**NÍCOLAS HENRIQUE VIEIRA TOLEDO**

**SEBASTIÃO BATISTA DE ANDRADE NETO**

**ESTATÍSTICAS DE REDES SOCIAIS**

**COM BIG DATA**

**UNIVERSIDADE DO VALE DO SAPUCAI**

**POUSO ALEGRE**

**2015**

# QUADRO TEÓRICO

No quadro teórico são descritos o histórico e os conceitos que se relacionam com o tema, bem como as tecnologias que são aplicadas no desenvolvimento do trabalho.

## *Big Data*

A história de como os dados se tornaram tão vastos, gerando o que hoje conhecemos como *Big Data*, vai muito além do atual estado estabelecido. Há nove anos atrás já houve tentativas de quantificar a taxa de crescimento no volume de dados ou o que ficou conhecido como a “explosão de informação” (PRESS, 2013).

Aqui será descrito um resumo de como tudo aconteceu desde 1944, quando Freemont Rider, em sua obra “The Scholar and the Future of the Research Library”[[1]](#footnote-1), estimava que “As bibliotecas universitárias americanas dobravam o tamanho a cada dezesseis anos” (RIDER, 1944, apud PRESS, 2013). Ele especulava que em 2040, a Biblioteca de Yale teria cerca de 200 milhões de volumes e precisaria de cerca de seis mil pessoas trabalhando no serviço de catálogo (PRESS, 2013).

Hal B. Becker, em sua publicação “Can users really absorb data at today’s rates? Tomorrow’s?”[[2]](#footnote-2), de 1986, estimava que a densidade de caracteres alcançada por Guttenberg era de aproximadamente de 500 caracteres por polegada cubica. Por volta do ano 2000, as memórias RAM armazenariam 1.25X10¹¹ bytes por polegada cubica (PRESS, 2013).

Em 1997 Michael Cox e David Ellsworth publicaram “Application-controlled demand paging for out-of-core visualization”[[3]](#footnote-3), no qual eles começaram o artigo afirmando que o tamanho extenso dos conjuntos de dados penalizam as capacidades da memória principal, disco local e até discos remotos, gerando assim um desafio interessante para os sistemas computacionais. A esse desafio, eles deram o nome de “problema do Big Data”. De acordo com eles, quando conjuntos de dados não cabem mais na memória principal, ou quando os dados não cabem nem nos discos locais, a solução mais comum é adquirir mais recursos (PRESS, 2013).

A publicação de 1999 “*Visually exploring gigabyte data sets in real time”*[[4]](#footnote-4), de Steve Bryson, David Kenwright, Michael Cox, David Ellsworth e Robert Haimes, dizia que computadores poderosos são uma dádiva em diversos campos de investigação, porém em contrapartida, suas operações computacionais rápidas geram quantidades massivas de dados (PRESS, 2013).

Eles também afirmavam que tempos atrás, alguns megabytes de conjuntos de dados eram considerados grandes. Entretanto, na época já eram encontrados conjuntos de dados na casa dos 300 gigabytes. Porém se torna um desafio entender os resultados dos dados de computações de alto desempenho. É simplesmente difícil olhar para todos os números. Além do mais, o objetivo da computação é o conhecimento, e não números (PRESS, 2013).

Na publicação “*How Much Information?*”[[5]](#footnote-5), de Peter Lyman e Hal R. Varian feita em 2000, foi apontado que em 1999 o mundo produziu cerca de 1.5 exabytes de informação original, ou cerca de 250 megabytes para cada homem, mulher e criança na terra. Uma vasta quantidade de informação é criada e armazenada por indivíduos (democratização dos dados), e isso cresce velozmente. Chamando essa descoberta de “A dominância do Digital”, Lyman e Varian diziam que maioria da informação textual já “nascia digital” e em poucos anos, a sentença também se tornaria verdadeira para imagens (PRESS, 2013).

Também em 2000, Francis X. Diebold apresentou um documento intitulado “*Big Data Dynamic Factor Models for Macroeconomic Measurement and Forecasting*”[[6]](#footnote-6), onde ele afirma que “Recentemente, muita ciência boa, seja física, biológica ou social, foi forçada a confrontar, além de obter benefícios, do fenômeno designado *Big Data*” (DIEBOLD, 2000, apud PRESS, 2013).

*Big Data* refere-se como a explosão na quantidade (e as vezes na qualidade) da informação relevante disponível, em grande parte, o resultado de avanços recentes e inéditas em gravação de dados e tecnologia de armazenamento. (PRESS, 2013).

Em 2001, Doug Laney, um analista no Meta Group, publicou uma nota de pesquisa intitulada “*3D Data Management: Controlling Data Volume, Velocity, and Variety.*”[[7]](#footnote-7) Uma década depois, os “3v's” se tornaram as três dimensões que definem o *Big Data*, mesmo que nas notas de Laney o termo em si não aparece (PRESS, 2013).

Em 2007 John F Gantz, David Reinsel e outros pesquisadores lançam um livro, intitulado “*The Expanding Digital Universe: A Forecast of Worldwide Information Growth through 2010*”[[8]](#footnote-8). Nele, era afirmado que em 2006, o mundo produziu 161 exabytes de dados, além de prever que entre 2006 e 2010, a informação adicionada anualmente ao universo digital aumentaria para 988 exabytes, ou dobrando a cada 18 meses. De acordo com os lançamentos de 2010 e 2012 do mesmo estudo, a quantidade de dados digitais ultrapassou esta previsão, atingindo 1.227 exabytes em 2010, e crescendo a 2.837 exabytes em 2012 (PRESS, 2013).

Em 2008 Randal E. Bryant, Randy H. Katz, e Edward D. Lazowska publicaram “*Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society*”[[9]](#footnote-9).

Para Bryant; Katz e Lazowska (2008, p.1 e seg. apud Press 2013):

Assim como os motores de busca têm transformado a forma de acessar as informações, outras formas de *Big Data* podem e vão transformar as atividades das empresas, pesquisadores científicos, médicos e operações de defesa e inteligência da nossa nação[...] *Big Data* é, talvez, a maior inovação na computação na última década. Nós apenas começamos a ver o seu potencial para coletar, organizar e processar dados em todas as esferas da vida. Um modesto investimento por parte do governo federal poderia acelerar consideravelmente o seu desenvolvimento e implementação.

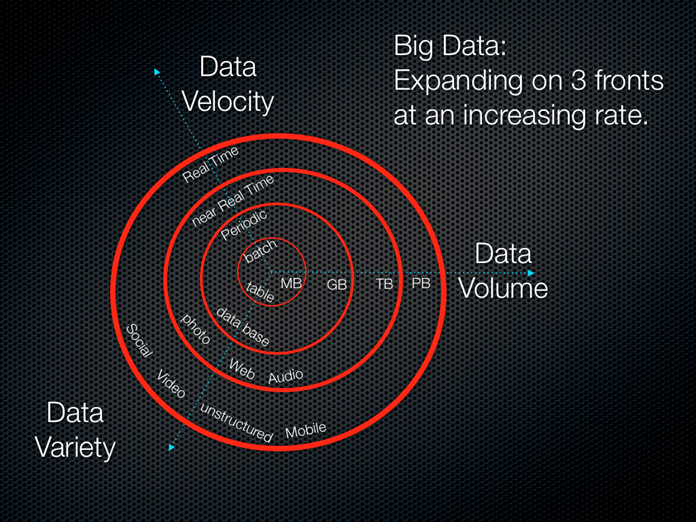
Em 2010 Kenneth Cukier informa em seu relatório “*Data, data everywhere*”[[10]](#footnote-10) que o mundo contém uma inimaginável quantidade de informação digital que está crescendo cada vez mais mais rapidamente, e que o efeito pode ser visto em todos os lugares. Cientistas e engenheiros de computação designaram termo para o fenômeno: “*Big Data*” (PRESS, 2013).

Na publicação de 2011 “*Big data: The next frontier for innovation, competition, and productivity*”[[11]](#footnote-11) feita por James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, e Angela Byers, foi estimado que até 2009, quase todos os setores da economia americana tinham, pelo menos, uma média de 200 terabytes de dados armazenados por empresa (o dobro do tamanho do *Data Warehouse*[[12]](#footnote-12) da varejista americana *Wal-Mart* em 1999). No total, o estudo estimou que 7,4 exabytes de novos dados foram armazenados pelas empresas e 6,8 exabytes por consumidores em 2010 (PRESS, 2013).

Na publicação de 2012 “*Critical Questions for Big Data in* Information, Communications, and Society**”**[[13]](#footnote-13), Danah Boyd e Kate Crawford definem *Big Data* como um fenômeno cultural, tecnológico e acadêmico que é formado da interação entre tecnologia (capacidade computacional e precisão algorítmica para coletar, analisar, unir e comparar dados), análise (identificação padrões, a fim de fazer reivindicações econômicas, sociais, técnicas e legais) e mitologia (crença de que grandes conjuntos de dados oferecem uma forma mais elevada de inteligência e conhecimento que podem gerar perspectivas que antes eram impossíveis, com verdade, objetividade e precisão) (PRESS, 2013).

De acordo com Soubra (2012), existem três propriedades que são capazes de definir a expansão do Big Data: Volume, Variedade e Velocidade. A junção destes é chamada de “os 3 v’s do Big Data”. Cada um deles pode ser definido da seguinte forma:

* Volume: Quantidades de dados realmente grandes, que tem crescimento exponencial;
* Variedade: O grande volume de dados se deve ao fato da existência de uma grande variedade de informações. Os dados podem ser classificados em estruturados, semi-estruturados e não estruturados.
* Velocidade: O tratamento dos dados deve ser feito em tempo hábil. O volume de informações, com uma variedade de dados diversa, deve ser processado e analisado no menor tempo possível.



**Figura 1** - Os 3 v’s que definem o *Big Data*. **Fonte:** Data Science Central (2012)

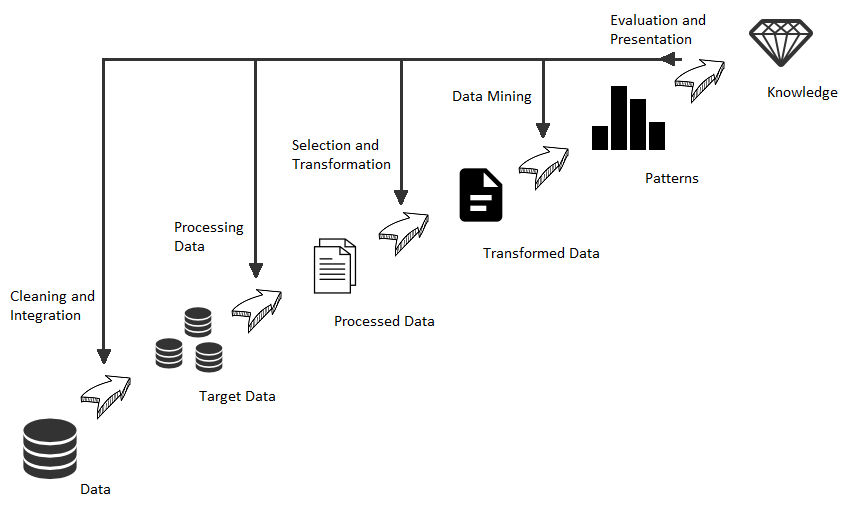
A cada dia que se passa o volume de dados cresce de forma exponencial. Desse modo, se vê necessária a existência de uma abordagem que tenha a capacidade de armazenar de maneira segura uma quantidade de dados crescente em tamanho e variadade. Além disso, existe a necessidade de processar e apresentar essas informações em tempo hábil para a tomada de decisões. Para o atendimento de todas essas necessidades, foi criado o conceito de *Big Data* (ALECRIM, 2015).

## *Data Mining*

Para Han, Kamber e Pei (2012), *Data Mining* (ou mineração de dados) pode ser definida de diversas formas. O termo “mineração” é constantemente usado para caracterizar o processo em que se encontra um conjunto de “pepitas” de uma grande quantidade de matéria-prima. Assim, nasceu o termo *Data Mining*, que em suma, representa a extração de conhecimento a partir da análise de dados.

De acordo com Han, Kamber e Pei (2012), o processo de descoberta do conhecimento segue a sequência iterativa dos seguintes passos:

* Limpeza dos dados: Remoção de lixo e dados inconsistentes;
* Integração dos dados: Combinação de múltiplas fontes de dados;
* Seleção dos dados: Os dados relevantes para a tarefa são obtidos do banco de dados;
* Transformação dos dados: Os dados são transformados e consolidados em formas apropriadas para a mineração por operações de resumo ou operações de agregação;
* Mineração de dados: Processo onde são aplicados métodos inteligentes para a extração de padrões de dados;
* Avaliação dos padrões: Identificação dos padrões verdadeiramente interessantes que representam conhecimento baseados em medidas de interesse;
* Apresentação do conhecimento: Técnicas de visualização e representação do conhecimento são usadas para apresentar o conhecimento extraído para os usuários.



**Figura 2** - *Data Mining* como um passo no processo da descoberta do conhecimento. **Fonte:** Elaborado pelos autores.

A mineração de dados é o processo de descobrir padrões e conhecimento de grandes quantidades de dados. As fontes de dados podem incluir bancos de dados, *data warehouses*, a Web e outros repositórios de informações ou dados que serão transmitidos para o sistema de forma dinâmica (HAN, KAMBER e PEI, 2012).

### *Data Mining* em Redes Sociais

De acordo com Russel (2013), algumas das redes sociais eram consideradas inicialmente como uma simples “moda”. Posteriormente, passaram a ser consideradas como um “*mainstream*”[[14]](#footnote-14). Atualmente, chegaram ao reconhecimento de uma espécie de “grife”, chegando a mudar a maneira de vida da população dentro e fora da Web.

Para Russel (2013), de um modo geral, as redes sociais, juntamente com *data mining*,análises, e técnicas de visualização para explorar dados buscam responder as seguintes perguntas:

* “Quem conhece quem, e quem são os contatos comuns em suas redes sociais?”;
* “Com que frequência pessoas específicas se comunicam com outras?”;
* “Qual rede social gera mais valor para um nicho específico?”;
* “Como a geografia afeta suas conexões sociais em um mundo online?”
* “Quem são as pessoas mais influentes/procuradas em uma rede social?”;
* “Sobre o que as pessoas estão conversando? (É valioso?)”;
* “Quem são as pessoas interessadas, com base na linguagem humana, e o que eles utilizam no mundo digital?”.

A resposta a essas perguntas muitas vezes deram oportunidades valiosas e lucrativas a empresários, cientistas sociais e outros profissionais, promovendo assim, o entendimento de problemas e a descoberta de soluções (RUSSEL, 2013).

## REST API

Para Massé (2011), *Representational State Transfer* (REST) é o nome de uma arquitetura Web que pode ser implementada usando diferentes tecnologias, além de instanciá-la utilizando diferentes valores para suas propriedades abstratas. O estilo arquitetural REST é comumente aplicado ao desenvolvimento de API’s[[15]](#footnote-15) para *Web Services*[[16]](#footnote-16). Uma API em conformidade com a arquitetura REST é uma REST API.

Ainda de acordo com Massé (2011), algumas das melhores práticas para o design REST API estão implícitas no padrão HTTP, enquanto outras abordagens “pseudo-padrão” surgiram ao longo dos anos. No entanto, nos dias atuais recomenda-se continuar a buscar as respostas para uma série de questões, tais como:

* “Quando um *path* URI deve ser nomeado com substantivo no plural?”;
* “Qual método de requisição deve ser usado para atualizar o estado de recurso?”;
* “Como fazer para mapear operações não-CRUD aos URI’s?”;
* “Qual é o código de status de resposta HTTP mais apropriado para um determinado cenário?”;
* “Como gerenciar as versões de representação do estado de um recurso?”;
* “Como estruturar um *hyperlink* em JSON?”.

O conhecimento dos princípios do REST trazem benefícios no desenvolvimento de aplicações Web, bem como na transformação da API em um *Web Server*[[17]](#footnote-17) robusto (MASSÉ, 2011).

### *Twitter Stream* API

Para Twitter Developers (2015), as *streamig* API’s do *Twitter* proporcionam aos desenvolvedores um acesso de baixa latência ao fluxo global de *tweets[[18]](#footnote-18)*. A implementação correta da API retorna os *tweets* e seus respectivos eventos, sem ocorrer qualquer sobrecarga nas extremidades da REST.

Continuando com Twitter Developers (2015), o *Twitter* oferece alguns tipos de serviço de *stream*, os quais podem ser citados:

* *Public Streams*: *Streams* de dados públicos que fluem através do *Twitter*. É mais adequado para seguir usuários ou tópicos específicos, bem como mineração de dados;
* *User Streams*: *Streams* de um único usuário. Retorna todos os dados de um usuário específico;
* *Site Streams*: *Streams* multiusuário. Os dados são destinados a servidores, os quais se conectam ao *Twitter* através de muitos usuários.

As API’s de *stream* do *Twitter* facilitam o acesso às informações dos seus usuários. Com a implementação correta, torna-se possível fazer a mineração dos dados necessários para desenvolver uma aplicação integrada com o *Twitter* (TWITTER DEVELOPERS, 2015).

### *Twitter4j*

Para Twitter4j (2015), o *Twitter4j* é uma biblioteca Java não-oficial para o consumo do *Web Service* do Twitter. A principal característica da *Twitter4j* é a facilidade de integração entre a aplicação Java e o *Twitter Service*.

De acordo com Twitter4j (2015), são características da biblioteca *Twitter4j*:

* Funciona em qualquer plataforma Java na versão 5 ou posterior;
* Plataformas *Android* e *Google* *App Engine* incluídas;
* Dependência zero - Não requer jar’s[[19]](#footnote-19) adicionais;
* Suporte a OAuth[[20]](#footnote-20) integrado;
* Suporte a gzip[[21]](#footnote-21);
* Compatível com a versão 1.1 do *Twitter* API.

A biblioteca *Twitter4J* possui uma série de funcionalidades simples de serem utilizadas, resultando em uma implementação rápida e sem maiores dificuldades (TWITTER4J, 2015).

## *Storm*

De acordo com Leibiusky, Eisbruch e Simonassi (2012), o *Storm* é um sistema distribuído, confiável e tolerante a falhas desenvolvido para processar fluxos de dados. Sua estrutura consiste em diferentes tipos de componentes. Desta forma o trabalho é dividido entre os componentes, cada qual é responsável por uma tarefa de processamento simples e específica. O fluxo de entrada do *cluster*[[22]](#footnote-22) do *Storm* é tratado por um componente chamado *spout*.

Segundo Apache (2015), um *spout* pode ser denominado como a fonte de fluxos em uma topologia. Geralmente, eles irão ler tuplas (sequências de objetos) de uma fonte externa (uma API de rede social, por exemplo), posterirormente encaminhando-os para a topologia.

Continuando com Leibiusky, Eisbruch e Simonassi (2012), o *spout* por sua vez direciona os dados para um componente denominado *bolt*, o qual irá transformá-los de alguma maneira. Um *bolt* pode tanto persistir os dados em algum tipo de armazenamento, quanto direcioná-lo para outro *bolt*. Para melhor entendimento, pode-se imaginar o *cluster* do *Storm* como uma corrente de *bolts*, cada qual executa algum tipo de transformação nos dados expostos pelo *spout*.

Leibiusky, Eisbruch e Simonassi (2012) afirmam que, dentre propriedades do *Storm*, as principais a serem citadas são:

* Simples de programar: A complexidade é reduzida;
* Suporte a várias linguagens de programação: Maior facilidade desenvolver em linguagens baseadas na JVM[[23]](#footnote-23), no entanto qualquer linguagem é suportada desde que se implemente uma pequena biblioteca intermediária;
* Tolerante a falhas: O *cluster* do *Storm* cuida dos trabalhos, realocando as tarefas quando necessário;
* Escalável: O *Storm* pode reatribuir as tarefas para as novas máquinas, caso seja necessário;
* Confiável: Todas as mensagens são obrigatoriamente processadas ao menos uma vez. Em caso de erros, a mensagem pode ser processada novamente. Assim, tem-se a garantia de nunca perder uma mensagem;
* Rápido: Velocidade é um dos fatores-chave do *Storm*;
* Transitivo: Pode-se obter exatamente a mesma semântica de mensagens para praticamente qualquer tipo de computação.

O *Storm* torna-se uma solução eficiente, rápida, confiável e de baixo custo para operações relacionadas à *Big Data*. Grandes massas de dados tem que ser processadas no menor tempo possível, com a garantia que nenhum dado se perca. A ferramenta possibilita este tipo de solução, através de seu design e suas funcionalidades (LEIBIUSKY, EISBRUCH e SIMONASSI, 2012).

### *Maven*

De acordo com Leibiusky, Eisbruch e Simonassi (2012) o *Maven* é uma ferramenta de gerenciamento de projetos de software e compreensão. Ele pode ser usado para gerenciar vários aspectos de um ciclo de desenvolvimento do projeto, a partir de dependências para o processo de compilação de lançamento.

Os objetivos principais do *Maven* de acordo com a *Apache Software Foundation* é permitir que um desenvolvedor para compreender o estado completo de um esforço de desenvolvimento no menor período de tempo. Para atingir este objetivo há várias áreas de preocupação que *Maven* tenta lidar com:

* Tornar o processo de compilação fácil.
* Fornecer um sistema de construção uniforme.
* Fornecer informações sobre o projeto de qualidade.
* Fornecendo orientações para melhor desenvolvimento de práticas.
* Permitindo a migração transparente para novas funcionalidades.

Embora de acordo com Leibiusky, Eisbruch e Simonassi (2012) não é necessário ser um especialista em *Maven* para usar o *Storm*, é útil para saber o básico de como funciona o *Maven*.

## Banco de dados NoSQL

Para Sadalage e Fowler (2013), o banco de dados do tipo NoSQL tem como característica o armazenamento de suas tabelas como arquivos do tipo ASCII[[24]](#footnote-24). Cada uma de suas tuplas é representada por uma linha, que tem os seus campos separados por guias.

Continuando com Sadalage e Fowler (2012), o nome “NoSQL” vem do fato de não ser usado SQL[[25]](#footnote-25) como uma linguagem de consulta aos seus registros. Ao invés disso, o banco de dados NoSQL é manipulado através de *shell scripts*[[26]](#footnote-26) que podem ser combinados nos *pipelines* do UNIX[[27]](#footnote-27).

Ainda de acordo com Sadalage e Fowler (2012), o uso do NoSQL que é reconhecido em dias atuais, remonta a um encontro que ocorreu em 11 de junho de 2009 em São Francisco, EUA, organizado pelo desenvolvedor de software Johan Oskarsson. Um exemplo que foi desenvolvido usando BigTable[[28]](#footnote-28) e Dynamo[[29]](#footnote-29) havia inspirado diversos projetos a experimentar um armazenamento de dados alternativo, e discussões a respeito do NoSQL haviam se tornado uma característica das melhores conferências de software em torno desse tempo.

De acordo DevMedia (2013), o movimento NoSQL promove soluções de armazenamento de dados não relacionais. Suas diversas ferramentas resolvem problemas relacionados a grandes volumes de dados, execução de consultas com baixa latência e modelos flexíveis de armazenamento de dados (como XML[[30]](#footnote-30) ou JSON[[31]](#footnote-31)).

O objetivo do NoSQL não é de substituir os bancos de dados relacionais, mas sim apresentar algumas soluções, que em determinados momentos, são mais adequadas e eficientes. Assim, nada se impede de se trabalhar com as tecnologias baseadas em NoSQL e os tradicionais bancos de dados relacionais em um mesmo projeto (DEVMEDIA, 2013).

Para DevMedia (2013), os tipos de bancos de dados NoSQL são:

* Chave-Valor: Armazenam objetos indexados por chaves, e possibilitam sua busca a partir de sua respectiva chave.
* Documentos: Conjunto de documentos no formato JSON. Os documentos são tratados como objetos únicos, os quais possuem campos com os respectivos valores.
* Coluna: Formado por colunas que contem um conjunto de informações, semelhante a uma tabela.
* Grafos: Formado por dados distribuídos em forma de vértices e arestas, os quais possuem atributos tanto nas arestas quanto nos vértices.

Os bancos de dados NoSQL são implantados quando os bancos de dados relacionais já não estão suportando a demanda de dados, ou seja, estão apresentando lentidão em consultas e processamentos. Um NoSQL pode substituir um banco de dados relacional por completo, ou simplesmente, ser implantado como um “suporte”, realizando as consultas/processamentos em um grande volume de dados onde a velocidade é necessária. O tipo de NoSQL é escolhido após uma análise crítica do gestor de TI da organização, dependendo da necessidade da mesma (DEVMEDIA, 2013).

### MongoDB

Segundo Chodorow (2013), o MongoDB é um banco de dados NoSQL poderoso, flexível e escalável. Ele combina a capacidade de dimensionamento com características tais como índices secundários, consultas por abrangência, classificação, agregações e índices geoespaciais.

O MongoDB não é um banco de dados relacional, mas sim um banco de dados orientado a documentos. Uma das principais razões para a migração dos tradicionais modelos relacionais é a necessidade de fazer um escalonamento de modo mais fácil (CHODOROW, 2013).

Um banco de dados orientado a documentos substitui o tradicional conceito de uma "linha" com um modelo mais flexível, o "documento". Ao permitir a incorporação de documentos e *arrays*[[32]](#footnote-32), o banco de dados orientado a documento permite representar relações hierárquicas mais complexas com um único registro. Isso se encaixa naturalmente a forma como os desenvolvedores de linguagens orientadas a objetos pensam sobre seus dados (CHODOROW, 2013).

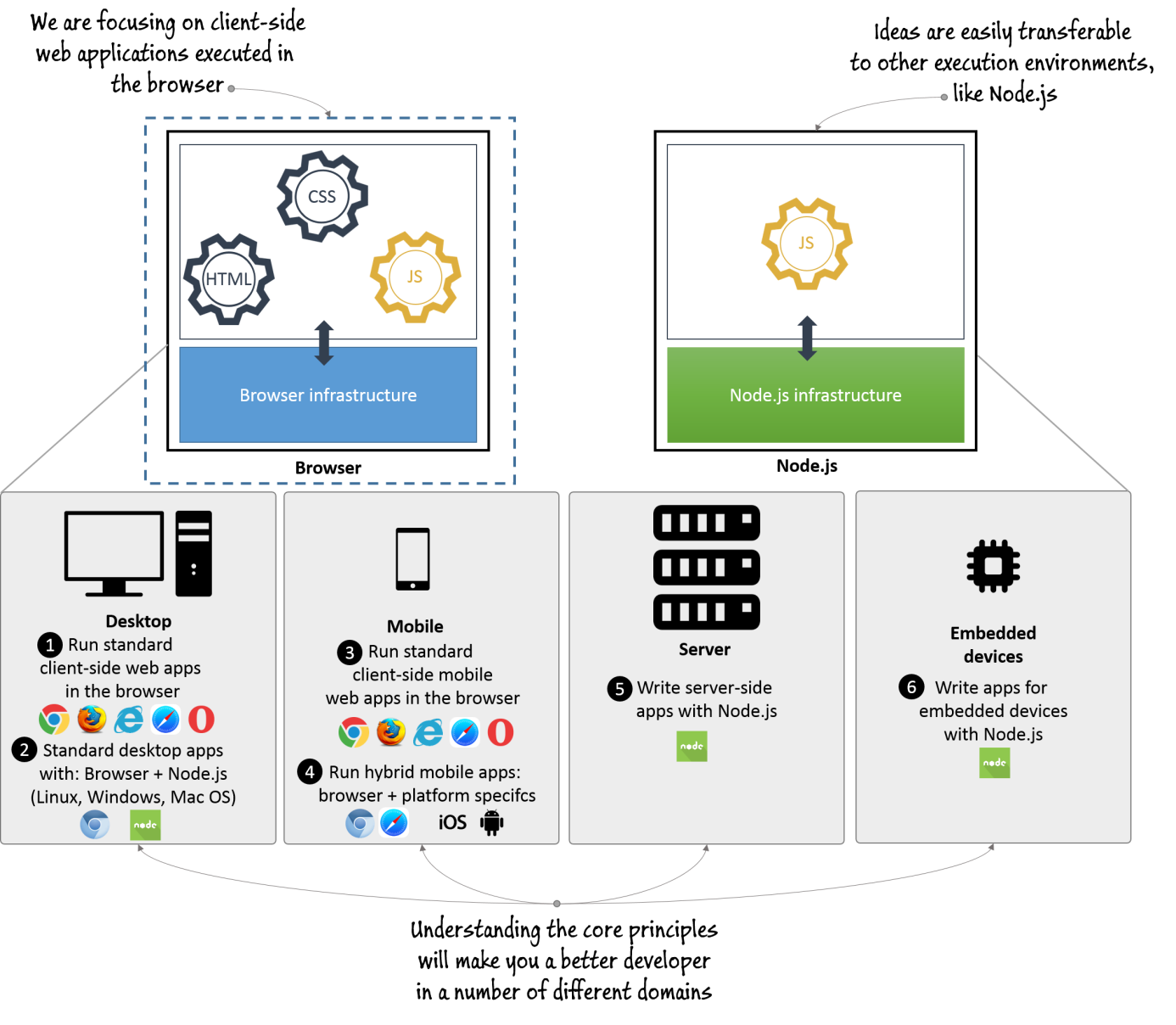
O MongoDB foi projetado para ser escalonável. O seu modelo de dados orientado a documentos o permite separar os dados entre vários servidores de maneira mais fácil. Ele se encarrega automaticamente de balancear a carga de dados através de um *cluster*. Assim, ele consegue redistribuir documentos automaticamente e rotear as solicitações dos usuários para as máquinas corretas. Esta característica permite que os desenvolvedores se concentrem no desenvolvimento da aplicação, não do escalonamento (CHODOROW, 2013).

Caso um cluster necessite de mais capacidade, novas máquinas podem ser adicionadas. Automaticamente, o MongoDB irá descobrir como os dados devem ser distribuídos entre elas (CHODOROW, 2013).

## Node.JS

De acordo com Pereira (2013), os sistemas Web desenvolvidos sobre plataforma .NET, Java, PHP, Ruby ou Pythoncompartilham da mesma característica da paralisação de processamento enquanto utilizam um I/O[[33]](#footnote-33) no servidor. Essa paralisação é conhecida com modelo bloqueante (*Blocking-Thread*). O problema é que essas arquiteturas se tornam ineficientes em alguns casos, pois elas mantêm uma fila ociosa enquanto as operações de I/O são executadas, bloqueando totalmente o sistema. Desse modo, o aumento de acessos no sistema geraram *gap’s*[[34]](#footnote-34)mais frequentes, aumentando a necessidade de fazer um *upgrade* de *hardware* nos servidores, o que é financeiramente inviável.

Baseado neste problema que em 2009, Ryan Dahl e mais 14 colaboradores criaram o Node.JS. O Node.JS tem um modelo inovador, sendo sua arquitetura não-bloqueante (*non-blocking thread*), apresentando uma boa performance e baixo consumo de memória (PEREIRA, 2013).



**Figura 3** - O ecossistema da linguagem *JavaScript*. **Fonte:** DZone (2015).

O Node.JS é uma plataforma escalável e de baixo nível, pois a programação é feita diretamente com protocolos de rede e bibliotecas que acessam os recursos do sistema operacional. O *JavaScript* é a sua linguagem de programação. Isso se deu graças à *engine* *JavaScript* V8, utilizada também no navegador Google Chrome (PEREIRA, 2013).

### *Express*

Para Brown (2014), o *Express* é um Web *framework*[[35]](#footnote-35) Node.JS minimalista e flexível que fornece um conjunto robusto de recursos para a criação das mais diversas aplicações Web.

De acordo com Brown (2014), o *Express* tem como características:

* Minimalista: A filosofia do *Express* é proporcionar a camada mínima entre o desenvolvedor e o servidor. Isso significa que ele libera o desenvolvedor para ter a plena expressão de suas ideias, e, ao mesmo tempo, proporciona algo útil para o mesmo.
* Flexível: Como o Express fornece um framework minimalista, fica a cargo do desenvolvedor adicionar ao *Express* diferentes partes de sua funcionalidade quando for necessário, substituindo assim o que não atender as suas necessidades.
* Web *Framework*: Possibilita a construção de páginas Web, sites e aplicações Web. É fácil imaginar que em poucos anos, não existirá mais a distinção entre eles.
* Suporte a aplicações Web de página única: Em vez de um site fazendo requisições de rede à medida que o usuário vai navegando por páginas diferentes, uma aplicação web de página única faz o *download* do site inteiro para o cliente do navegador. O desenvolvimento desse tipo de aplicação é facilitado pelo *Express*.
* Suporte a aplicações Web Multi-páginas e Aplicações Web híbridas: As aplicações Web Multi-páginas usam a abordagem tradicional para o desenvolvimento de websites, enquanto as aplicações Web híbridas usam em conjunto as aplicações Web de página única e as aplicações Web Multi-páginas. O desenvolvimento de ambas é facilitado pelo *Express*.

O *Express* facilita o desenvolvimento de aplicações Web que utilizam o Node.JS como servidor. Ele é minimalista, robusto e flexível, além de ter um eficiente sistema de roteamento, um executável para a geração de aplicações, dentre outras vantagens (NODEBR, 2013).

## *Cloud Computing*

Para Marinescu (2013), o ideal que a computação pode ser livremente distribuída como uma utilidade pública, do mesmo modo que a água e a eletricidade, foi formulado na década de 1960 pelo cientista da computação e visionário John McCarthy, o mesmo que defendeu a lógica matemática na inteligência artificial. Cerca de quatro décadas depois, com a internet já consolidada, a causa da “utilidade computacional” foi resgatada por grandes empresas de TI, tais como Amazon, Apple, Google, HP, IBM, Microsoft e Oracle.

O movimento da computação em nuvem foi um movimento iniciado em algum período em meados da primeira década do novo milênio, cuja ideia principal é que o processamento de informações pode ser feito com mais eficiência em métodos de armazenamento via internet, sendo assim alocados em um “*server farm*”, ou seja, um conjunto de servidores com capacidades computacionais extremamente superiores a de um computador convencional (MARINESCU, 2013).

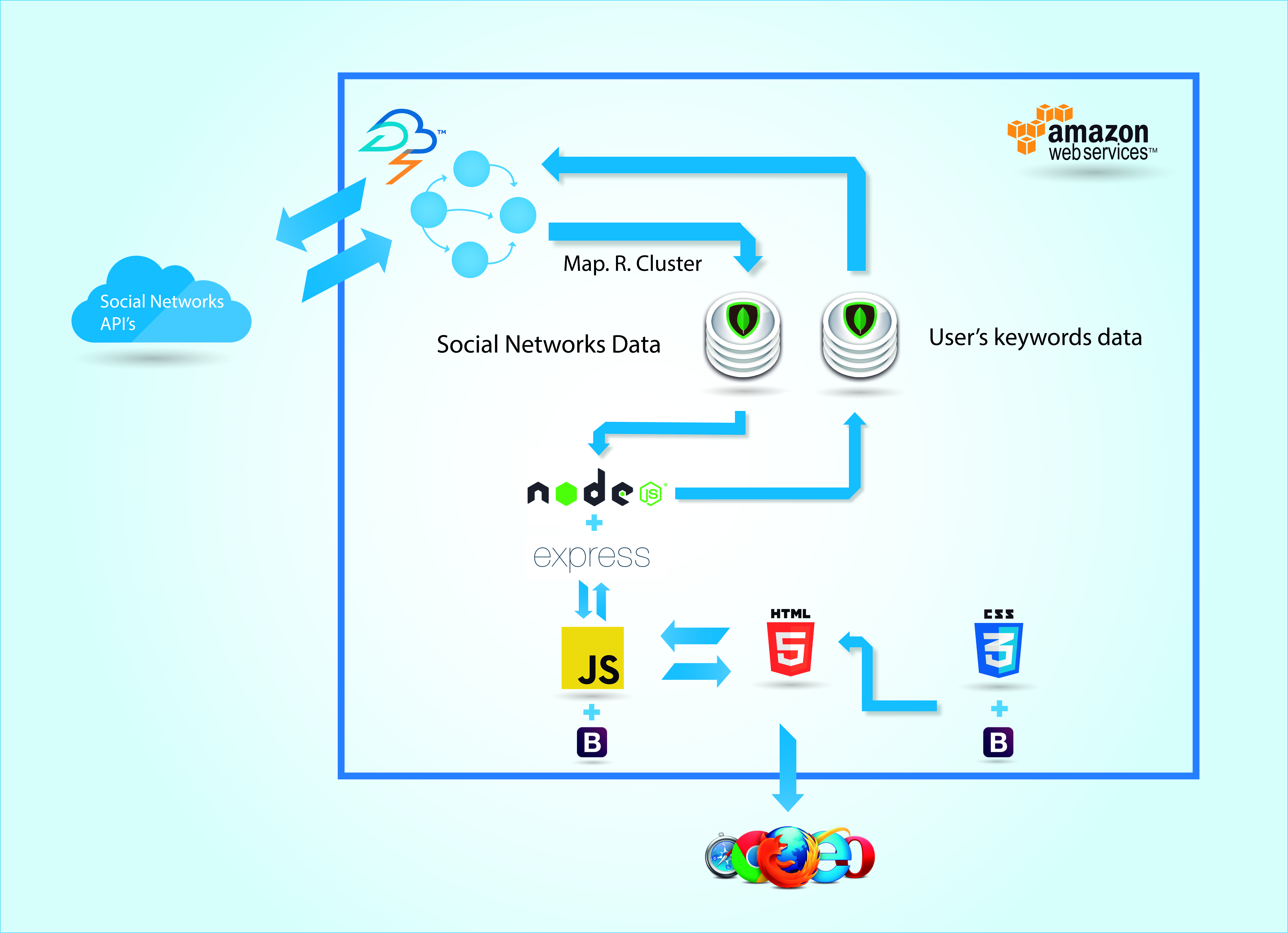
Uma das principais vantagens da computação em nuvem é que ela oferece serviços computacionais e de armazenamento escaláveis e elásticos. Todos os recursos utilizados para estes serviços são medidos. Com isso, os usuários serão somente cobrados pelos serviços que eles consomem. Aplicações científicas e de engenharia, mineração de dados, financiamento computacional, jogos, redes sociais, e muitas outras atividades computacionais de uso intensivo de dados podem se beneficiar da computação em nuvem (MARINESCU, 2013).

A arquitetura da computação em nuvem oferece três tipos de serviços: Software como serviço (SaaS), Plataforma como serviço (PaaS) e Infraestrutura como Serviço (IaaS). Os serviços baseados em SaaS são mais designados ao usuário final. Os serviços baseados em PaaS são mais voltados a área de desenvolvimento, oferecendo assim poderosas ferramentas ao desenvolvedor. Já os serviços baseados em IaaS oferecem serviços de infraestrutura de TI completos, liberando assim a organização para focar nos negócios (MARINESCU, 2013).

A computação em nuvem já é uma realidade empresarial em dias atuais. O número de organizações que estão adotando o paradigma é crescente (MARINESCU, 2013).

### *Amazon Web Services*

De acordo com Murty (2008), o *Amazon Web Services* é um conjunto de *web services* disponibilizados pela *Amazon*[[36]](#footnote-36) que permitem aos desenvolvedores acessar e desenvolver sobre a sua plataforma tecnológica. Neste conjunto de *web services*, estão inclusos diversos serviços de infraestrutura, os quais têm a finalidade de expansão ou substituição da infraestrutura física tradicional de aplicações Web (MURTY, 2008).



**Figura 4** - Fronteira entre a *Amazon Web Services e* as tecnologias empregadas no trabalho. **Fonte:** Elaborado pelos autores.

Estes serviços de infraestrutura fornecem armazenamento, capacidade computacional, um sistema de mensagens, um sistema de pagamento e um banco de dados que pode ser acessado por qualquer pessoa, desde que possua uma conta na Amazon.com e um cartão de crédito. A principal característica, é que os usuários pagam somente pelo que usam (MURTY, 2008).

Esses serviços oferecem uma alternativa ao tradicional desenvolvimento de aplicações sob o hardware físico, pois eles fornecem componentes de infraestrutura escalável, confiável e rentável sem exigir muito esforço, ao contrário da montagem do hardware proprietário. Desse modo, o foco se torna a aplicação em vez da plataforma física (MURTY, 2008).

Outra vantagem desses serviços é a redução significativa do investimento inicial exigido para construir e oferecer um serviço. A partir do momento em que a gestão de infraestrutura está sob a responsabilidade da *Amazon*, o foco se torna o desenvolvimento da aplicação e o crescimento dos negócios. Desse modo, a aplicação web está liberada para competir e ter sucesso com base em suas ideias, inovação e execução (MURTY, 2008).

## *Open Web Plataform*

De acordo com W3C (2015), a *Open Web Plataform* é uma coleção de tecnologias abertas que podem ser implementadas sem a necessidade de qualquer aprovação ou pagamento de taxas de licença.

Seguindo esse conceito, neste tópico serão descritas as tecnologias da *Open Web Plataform* que serão utilizadas neste projeto.

### HTML5

Segundo W3C (2015), a WWW (*World Wide Web*) foi criada em 1989, por Tim Berners-Lee no CERN (*European Organization for Nuclear Research*[[37]](#footnote-37)) em Genebra, na Suíça. Berners-Lee estava trabalhando em uma seção de serviços de computação no CERN, quando ele teve a ideia de reunir e organizar as pesquisas de diversos cientistas do mundo todo. Mais do que somente reunir, ele propôs que os textos poderiam ter *links*[[38]](#footnote-38) entre si, ou seja, durante a leitura de uma pesquisa, seria possível rapidamente exibir parte de outro artigo que contém texto ou diagramas diretamente relevantes.

Como estava tudo muito recente, era necessário ter uma implementação relativamente simples. Seguindo esse conceito, Berners-Lee desenvolveu o protocolo HTTP. Desse modo, o formato de texto escrito para o HTTP foi nomeado de HTML. Entre 1993 e 1997, o HTML iria passar pelas versões HTML+, HTML 2.0, HTML 3.0, HTML 4.0 e HTML 4.0.1, todas abrangendo diversas propostas para enriquecer as possibilidades da linguagem. Em 2009, foi lançado o HTML5, sua atual versão (W3C, 2015).

Ainda de acordo com W3C (2015), o HTML5 tem como uma de suas principais funções a facilidade de manipulação do elemento. Com isso, o desenvolvedor pode modificar as características dos objetos de forma não intrusiva e transparente para o usuário final. Contrariamente as suas versões anteriores, o HTML5 fornece um número de ferramentas para o CSS[[39]](#footnote-39) e o *JavaScript*[[40]](#footnote-40) executarem suas funções de maneira mais eficiente possível.

Continuando com W3C (2015), HTML é uma abreviação de *Hypertext* *Markup* *Language* - Linguagem de Marcação de Hypertexto. A função do HTML é a publicação de conteúdo (texto, imagem, vídeo, áudio e etc) na Web.

### CSS3

Segundo Pereira (2009), o CSS nasceu das complicações de aparência e da diferença de compatibilidade dos navegadores das páginas HTML. A principal função do CSS é separar o conteúdo e formato de um documento de sua apresentação, incluindo elementos como cores, formatos de fontes e layout, além de ser responsável por definir animações e transições. Com isso, foi proporcionada uma maior flexibilidade e controle na especificação de como as características serão exibidas, além de permitir um compartilhamento de formato e reduzir a repetição no conteúdo estrutural de uma página.

Para Mozilla Developer Network (2015), o CSS3 é a mais recente evolução do CSS. Dentre suas novidades, estão cantos arredondados, sombras, gradientes, transições ou animações, bem como novos *layouts* como multiplas colunas, caixa flexível ou *layouts* de grade. As partes experimentais devem ser usadas com cautela, pois tanto sua sintaxe quanto sua semântica podem mudar no futuro.

### *JavaScript*

De acordo com Powers (2009), a linguagem *JavaScript* foi inicialmente destinada a ser uma interface *script* entre uma página Web carregada no cliente do navegador (na época, o Netscape) e a aplicação no servidor. Desde a sua introdução em 1995, *JavaScript* se tornou um componente chave do desenvolvimento Web, além da descoberta seu uso em outros lugares.

Segundo Microsoft (2015), a linguagem *JavaScript* é uma linguagem de *script* interpretada com base em objetos. Sua sintaxe é semelhante a sintaxe da linguagem C.

Continuando com Powers (2009), *JavaScript* é uma linguagem fracamente tipada, ou seja, os tipos de dados das variáveis não são declarados de forma explícita. Na maioria das vezes, o *JavaScript* executa conversões de tipo das varáveis automaticamente.

## *Bootstrap*

De acordo com Magno (2012), o *Bootstrap* é um *framework* *front-end*[[41]](#footnote-41) intuitivo criado para facilitar o desenvolvimento dos elementos de interface em páginas Web. Pode ser também utilizado como um guia para reproduzir de forma consistente os padrões de desenvolvimento consolidados pelo *Twitter*. O *Bootstrap* também pode ser utilizado para facilitar na padronização e nas melhores práticas de desenvolvimento HTML/CSS e *JavaScript*, tanto para iniciantes, quanto para desenvolvedores com habilidades mais avançadas que desejam dar um passo além em interações mais complexas.

## *Google Maps API*

De acordo com Svennerberg (2010), O *Google Maps* foi inicialmente introduzido em um post de um *blog* em 2005, onde, até então, as soluções de mapa da época eram caras, necessitando servidores especiais exclusivos para os sistemas de mapas. A partir de então, ocorreu uma revolução do modo como os mapas em páginas web funcionavam, trazendo um maior nível de interatividade e menor custo. Desse modo, em junho de 2005 foi lançada a primeira versão pública da API do *Google Maps*. Atualmente, o *Google Maps* API já se encontra em sua terceira versão, lançada em Maio de 2010.

Continuando com Svennerberg (2010), o funcionamento do *Google Maps* API é uma interação entre HTML, CSS e *JavaScript*. O mapa é composto por imagens que são carregadas em segundo plano por meio de chamadas AJAX. Posteriormente, o conteúdo é inserido em uma *div* na página HTML. Conforme é feita a navegação no mapa, a API envia informações sobre novas coordenadas e intensidade de zoom no mapa, renderizando assim novas imagens.

Ainda de acordo com Svennerberg (2010), a API consiste em arquivos *JavaScript* que contém classes com os métodos e propriedades que definem o comportamento do mapa, fazendo assim com que o mesmo tenha um comportamento dinâmico.

1. A escola e o futuro da biblioteca de pesquisa. [↑](#footnote-ref-1)
2. Os usuários podem realmente absorver dados nas taxas de hoje? Amanhã? [↑](#footnote-ref-2)
3. Paginação por demanda controlada por aplicativo para visualização externa. [↑](#footnote-ref-3)
4. Explorar visualmente conjuntos de dados gigabyte em tempo real.

   Exabyte - Unidadede medida computacional. [↑](#footnote-ref-4)
5. Quanta informação? [↑](#footnote-ref-5)
6. Modelos de fator dinâmico de Big Data para medição e previsão macroeconômica. [↑](#footnote-ref-6)
7. Gestão de dados 3D: controlar o volume de dados, velocidade e variedade. [↑](#footnote-ref-7)
8. A expansão do universo digital: uma previsão de crescimento da informação Worldwide até 2010. [↑](#footnote-ref-8)
9. Computação Big-Data: Criando avanços revolucionários no comércio, ciência e sociedade. [↑](#footnote-ref-9)
10. Dados, dados em toda parte. [↑](#footnote-ref-10)
11. Big data: a próxima fronteira para a inovação, concorrência e produtividade. [↑](#footnote-ref-11)
12. Termo usado para definir o conceito de armazenamento dos dados relativos às atividades de uma organização. [↑](#footnote-ref-12)
13. Questões críticas para Big Data em Informação, Comunicação e Sociedade. [↑](#footnote-ref-13)
14. Termo inglês usado para designar o pensamento ou gosto atual da maioria da população. [↑](#footnote-ref-14)
15. Conjunto de instruções e padrões de programação para acesso a um aplicativo baseado na Web. [↑](#footnote-ref-15)
16. Solução de integração de sistemas, bom como na comunicação entre aplicações diferentes. [↑](#footnote-ref-16)
17. Tecnologia de processamento de requisições HTTP. [↑](#footnote-ref-17)
18. Publicação de um usuário do *Twitter*. [↑](#footnote-ref-18)
19. Arquivo compactado usado para distribuir um conjunto de classes ou um aplicativo Java, entre outros. [↑](#footnote-ref-19)
20. Protocolo aberto que permite uma autorização segura com um método simples e padronizado para aplicações web, *mobile* e *desktop*. [↑](#footnote-ref-20)
21. Abreviação de GNU zip, um Software Livre de compressão sem perda de dados. [↑](#footnote-ref-21)
22. Relacionamento entre dois ou mais computadores para que estes trabalhem de maneira conjunta no intuito de processar uma tarefa. [↑](#footnote-ref-22)
23. *Java Virtual Machine* - Máquina Virtual Java. [↑](#footnote-ref-23)
24. *American Standard Code for Information Interchange* - Codificação usada para representar textos em computadores, equipamentos de comunicação, entre outros dispositivos que trabalham com texto. [↑](#footnote-ref-24)
25. *Structured Query Language* - Linguagem de consulta estruturada, usada em bancos de dados. [↑](#footnote-ref-25)
26. Linguagem baseada em *scripts*. [↑](#footnote-ref-26)
27. Sistema operacional. [↑](#footnote-ref-27)
28. Banco de dados criado pelo Google. [↑](#footnote-ref-28)
29. Serviço de banco de dados NoSQL. [↑](#footnote-ref-29)
30. *Extensible Markup Linguage* - Linguagem de Marcação Extensiva. [↑](#footnote-ref-30)
31. *JavaScript Object Notation* - Notação de Objetos *JavaScript*. [↑](#footnote-ref-31)
32. Termo da computação que significa uma lista que armazena uma coleção de elementos. [↑](#footnote-ref-32)
33. Input/Output – Entrada/Saída. [↑](#footnote-ref-33)
34. Gargalos. [↑](#footnote-ref-34)
35. Abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica. [↑](#footnote-ref-35)
36. Empresa multinacional de comércio eletrônico. [↑](#footnote-ref-36)
37. Organização europeia de pesquisas nucleares. [↑](#footnote-ref-37)
38. Ligações, referências. [↑](#footnote-ref-38)
39. *Cascading Style Sheets*, formata a informação entregue pelo HTML [↑](#footnote-ref-39)
40. Linguagem de programação interpretada pelo navegador. [↑](#footnote-ref-40)
41. Toda parte incluída na apresentação visual de um site. [↑](#footnote-ref-41)