-admin-server.herokuapp.com/recipes/<id>

**Method**: PUT

**Body Object**: Receita

**Retorno**: Status Code 204

**DeleteRecipe**:

**URL**: https://tcc-recipes-admin-server.herokuapp.com/recipes/<id>

**Method**: DELETE

**Syncronize**:

**URL**: https://tcc-recipes-admin-server.herokuapp.com/recipes/sync

ronize

**Method**: POST

ANEXO C

Estrutura do *Mapping* para o nosso trabalho



1. *Website* do tipo buscador, onde tem o objetivo de trazer os resultados relevantes a pesquisa feita. Ver em: http://www.google.com.br [↑](#footnote-ref-1)
2. Sistema Operacional da Microsoft. Ver em: https://www.microsoft.com/pt-br/windows [↑](#footnote-ref-2)
3. Sistema de Recurso e Planejamento Empresarial. Ver em: http://go.sap.com/brazil/index.html [↑](#footnote-ref-3)
4. Ver em: https://www.elastic.co/products/elasticsearch [↑](#footnote-ref-4)
5. Enciclopédia Livre. Ver em: https://pt.wikipedia.org [↑](#footnote-ref-5)
6. Jornal Britânico. Ver em: http://www.theguardian.com/ [↑](#footnote-ref-6)
7. Repositório de código-fonte de softwares. Ver em: https://github.com [↑](#footnote-ref-7)
8. Repositório online de código-fonte. Ver em: http://www.sourceforge.net [↑](#footnote-ref-8)
9. Organização sem fins lucrativos criada para suportar os projetos de código aberto. Ver em: http://www.apache.org [↑](#footnote-ref-9)
10. Ver em: http://lucene.apache.org. [↑](#footnote-ref-10)
11. Um ciclo completo de comunicação entre Cliente e Servidor. [↑](#footnote-ref-11)
12. Comunicação bidirecional entre *sockets* ocorrendo através do protocolo TCP.

    Ver em: http://www.html5rocks.com/pt/tutorials/websockets/basics/ [↑](#footnote-ref-12)