# QUADRO METODOLÓGICO

No quadro metodológico serão apresentadas as informações e procedimentos definidos para realização e conclusão do projeto. Serão apresentados o tipo de pesquisa, contexto, instrumentos, procedimentos, orçamento e cronograma do projeto.

## Tipo de Pesquisa

A pesquisa tem como objetivo a obtenção de novos conhecimentos, tomando por base a utilização de procedimentos científicos. Contribui para a solução dos problemas e processos eventuais nas mais diversas atividades humanas, em ações comunitárias, no processo de formação e outros. Dessa forma, o conhecimento se torna uma ferramenta para o desenvolvimento do ser humano e a pesquisa uma consolidação da ciência (SILVA, 2008).

Para Oliveira (2002, p. 62), “A pesquisa, tanto para efeito científico como profissional, envolve a abertura de horizontes e a apresentação de diretrizes fundamentais, que podem contribuir para o desenvolvimento do conhecimento”.

Abrangendo o contexto de pesquisa, este projeto tomou por base a metodologia de pesquisa aplicada, que é utilizada quando o resultado final é um produto real, o qual pode ser aplicado em um determinado contexto.

De acordo com Marconi; Lakatos (2009, p. 6), “Pesquisa aplicada caracteriza-se por seu interesse prático, isto é, que os resultados sejam aplicados ou utilizados, imediatamente, na solução de problemas que ocorrem na realidade”.

Após a análise e aplicação os conceitos de pesquisa, foi desenvolvida uma aplicação web que tem como objetivo a mineração, análise e demonstração dos posts do Twitter, além do objetivo de demonstrar como foi feita integração das tecnologias utilizadas para análise de *Big Data*.

## Contexto de pesquisa

Atualmente, devido à facilidade de acesso à internet, dia após dia são geradas quantidades imensas de dados. Esses dados contém uma variedade imensa de informações. Eles podem ser gerados por bancos, operadoras de telefonia, redes varejistas ou até mesmo uma pessoa comum. Mas de nada serve uma quantidade inimaginável de dados se ela não for analisada. Para suprir essa necessidade, foi criado o conceito de *Big Data*.

Não é difícil imaginar o cenário em que este conceito se aplica. Por exemplo, podem ser citados os milhares de e-mails trocados por dia, bem como o número de transações bancárias e de posts de uma rede social.

A esta pesquisa demonstra como soluções baseadas em *Big Data* tem o poder de influenciar e auxiliar determinadas áreas de conhecimento do ser humano, através da análise, gerenciamento e demonstração dos dados.

As soluções de *Big Data* possibilitam aos analistas um melhor entendimento de determinado produto, serviço ou situação. Desse modo, torna-se capaz de ser feita uma melhoria ou reestruturação para evitar desperdício de recursos, tal como melhorar uma produção em quantidade, qualidade e tempo. Esses fatores podem ser decisivos para o futuro de uma empresa.

Desse modo, o objetivo deste projeto é demonstrar como uma solução baseada em *Big Data* se torna eficiente e decisória no âmbito político, apresentando as opiniões públicas dos usuários de redes sociais de forma rápida e clara, além da geração de gráficos para ser feita a análise de satisfação pública por região. Esta pesquisa, por envolver diversas tecnologias relacionadas à *Big Data*, servirá de base para a formação do conhecimento de estudantes da área de tecnologia, bem como o aprimoramento do conhecimento de profissionais da área.

## Instrumentos

Para ser feita a escolha do tipo de instrumento, se faz necessária a observação do que será estudado.

Na realização de uma pesquisa, segundo Oliveira (2002, p. 66):

Depois de definidas as fontes de dados e o tipo de pesquisa, que pode ser de campo ou de laboratório, devemos levantar as técnicas a serem utilizadas para a coleta de dados, destacando-se: questionários, entrevistas, observação, formulários e discussão em grupo.

A coleta de dados foi realizada por meio de livros e materiais relacionados com o tema, trabalhos e artigos acadêmicos, páginas na internet, pesquisas bibliográficas e reuniões com pessoas com conhecimentos relacionados com o tema.

Também foram feitas reuniões semanais com o orientador do projeto, com o objetivo de mostrar pesquisas e resultados, definir o foco do projeto até a reunião seguinte, tomar as devidas decisões relacionadas ao desenvolvimento da aplicação e ser feita a escolha da forma da apresentação das tecnologias abordadas no projeto.

## Procedimentos

Neste projeto, foi desenvolvida uma aplicação web que demonstra como soluções baseadas no conceito de *Big Data* podem impactar nas diferentes áreas do conhecimento, mais especificamente na área política. Foram utilizadas tecnologias de mineração, armazenamento e demonstração de grandes quantidades de dados.

Para organizar e facilitar no desenvolvimento do projeto, foi previamente definido um cronograma estimando a quantidade de tempo que cada uma das atividades durariam até serem finalizadas. As atividades progrediam conforme era feito o estudo, implementação e análise dos resultados de cada tecnologia empregada no projeto.

### Configuração do ambiente

Para o desenvolvimento deste projeto, foi necessário configurar o ambiente de desenvolvimento. O sistema operacional usado para o desenvolvimento deste projeto foi o Ubuntu 14.04 LTS, um sistema operacional de código aberto baseado no Linux. O sistema operacional já se encontrava previamente instalado na máquina de desenvolvimento.

O primeira ação tomada foi a instalação do Node.JS e de seu gerenciador de pacotes, o NPM, obtidos e instalados através da linha de comando “sudo apt-get install nodejs npm”. Após feita a instalação, automaticamente foram configurados o compilador e gerenciador de pacotes NPM, habilitando, assim, seus comandos via terminal, conforme demonstrado na Figura 1.

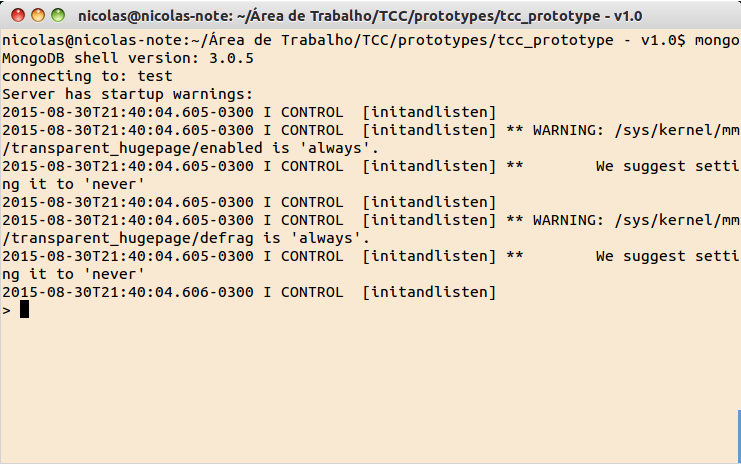
****

**Figura 1** – Demonstração da versão do Node.JS e NPM usadas, respectivamente. **Fonte: Elaborado pelos autores**.

Para se fazer real o armazenamento da massa de dados, foi feito o *download* e instalação do banco de dados MongoDB, através das seguintes linhas de comando:

* **Importação da chave pública usada no sistema de gerenciamento de pacotes:** sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10;
* **Criação arquivo de listas para o MongoDB:** sudo touch /etc/apt/sources.list.d/mongodb-org-3.0.list && echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb\_release -sc)"/mongodb-org/3.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list;
* **Atualização do repositório do sistema operacional:** sudo apt-get update;
* **Instalação do MongoDB:** sudo apt-get install -y mongodb-org.

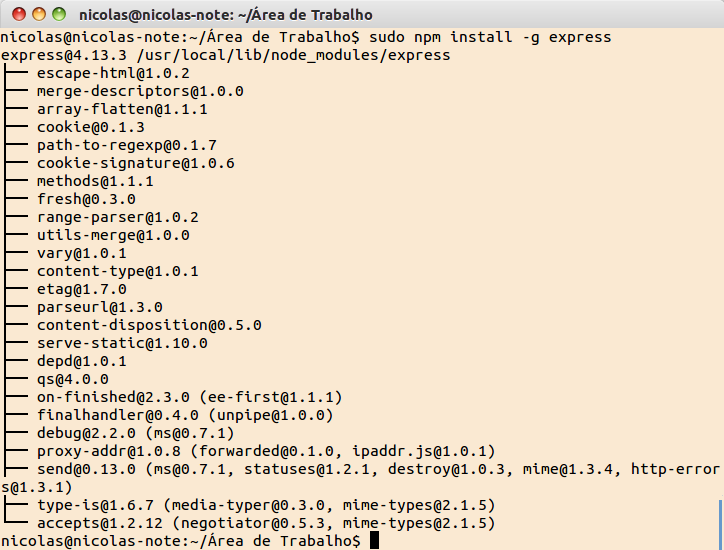
Automaticamente, seu executável foi adicionado na inicialização do sistema operacional, tornando-se assim disponível para a aplicação a qualquer momento. Logo após sua instalação, através do seu próprio terminal, foram criados respectivamente o banco de dados da aplicação e suas *collections*, para a posterior consulta, inserção e atualização dos dados. Sua inicialização, bem como seu terminal, são mostrados na Figura 2.

****

**Figura 2** – Terminal do MongoDB. **Fonte: Elaborado pelos autores**.

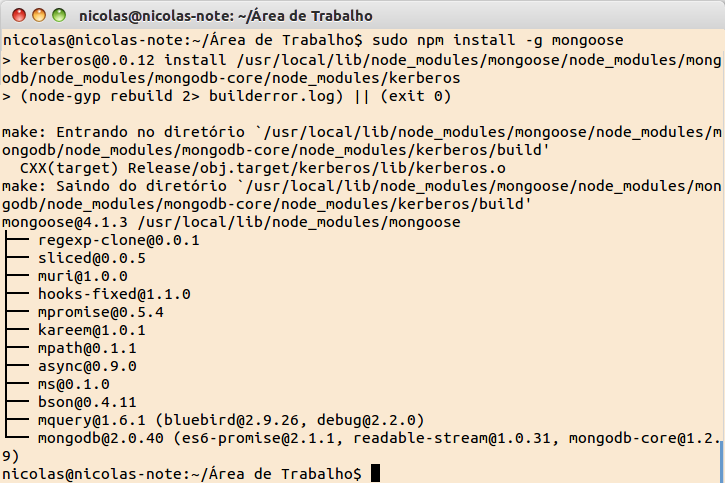
Para o desenvolvimento do *front-end* e *back-end* da aplicação, foi feito o *download* da IDE WebStorm, através de seu próprio site. Após feito o *download*, o arquivo foi descompactado em uma pasta. Logo em seguida, a pasta foi movida para o diretório “/Documents”, na intenção de facilitar sua localização. Posteriormente, foi executada a linha de comando “~/Documents/WebStorm-10.0.0/bin/WebStorm.sh”, que consistia na execução de um *script* Linux para a instalação e configuração da IDE.

Logo após a escolha e instalação da IDE de desenvolvimento, foi feita a instalação e configuração do *framework* Express através da linha de comando “npm install –g express” no terminal do Node.JS, instalando-o assim como um módulo global através do parâmetro “-g”, conforme mostrado na Figura 3.



**Figura 3** – Demonstração da instalação do*framework Express*. **Fonte: Elaborado pelos autores**.

Depois do *framework* *Express*, foi necessário instalar o *Mongoose*, módulo responsável pela comunicação com o MongoDB. Ele foi instalado atavés do comando “npm install –g mongoose”, também como módulo global, conforme demonstrado na Figura 4.



**Figura 4** – Demonstração da instalação do *mongoose*. **Fonte: Elaborado pelos autores**.

Depois de todos os pré-requisitos prontos, viu-se a necessidade de criar a arquitetura do projeto, visando melhor organização e respeito de padrões. Ao executar a linha de comando “express politistatus -ejs”, foi criada a árvore de diretórios do projeto, no padrão MVC. O parâmetro “-ejs” da linha de comando ativou o projeto para o uso do EJS *template engine.*

A estrutura de diretórios gerada pelo Express ficou da seguinte forma:

* ***Views*:** Diretório responsávelpelos arquivos de tela da aplicação;
* ***Public*:** Diretório responsável pelos arquivos estáticos da aplicação web, como *JavaScript’s*, *stylesheets*, imagens, dentre outros;
* ***Routes*:** Diretório responsável pelos arquivos de rota da aplicação;
* ***App.js*:** Arquivo JavaScript que contém a *stack* de configurações responsável pela inicialização da aplicação, através do comando “npm start”;
* ***package.json*:** Arquivo JSON contendo as informações da aplicação, tais como versão do projeto, controle de módulos, dependências entre outros. É através deste arquivo que se torna possível a atualização dos módulos a aplicação, através do comando “npm install”.

Para deixar o arquivo *package.json* com as informações corretas sobre o projeto, foram configurados os atributos *name*, *version*, *description*, *author*, informando respectivamente o nome, a versão, a descrição e os autores da aplicação. Também foi setado o atributo *private* como *false*, informando que a aplicação desenvolvida é *open-souce*. Dentre todas as configurações de atributos, a mais importante do arquivo *package.json* é a versão, pois sem ela é impossível instalar/atualizar os módulos com o comando “npm”. A estrutura do arquivo é demonstrada na Figura 5.

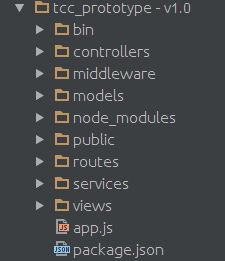


**Figura 5** – Demonstração do arquivo package.json. **Fonte: Elaborado pelos autores**.

Após todas estas etapas, o ambiente básico para o desenvolvimento da aplicação estava criado. Porém, devido à necessidade da pesquisa, foram adicionados ao projeto os seguintes diretórios:

* ***Controller*:** Diretório responsável pelos arquivos controladores das *view’s*;
* ***Model*:** Diretório responsável pelos arquivos de *persistence* dos modelos do banco de dados;
* ***Service*:** Diretório responsável pelos arquivos de conexão com o banco de dados, possibilitando assim as operações de *create*, *retrieve*, *update* e *delete*.

A estrutura de diretórios é demonstrada na Figura 6.

****

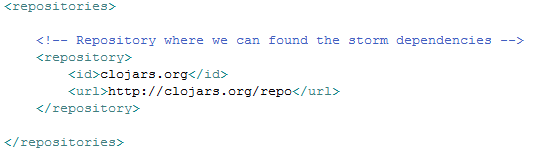
**Figura 6** – Demonstração da árvore de diretórios da aplicação. **Fonte: Elaborado pelos autores**.

Na árvore de diretórios, encontram-se também as pastas *bin* e *node\_modules*, as quais contém respectivamente o arquivo de inicialização do servidor web e os módulos Node.JS necessários para a execução do sistema.

Ao final das configurações da aplicação web, se viu necessário o *download* e configuração da IDE de desenvolvimento *Eclipse Luna*. Essa IDE é específica para desenvolvimento de projetos na linguagem Java. Ela foi empregada no desenvolvimento do *framework* de mineração de dados *Apache Storm*.

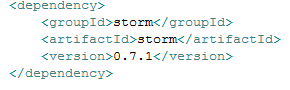
Após feito o *download*, extração e a movimentação da pasta do Eclipse para o diretório “/Documents”, foi necessária a instalação do plug-in de integração do Maven “m2e – *Maven Integration for Eclipse*”. Foi necessária também a instalação do Maven2 através da linha de comando “sudo apt-get install maven2” no terminal do sistema. Desse modo, a instalação e configuração foram automáticas.

Para o funcionamento correto do *Maven*, foi primeiramente necessário configurar o repositório do clojars.org, por este não ser um repositório padrão do *Maven*. No arquivo “pom.xml”, foram adicionadas as linhas de código demonstradas na Figura 7:



**Figura 7** - Adição da dependência CloarJS.org no arquivo “pom.xml”. **Fonte: Elaborado pelos autores**.

A integração do *Apache Storm* também foi feita através do *Maven*. No arquivo “pom.xml” foram adicionadas as linhas de código demonstradas na Figura 8:



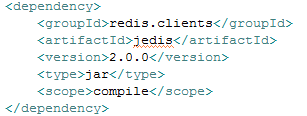
**Figura 8** - Adição da dependência do *Apache Storm* no arquivo “pom.xml”. **Fonte: Elaborado pelos autores**.

Também através do *Maven*, foi feita a integração do *Commons-*HTTP. No arquivo “pom.xml” foram adicionadas as linhas de código demonstradas na Figura 9:



**Figura 9** - Adição da dependência *Commons-*HTTP no arquivo “pom.xml”. **Fonte: Elaborado pelos autores**.

Do mesmo modo, foi feita integração do *Redis.clients*. No arquivo “pom.xml” foram adicionadas as linhas de código demonstradas na Figura 10:



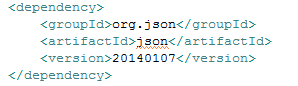
**Figura 10** - Adição da dependência *redis.clients* no arquivo “pom.xml”. **Fonte: Elaborado pelos autores**.

A integração da biblioteca Twiter4j foi feita do mesmo modo. Também no arquivo “pom.xml”, adicionou-se as linhas de código demonstradas na Figura 11:



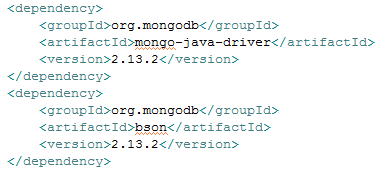
**Figura 11** - Adição da dependência da biblioteca *Twitter4j* no arquivo “pom.xml”. **Fonte: Elaborado pelos autores**.

Do mesmo modo foi feita a integração do Org.JSON, demonstrada na Figura 12:



**Figura 12** - Adição da dependência Org.JSON no arquivo “pom.xml”. **Fonte: Elaborado pelos autores**.

A integração do MongoDB e seu driver de acesso também foi feita também através do *Maven*. No arquivo “pom.xml” foram adicionadas as linhas de código demonstradas na Figura 13:



**Figura 13** - Adição das dependênncias banco de dados MongoDB no arquivo “pom.xml”. **Fonte: Elaborado pelos autores**.

Após serem feitas as configurações das dependências, executou-se o comando “maven install” no Eclipse para ser feito o *download* das bibliotecas necessárias para o desenvolvimento do *framework* de mineração de dados do *Twitter*.

Ao final de todas essas configurações, a estrutura-base e o ambiente de desenvolvimento estavam prontos para se iniciar a fase de protótipos do projeto.

### Desenvolvimento dos protótipos

O desenvolvimento dos protótipos da aplicação deu-se através dos graduais estudos sobre as tecnologias envolvidas. O desenvolvimento dos protótipos serviu de base prática para a geração de conhecimentos relacionados ao projeto, bem como testes de funcionamento e integração das tecnologias.

Gradualmente, ao serem desenvolvidos os protótipos da aplicação web, foi se consolidando que a mesma teria a necessidade de ser *single-page*, onde todas as informações principais do sistema estariam disponíveis em sua página inicial. Do mesmo modo, foi definido que o sistema deveria ser multiusuário, devido a aplicação estar em um ambiente livre, a internet.

Depois de feitas essas definições, começou-se a desenvolver versões finais dos protótipos das telas, usando as tecnologias HTML5, CSS3, *JavaScript* e o *framework Bootstrap*. As telas desenvolvidas foram:

* **Tela de login:** Desenvolvida para sanar a necessidade multiusuário da aplicação;
* ***Partial View* de cadastro de usuário:** Desenvolvida, juntamente com a tela de login, para ser feito o cadastro dos usuários do sistema;
* **Página incial da aplicação:** É a página principal da aplicação, onde estão todas as suas informações importantes;
* ***Partial View* de cadastro de palavra-chave:** Desenvolvida para ser feito o cadastro dos políticos, os quais serão buscados como palavras-chave no nos comentários do Twitter;
* ***Partial View* de edição de palavra-chave:** Desenvolvida para ser feita a edição das palavras-chave (nome dos políticos);
* ***Partial View* de exclusão de palavra-chave:** Desenvolvida para ser feita a exclusão das palavras-chave (nome dos políticos);
* ***Partial View* de *Stream* e filtragem dos *tweets* minerados:** Desenvolvida em conjunto com a página inicial, essa modal tem como finalidade listar os *tweets* retornados pelo *Apache Storm*, tanto em tempo de ocorrência dos mesmos, quanto filtrados por data*.* Esse último lista os *tweets* mais antigos de acordo com a data inicial e final informadas pelo usuário.
* **Página “Sobre”:** Página onde existem as informações relacionadas ao projeto, como autores, contato com os desenvolvedores e o *link* para ser feito o acesso ao controlador de versão do projeto.

Na página inicial da aplicação foi inserida uma *grid*, que corresponde à listagem de todas as palavras-chave cadastradas pelo usuário. Também foi inserida na página inicial algumas informações do usuário logado, como nome e sobrenome previamente cadastrados.

Também inserido na página inicial da aplicação, está inserido um mapa constantemente atualizado, que informa através de pontos ao usuário da aplicação quais são os locais do Brasil onde estão sendo realizados os *tweets*.

Paralelamente ao desenvolvimento dos protótipos de *front-end*, foram desenvolvidos os protótipos de *back-end* Node.JS. À medida que o as telas precisavam das informações a serem mostradas, eram criadas rotinas no *server-side* da aplicação. Nas versões finais dos protótipos, cada requisição de feita pelo *font-end* respeitava a seguinte sequencia lógica:

* ***Routes:*** Cada requisição tinha sua própria rota, onde tinha registrado o seu método a ser chamado no *Controller*;
* ***Middleware:*** Antes de ser feita a chamada do *Controller*, foi feita a implementação de checagem de usuário logado. Caso o usuário estivesse logado, a aplicação redirecionava a requisição para o método no *Controller*;
* ***Controller:*** Ao chegar nesse ponto, era feita toda a lógica necessária para atender as necessidades do usuário da aplicação. Para atender essas necessidades, era feita uma chamada no *Service*, o qual retornava as informações necessárias para o usuário;
* ***Service:*** As classes implementadas no *Service* fazem a conexão com o banco de dados, persistindo os *Schemas* definidos nos *Modules* da aplicação.

Depois de feita toda essa sequência lógica, os dados do usuário retornavam para o *front-end* da aplicação, assim mostrando para o usuário o que ele havia previamente requisitado.

Em relação ao *Apache Storm*, a fase de prototipação consistiu inicialmente na implementação e adequação de necessidade das classes Java disponibilizadas pela biblioteca *Twitter4j*, onde foram realizados testes de conexão com o *Twitter Sream* API e visualização de resultados retornados pelo mesmo. Posteriormente, à medida que foi aprimorada a aprendizagem da tecnologia, foi feita a conexão do *framework* com o banco de dados MongoDB, para ser realizada a busca dos dados cadastrados pela aplicação web. Desse modo, os dados retornados são salvos no banco e consumidos pela aplicação web.

O desenvolvimento dos protótipos do sistema foi uma etapa essencial para a elaboração do projeto. Através dela, foram adquiridos os conhecimentos necessários para desenvolver a versão final da aplicação, além de ser feita a comprovação que interação entre as tecnologias do projeto tornam possível a realização do conceito de *Big Data*.