CENTRO UNIVERSITÁRIO UNIVATES
CURSO DE SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE UMA FERRAMENTA DE COLETA E ARMAZENAMENTO DE DADOS PARA BIG DATA

Bruno Edgar Fuhr

Lajeado, novembro de 2014.

Bruno Edgar Fuhr

DESENVOLVIMENTO DE UMA FERRAMENTA DE COLETA E ARMAZENAMENTO DE DADOS PARA BIG DATA

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação do Centro Universitário Univates, para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Evandro Franzen

Lajeado, novembro de 2014.

RESUMO

O crescimento da internet na última década fez dela a maior fonte de dados de acesso público em todo mundo. Pessoas colocam nela suas opiniões, comportamentos, relações e diversas outras informações potencialmente úteis para uso em sistemas de análise de dados. Estes dados formam o que se chama de “Big Data”, termo que se refere ao enorme volume de dados que existem hoje nas mais diversas fontes. As organizações podem aproveitar essas informações disponíveis e utilizá-las para benefício de seus negócios, tendo a oportunidade de analisar o comportamento de seus clientes, o que o público pensa de seus produtos ou serviços, e muitas outras análises possíveis. Porém, para um sistema conseguir realizar alguma análise, primeiro é necessário obter e armazenar um grande volume de dados, sendo este o principal objetivo do presente trabalho: construir uma ferramenta, através da qual seja possível definir uma estrutura de armazenamento e também configurar coletores de dados. Esta ferramenta permitirá a coleta de dados de diversas fontes, tais como redes sociais, através de diferentes métodos de busca, como palavras-chave. Para atingir este objetivo, é apresentado neste trabalho o projeto e o desenvolvimento desta ferramenta.

Palavras-chave: Big Data, coleta de dados, ferramenta.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

TB:	Terabyte
GB:	Gigabyte
SGDB:	Sistema Gerenciador de Banco de Dados
SQL:	Structured Query Language
NoSQL:	Not Only SQL
TI:	Tecnologia da Informação
URL:	Uniform Resource Locator
RDBMS:	Relational Database Management System
BSD:	Berkeley Software Distribution
JSON:	JavaScript Object Notation

SUMÁRIO

1 INTRODUÇÃO

Gerenciar e analisar dados oferece grandes desafios e benefícios para organizações de todos os setores e tamanhos, que têm buscado encontrar métodos eficazes de capturar informações de seus clientes, fornecedores, produtos e serviços (HURWITZ *et al*, 2013). Nos dias de hoje, há uma imensa quantidade de dados disponíveis nos mais diversos locais. Sistemas de informação, páginas de notícias, *blogs* e redes sociais são alguns exemplos de fontes de dados que podem ser utilizadas para extrair informações relevantes.

O rápido crescimento da internet na última década fez dela a maior fonte de dados de acesso público em todo o mundo (LIU, 2011).

No entanto, o volume com que essas informações são criadas e atualizadas diariamente excedem a capacidade de armazenamento e processamento das tecnologias de informação tradicionais. Para se ter uma ideia, as redes sociais Twitter e Facebook geram, por dia, em torno de 17 terabytes (TB) de dados (EATON *et al*, 2012).

A este grande volume de dados, que não pode ser processado e analisado utilizando-se processos e ferramentas tradicionais, atribui-se o termo “Big Data” (EATON *et al*, 2012). Big Data refere-se ao enorme crescimento no volume, variedade e velocidade com que os dados estão sendo produzidos e ao conjunto de aplicações que geram, armazenam, processam e analisam estes dados.

Segundo Hurwitz (2013), as pessoas estão se tornando emissoras de informações. Ao navegar pela rede, deixam rastros digitais com inúmeras informações de valor, como por exemplo: como tomam decisões de compra, como influenciam seus amigos, que assuntos procuram nos sites de busca, porque amam

ou odeiam as marcas; e estão transformando a internet em uma imensa plataforma de pesquisa, que pode ser utilizada para reduzir custos e até como principal fonte de coleta de opiniões.

Porém, lidar com esta imensa quantidade de dados gerados traz novos desafios, principalmente na área de banco de dados. Neste contexto de dezenas, ou centenas de terabytes de dados não estruturados, os bancos de dados tradicionais não são os mais adequados, principalmente por possuírem uma estrutura rígida, não permitindo armazenar modelos de dados flexíveis (VIEIRA *et al*, 2012).

Além de requerer uma estrutura de armazenamento mais flexível e robusta, este grande volume de dados também necessita de um ambiente onde possam ser aplicados algoritmos de processamento e de análise, para que dele extraiam-se informações úteis e de valor.

Em vista destes novos desafios, este trabalho visa criar uma ferramenta de apoio à criação, obtenção e armazenamento de um grande volume de dados, de modo que possa servir como base para o futuro desenvolvimento de aplicações que utilizem técnicas de mineração de dados. Também neste trabalho é proposta uma arquitetura para este futuro sistema, com módulos de processamento e análise de grande volume de dados.

1.1 Objetivos

O presente trabalho tem como objetivo principal construir uma estrutura de coleta e armazenamento de dados, utilizando-se de coletores de dados (*crawlers)*. Estes coletores formarão um banco de dados com as informações obtidas, que poderá servir de base para um sistema de análise de grandes volumes de dados (Big Data).

Para atingir o objetivo principal definido, os seguintes objetivos específicos também deverão ser cumpridos:

- Compreender as tecnologias utilizadas para coletar dados da internet;
- Compreender os princípios de armazenamento em banco de dados NoSQL;
- Permitir a busca de dados através de rotinas automatizadas;
- Contribuir para o desenvolvimento de aplicações de mineração de dados em

Big Data;

- Propor uma arquitetura de uma ferramenta de coleta, processamento e análise de grande volume de dados.

1.2 Justificativa e relevância

Big Data está se tornando uma das tendências tecnológicas mais importantes (HURWITZ *et al*, 2013), com o potencial de mudar radicalmente a forma como as organizações usam a informação para melhorar a experiência do cliente e transformar seus modelos de negócio. É importante pois permite que as organizações manipulem grandes volumes de dados na velocidade e no tempo certo, a fim de gerar conhecimento para seus modelos de negócio.

Cada fonte de dados, como as redes sociais, possui uma estrutura e um modelo de acesso aos dados diferente e, portanto, demandam de técnicas de coleta diferentes. Com o desenvolvimento da ferramenta proposta, o desenvolvimento de um sistema de Big Data é simplificado, uma vez que os dados a serem analisados podem ser coletados de uma forma simples e fácil, deixando o desenvolvedor preocupado apenas com o que realmente interessa em um sistema deste tipo: como analisar e quais informações extrair desta grande massa de dados.

A partir da extração de conhecimento deste montante de dados a princípio discrepantes, empresas podem descobrir o que as pessoas pensam de seus produtos, podem coletar *feedback* sobre algum evento e também relacionar fatos do mundo real com dados internos de seus sistemas de gestão.

Outro ponto relevante e que justifica o esforço proposto neste trabalho é que pesquisas sobre Big Data e bancos de dados NoSQL ainda carecem de investigação e estudos. O presente trabalho, a partir de seu objetivo, vem a fazer parte de uma iniciativa maior do autor, que é, futuramente, desenvolver um ecossistema para mineração de dados em Big Data.

1.3 Delimitação

O presente trabalho visa a estruturação de uma ferramenta que auxilie na

montagem e estruturação de uma base de grande volume de dados, através de coletores que podem ser configurados para os mais diversos objetivos. Esta base de dados poderá servir para o futuro desenvolvimento de aplicações em Big Data, que utilizem técnicas de mineração nestes dados coletados. Porém, o trabalho proposto trata apenas da coleta e armazenamento de informações, com possibilidade de efetuar algumas consultas básicas para demonstrar os dados armazenados. É proposta uma arquitetura de um sistema maior, com módulos de processamento e análise de grande volume de dados, que pode ser utilizada no desenvolvimento de futuros trabalhos utilizando a ferramenta desenvolvida.

Não é o foco deste trabalho tratar de questões como: análise quantitativa e qualitativa dos dados, pesquisas semânticas de texto e/ou outros tipos de análises avançadas dos dados coletados.

O sistema proposto irá armazenar e organizar os dados obtidos das fontes, contendo uma tela para consultas dos dados.

1.4 Estrutura do documento

No capítulo seguinte são apresentados mais detalhes sobre o conceito de Big Data. A seguir, no capítulo 3, são expostos alguns conceitos de mineração de dados. No capítulo 4 fala-se sobre bancos de dados relacionais e NoSQL, suas características e diferenças e são apresentados algumas implementações de sistemas NoSQL. No 5º capítulo é apresentado o projeto da solução proposta e, no 6º, a implementação deste projeto. Em seguida são apresentados os resultados obtidos através da ferramenta desenvolvida, no capítulo 7. Por fim, a conclusão e as considerações finais são apresentadas no capítulo 8.

2 BIG DATA

"Big Data" é o conjunto de dados que excede a capacidade dos sistemas de banco de dados tradicionais (DUMBILL, 2012), enquanto que um "sistema de Big Data" é um conjunto de recursos computacionais baseados em programação, algoritmos, ferramentas de busca e banco de dados, que permitem agregar todos os dados estruturados e não estruturados. Quando unidas, estas fontes de dados geram um enorme volume, do qual uma aplicação pode extrair conhecimentos, buscar padrões e fazer análise que não seriam possíveis sem essa agregação. Não é uma tecnologia única, mas uma combinação de velhas e novas técnicas que ajudam as organizações a obterem conhecimentos práticos. Portanto, um sistema de Big Data deve ter a capacidade de gerenciar um vasto volume de dados discrepantes, para permitir análises e decisões em tempo real.

Conforme Hurwitz (2013), pode-se dividir Big Data em três características, os chamados três V's, que são detalhados a seguir:

- Volume: os benefícios adquiridos com a capacidade de processar grandes quantidades de informação é o grande atrativo do Big Data. Quanto maior o volume de dados disponíveis para análise, mais precisos serão os resultados. E é este grande volume de dados o maior desafio para as estruturas de TI (Tecnologia da Informação) convencionais, pois é necessário um sistema de armazenamento escalável e uma abordagem de consulta eficaz. Muitas organizações já tem um grande número de dados armazenados, porém não tem ainda a capacidade de processá-los (DUMBILL, 2012).

- Velocidade: velocidade dos dados refere-se ao aumento da taxa com que os dados crescem em uma organização e em quanto tempo são analisados. A

importância da velocidade com que esses dados são compilados e analisados traz um grande diferencial estratégico para a organização. Por exemplo, uma empresa de operações financeiras necessita lidar com mudanças rápidas nos dados analisados para ter alguma vantagem a seu favor. Ou então uma loja on-line que a cada clique e interação do usuário pode mostrar recomendações e promoções diferentes.

- Variedade: raramente os dados coletados em um sistema de Big Data encontram-se ordenados e prontos para processamento. A maioria das fontes de dados não se enquadra em estruturas relacionais puras, como por exemplo: textos de redes sociais, dados de imagens e páginas de internet. Nada disso vem pronto para a integração em uma aplicação de análise.

A grande novidade das soluções de Big Data é lidar não apenas com dados estruturados de bancos de dados relacionais, mas também com os chamados dados não-estruturados, que até então só podiam ser compreendidos por pessoas. São mensagens em redes sociais, comentários em blogs e comportamento de clientes que dependem de contexto para terem sentido.

2.1 Tipos de dados em um Big Data

Existe uma grande variedade de dados a serem gerenciados e analisados por um sistema de Big Data, desde transações financeiras até mensagens em redes sociais. Porém existem apenas dois tipos principais de dados definidos quando se trabalha com Big Data: estruturados e não-estruturados.

2.1.1 Dados estruturados

O termo “dados estruturados” refere-se aos dados que tem um tamanho e formato definidos. Exemplos de dados estruturados incluem números, datas e palavras. Estes dados normalmente estão armazenados em um banco de dados, sendo possível consultá-los utilizando linguagens como o SQL (*Structured Query Language*).

As fontes dos dados estruturados são divididas em duas categorias:

- Gerados por máquinas: geralmente refere-se aos dados criados por uma máquina sem a intervenção humana, como *logs* e dados de sensores.

- Gerados por humanos: dados gerados por pessoas, em interação com sistemas computacionais, como informações de um cliente inseridas em um aplicativo, cliques em um site e palavras em um mecanismo de busca.

2.1.2 Dados não-estruturados

Dados não-estruturados são aqueles que não seguem um formato específico. É o tipo de dado encontrado em maior abundância. No entanto, até pouco tempo, a tecnologia não conseguia fazer muito com eles, exceto armazená-los ou analisá-los manualmente.

Este tipo de dado está presente em todos os lugares, e assim como os dados estruturados, dividem-se em dados gerados por máquinas e por pessoas.

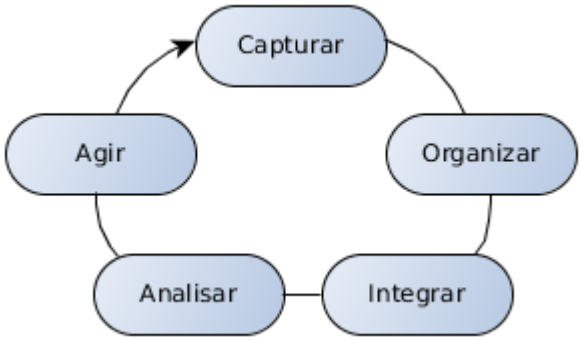
Alguns exemplos de dados não-estruturados gerados por máquina são: dados de satélite, fotos e vídeos de monitoramento e dados de radares ou sonares. Dados não-estruturados gerados por humanos podem ser textos internos das organizações, como e-mails, históricos de conversações e atas de reuniões; dados gerados nas diferentes mídias sociais e conteúdo de *websites* e *blogs*.

2.2 Requisitos de um sistema de Big Data

Para construir uma arquitetura funcional para Big Data, primeiro é necessário conhecer seus requisitos funcionais.

Como pode-se observar na figura 1, os dados precisam primeiramente serem capturados, e então organizados e integrados. Após essa fase, os dados podem ser analisados com base no problema a ser resolvido. Através dos resultados destas análises, o tomador de decisão pode decidir a melhor ação a ser executada para determinada situação.

Figura 1 - Ciclo de gerenciamento de dados em um sistema de Big Data



Fonte: Modificado de Hurwitz (2013)

Um exemplo é uma loja de departamentos on-line, que recomenda um produto a um cliente baseado em compras e visitas anteriores deste, ou oferece descontos em produtos de interesse do usuário.

Embora isso pareça simples, certos aspectos destas funções são complicadas, como por exemplo a validação. A validação dos dados é um ponto importante na combinação dos dados das diversas fontes, pois é crucial que os dados façam sentido quando combinados (HURWITZ *et al*, 2013).

2.3 Exemplos de aplicação de um sistema de Big Data

Um uso comum para sistemas de Big Data é extrair de dados desestruturados alguma informação que faça sentido para uso das pessoas ou para servir de entrada para outra aplicação. O processo de mover estas fontes de dados para aplicações estruturadas pode envolver a perda de informações, por isso deve-se analisar o que é descartado, pois podem haver dados úteis no que foi desprezado.

Já existem aplicações que atendem a problemas reais utilizando-se do Big Data. A seguir, é apresentada uma lista de alguns sistemas implementados a partir deste conceito (GLOBO, 2013):

- uma empresa tira fotos de satélite e vende a seus clientes informações em tempo real sobre a disponibilidade de vagas de estacionamento livres em uma cidade, em determinada hora;

- o projeto “Global Pulse” das Nações Unidas utiliza-se de um programa que decifra a linguagem humana na análise de textos e comentários em redes sociais para prever o aumento do desemprego, o esfriamento econômico e epidemias de doenças;

- a rede varejista americana Dollar General monitora as combinações de produtos que seus clientes põem nos carrinhos. Ganhou eficácia e ainda descobriu curiosidades: quem bebe Gatorade tem mais chances de comprar também laxante;

- a Sprint Nextel saltou da última para a primeira posição no ranking de satisfação dos usuários de celular nos EUA ao integrar os dados de seus canais de relacionamento. De quebra cortou pela metade gastos com o setor de atendimento ao cliente.

3 MINERAÇÃO DE DADOS (*DATA MINING*)

Neste capítulo são apresentados alguns conceitos sobre mineração de dados, que é a base da mineração da internet. Estes conceitos não serão aplicados diretamente no presente trabalho, porém são muito importantes para o entendimento da arquitetura proposta e para os próximos módulos a serem construídos a partir deste trabalho.

Data Mining, termo em inglês que significa “Mineração de dados”, compreende os principais algoritmos que permitem obter percepções e conhecimentos a partir de uma massa de dados. É um campo de pesquisa interdisciplinar, que utiliza conceitos de sistemas de banco de dados, estatística, aprendizado de máquina e reconhecimento de padrões (ZAKI; MEIRA, 2013). Em outras palavras, mineração de dados significa obter dados relevantes e com algum significado a partir de um aglomerado de dados que inicialmente não possuem relações entre si.

Segundo Carvalho (2005), embora as técnicas de mineração de dados sejam antigas, foi apenas nos últimos anos que passaram a ser usadas como exploração de dados, pelos mais diversos motivos:

- O volume de dados disponível atualmente é enorme: mineração de dados é uma técnica que apenas com grandes volumes de dados fornece resultados confiáveis;

- Os dados estão sendo organizados: os dados das mais diversas fontes estão sendo organizados e padronizados, de modo a possibilitar sua organização dirigida ao auxílio da decisão;

- Os recursos computacionais estão cada vez mais potentes: a mineração de

dados necessita de muitos recursos computacionais para operar seus algoritmos sobre grande quantidade de dados;

- A competição empresarial exige técnicas mais modernas de decisão: as empresas buscam na mineração de dados rapidez e velocidade no processo de tomada de decisão.

3.1 Tipos de aplicações em mineração de dados

No geral existem cinco tipos de aplicações em mineração de dados: classificação, estimativa, previsão, análise de afinidades e análise de agrupamentos (CARVALHO, 2005).

3.1.1 Classificação

A classificação é uma discriminação de unidades, classes ou categorias. Por exemplo, classificam-se pessoas como boas/más, eventos como legais/chatos e sabores em doces/salgados.

É uma das técnicas mais utilizadas, pois o ser humano está sempre classificando o que está à sua volta, para auxílio e compreensão do ambiente em que vive.

Em um processo de mineração de dados, a classificação está especificamente voltada à atribuição de uma das classes pré-definidas pelo analista a novos fatos ou objetos submetidos à classificação. Essa técnica pode ser utilizada tanto para entender dados existentes quanto para prever como novos dados irão se comportar.

3.1.2 Estimativa

A estimativa, ao contrário da classificação, está associada a respostas contínuas. Estimar algum índice é determinar seu valor mais provável diante de dados do passado ou de dados de outros índices semelhantes sobre os quais se tem conhecimento.

3.1.3 Previsão

A previsão está associada à avaliação de um valor futuro de uma variável a partir de dados históricos de seu comportamento passado. Por exemplo, é possível prever se um índice da bolsa de valores irá subir ou descer no dia seguinte, de acordo com os seus dados históricos.

A única maneira de avaliar se a previsão foi bem feita é aguardar o acontecimento e verificar o quanto foi acertada ou não a previsão realizada.

3.1.4 Análise de afinidades

A análise de afinidades preocupa-se em reconhecer padrões de ocorrência simultânea de determinados eventos nos dados em análise. Determinar que fatos ocorrem simultaneamente com probabilidade razoável (co-ocorrência) ou que itens de uma massa de dados estão presentes juntos com uma certa chance (correlação).

O exemplo mais clássico de análise de afinidades é o do carrinho de supermercado, do qual deseja-se conhecer quais os produtos que são comumente comprados em conjunto pelos consumidores. Isto possibilita a otimização da organização interna dos supermercados e a realização de vendas dirigidas nas quais os itens são oferecidos já em conjuntos com preços menores.

3.1.5 Análise de agrupamentos

A análise de agrupamentos visa formar grupos de objetos ou elementos mais homogêneos entre si. Pode ser estabelecido previamente um número de grupos a ser formado, ou então se pode admitir ao algoritmo de agrupamento uma livre associação de unidades, de forma que a quantidade de grupos resultante seja conhecida somente ao final do processo.

A diferença entre agrupamento e classificação é que na classificação as classes são pré-definidas pelo pesquisador, enquanto que na análise de agrupamentos não existe tal requisito, o que torna esta técnica muito mais complexa do que a classificação.

3.2 Mineração da internet (*Web Minning*)

Devido à grande quantidade de informações disponíveis, a internet é um campo fértil para a pesquisa e mineração de dados. Desse modo, *Web Mining*, ou mineração na web pode ser entendida como uma extensão da mineração de dados para a internet. Ou seja, é o uso de técnicas de mineração de dados para descobrir e extrair automaticamente informações e conhecimentos relevantes da estrutura e/ou conteúdo de uma página de internet (LIU, 2011).

O processo de mineração na web é similar à mineração de dados. A principal diferença está na coleção de dados utilizada. Na mineração de dados tradicional, os dados geralmente estão armazenados e são coletados em um armazém de dados (*Data Warehouse*[1]). Na mineração na web, os dados não possuem uma estrutura definida e obtê-los envolve coletar um grande número de páginas de internet (LIU, 2011). No item 3.2.1 serão apresentados alguns conceitos de coleta de dados da web.

Segundo Liu (2011), a mineração na web pode ser categorizada em três tipos:

- Mineração da estrutura da web: a mineração da estrutura da web encontra informações de relacionamento entre páginas através de seus *links*. Por exemplo, através dos *links* é possível descobrir páginas importantes, que é uma tecnologia usada em motores de busca. Também é possível descobrir comunidades de usuários que compartilham interesses em comum. A mineração de dados tradicional não executa este tipo de processo porque não há uma estrutura de *links* em uma tabela relacional.

- Mineração do conteúdo da web: extrai informações e conhecimentos do conteúdo das páginas. Por exemplo, pode-se automaticamente classificar páginas da web de acordo com seus tópicos, ou obter opiniões de clientes sobre algum produto em um blog ou fórum.

- Mineração do uso da web: mineração do uso da web se refere à descoberta de padrões em como o usuário age navegando na internet, como por exemplo monitorar cada clique feito por um usuário.

1 Um armazém de dados, ou ainda depósito de dados, é um sistema de computação utilizado para armazenar informações relativas às atividades de uma organização. O desenho da base de dados favorece os relatórios e a obtenção de informações estratégicas que podem facilitar a tomada de decisão.

3.2.1 Coletores de dados da web

Web crawlers, ou coletores de dados da web, são programas que obtêm automaticamente informações de páginas da internet (LIU, 2011). Um coletor pode visitar muitos *sites* para coletar informações que podem ser analisadas e armazenadas em um local único.

Enquanto a internet era uma coleção de páginas estáticas, a vida útil de um coletor era pequena, uma vez que se todas as páginas fossem analisadas e salvas em um repositório, não havia mais o que coletar. Entretanto, a *web* é uma entidade dinâmica, que é atualizada rapidamente, necessitando assim do uso contínuo de coletores.

Há muitas aplicações para coletores da web. Uma delas é inteligência de negócios (*business intelligence*), onde organizações coletam informações sobre seus competidores e potenciais colaboradores. Outro uso comum é o monitoramento de *websites* de interesse, então um usuário ou grupo pode ser notificado quando novas informações estiverem disponíveis em certos lugares. Porém o uso mais difundido dos coletores é a coleta de páginas para motores de busca montarem seus índices. Os bem conhecidos mecanismos de busca do Google, Yahoo e Microsoft rodam coletores muito eficientes, projetados para reunir todas as páginas, independentemente de seu conteúdo (LIU, 2011).

3.2.1.1 Algoritmo básico de um coletor

De uma forma simples, pode-se dizer que um coletor inicia com um conjunto inicial de endereços a visitar, e então usa os *links* contidos nessas páginas para buscar outros endereços. Este processo se repete até que um número suficiente de páginas forem visitadas, ou outro objetivo for alcançado (LIU, 2011).

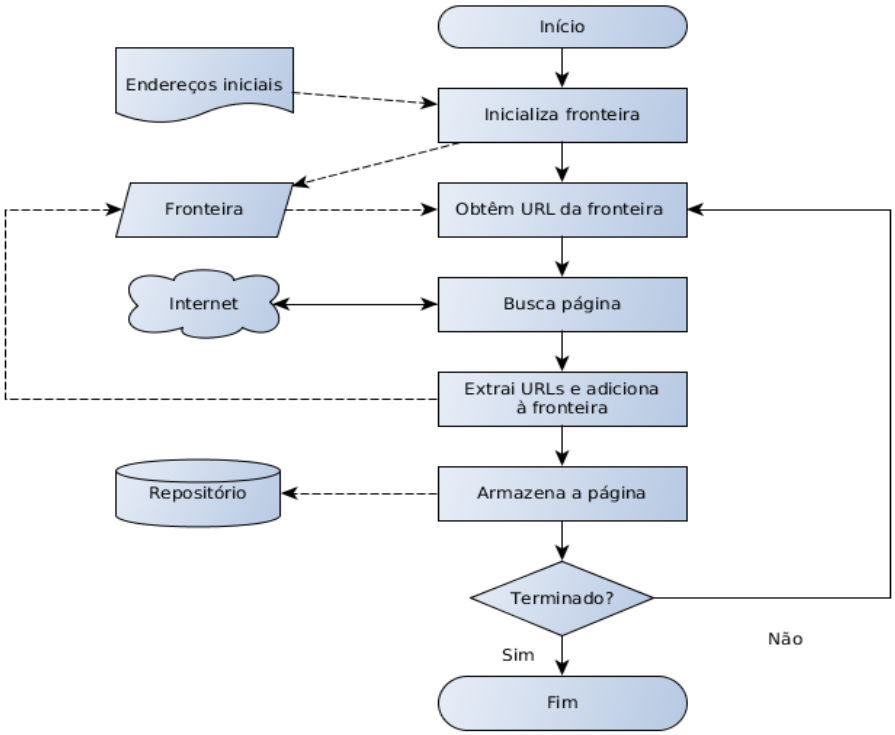
Segundo Liu (2011), a internet pode ser vista como um grande grafo, com páginas sendo os nós e *links* como sendo as bordas (ou arestas). Um coletor inicia através de alguns nós (sementes) e então segue pelas bordas para alcançar outros nós.

A “fronteira” é a principal estrutura de dados, que contém as URLs (*Uniform*

Resource Locator) de páginas ainda não visitadas. Coletores geralmente armazenam a fronteira na memória para ganhar eficiência. O programador de um coletor pode também decidir quais URLs tem alta e baixa prioridade, e assim saber quais descartar quando o espaço da fronteira for preenchido.

A seguir, na figura 2, é apresentado um fluxograma que representa o algoritmo básico de um coletor.

Figura 2 - Algoritmo básico de um coletor de dados da web



Fonte: Modificado de Liu (2011).

4 BANCO DE DADOS

Pode-se definir banco de dados como um conjunto de dados interligados com o objetivo de atender a um grupo de usuários (HEUSER, 1998). Um banco de dados é projetado, construído e preenchido com dados para um propósito específico.

Nos primórdios da programação de aplicações em computadores, os programadores incorporavam em seus sistemas todas as funcionalidades necessárias, desde a interação com o usuário até o acesso e gravação dos dados. Porém, com a evolução dos sistemas e a necessidade de compartilhamento de informações entre aplicações, percebeu-se que esse método dificultava muito a manutenção e a troca de informações com outros sistemas, uma vez que o código de gerência dos dados ficava dentro da aplicação.

A solução para este problema foi o desenvolvimento de uma aplicação específica para o gerenciamento de conjuntos de dados, conhecido por SGBD. Segundo Heuser (1998), SGBD é um *software* que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados.

Os SGBDs surgiram em meados da década de 70 com o intuito de facilitar o desenvolvimento de programas que utilizem dados armazenados (HEUSER, 1998). Os primeiros sistemas desse tipo eram caros e difíceis de utilizar, demandando especialistas treinados para cada SGBD específico. Mais tarde, com a popularização da informática e o barateamento dos equipamentos necessários para executar um SGBD, houve muita pesquisa na área de bancos de dados. Este investimento em pesquisas resultou em um tipo de SGBD, que praticamente tornou-se o padrão para todas as implementações do mercado: o SGBD relacional.

4.1 Sistemas de banco de dados relacionais

Um sistema gerenciador de banco de dados relacional, ou RDBMS (*Relational Database Management System*), é aquele em que os dados estão organizados em uma coleção de relações, comumente chamadas de “tabelas”. “Relação” é um termo formal em lógica matemática, que remete à noção de “relação entre as coisas” e que, em geral, pode relacionar qualquer número de coisas (DARWEN, 2009).

Um RDBMS é composto principalmente pela especificação de relações, chaves e restrições de integridade. A seguir são detalhados estes conceitos:

- Relação: mais conhecidas como tabelas, são conjuntos ordenados de tuplas. Tuplas, também chamadas de “linhas”, possuem um conjunto de atributos, e cada atributo possui um nome e um tipo específico de dados. Estes atributos também são conhecidos como “colunas”.

- Chaves: servem como identificador único de uma tupla e para relacionar tuplas e relações. Podem ser do tipo primária, que distingue uma tupla de outra em uma relação, ou estrangeira. Uma chave estrangeira é uma coluna ou conjunto de colunas cujos valores aparecem como chave primária em outra relação, implementando assim o relacionamento entre as relações.

- Restrições de integridade: são regras de consistência de dados. Podem ser classificadas como: integridade de domínio, que define que o valor de uma coluna deve obedecer ao domínio desta coluna; integridade de valores nulos, que define se uma coluna pode ou não ter valores vazios (nulos); integridade de chave, que define que os valores para chaves primária e alternativas devem ser únicos; e integridade referencial que define que os valores em uma chave estrangeira devem aparecer na chave primária da tabela referida.

Dentre os principais sistemas gerenciadores de bancos de dados relacionais, pode-se citar o Orcale, o MySQL, o Microsoft SQL Server e o PostgreSQL. A seguir é apresentado o sistema PostgreSQL, que será o RDBMS utilizado no desenvolvimento do presente trabalho, principalmente por dois motivos: é um sistema de código aberto, não sendo necessário adquirir licenças de uso, e por ser reconhecidamente um sistema estável e poderoso.

4.1.1 PostgreSQL

Segundo Milani (2008), o PostgreSQL é um RDBMS encarregado de armazenar e gerenciar dados de acordo com regras previamente definidas. Teve origem na Universidade Berkeley, na Califórnia, EUA, e sua primeira versão estável foi lançada no ano de 1989. Atualmente encontra-se em uma versão estável e confiável, disponibilizando os principais recursos existentes nos bancos de dados pagos do mercado e com capacidade para suprir pequenas, médias e grandes aplicações.

Este RDBMS é compatível com diversos sistemas operacionais como o Linux, Windows, MacOS, Solaris, entre outros. Também oferece suporte para as principais plataformas e linguagens de programação, podendo-se citar: C, Java, PHP, .NET, Python e Ruby.

A seguir, algumas características e recursos existentes no PostgreSQL:

- suporte a transações;

- replicação – oferece recursos necessários para a replicação entre servidores;

- alta disponibilidade;

- criptografia;

- incorporável em aplicações gratuitamente – por utilizar a licença BSD (*Berkeley Software Distribution*), pode ser livremente incorporado em aplicações pessoais e/ou comerciais;

O PostgreSQL não tem limite de tamanho para seus bancos de dados, sendo limitado apenas pelo hardware disponível. Sua limitação está a nível de tabela, com um limite máximo de 32TB e 1600 campos por tabela. Cada linha de uma tabela pode ter até 1.6TB e campos de até 1GB (MILANI, 2008).

4.2 Sistemas de banco de dados NoSQL

Devido ao intenso crescimento do número de aplicações, soluções, recursos e tudo o que se refere a sistemas computacionais nas últimas décadas, o volume de dados associados a tais tipos de sistemas também teve um ritmo de crescimento

acelerado.

Atualmente o volume de dados de determinadas organizações atinge o nível de *petabytes*, o que corresponde a 10^{15} *bytes*. Em cenários como este, os bancos de dados do modelo relacional já não se mostram tão eficientes, principalmente pela demanda cada vez mais frequente de escalabilidade. Por exemplo, uma aplicação sendo executada em um banco de dados relacional começa a ter uma queda de performance à medida em que o volume de dados aumenta. Para resolver isso, restam duas alternativas: aumentar a capacidade do servidor ou aumentar o número de servidores.

Porém, estas opções não são suficientes se o volume de dados continuar a crescer exponencialmente, pois então o problema passa a ser no próprio acesso à base de dados. A solução passa a ser distribuir o banco de dados em diversos servidores, em diversas máquinas, o que é possível de ser realizado em um SGBD relacional, porém não de uma maneira simples, pois por sua natureza estruturada, a organização dos dados em um sistema distribuído é complexa (LEAVITT, 2010).

Assim, começam a surgir nas empresas de grande porte, o desenvolvimento de soluções de bancos de dados livres de certas estruturas e regras presentes no modelo relacional, e que poderiam ganhar em performance flexibilizando-se para as particularidades de cada organização. Estas soluções passaram a ficar conhecidas como NoSQL, que é uma abreviação de *Not Only SQL* (não apenas SQL), e utilizadas principalmente quando um SGBD do modelo relacional não apresentava a performance adequada.

O propósito das soluções NoSQL não é substituir o modelo relacional, e sim ser utilizado em casos em que se necessite de uma maior flexibilidade da estruturação da base de dados (CHANG *et al*, 2010).

4.2.1 Características dos sistemas NoSQL

Os sistemas de bancos de dados NoSQL, apresentam algumas características que os diferenciam dos sistemas baseados no modelo relacional. A seguir são apresentadas algumas destas características.

4.2.1.1 Escalabilidade horizontal

Escalabilidade é a característica de um sistema que pode continuar a servir um grande número de requisições com pouca perda de performance.

Existem duas maneiras de se alcançar a escalabilidade em um sistema. A primeira, chamada de escalabilidade vertical, é adicionar ou aumentar os recursos dos servidores já existentes. Porém esta é uma solução cara e, geralmente, de tecnologia proprietária. A outra solução é simplesmente adicionar mais servidores para o armazenamento e o processamento de dados. É chamada de escalabilidade horizontal (HEWITT, 2011).

O advento do Big Data e a necessidade de processamento paralelo em larga escala para manipular os dados levou à adoção generalizada de infraestruturas escaláveis horizontalmente. Em grandes corporações, como Google, Facebook e Yahoo, estas infraestruturas envolvem um número muito grande (centenas de milhares) de servidores, processando dados em tempo real (TIWARI, 2011).

4.2.1.2 Alta disponibilidade

Disponibilidade é a característica de um sistema resistente a falhas de qualquer natureza (*hardware*, *software*, energia), e tem como objetivo manter os serviços disponíveis o máximo de tempo possível.

Nos bancos de dados NoSQL, os dados são distribuídos nos vários servidores do sistema, propiciando que um maior número de solicitações sejam atendidas e que o sistema fique menos tempo não-disponível.

4.2.1.3 Ausência de esquema ou esquema flexível

Uma característica marcante dos SGBD's NoSQL é a ausência quase completa de um esquema que define a estrutura dos dados modelados. Esta ausência facilita a escalabilidade e contribui para uma maior disponibilidade. Porém, não existem garantias de integridade dos dados, o que ocorre nos sistemas relacionais devido à sua estrutura rígida.

Em sua maioria, as estruturas são baseadas em um conceito de chave-valor, o que permite uma alta flexibilidade na forma como os dados são organizados.

4.2.2 Modelos de dados NoSQL

Segundo Chodorow (2013), os bancos de dados NoSQL atualmente podem ser agrupados em quatro modelos principais: chave-valor, orientado a colunas, orientado a documentos, orientado a grafos.

A seguir são apresentadas algumas características destes modelos.

4.2.2.1 Modelo chave-valor

Os dados são armazenados como pares chave-valor e são recuperados através da chave.

Este modelo é considerado bastante simples e permite a visualização do banco de dados como uma grande tabela. Por ser de fácil implementação, permite que os dados sejam rapidamente acessados pela chave, através de operações simples como *get()* e *set()*, que permitem recuperar e definir valores, respectivamente.

A desvantagem deste modelo é não ser possível recuperar informações através de consultas mais complexas.

Como exemplos de bancos de dados deste modelo, existem o Amazon Dynamo, o Redis e o Riak.

4.2.2.2 Modelo orientado a colunas

Os bancos de dados deste modelo organizam os dados em tabelas, similarmente a um SGBD relacional, porém armazenando o conteúdo por colunas ao invés de linhas.

Este modelo permite que os dados sejam armazenados de forma eficiente, impedindo consumo de espaço quando um valor for nulo, simplesmente não o

gravando. Cada unidade do dado pode ser pensado como um conjunto de pares chave-valor, onde a unidade é identificada por um par chave-valor identificador, comumente chamado de chave primária (TIWARI, 2011).

Alguns exemplos de bancos de dados orientados a colunas são Hbase, Cassandra e Hypertable.

4.2.2.3 Modelo orientado a documentos

Segundo Tiwari (2011), um banco de dados orientado a documentos substitui o conceito de “linha” dos modelos relacionais com um modelo mais flexível: o “documento”. Ao permitir incorporar documentos e matrizes, é possível representar estruturas hierárquicas complexas em um único registro. Os dados são armazenados e organizados como uma coleção de documentos, que são flexíveis: cada documento pode ter qualquer número de campos.

Também não existem esquemas pré-definidos: as chaves e valores de uma tabela não tem tipos ou tamanhos fixos. Sem um esquema fixo, adicionar e remover campos se torna fácil. Geralmente isso torna o desenvolvimento mais rápido e também mais fácil de experimentar. Os desenvolvedores podem testar dezenas de modelos para os dados e escolher o melhor para prosseguir (CHODOROW, 2013).

Os principais SGDB's deste modelo são o Apache CouchDB e o MongoDB (TIWARI, 2011).

4.2.2.4 Modelo orientado a grafos

Estes bancos de dados aplicam a teoria dos grafos para armazenar e recuperar dados. Eles se concentram na interligação das diferentes partes de dados. Unidades de dados são visualizados como nós e as relações entre eles são definido pelas arestas que ligam os nós. Neo4j é um exemplo de banco de dados orientado a grafos.

4.2.3 Implementações de sistemas NoSQL

Uma das primeiras implementações de um sistema não-relacional foi o BigTable da Google. Lançado em 2004, esse banco de dados proprietário tinha o objetivo de promover maior escalabilidade e disponibilidade, além de flexibilizar a forte estruturação utilizada no modelo relacional.

Outras implementações surgiram também em outras empresas, como o Dynamo, da Amazon em 2007 e o Cassandra, desenvolvido pelo Facebook em 2008. O Cassandra foi projetado para lidar com grandes volumes de dados e licenciado sob código livre. Em 2010, passou a ser também o banco de dados do Twitter, no lugar do tradicional MySQL.

Além do Cassandra, diversos outros projetos NoSQL de código livre estão disponíveis, como por exemplo o Apache CouchDB, o MongoDB, HyperTable e Hbase. A seguir são apresentadas mais características sobre os sistemas Cassandra, Apache CouchDB e MongoDB.

4.2.3.1 Cassandra

O Cassandra é um banco de dados orientado a colunas, de código livre, distribuído, descentralizado, escalável e de alta disponibilidade, baseado no Amazon Dynamo e no Google BigTable. Foi criado pelo Facebook e hoje é utilizado em alguns dos maiores sites da *web*.

É um sistema distribuído, pois tem a capacidade de rodar em diversas máquinas enquanto que pareça um processo único para o usuário, e descentralizado, pois não possui um único ponto de falha.

O Cassandra utiliza um tipo especial de escalabilidade horizontal, chamado de “escalabilidade elástica”, que significa que um *cluster* pode tanto ser escalável para cima como para baixo. Para fazer isso, o *cluster* deve ser capaz de aceitar remover e adicionar novos nós que podem começar a participar obtendo uma cópia de alguns ou todos os dados e servir novas requisições, sem impactar no sistema como um todo.

Segundo Hewitt (2011), a disponibilidade de um sistema é mensurada de

acordo com a capacidade de executar requisições. Porém, computadores podem falhar de diversas maneiras diferentes, desde problemas em seus componentes até conflitos na rede. Então, para um sistema ser altamente disponível, ele geralmente deve conter múltiplos computadores interligados por rede, e um *software* capaz de operar em um *cluster* e ter a facilidade de reconhecer nós com problema e redirecionar as requisições para outra parte do sistema. Com o Cassandra é possível substituir nós sem parada no serviço e replicar os dados em múltiplos *data centers* para oferecer melhor performance local e prevenir paradas devido a catástrofes em um *data center*.

4.2.3.2 MongoDB

MongoDB é um sistema de banco de dados orientado a documentos, poderoso, flexível e escalável, que combina a capacidade de dimensionamento com características como índices secundários, consultas, classificação e agregações (CHODOROW, 2013).

Os documentos no MongoDB são do tipo JSON (*JavaScript Object Notation*), que é uma representação simples de dados, compreensível para seres humanos e de fácil interpretação para máquinas. JSON utiliza formato texto e baseia-se em convenções familiares a muitas linguagens de programação, o que faz com que JSON seja um formato ideal de troca de dados (JSON, 2013). A figura 3 mostra a estrutura de um documento JSON.

Figura 3 - Estrutura de um documento JSON

Fonte: Modificado de Chodorow (2013)

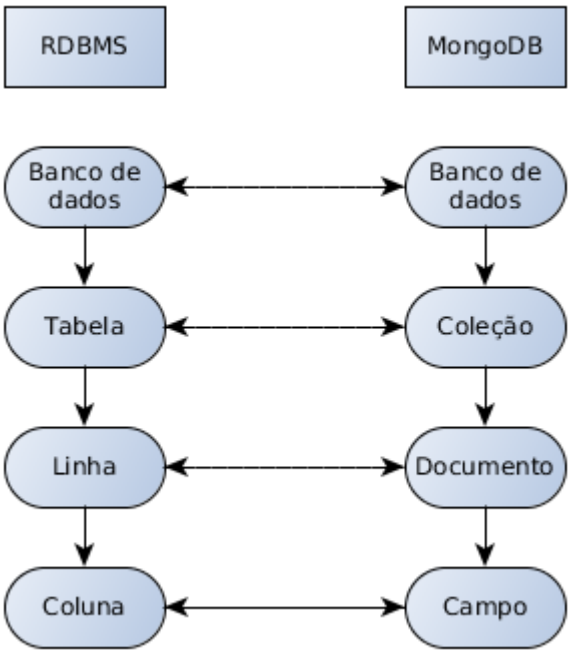
Um servidor MongoDB pode conter inúmeros bancos de dados, ou *databases*, e cada banco de dados atua como um conjunto de dados que são independentes um do outro. Uma *database* MongoDB contêm uma ou mais coleções, como por exemplo uma base de dados para um sistema de *blog* geralmente é formada por coleções de artigos, autores, comentários e categorias.

Uma coleção é um conjunto de documentos, e pode ser entendida como uma tabela em um banco de dados relacional. Porém, diferentemente de uma tabela tradicional, não é necessário definir a estrutura dos dados que serão armazenados na coleção previamente.

Um documento contêm um conjunto de campos ou pares chaves-valor. As chaves são *strings* e os valores podem ser de vários tipos: *strings*, números, datas e diversos outros. É possível também armazenar um documento como valor de um campo em outro documento.

Na figura 4 é mostrada uma comparação dos componentes do MongoDB com bancos de dados relacionais, que ajuda no entendimento da estrutura deste SGDB (ISLAM, 2011).

Figura 4 – Componentes do MongoDB



Fonte: Modificado de Islam (2011)

Mongo é uma excelente escolha para projetos *web* que dispõe de um pequeno orçamento e que não podem comprar uma solução de *hardware*. Devido à sua falta de esquema definido, o banco de dados Mongo pode crescer e mudar de acordo com o modelo de dados, utilizando-se da escalabilidade horizontal para isso (REDMOND; WILSON, 2013).

4.2.3.3 Apache CouchDB

Lançado em 2005, o Apache CouchDB é um banco de dados de código aberto, orientado a documentos, e, assim como o MongoDB, armazena os dados em documentos JSON. O acesso aos dados é feito através de requisições HTTP e também através de consultas.

Este sistema visa atender diferentes problemas, de diferentes tamanhos, e suporta diversos cenários de desenvolvimento, desde grandes *datacenters* até um *smartphone*, pois é flexível na estruturação e distribuição dos dados (REDMOND; WILSON, 2013).

5 PROJETO DA SOLUÇÃO DESENVOLVIDA

Neste capítulo são apresentados os artefatos, documentos e técnicas que foram utilizados na implementação da solução proposta.

O presente trabalho tem, como uma de suas justificativas, fazer parte de um ecossistema para mineração de dados em Big Data (como citado no item 1.2). Este ecossistema será formado por três módulos principais: aquisição, processamento e análise.

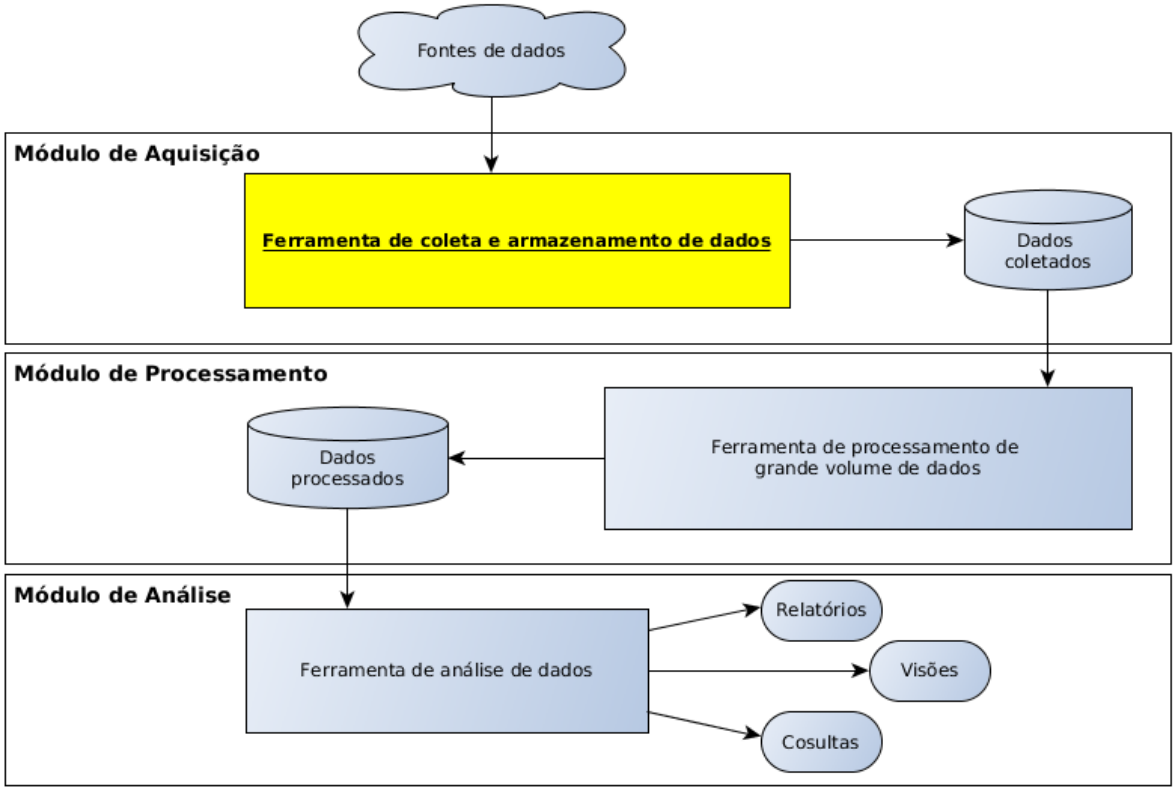
O módulo de aquisição é o responsável pela obtenção e armazenamento dos dados. Este foi o módulo desenvolvido e será detalhado no decorrer deste trabalho.

O módulo de processamento deverá conter uma ferramenta que prepare os dados armazenados pelo módulo de aquisição para serem utilizados no módulo de análise. Essa preparação de dados pode abranger atividades como remover tipos de palavras dos textos, como preposições e artigos, encontrar e agrupar termos semelhantes e classificar os registros de acordo com seu contexto.

E o módulo de análise deverá aplicar algoritmos e técnicas de mineração nos dados processados, possibilitando a geração de conhecimento a partir deste grande volume de dados obtidos.

A imagem a seguir demonstra esta estrutura e destaca o papel da ferramenta proposta neste trabalho.

Figura 5 – Ecossistema para mineração de dados em Big Data



Fonte: Elaborado pelo autor.

A proposta de arquitetura de sistema de mineração de dados apresentado na figura 5 tem o objetivo de servir como base para a implementação dos demais módulos deste sistema, uma vez que as etapas de processamento e análise não fazem parte dos objetivos do presente trabalho.

A seguir são apresentadas as definições das tecnologias que foram utilizadas para a implementação do trabalho.

5.1 Tecnologias utilizadas

Para o desenvolvimento da solução, diferentes tecnologias foram empregadas em cada parte do sistema.

Os coletores de dados foram implementados na linguagem de programação PHP, pois é uma linguagem voltada para aplicações *web*, e armazenam os dados coletados em um servidor de banco de dados orientado a documentos. O SGDB

escolhido para isso foi o MongoDB, principalmente por três motivos, citados por Islam (2011):

- implementa a ideia de esquema flexível. Não é preciso definir uma estrutura de dados antes de se começar a armazenar informações, o que o torna adequado para armazenar dados não-estruturados;

- é altamente escalável. O MongoDB vem com muitos recursos para ajudar a manter um bom desempenho, enquanto o tráfego e o volume de dados cresce, com pouca ou nenhuma alteração na aplicação;

- é de fácil aprendizagem por ter muitos conceitos parecidos com bancos de dados relacionais.

A escolha do MongoDB também deve-se ao fato de ser livre para uso e não requerer um *hardware* específico para seu funcionamento, facilitando assim o desenvolvimento de demais módulos e ferramentas de análise, que utilizarão este banco de dados.

Cada coletor é um programa independente, utilizando recursos específicos conforme sua finalidade.

Dentre alguns recursos que foram utilizados pelos coletores, pode-se citar as API's (*Application Programming Interface*, em português “Interface de Programação de Aplicativos”) do Twitter, do Facebook e do Google Plus. Através destas interfaces, os coletores acessam e coletam dados destas redes sociais.

O módulo de configuração utiliza *a* biblioteca Sencha ExtJS, escrita em *javascript*[2]*,* para as interfaces de interação com o usuário. A escolha desta biblioteca para a parte gráfica da aplicação deve-se ao fato dela prover uma interface bastante amigável ao usuário, além de haver muita documentação disponível. Com mais de 100 exemplos, centenas de componentes, uma suíte de documentação completa e construído em temas, o ExtJS fornece as ferramentas necessárias para construir aplicações robustas (SENCHA, 2013).

Os dados do módulo de configuração, são armazenados em um banco de dados PostgreSQL. A escolha por este *software* se deve por ser o PostgreSQL um SGDB que pode ser usado livremente em qualquer aplicação, além de ser reconhecidamente estável e robusto.

2 Linguagem de programação executada pelo lado do cliente, no próprio navegador. Ou seja, através do javascript podem ser executados códigos sem a necessidade de comunicação com o servidor.

5.2 Descrição do sistema

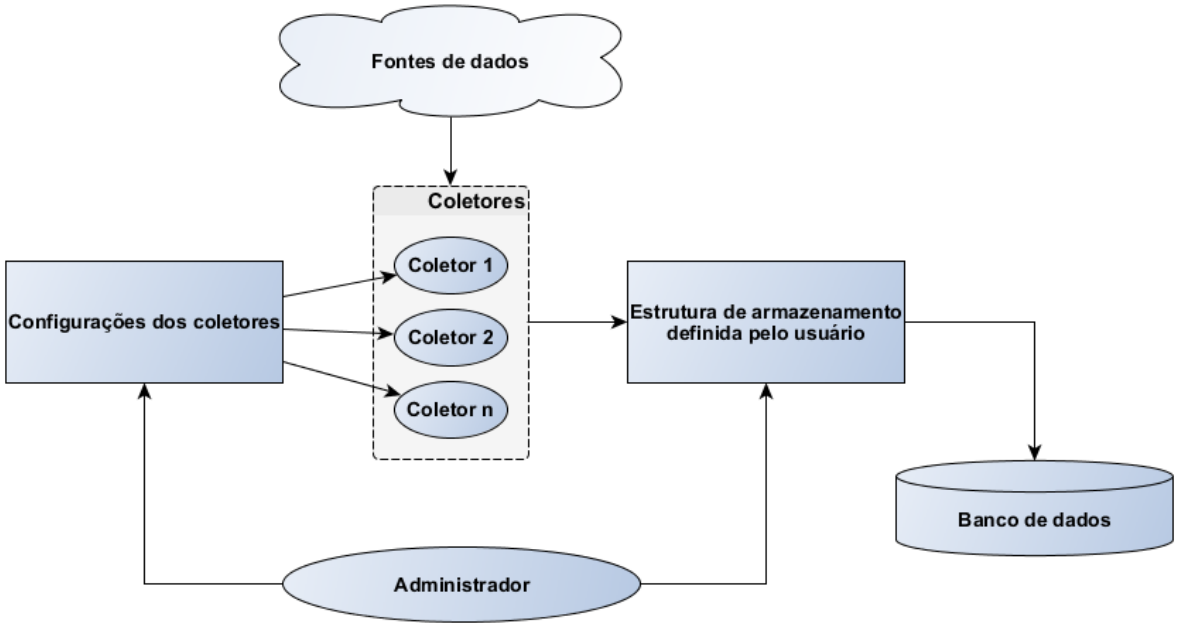
O sistema proposto é composto por dois módulos: configuração e coleta.

Através do módulo de configuração é possível, por meio de uma interface gráfica, configurar e definir diferentes coletores de dados e os parâmetros para a execução destes, além de definir a estrutura de armazenamento dos dados coletados pelos coletores. Neste módulo também é possível cadastrar usuários para a utilização do programa e acessar uma tela para efetuar consultas nos dados.

O módulo de coleta é o responsável pela execução dos coletores. Cada coletor é um processo isolado, ou seja, pode haver mais de um coletor do mesmo tipo coletando dados ao mesmo tempo. Os coletores armazenam as informações obtidas conforme as configurações definidas pelo usuário.

A figura 6 ilustra um esboço da estrutura do sistema descrito.

Figura 6 – Esboço da estrutura do sistema proposto



Fonte: Elaborado pelo autor.

A seguir são apresentados os requisitos da aplicação.

5.3 Requisitos da aplicação

Este item tem o objetivo de descrever os requisitos funcionais e não funcionais desejáveis para o desenvolvimento do trabalho proposto.

5.3.1 Requisitos funcionais

Os requisitos funcionais, que são as funcionalidade esperadas do sistema, estão classificados em três níveis de prioridade: obrigatório, importante e desejável. Estes níveis serão representados pelas letras 'O', 'I', e 'D', respectivamente.

A seguir, é apresentada a lista de requisitos funcionais do sistema proposto:

Tabela 1 – Lista de requisitos funcionais

Código	Descrição do requisito	Prioridade
RF001	Configurar estrutura do banco de dados	O
RF002	Configurar comportamento dos coletores	O
RF003	Iniciar a coleta de dados	O
RF004	Interromper a coleta de dados	O
RF005	Cadastrar usuários	I
RF006	Consultar dados coletados	D

Fonte: Elaborado pelo autor

RF001 (Obrigatório) – Configurar estrutura do banco de dados: através deste requisito, o usuário poderá construir a estrutura desejada para o banco de dados.

RF002 (Obrigatório) – Configurar comportamento dos coletores: através deste requisito, o usuário poderá configurar os parâmetros de cada coletor, informando dados como: o que coletar, onde armazenar e durante qual período coletar.

RF003 (Obrigatório) – Iniciar a coleta de dados: através deste requisito, o usuário por meio de uma ação, informa quando iniciar a execução dos coletores.

RF004 (Obrigatório) – Interromper a coleta de dados: através deste requisito, o usuário por meio de uma ação, informa que deseja parar a execução dos coletores.

RF005 (Importante) – Cadastrar usuários: através deste requisito, o usuário

pode cadastrar outros usuários para utilizarem o sistema.

RF006 (Desejável) – Consultar dados coletados: através deste requisito, o usuário pode realizar consultas nos dados armazenados na base.

5.3.2 Requisitos não funcionais

Os requisitos não funcionais são aqueles que caracterizam o meio-ambiente do sistema, definindo suas características e restrições. Assim como os requisitos funcionais, os níveis de prioridade são classificados como obrigatório, importante e desejável. A seguir são apresentados os requisitos não funcionais do sistema proposto:

Tabela 2 – Lista de requisitos não funcionais

Código	Descrição do requisito	Prioridade
RNF001	Estar disponível 24h	D
RNF002	Ser desenvolvido na linguagem de programação PHP	D
RNF003	Utilizar SGBD MongoDB para grande volume de dados	D
RNF004	Utilizar SGBD PostgreSQL para dados da aplicação	D

Fonte: Elaborado pelo autor

RNF001 – Estar disponível 24h: através deste requisito, é desejável que a aplicação seja acessível a qualquer momento, sem sofrer interrupções ou paradas.

RNF002 – Ser desenvolvido na linguagem de programação PHP: este requisito define qual a linguagem a ser utilizada na escrita do programa.

RNF003 – Utilizar SGBD MongoDB para grande volume de dados: este requisito define o sistema gerenciador de banco de dados a ser utilizado para o armazenamento dos dados obtidos pelos coletores.

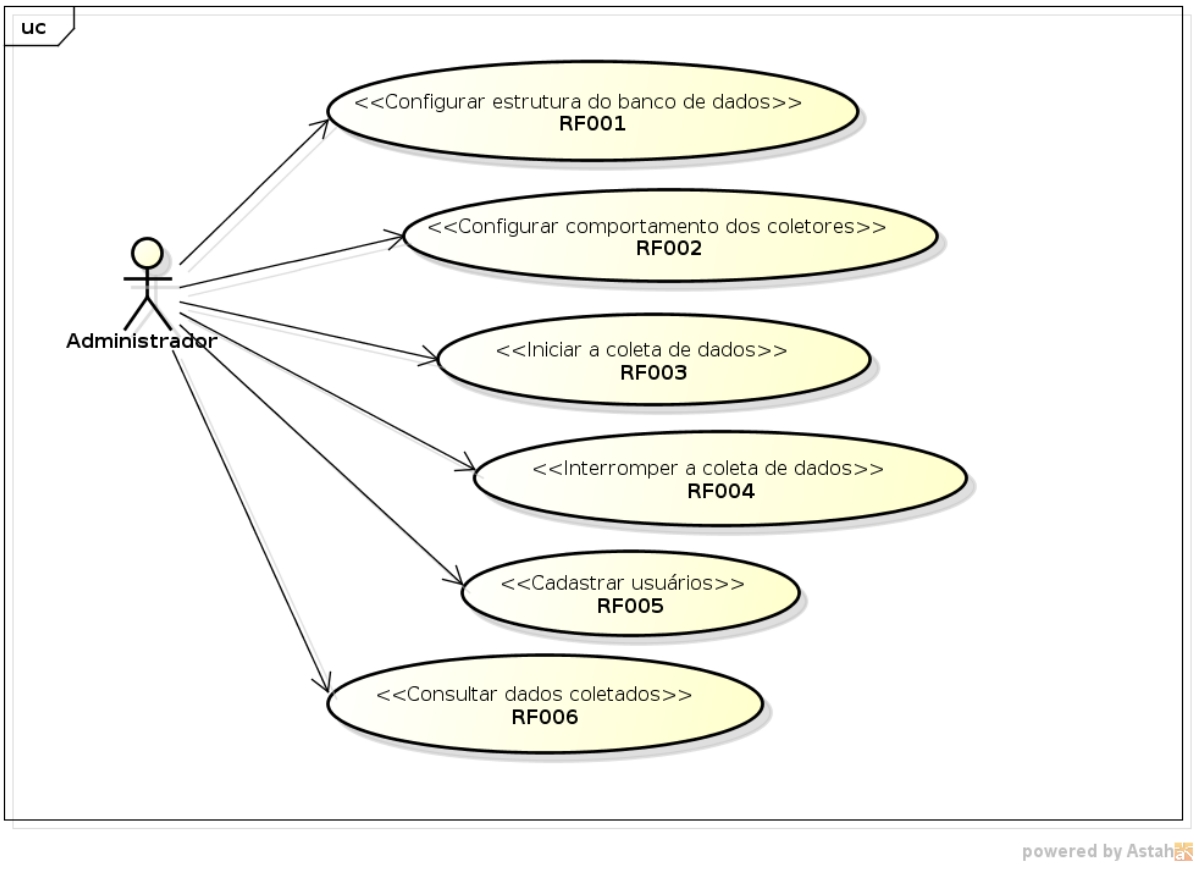
RNF004 – Utilizar SGBD PostgreSQL para dados da aplicação: este requisito define o sistema gerenciador de banco de dados a ser utilizado para armazenar os dados de configurações, usuários e histórico.

5.4 Casos de uso

Os casos de uso de um sistema são a definição de qual requisito cada papel deve executar para que o sistema alcance seus objetivos.

No caso do sistema proposto, existe apenas o papel do administrador, que executa todas as tarefas. No diagrama de casos de uso apresentado na figura 7, pode-se verificar as ações de cada papel.

Figura 7 – Diagrama de casos de uso

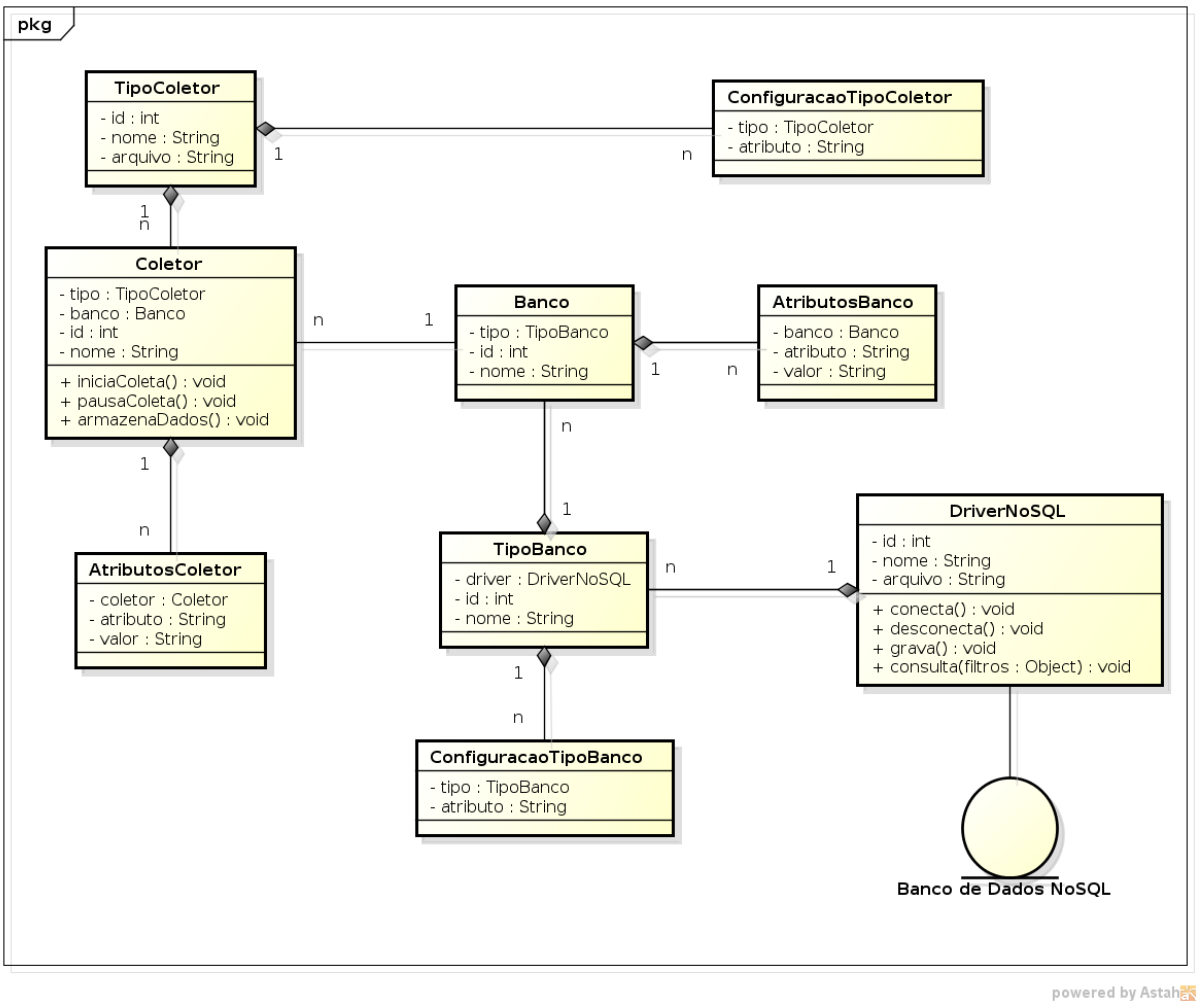


Fonte: Elaborado pelo autor

5.5 Diagrama de classes

O diagrama de classes (figura 8) fornece uma visão geral sobre todo o sistema e é a base para a construção dos demais diagramas. Este diagrama apresenta as classes que farão parte do projeto e as relações entre elas.

Figura 8 – Diagrama de classes



Fonte: Elaborado pelo autor.

Nas tabelas a seguir, são detalhados os atributos e métodos de cada classe do modelo apresentado no diagrama.

Tabela 3 – Classe TipoColetor

TipoColetor	
Esta classe representa um tipo de coletor disponível na ferramenta. O usuário irá utilizar-se desta classe para adicionar novos tipos de coletores ou remover tipos de coletores já existentes. Cada tipo de coletor terá um programa associado que realizará a coleta de dados.	
Atributos	
id	Número identificador do tipo de coletor.
nome	Nome do tipo do coletor.
arquivo	Caminho do arquivo ou um texto com o script PHP que será executado ao iniciar a coleta.

Fonte: Elaborado pelo autor

Tabela 4 – Classe ConfiguracaoTipoColetor

ConfiguracaoTipoColetor	
Esta classe define os atributos que um tipo de coletor utiliza na execução do programa de coleta.	
Atributos	
tipo	Tipo de coletor associado ao atributo.
atributo	Nome do atributo utilizado pelo tipo de coletor.

Fonte: Elaborado pelo autor

Tabela 5 – Classe Coletor

Coletor	
Esta classe representa um coletor de dados, que tem um tipo e um banco de dados associado.	
Atributos	
tipo	Define de qual tipo é o coletor
banco	Define em qual banco de dados o coletor deve armazenar os dados coletados.
id	Número identificador do coletor.
nome	Nome do coletor.
Métodos	
iniciaColeta()	Este método é reponsável por iniciar o programa de coleta de dados.
pausaColeta()	Este método é responsável por interromper o programa de coleta de dados.
armazenaDados()	Através deste método, o programa coletor irá armazenar os dados coletados no banco de dados.

Fonte: Elaborado pelo autor

Tabela 6 – Classe AtributosColetor

AtributosColetor	
Esta classe representa um atributo de um coletor de dados.	
Atributos	
coletor	Define de qual coletor é este atributo.
atributo	Define qual é o atributo representado por esta classe.
valor	Valor do atributo desta classe.

Fonte: Elaborado pelo autor

Tabela 7 – Classe DriverNoSQL

DriverNoSQL	
Esta classe é a responsável pela interação com o banco de dados NoSQL.	
Atributos	
id	Número identificador do driver.
nome	Nome do driver.
arquivo	Caminho do arquivo ou um texto com o script PHP que executa as interações com o banco de dados.
Métodos	
conecta()	Este método, utilizando os atributos do tipo do banco, é responsável pela conexão com o banco de dados. Em caso de sucesso, retorna um objeto de conexão.
desconecta()	Este método é responsável por encerrar uma conexão com o banco de dados. Retorna verdadeiro em caso de sucesso, e falso caso contrário.
grava()	Através deste método, o programa irá armazenar os dados enviados pelo coletor no banco de dados. Em caso de gravação com sucesso, retorna o identificador do registro salvo. Caso contrário, retorna nulo.
consulta()	Através deste método será possível executar buscas nos dados armazenados no banco de dados, informando filtros de pesquisa. Este método retorna os registros encontrados na consulta aos dados.

Fonte: Elaborado pelo autor

Tabela 8 – Classe TipoBanco

TipoBanco	
Esta classe representa um tipo de banco de dados disponível na ferramenta. Através desta classe pode-se ter diferentes sistemas de bancos de dados disponíveis para armazenamento dos dados coletados.	
Atributos	
id	Número identificador do tipo de banco de dados.
nome	Nome do tipo de banco de dados.
driver	Identificador do objeto responsável pela interação com o banco de dados.

Fonte: Elaborado pelo autor

Tabela 9 – Classe ConfiguracaoTipoBanco

ConfiguracaoTipoBanco	
Esta classe define os atributos que um tipo de banco de dados necessita para estruturar como irá armazenar as informações coletadas.	
Atributos	
tipo	Identificador do tipo de banco a que o atributo pertence.
atributo	Nome do atributo.

Fonte: Elaborado pelo autor

Tabela 10 – Classe Banco

Banco	
Esta classe representa um banco de dados NoSQL que será utilizado para armazenar as informações obtidas pelos coletores.	
Atributos	
tipo	Tipo de banco de dados deste banco.
id	Número identificador do banco de dados.
nome	Nome do banco de dados.

Fonte: Elaborado pelo autor

Tabela 11 – Classe AtributosBanco

AtributosBanco	
Esta classe define o valor dos atributos de um banco de dados. Através destes atributos, o driver do banco saberá a estrutura de armazenamento dos dados coletados.	
Atributos	
banco	Identificador do banco de dados a que o atributo pertence.
atributo	Identificador do atributo.
valor	Valor do atributo.

Fonte: Elaborado pelo autor

5.6 Captura de dados

Os dados armazenados através do presente trabalho primeiramente têm origem em três fontes: as redes sociais Twitter, Facebook e Google Plus.

Para coletar os dados destas fontes, foram desenvolvidos três coletores, um para cada fonte. Cada coletor é um processo, podendo ser executado em paralelo

aos demais. Assim é possível, por exemplo, rodar ao mesmo tempo dois ou mais coletores do Twitter, ao mesmo tempo que são executados dois ou mais coletores do Facebook e do Google Plus.

A arquitetura aberta da ferramenta suporta a criação de coletores para outras fontes de dados, que podem ser acoplados ao sistema do mesmo modo que os programas coletores desenvolvidos.

Em seguida são apresentados mais detalhes de cada um dos coletores.

5.6.1 Coletor de dados do Twitter

Twitter é um serviço de *microblogging*, de caráter social que permite postagens de mensagens com 140 caracteres ou menos. Essas mensagens são chamadas de *tweets*. O Twitter oferece API's, que permitem a coleta de dados de seus usuários e *tweets*. A documentação das API's oferecidas pelo Twitter podem ser encontradas no seguinte endereço: *https://dev.twitter.com*.

O coletor responsável pelas mensagens do Twitter irá utilizar-se da API que o mesmo disponibiliza para coleta de dados, utilizando-se das palavras-chave informadas pelo usuário para filtrar os resultados.

5.6.2 Coletor de dados do Facebook

O Facebook é um site e serviço de rede social que permite o compartilhamento de fotos, vídeos e áudios, além da troca de mensagens públicas e privadas entre os usuários. Cada postagem de um usuário é chamada de *post*.

O programa coletor de postagens do Facebook irá utilizar a API oficial da plataforma, disponibilizada no endereço *https://developers.facebook.com/*.

5.6.3 Coletor de dados do Google Plus

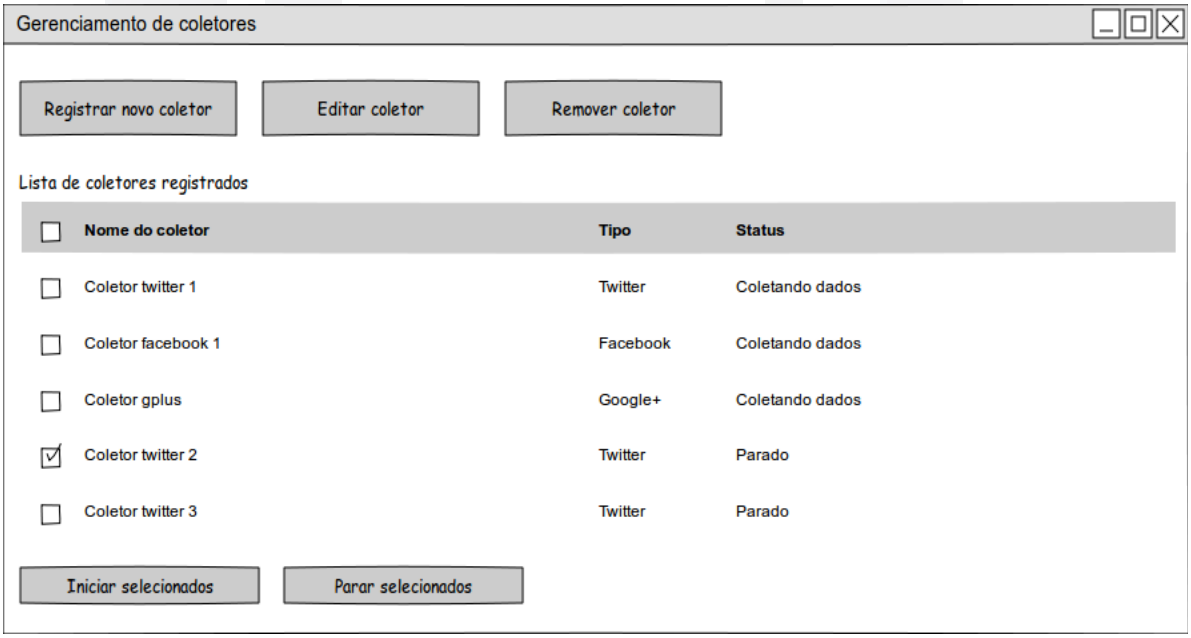
A rede social Google Plus possui vários recursos de interação entre os usuários, tais como: organização dos contatos em grupos, bate-papo, comunidades, compartilhamento de mídias e jogos multi-usuários.

Para efetuar a coleta dos dados desta fonte, o coletor utilizará a biblioteca de código aberto "*google-api-php-client*", pois não há uma API oficial do Google na linguagem PHP.

5.7 Interfaces do sistema

Nesta seção são apresentados os esboços de algumas telas principais do sistema proposto. Na figura 9 é apresentado o esboço da tela de gerenciamento de coletores.

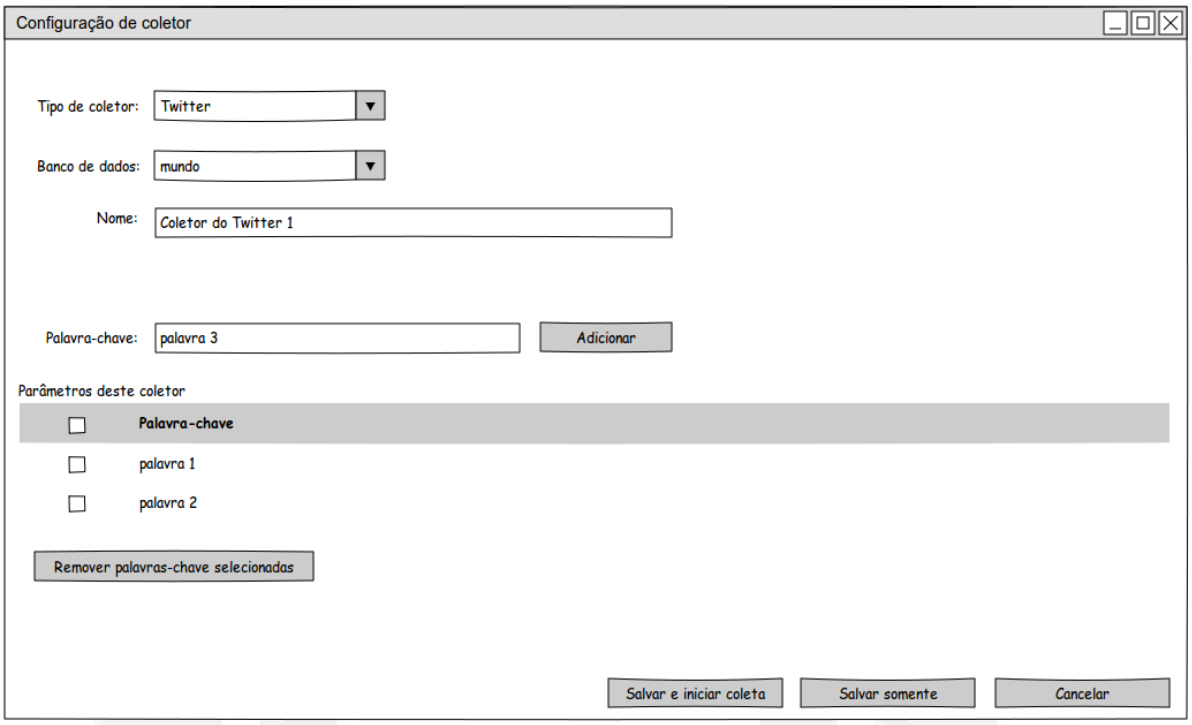
Figura 9 – Esboço da tela de gerenciamento de coletores.



Fonte: Elaborado pelo autor.

O esboço da tela de configuração de um coletor é visto na figura 10.

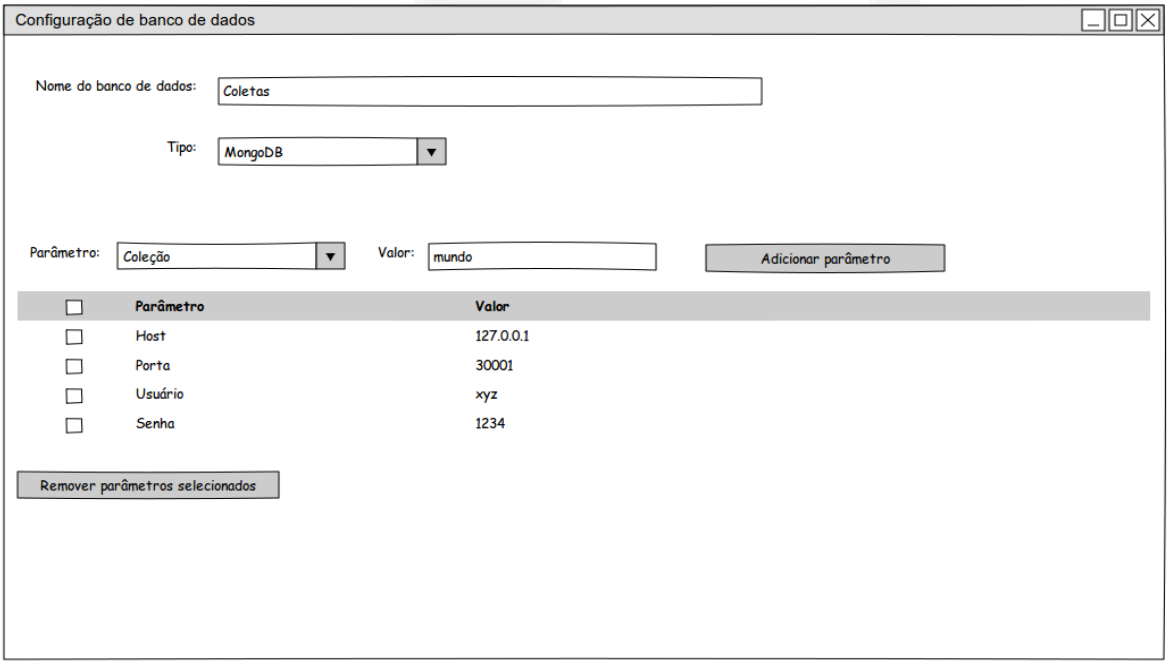
Figura 10 – Esboço da tela de configuração de um coletor.



Fonte: Elaborado pelo autor.

Na figura 11 é apresentada a interface de configuração de um banco de dados.

Figura 11 – Esboço da tela de configuração de banco de dados.



Fonte: Elaborado pelo autor.

6 DESENVOLVIMENTO

Nesta seção, é apresentado o desenvolvimento da ferramenta com base no projeto apresentado no capítulo anterior.

Conforme descrito no item 5.2 deste documento, o sistema foi implementado em dois módulos principais: coleta e configuração. A seguir é apresentado o desenvolvimento de cada um destes módulos e das partes que os compõem.

6.1 Módulo de coleta

O módulo de coleta é responsável por consultar as fontes de dados e armazenar os dados obtidos, conforme configurado pelo usuário. É composto por três programas principais, responsáveis, respectivamente, por obter dados de três fontes: Twitter, Facebook e Google Plus. Existe também um programa que é executado em segundo plano (processo) responsável por executar os programas de coleta paralelamente.

A seguir serão apresentados os programas coletores de cada uma das fontes de dados e o processo responsável pela execução dos mesmos.

6.1.1 Coletor para Twitter

O programa coletor para a rede social Twitter foi escrito na linguagem PHP e utiliza-se de uma API para interação com a fonte de dados. Esta API, que está na versão 1.1, é disponibilizada pelo próprio Twitter e a documentação de todos os métodos e funções existentes está disponível no endereço *https://dev.twitter.com/*.

6.1.1.1 Pré-requisitos

A API do Twitter exige uma autenticação OAuth (protocolo aberto que permite a autenticação segura em um método padrão e simples) para permitir a coleta de dados. Para possibilitar esta autenticação, é necessário criar uma aplicação no Twitter, o que deve ser feito no endereço *https://apps.twitter.com/*.

6.1.1.2 Recursos utilizados

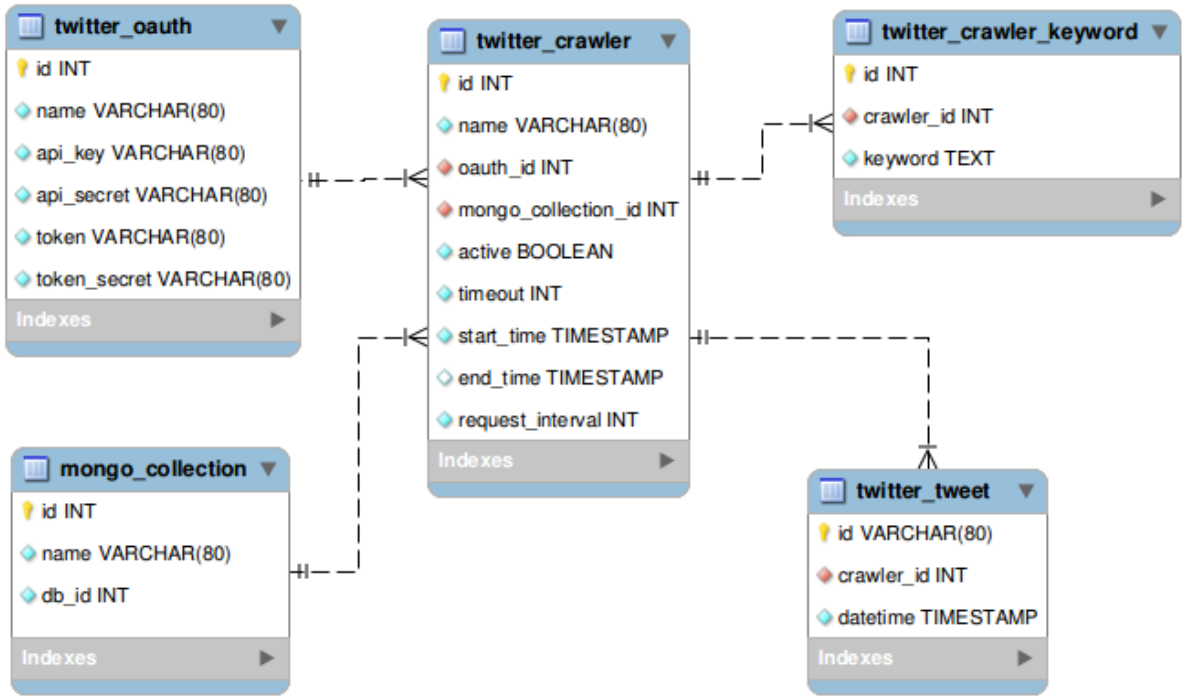
Para efetuar a autenticação na API via OAuth, o programa coletor do Twitter utiliza-se da biblioteca "*twitteroauth.php*" de autoria de Abraham Williams (abraham@abrah.am), disponível em *https://github.com/abraham/twitteroauth*.

Os demais métodos do programa, como acesso aos bancos de dados PostgreSQL e MongoDB, foram desenvolvidos pelo autor, utilizando a linguagem PHP.

6.1.1.3 Modelo de dados – Configuração

O coletor do Twitter se baseia nas configurações cadastradas pelo usuário para coletar os dados da fonte. A seguir é apresentado o modelo ER (Entidade Relacionamento) utilizado para armazenar a configuração dos coletores.

Figura 12 – Modelo ER do coletor de dados do Twitter



Fonte: Elaborado pelo autor

Os dados para autenticação na API são cadastrados na tabela “twitter_oauth”, que é referenciada pela coluna “oauth_id” da tabela “twitter_crawler”, que é a tabela de configuração principal do coletor. Nesta tabela também é informada a coleção de dados na qual o coletor irá gravar os dados coletados, através da coluna “mongo_collection_id”, que referencia a tabela “mongo_collection”.

Na tabela “twitter_crawler_keyword” são armazenadas as palavras-chave que o coletor irá utilizar para filtrar os dados a serem obtidos da fonte. E a tabela “twitter_tweet” armazena o identificador e a data em que o *tweet* foi coletado, e serve como controle para o programa não armazenar a mesma informação mais de uma vez.

6.1.1.4 Modelo de dados – Armazenamento

Os dados coletados pelo programa de coleta do Twitter são armazenados em uma coleção de um banco de dados de um servidor MongoDB. Estes dados são, individualmente, *tweets*, que são as postagens dos usuários na rede social Twitter.

Cada *tweet* é composto de uma mensagem principal de até 140 caracteres e de dados adicionais, como usuário, localização e horário da atualização. A próxima tabela mostra um exemplo da estrutura de um *tweet*, listando algumas de suas principais propriedades:

Tabela 12 – Estrutura de um *tweet*

Campo	Descrição	Tipo
id	O identificador do post	numérico
id_str	O mesmo que “id”, porém de tipo texto	texto
text	Texto principal do tweet	texto
coordinates	As coordenadas (latitude e longitude) de onde ocorreu o tweet	texto
created_at	Horário de criação do tweet	data/hora
hashtags	Textos marcados com o caractere '#' presentes no texto do tweet	objeto
in_reply_to_user_d_str	Identificador do usuário ao qual o tweet foi respondido (se for um tweet do tipo 'reply')	texto
retweet_count	Contagem de vezes em que o tweet foi 'retuitado'	numérico
retweeted	Se o tweet foi ou não 'retuitado'	boolean
user	Usuário que efetuou o tweet	objeto
user.favourites_count	Contagem de vezes em que o usuário foi marcado como 'favorito'	numérico
user.followers_count	Contagem de seguidores do usuário	numérico
user.description	Descrição do usuário	texto
user.friend_count	Contagem de amigos do usuário	numérico
user.screen_name	Nome do usuário como é mostrado no Twitter	texto
user.lang	Linguagem do usuário	texto
source	Origem do tweet (web, android, ios)	texto

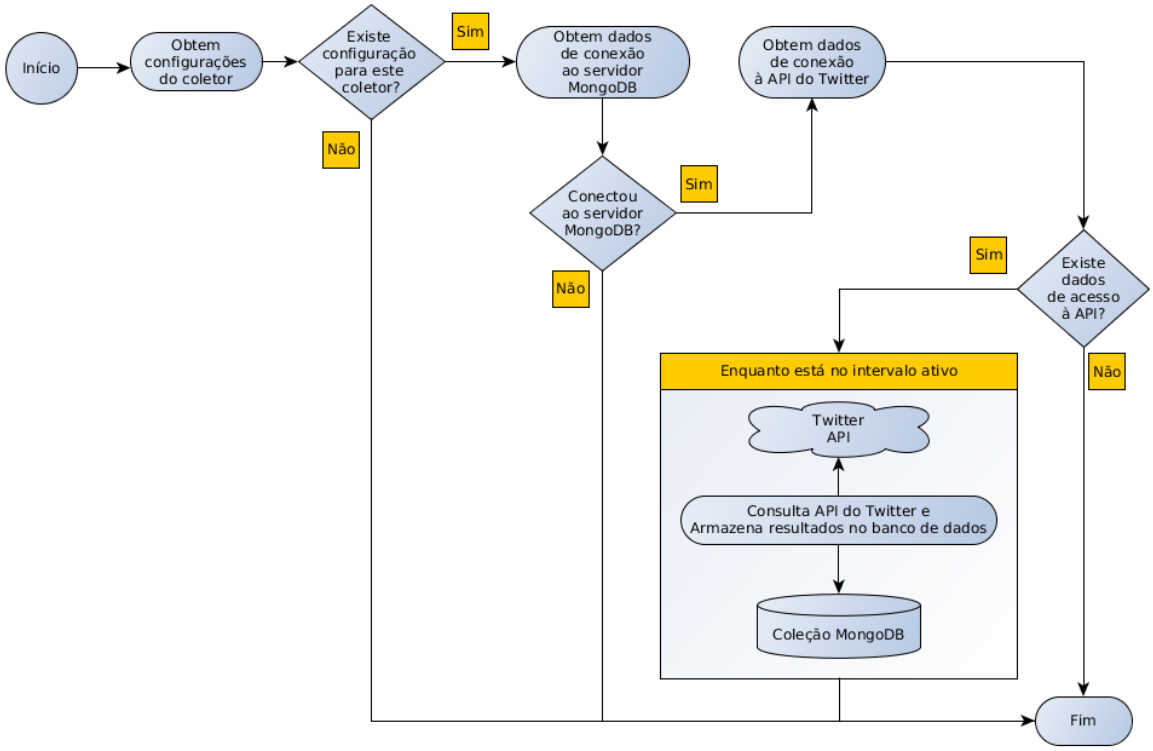
Fonte: https://dev.twitter.com/docs/api/1.1

6.1.1.5 Estrutura do coletor

O coletor do Twitter é um programa que basicamente lê as configurações definidas pelo usuário, consulta a API do Twitter utilizando a autenticação cadastrada

e grava os dados obtidos em uma base de dados. A figura a seguir ilustra o fluxo de funcionamento do programa.

Figura 13 – Fluxo do programa coletor para Twitter



Fonte: Elaborado pelo autor.

O programa recebe como parâmetro inicial o código do coletor que foi cadastrado pelo usuário. Se for passado como parâmetro o código de um coletor que não estiver cadastrado, o programa é finalizado. Caso contrário o código é utilizado para carregar as configurações de conexão ao servidor MongoDB. Com estes dados, é feita uma tentativa de conexão ao servidor. Em caso de não houver sucesso nesta conexão, o programa é finalizado, caso contrário são carregadas as informações de acesso à API. Se não houverem dados para este acesso, o programa é finalizado. Se existirem, o programa verifica se o horário está entre o intervalo cadastrado pelo usuário para o funcionamento deste coletor.

Se o horário em que o coletor estiver sendo executado estiver no intervalo ativo definido pelo usuário, o programa utilizará os dados de consulta, como as palavras-chave, para efetuar consultas à API do Twitter. O resultado destas

consultas são armazenados na coleção definida pelo usuário para este coletor. Estas consultas são realizadas em um intervalo de tempo definido pelo usuário (não menor do que um minuto), enquanto o programa estiver no intervalo ativo do coletor. Assim que o horário não estiver mais entre o intervalo ativo do coletor, o programa é finalizado.

6.1.2 Coletor para Facebook

O programa coletor para a rede social Facebook foi escrito, assim como o coletor para Twitter, na linguagem PHP e utiliza-se de uma API para interação com a fonte de dados. Esta API utilizada é disponibilizada pelo Facebook e a documentação de todos os métodos e funções existentes está disponível no endereço *https://developers.facebook.com/*.

6.1.2.1 Pré-requisitos

A API oferecida pelo Facebook exige um código de aplicação (*App ID*) e uma chave de acesso (*App Secret*) para efetuar as consultas nas postagens públicas da rede social. Para obter estas informações, é necessário criar uma aplicação no Facebook, através do endereço *https://developers.facebook.com/*.

6.1.2.2 Recursos utilizados

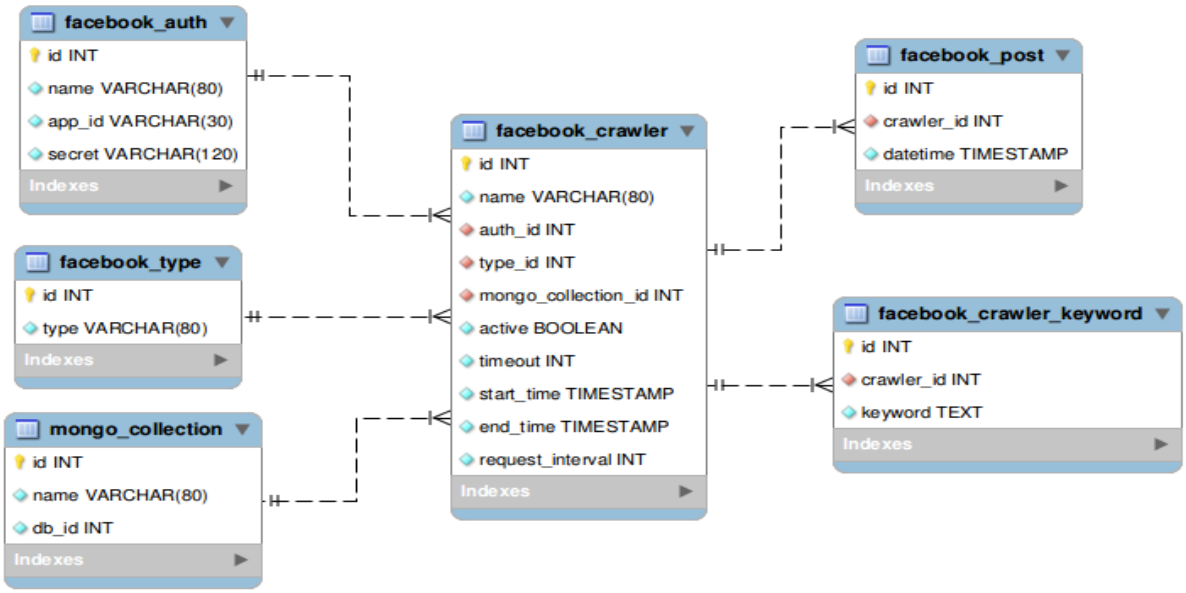
Para o desenvolvimento do programa de coleta para Facebook, foi utilizada a API "*facebook-php-sdk-master*" fornecida pelo próprio Facebook, disponível no endereço *https://developers.facebook.com*. Com esta ferramenta é possível consultar dados desta rede social através do código e chave da aplicação, citados no item anterior.

Os demais métodos do programa, como acesso aos bancos de dados PostgreSQL e MongoDB, foram desenvolvidos pelo autor, utilizando a linguagem PHP.

6.1.2.3 Modelo de dados – Configuração

O coletor do Facebook baseia-se nas configurações cadastradas pelo usuário para coletar os dados da fonte. A seguir é apresentado o modelo ER utilizado para armazenar a configuração dos coletores.

Figura 14 – Modelo ER do coletor de dados do Facebook



Fonte: Elaborado pelo autor

Os dados de autenticação na API são armazenados na tabela “facebook_auth”. Em “facebook_type” se encontram os tipos de registros possíveis de obtenção pela API, tais como *posts*, imagens e vídeos. Estas tabelas são referenciadas, respectivamente, pelas colunas “auth_id” e “type_id” da tabela “facebook_crawler”, que é a tabela central dos dados do coletor. Nela encontra-se também uma referência à uma coleção MongoDB, na qual serão armazenados os dados coletados por este programa.

Na tabela “facebook_crawler_keyword” são armazenadas as palavras-chave que o coletor irá utilizar para filtrar os dados a serem obtidos da fonte. E a tabela “facebook_post” armazena o identificador e a data em que o *post* foi coletado, e serve como controle para o programa não armazenar a mesma informação mais de uma vez.

6.1.2.4 Modelo de dados – Armazenamento

Os dados coletados pelo programa de coleta do Facebook são armazenados em uma coleção de um banco de dados de um servidor MongoDB, com a mesma estrutura como são coletados.

Um *post* do facebook, obtido através da API do Facebook, tem a seguinte estrutura de campos:

Tabela 13 – Estrutura de um *post* do Facebook

Campo	Descrição	Tipo
id	O identificador do post	texto
caption	Título de um link no post	texto
created_time	O horário em que o post foi inicialmente publicado	data/hora
description	A descrição de um link no post	texto
from	Informações sobre o perfil que postou a mensagem	objeto
icon	Um link para um ícone representando o tipo do post	texto
is_hidden	Se o post está marcado como oculto	boolean
link	O link anexado no post	texto
message	A mensagem de status no post	texto
message_tags	Perfis marcados na mensagem	objeto
name	O nome do link	texto
object_id	O identificador de qualquer foto ou vídeo anexado ao post	texto
picture	A imagem de algum link incluído com o post	texto
place	Qualquer informação de localização anexado ao post	página
source	Uma URL para qualquer vídeo anexado ao post	texto
status_type	Descrição do tipo de uma atualização de status	texto
story	Texto com a história de um post	texto
to	Perfis mencionados ou marcados no post	objeto
type	O tipo de objeto de um post	texto

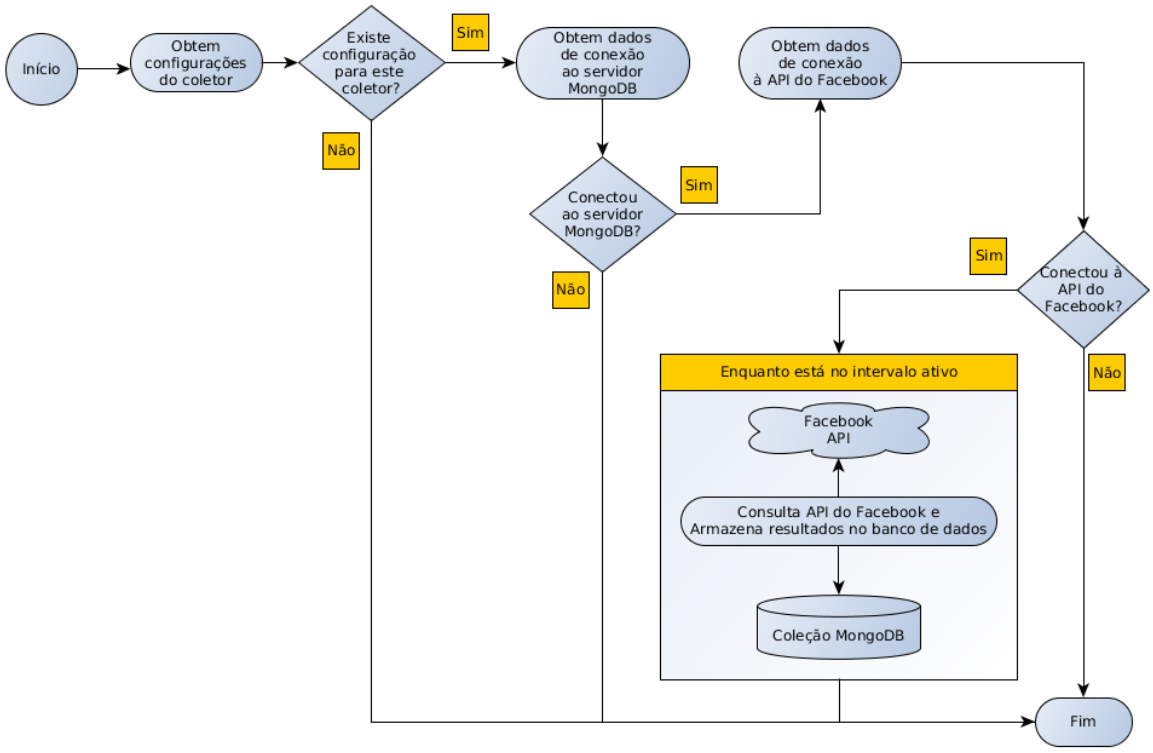
updated_time	O horário da última atualização do post	data/hora
with_tags	Perfis marcados com 'esteve com' o publicante do post	objeto

Fonte: https://developers.facebook.com/docs/graph-api/reference/v2.1/post

6.1.2.5 Estrutura do coletor

O coletor do Facebook é um programa que basicamente lê as configurações definidas pelo usuário, consulta a API utilizando a autenticação cadastrada e grava os dados obtidos em uma base de dados. A figura a seguir ilustra o fluxo de funcionamento do programa.

Figura 15 – Fluxo do programa coletor para Facebook



Fonte: Elaborado pelo autor

O programa recebe como parâmetro inicial o código do coletor que foi cadastrado pelo usuário. Se for passado como parâmetro o código de um coletor que não estiver cadastrado, o programa é finalizado. Caso contrário o código é utilizado para carregar as configurações de conexão ao servidor MongoDB. Com

estes dados, é feita uma tentativa de conexão ao servidor. Em caso de não houver sucesso nesta conexão, o programa é finalizado, caso contrário são carregadas as informações de acesso à API. Se não houverem dados para este acesso, o programa é finalizado. Se existirem, o programa verifica se o horário está entre o intervalo cadastrado pelo usuário para o funcionamento deste coletor.

Se o horário em que o coletor estiver sendo executado estiver no intervalo ativo definido pelo usuário, o programa utilizará os dados de consulta, como as palavras-chave, para efetuar consultas à API do Facebook. O resultado destas consultas são armazenados na coleção definida pelo usuário para este coletor. Estas consultas são realizadas em um intervalo de tempo definido pelo usuário (não menor do que um minuto), enquanto o programa estiver no intervalo ativo do coletor. Assim que o horário não estiver mais entre o intervalo ativo do coletor, o programa é finalizado.

6.1.3 Coletor para Google Plus (Google+)

O coletor de dados da rede social Google Plus (ou Google+), a exemplo dos outros coletores, foi escrito na linguagem PHP. Porém, até a data deste trabalho, não há uma API oficial da Google nesta linguagem. Por este motivo, foi utilizado o projeto de código aberto "*google-api-php-client*" para efetuar as requisições de dados, disponível no endereço *https://github.com/google/google-api-php-client*.

6.1.3.1 Pré-requisitos

Para ter acesso aos dados do Google+, é necessária a criação de um projeto no console de desenvolvimento da Google, para obtenção de uma chave de acesso, que será utilizada pela API para conectar ao serviço. A criação deste projeto deve ser efetuada no endereço *https://console.developers.google.com/project*.

6.1.3.2 Recursos utilizados

Como citado anteriormente, não há uma API oficial da empresa Google na

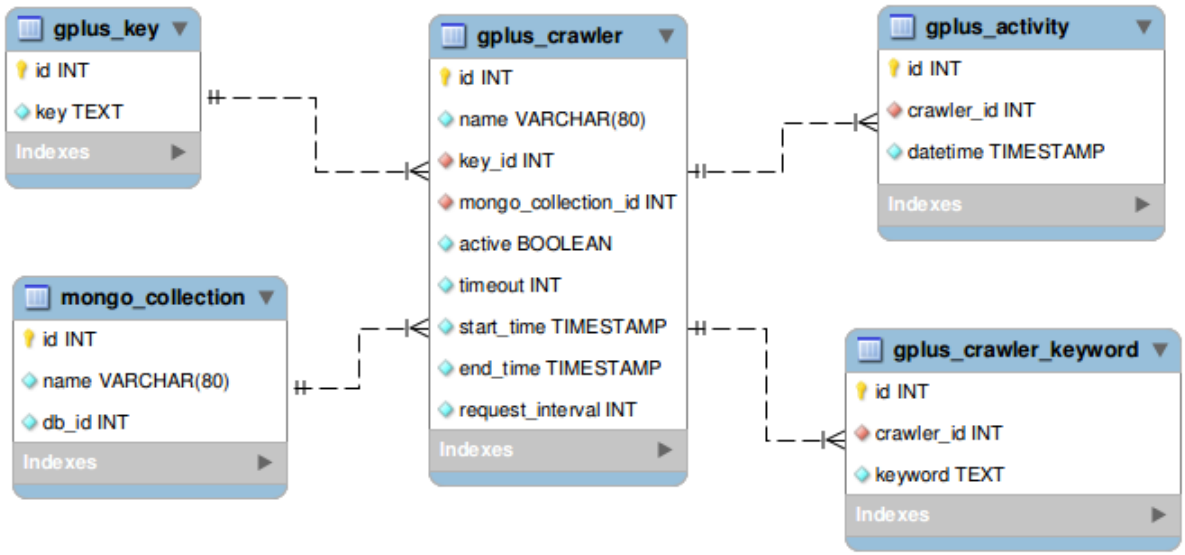
linguagem de programação PHP para o acesso aos dados da rede social Google+. Sendo assim, foi utilizado o projeto "*google-api-php-client*", que é uma biblioteca cliente em PHP para o acesso às APIs da Google.

Os demais métodos do programa, como acesso aos bancos de dados PostgreSQL e MongoDB, foram desenvolvidos pelo autor, utilizando a linguagem PHP.

6.1.3.3 Modelo de dados – Configuração

O coletor do Google+ baseia-se nas configurações cadastradas pelo usuário para coletar os dados da rede social. A seguir é apresentado o modelo ER utilizado para armazenar a configuração dos coletores.

Figura 16 – Modelo ER do coletor de dados do Google Plus



Fonte: Elaborado pelo autor

Os dados de autenticação na API são armazenados na tabela "gplus_key", que é referenciada pelo atributo "key_id" na tabela "gplus_crawler", utilizada para armazenar as configurações dos coletores do Google+. Nela encontra-se também uma referência à uma coleção MongoDB, na qual serão armazenados os dados coletados por este programa.

Na tabela "gplus_crawler_keyword" são armazenadas as palavras-chave que

o coletor irá utilizar para filtrar os dados a serem obtidos da fonte. E a tabela “gplus_activity” armazena o identificador e a data em que a atividade foi coletada, e serve como controle para o programa não armazenar a mesma informação mais de uma vez.

6.1.3.4 Modelo de dados – Armazenamento

Os dados coletados pelo programa de coleta do Google+ são chamados de “Atividades” (*Activities*), e são armazenados em uma coleção de um banco de dados de um servidor MongoDB.

Cada atividade é uma postagem de um usuário na rede social, e possui um sujeito (pessoa que realizou a atividade), um verbo (ação que foi realizada: postagem ou compartilhamento) e um objeto (objeto da atividade: texto ou atividade). O texto é especificado no conteúdo, e a foto, o vídeo ou o local são especificados nos anexos. A tabela a seguir apresenta a estrutura de uma atividade do Google+, listando alguns de seus principais atributos:

Tabela 14 – Estrutura de uma atividade no Google+

Campo	Descrição	Tipo
id	O identificador da atividade	texto
title	O título da atividade	texto
content	O conteúdo principal da atividade	texto
originalContent	O conteúdo original de uma atividade (no caso de um comentário)	texto
published	O horário em que a atividade foi inicialmente publicada	data/hora
url	O endereço para a atividade	texto
actor	A pessoa que realizou a atividade	objeto
verb	A ação realizada	texto
object	O objeto da atividade	objeto
type	Tipo da atividade	texto
replies	Respostas a esta atividade	objeto
resharers	Compartilhamentos desta atividade	objeto

attachments	Anexos desta atividade	objeto
geocode	A latitude e longitude em que a atividade ocorreu	texto
placeId	Código do lugar em que a atividade ocorreu	texto
placeName	Nome do lugar em que a atividade ocorreu	texto

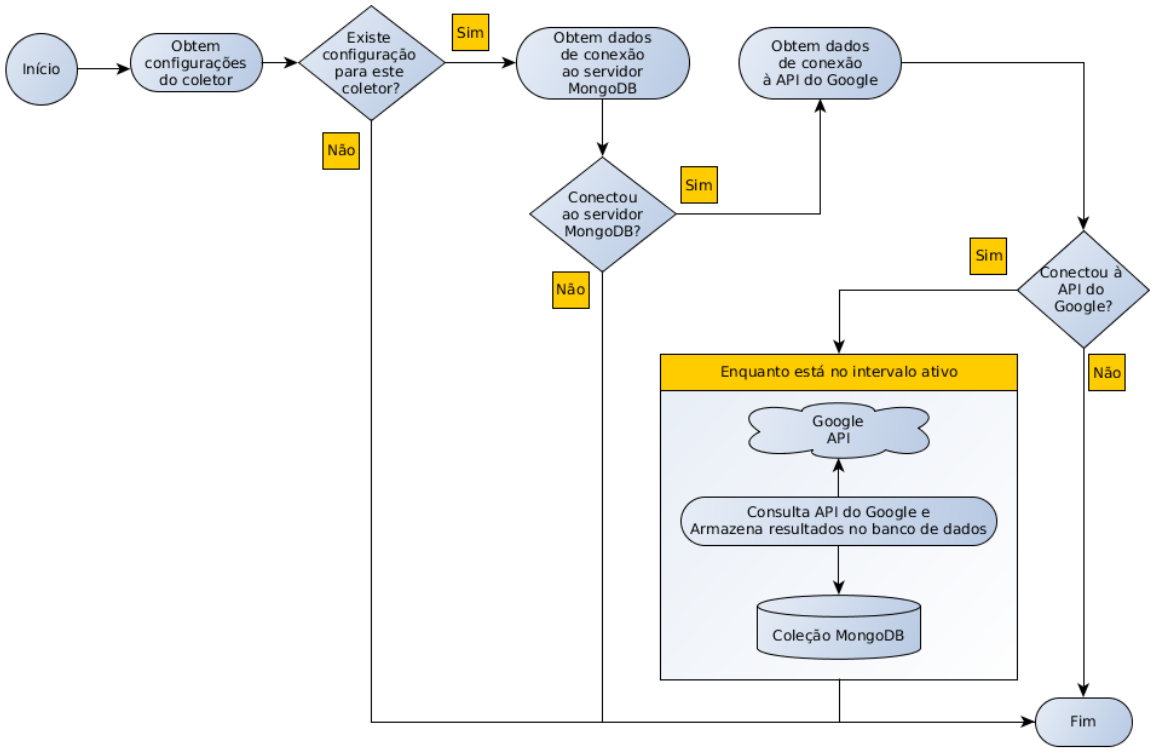
Fonte: https://developers.google.com/+/api/latest/activities#resource

A atividade é armazenada na coleção exatamente como é obtida da consulta à API, ou seja, como é representada na tabela 14.

6.1.3.5 Estrutura do coletor

O coletor do Google+ é um programa que basicamente lê as configurações definidas pelo usuário, consulta a API utilizando a autenticação cadastrada e grava os dados obtidos em uma base de dados. A figura a seguir ilustra o fluxo de funcionamento do programa.

Figura 17 – Fluxo do programa coletor para Google Plus



Fonte: Elaborado pelo autor

O programa recebe como parâmetro inicial o código do coletor que foi cadastrado pelo usuário. Se for passado como parâmetro o código de um coletor que não estiver cadastrado, o programa é finalizado. Caso contrário o código é utilizado para carregar as configurações de conexão ao servidor MongoDB. Com estes dados, é feita uma tentativa de conexão ao servidor. Em caso de não houver sucesso nesta conexão, o programa é finalizado, caso contrário são carregadas as informações de acesso à API. Se não houverem dados para este acesso, o programa é finalizado. Se existirem, o programa verifica se o horário está entre o intervalo cadastrado pelo usuário para o funcionamento deste coletor.

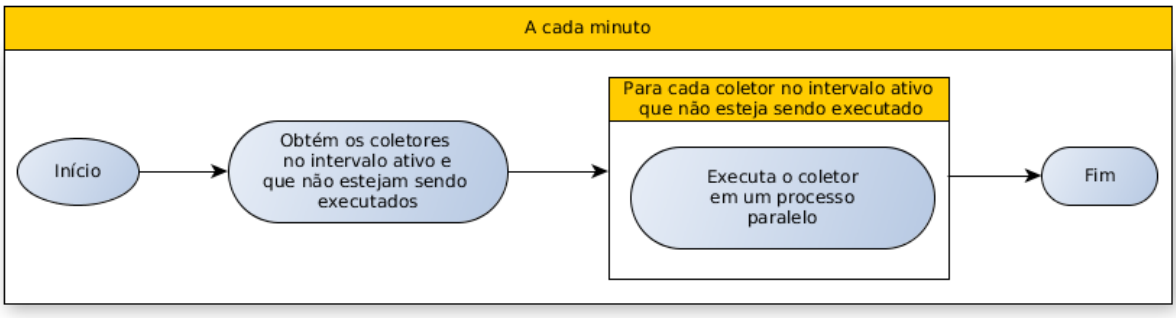
Se o horário em que o coletor estiver sendo executado estiver no intervalo ativo definido pelo usuário, o programa utilizará os dados de consulta, como as palavras-chave, para efetuar consultas à API do Google. O resultado destas consultas são armazenados na coleção definida pelo usuário para este coletor. Estas consultas são realizadas em um intervalo de tempo definido pelo usuário (não menor do que um minuto), enquanto o programa estiver no intervalo ativo do coletor. Assim que o horário não estiver mais entre o intervalo ativo do coletor, o programa é finalizado.

6.1.4 Processo de execução dos coletores

Para possibilitar a execução automática e paralela dos coletores, foi desenvolvido um programa que é executado em segundo plano no servidor onde a ferramenta foi instalada.

Este programa tem a finalidade de, através das tabelas de configuração, obter a lista de todos os coletores cadastrados no sistema e, se não estiverem ativos, executá-los em um processo separado. Desse modo, garante-se que os coletores sempre estarão em execução, se estiverem no intervalo de funcionamento configurado. A seguir é apresentado um diagrama mostrando a lógica de funcionamento deste processo:

Figura 18 – Estrutura de funcionamento do processo de execução dos coletores



Fonte: Elaborado pelo autor

Através deste processo, são atendidos os requisitos funcionais RF003 (Iniciar a coleta de dados) e RF004 (Interromper a coleta de dados).

6.2 Módulo de configuração dos coletores

O módulo de configuração dos coletores é uma interface gráfica onde o usuário pode gerenciar toda a ferramenta de coleta de dados. Neste módulo é possível manter o cadastro de servidores, banco de dados e coleções MongoDB, assim como configurar coletores para todas as fontes disponíveis no sistema desenvolvido. Também é possível acessar uma tela de consulta dos dados coletados pela ferramenta.

Esta interface é do tipo *web*, ou seja, é acessada através de um programa navegador, tal como o Mozilla Firefox ou o Google Chrome, sendo necessária a sua hospedagem em um servidor de páginas. A escolha por esse tipo de interface deu-se em virtude de ser possível acessá-la de qualquer lugar e através de qualquer dispositivo que possua um navegador e acesso à *internet*.

Através desta interface, os requisitos funcionais RF001 (Configurar estrutura do banco de dados), RF002 (Configurar comportamento dos coletores), RF005 (Cadastrar usuários) e RF006 (Consultar dados coletados) são atendidos.

6.2.1 Tecnologias utilizadas

Conforme citado no item 5.1 deste documento, as tecnologias utilizadas para

o desenvolvimento do módulo de configuração dos coletores foram as seguintes:

- para a interface gráfica foi utilizada a biblioteca javascript ExtJs, em sua versão 4.1;

- para executar as operações no lado servidor foi utilizada a linguagem de programação PHP, versão 5.3;

- para o armazenamento das informações de configuração dos coletores utilizou-se o SGDB PostgreSQL, na versão 9.1.

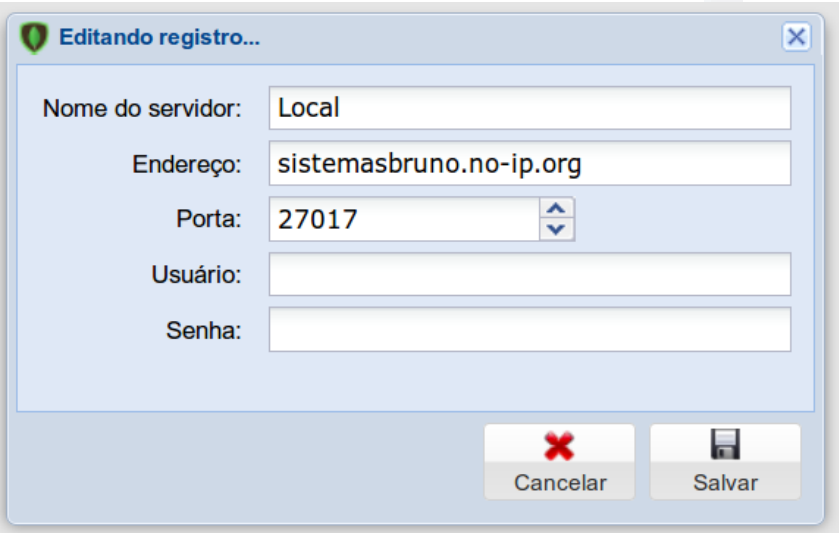
6.2.2 Interfaces

Nesta seção são apresentadas as telas de cadastro e configuração dos componentes que formam a ferramenta de coleta.

6.2.2.1 Configuração do MongoDB

Foram desenvolvidas três telas para a configuração do MongoDB: cadastro de servidores, cadastro de banco de dados e cadastro de coleções. A figura a seguir apresenta a interface de cadastro de servidores:

Figura 19 – Tela de cadastro de servidores MongoDB

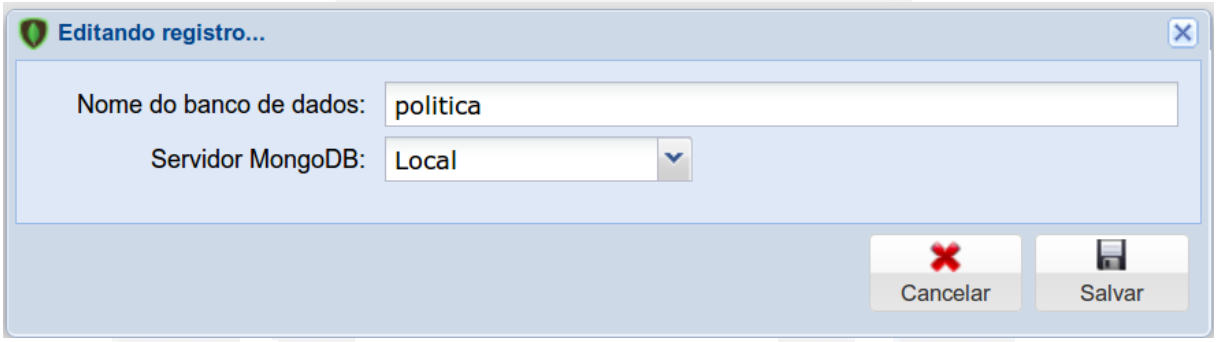


Fonte: Elaborado pelo autor.

Através desta interface, é possível cadastrar um ou mais servidores MongoDB, informando um nome, o endereço e a porta de conexão, e o usuário e a senha para o acesso ao servidor.

A próxima imagem mostra o cadastro de banco de dados, em que é necessário apenas definir um nome para o banco e escolher em qual servidor ele será utilizado:

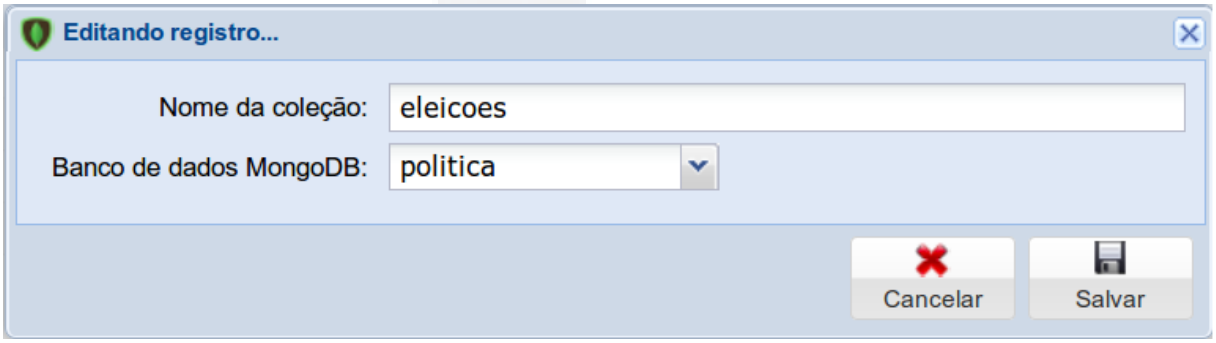
Figura 20 – Tela de cadastro de banco de dados MongoDB



Fonte: Elaborado pelo autor

A última interface de configuração do MongoDB é o cadastro de coleções. Um banco de dados pode ter várias coleções, assim, para cadastrar uma coleção deve-se definir um nome e o banco de dados em que a coleção será armazenada. A figura a seguir ilustra esta tela de cadastro.

Figura 21 – Tela de cadastro de coleções MongoDB

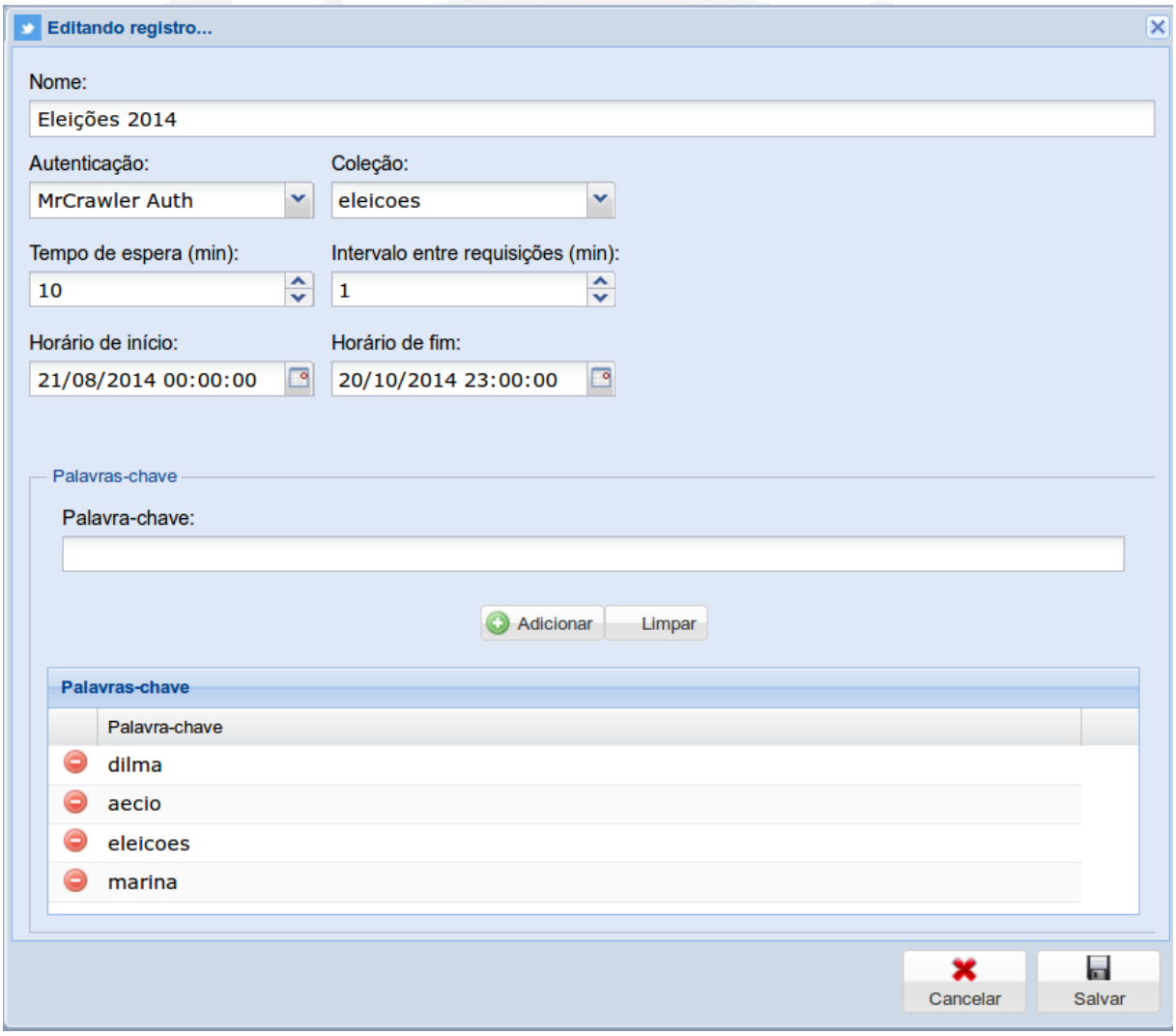


Fonte: Elaborado pelo autor

6.2.2.2 Interface de configuração do coletor para twitter

Para a configuração dos coletores de dados para o Twitter foi desenvolvida a interface exemplificada na imagem 22. Nela é possível informar o nome do coletor, qual a autenticação a ser utilizada, a coleção onde serem armazenados os dados coletados, o tempo de espera por uma resposta à uma requisição à API, o tempo entre requisições e o horário de início e fim de funcionamento do programa, além das palavras-chave a serem utilizadas para filtrar os dados do Twitter.

Figura 22 – Interface de configuração do coletor para Twitter

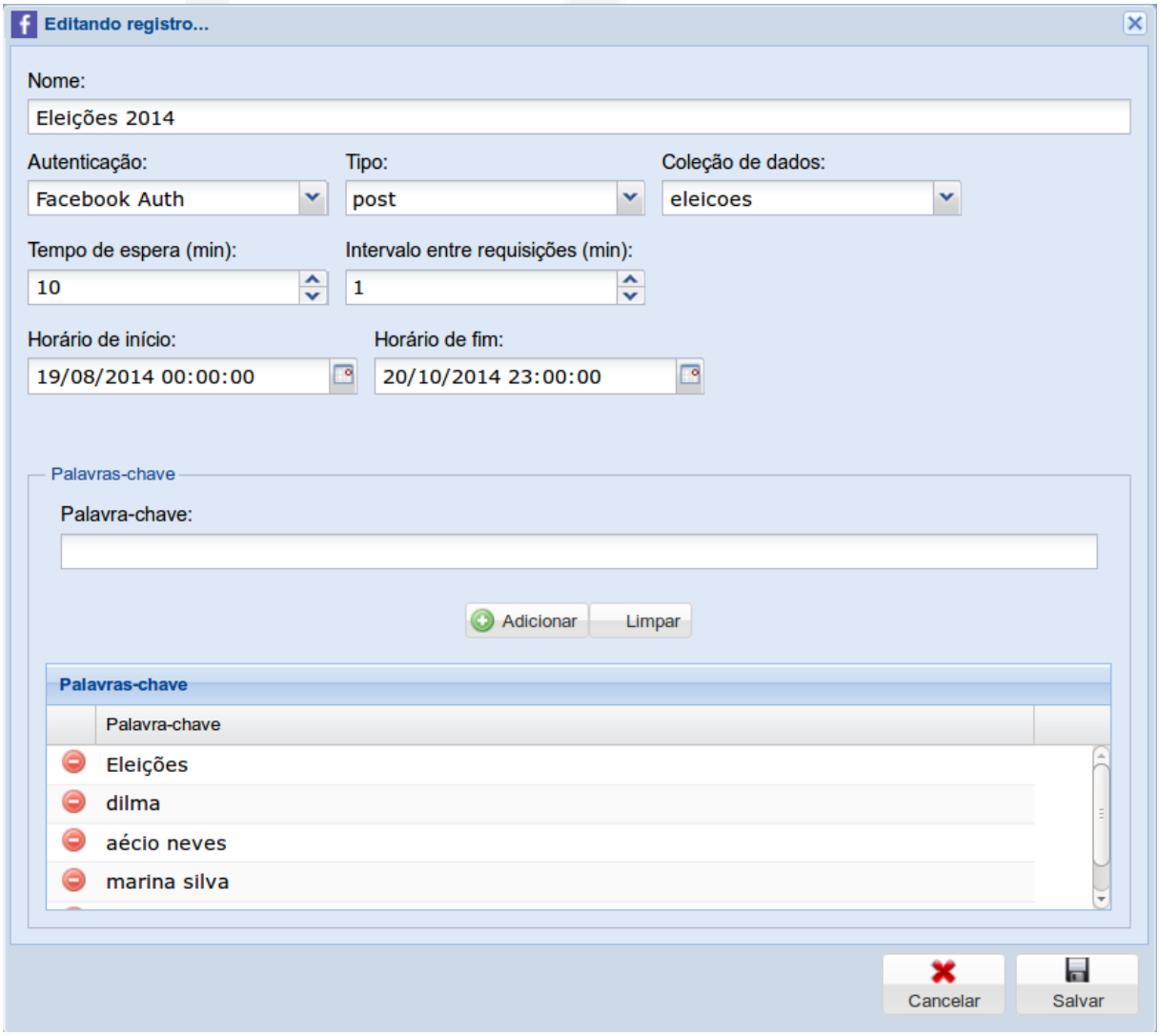


Fonte: Elaborado pelo autor

6.2.2.3 Interface de configuração do coletor para Facebook

A interface de configuração de coletores para Facebook é semelhante à do Twitter, sendo possível informar um nome, a autenticação, o tipo de conteúdo, a coleção onde os dados serão armazenados, o tempo de espera, o intervalo entre as requisições e o horário de funcionamento do programa, além das palavras-chave desejadas na consulta.

Figura 23 – Interface de configuração do coletor para Facebook

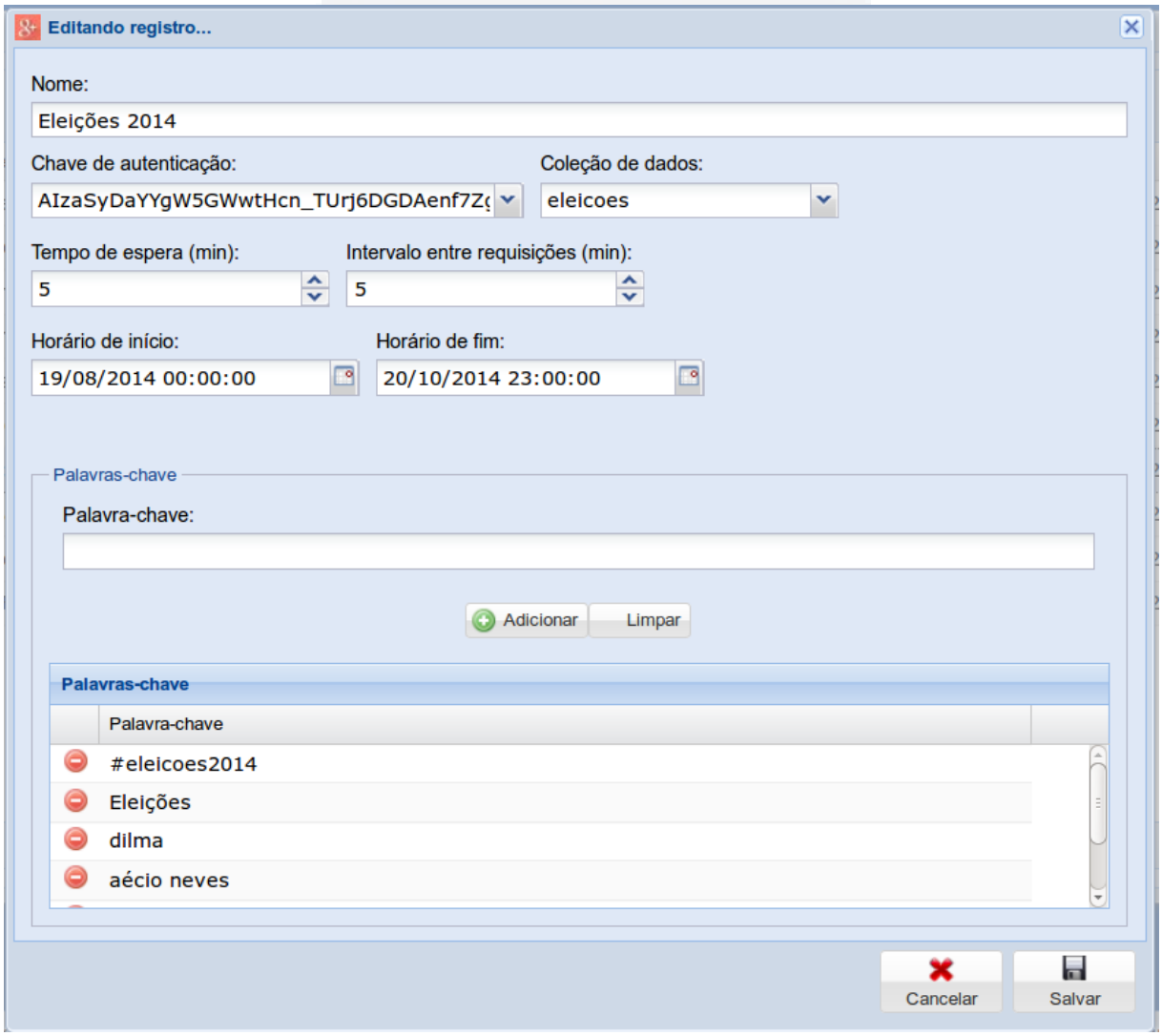


Fonte: Elaborado pelo autor

6.2.2.4 Interface de configuração do coletor para Google Plus

A tela de configuração do coletor para Google+ segue o padrão das demais interfaces. É possível informar o nome, a coleção, o tempo de espera, o intervalo entre requisições, o horário de funcionamento e as palavras-chave para a busca na rede social.

Figura 24 – Interface de configuração do coletor para Google Plus



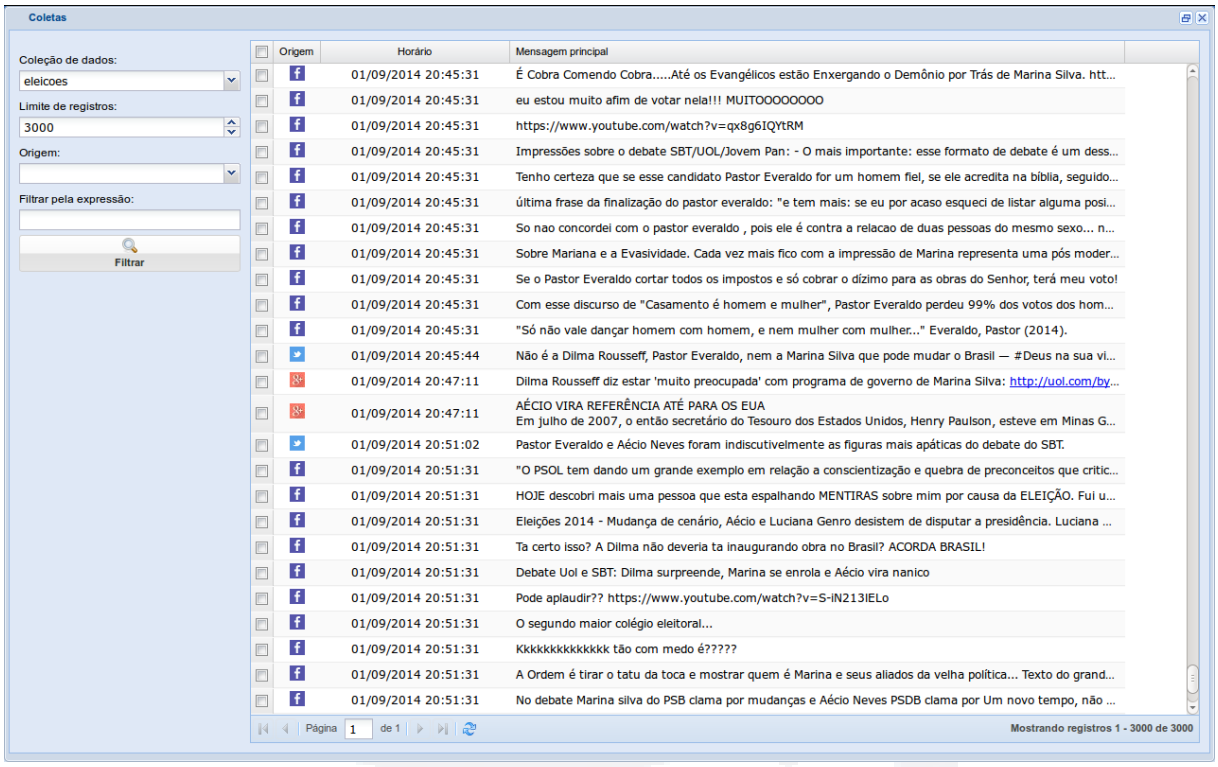
Fonte: Elaborado pelo autor

6.2.2.5 Interface de busca de dados coletados

Para visualizar os dados coletados pelos programas de coleta, foi

desenvolvida uma tela de busca. Nela pode-se verificar os dados armazenados nas coleções do(s) servidor(es) MongoDB, aplicando filtros. A imagem a seguir apresenta esta interface de consulta.

Figura 25 – Interface de busca de dados coletados



Fonte: Elaborado pelo autor

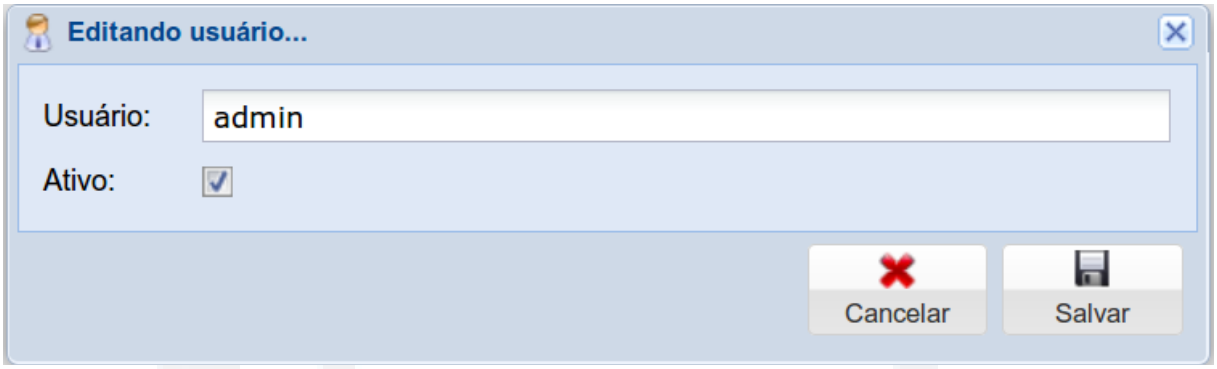
Para efetuar uma busca através desta tela, deve-se escolher a coleção na qual deseja pesquisar. Opcionalmente, pode ser aplicado um termo de pesquisa e um filtro de origem (informando de qual fonte de dados os registros foram obtidos). Também pode-se informar um limite de registros a serem listados, com a finalidade de não causar um travamento no sistema no caso de haverem muitos registros.

6.2.2.6 Interface de cadastro de usuários

Através da interface de cadastro de usuários é possível inserir novos acessos ao sistema. Em um primeiro momento não há uma distinção entre os usuários (todos tem acesso total a todas as funcionalidades), sendo esta uma melhoria a ser

implementada em uma nova versão da ferramenta. A seguir, na figura 26, é ilustrada a tela de cadastro de usuários.

Figura 26 – Tela de cadastro de usuários



Fonte: Elaborado pelo autor

7 TESTES E VERIFICAÇÕES

Com a ferramenta implementada, foram executadas coletas para verificar seu funcionamento. Nesta seção serão descritas algumas destas, além dos resultados obtidos.

A primeira coleta foi realizada em um período de aproximadamente 238 horas, durante os meses de setembro e outubro de 2014 e teve como tema as eleições do mesmo ano. Esta coleta utilizou os coletores de dados para Twitter, Facebook e Google+, com as mesmas palavras-chave para todos: “dilma”, “aécio” e “eleições”. Na tabela a seguir, são apresentados os totalizadores de registros desta coleta:

Tabela 15 – Totalizadores da primeira coleta

Coletor	Nº Registros
Twitter	285.879
Facebook	63.645
Google+	2.185
Total	351.709

Fonte: Elaborado pelo autor

Com o total de 351.709 registros coletados, esta coleta teve a média de 20.689 registros coletados por dia. O coletor de dados do Twitter foi o que mais coletou atividades dos usuários, principalmente pelo fato de todos os *tweets* serem de acesso público, um pouco diferente dos *posts* do Facebook, que dependem das configurações dos usuários para serem públicos ou não. Também deve-se ao fato da API do Facebook restringir o número de registros por requisição em 100. A API do Twitter retorna 150 registros por requisição. E a baixa coleta do coletor do Google

Plus deve-se ao fato de esta rede social ainda não ser muito utilizada e não possuir uma grande atividade por parte de seus usuários. Esta primeira coleta evidenciou que a ferramenta é capaz de adquirir um grande volume de dados.

Com estes dados coletados, foi possível fazer algumas análises e descobrir informações interessantes, que corroboram a importância da ferramenta para geração de conhecimento. Por exemplo, fazendo uma pesquisa nestes registros utilizando simplesmente as expressões “vou votar na dilma” e “vou votar no aécio”, chegou-se a um resultado muito parecido com o que realmente ocorreu nas urnas. Utilizando estas expressões foram encontrados 672 registros, sendo que com a expressão “vou votar na dilma”, foram coletados 349 registros (51,93%) e, com a expressão “vou votar no aécio”, 323 registros (48,07%). Estes números são muito próximos ao que realmente aconteceu nas eleições, com a candidata Dilma recebendo 51,64% dos votos e o candidato Aécio Neves recebendo 48,36% dos votos. Na imagem a seguir, é apresentada a consulta realizada no banco de dados MongoDB.

Figura 27 – Consulta na coleção de dados “eleicoes”

```
>
> db.eleicoes.count({$or: [{"text": {$regex: /.*vou votar na dilma.*/, $options: "si"}}, {"message"
: {$regex: /.*vou votar na dilma.*/, $options: "si"}}, {"description": {$regex: /.*vou votar na dil
ma.*/, $options: "si"}}]});
349
> db.eleicoes.count({$or: [{"text": {$regex: /.*vou votar no aecio.*/, $options: "si"}}, {"message"
: {$regex: /.*vou votar no aecio.*/, $options: "si"}}, {"description": {$regex: /.*vou votar no aec
io.*/, $options: "si"}}, {"text": {$regex: /.*vou votar no aécio.*/, $options: "si"}}, {"message":
{$regex: /.*vou votar no aécio.*/, $options: "si"}}, {"description": {$regex: /.*vou votar no aécio
.*/, $options: "si"}}]});
323
>
```

Fonte: Elaborado pelo autor

Este resultado mostra que o sistema desenvolvido pode ser utilizado como ferramenta de pesquisa por amostragem. A seguir, na figura 28, são exibidos dois registros no formato JSON, que estão armazenados na coleção de dados “eleicoes”, um filtrado com a expressão “vou votar na dilma” e outro com a expressão “vou votar no aécio”.

Figura 28 – Registros da coleção de dados “eleicoes”


```
{
    "_id" : ObjectId("5432b8f30b16375b0a8b6e5e"),
    "metadata" : { "iso_language_code" : "pt", "result_type" : "recent" },
    "created_at" : "Mon Oct 06 15:44:26 +0000 2014",
    "id" : NumberLong("519151200537944065"),
    "id_str" : "519151200537944065",
    "text" : "Vou votar na Dilma sim e não escondo. Sou do estado que tem Beto no governo e Álvaro Dias no senado.",
    "source" : "twitter",
    "truncated" : false,
    "user" : { "id" : NumberLong(53793436), "id_str" : "53793436", "name" : "cavanhinho", "screen_name" : "alcavanha", "location" : "Curitiba", "description"
    "geo" : null,
    "coordinates" : null,
    "place" : null,
    "contributors" : null,
    "retweet_count" : NumberLong(0),
    "favorite_count" : NumberLong(0),
    "entities" : { "hashtags" : [ ], "symbols" : [ ], "urls" : [ ], "user_mentions" : [ ] },
    "favorited" : false,
    "retweeted" : false,
    "lang" : "pt"
}

{
    "_id" : ObjectId("544c0ad40b163702078b706a"),
    "metadata" : { "iso_language_code" : "pt", "result_type" : "recent" },
    "created_at" : "Sat Oct 25 20:40:26 +0000 2014",
    "id" : NumberLong("526111060244586496"),
    "id_str" : "526111060244586496",
    "text" : "Eu não sou influenciado por pesquisas eleitorais. Eu vou votar no Aécio para presidente. #VotoAecioPeloBR45IL",
    "source" : "twitter",
    "truncated" : false,
    "user" : { "id" : NumberLong(294261353), "id_str" : "294261353", "name" : "marcos ", "screen_name" : "marcos_bezerra_", "location" : "Rio de Janeiro", "p
    "geo" : null,
    "coordinates" : null,
    "place" : { "id" : "e433fbca595f29e5", "url" : "https://api.twitter.com/1.1/geo/id/e433fbca595f29e5.json", "place_type" : "admin", "name" : "Rio de Janei
    "contributors" : null,
    "retweet_count" : NumberLong(0),
    "favorite_count" : NumberLong(0),
    "entities" : { "hashtags" : [ { "text" : "VotoAecioPeloBR45IL", "indices" : [ NumberLong(89), NumberLong(109) ] } ], "symbols" : [ ], "urls" : [ ], "user
    "favorited" : false,
    "retweeted" : false,
    "lang" : "pt"
}
```


Fonte: Elaborado pelo autor

No exemplo desta imagem, temos duas pessoas de diferentes lugares do país, que, ao expressarem sua opinião em uma rede social, fizeram parte da pesquisa de intenção de votos realizada como teste da ferramenta. Pesquisas como estas, podem ser aprimoradas e em alguns casos podem vir a substituir as tradicionais pesquisas de entrevistas.

A segunda coleta de validação do sistema desenvolvido foi efetuada nos dias 25, 26 e 27 de outubro de 2014 e buscou registros de duas marcas de refrigerantes, utilizando as palavras-chave “pepsi” e “coca-cola” para os três tipos de coletores. A tabela 16 apresenta os resultados desta coleta:

Tabela 16 – Totalizadores da segunda coleta

Coletor	Nº Registros
Twitter	5.349
Facebook	1.891
Google+	16
Total	7.256

Fonte: Elaborado pelo autor

Esta coleta obteve um total de 7.256 registros utilizando as duas palavras-chave. Destes, 5.900 referiam-se ao termo “coca-cola” e, 1.356 ao termo “pepsi”. A partir dos dados desta coleta, uma ferramenta de análise pode descobrir informações sobre os consumidores destas marcas, tais como se estão satisfeitos ou não com estes produtos, o quanto consumem ou o quanto gostariam de consumir, entre muitas outras análises possíveis. Na figura a seguir são apresentados alguns destes registros coletados, onde pessoas expressam suas opiniões sobre estas duas marcas.

Figura 29 – Registros de opiniões de pessoas sobre marcas de refrigerantes

Fonte: Elaborado pelo autor.

A última coleta de validação da ferramenta buscou registros da palavra “univates” nas fontes de dados e foi efetuada em 5 dias, durante os meses de outubro e novembro de 2014. Esta coleta evidenciou a facilidade da obtenção de assuntos que podem ser relevantes a uma instituição em uma determinada região, podendo-se obter um *feedback* de seu público quase que instantaneamente. A tabela 17 apresenta os totalizadores desta coleta.

Tabela 17 – Totalizadores da terceira coleta

Coletor	Nº Registros
Twitter	801
Facebook	264
Google+	0
Total	1.065

Fonte: Elaborado pelo autor

No período desta coleta foram obtidos 1.065 registros mencionando a palavra “univates”. Neste conjunto de dados foi executada ma consulta utilizando como filtro a palavra “biblioteca”, com a ideia de verificar a opinião das pessoas sobre a biblioteca da instituição. Os resultados desta consulta são apresentados na imagem a seguir.

Figura 30 – Registros de opiniões sobre a biblioteca da Univates

Fonte: Elaborado pelo autor.

8 CONSIDERAÇOES FINAIS

O estudo realizado com base em diversas fontes bibliográficas evidenciou a oportunidade de organizações adquirirem conhecimentos importantes para a melhoria contínua de seus processos de negócio, através da análise de grandes volumes de dados por sistemas de Big Data.

O exemplo de sucesso de diversas empresas apresentadas no referencial teórico corrobora a importância cada vez maior de utilizar-se destas fontes de dados no processo de tomada de decisão.

Ao propor o desenvolvimento de uma ferramenta de coleta de dados de diversas fontes, o autor busca atender a um requisito comum a todos os sistemas de Big Data: a coleta e armazenamento de um grande volume de dados, para posterior análise e mineração de informações relevantes.

O desenvolvimento da ferramenta apresentada nos capítulos anteriores e os dados coletados através de programas coletores evidenciam que o objetivo principal do presente trabalho foi atingido. Através dos programas coletores e da interface de configuração destes, foi implementada uma estrutura que coleta e armazena dados provenientes de diversas fontes, formando assim um banco de dados a ser utilizado por um sistema de mineração de dados.

Entende-se também que, além do objetivo principal, os objetivos secundários também foram alcançados. Com o uso das API's citadas no desenvolvimento do trabalho, foi possível compreender e utilizar ferramentas de coleta de dados das fontes utilizadas. Através do desenvolvimento do sistema, foi necessário compreender e desenvolver métodos para armazenar os dados em um banco de dados NoSQL (MongoDB). A implementação dos programas de coleta possibilitou

alcançar o objetivo de buscar dados através de rotinas automatizadas.

No capítulo 5 é apresentada uma proposta de arquitetura de um sistema de análise e mineração de grande volume de dados, composto dos módulos de aquisição, processamento e análise. Esta proposta pode ser seguida em trabalhos vindouros. Sendo assim, entende-se que o objetivo de propor uma arquitetura de sistema de coleta, processamento e análise de dados tenha sido alcançado.

Através dos resultados obtidos entende-se que este sistema pode ser utilizado para a coleta e o armazenamento de um grande volume de dados, sendo esta ferramenta uma contribuição para a criação de um sistema de mineração de dados em trabalhos futuros. Cabe ressaltar aqui que este trabalho é parte de uma iniciativa maior, com o objetivo de criar um ambiente para análise de grande volume de dados.

REFERÊNCIAS

CARVALHO, L. A. V. **Data Mining**: A Mineração de dados no Marketing. Rio de Janeiro: Editora Ciência Moderna, 2005.

CHANG, F.; DEAN, J.; GHEMAWAT, S.; HSIEH, W. C.; WALLACH, D. A.; BURROWS, M.; CHANDRA, T.; FIKES, A.; GRUBER, R. E. Bigtable: A distributed storage system for structured data. **In Proceedings of the 7th Conference on Usenix Symposium.** 2010.

CHODOROW, K. **MongoDB: The Definitive Guide**. 2.ed. Sebastopol: O'Reilly Media, 2013.

DARWEN, H. **An Introduction to Relational Database Theory**. Hugh Darwen & Ventus Publishing, 2009.

DUMBILL, E. **Big Data Now**: 2012 Edition. Sebastopol: O'Reilly Media, 2012.

EATON, C.; DEROOS, D.; DEUTSCH, T.; LAPIS, G.; ZIKOPOULOS, P. **Understanding Big Data**: Analytics for Enterprise Class Hadoop and Streaming Data. The McGraw-Hill Companies, 2012.

GLOBO. **Como funciona o Big Data**. O GLOBO DIGITAL E MÍDIA. Disponível em: http://oglobo.globo.com/infograficos/bigdata/ Acesso em: 25 set. 2013.

HEUSER, Carlos A. **Projeto de Banco de Dados**. 6. ed. Porto Alegre: Artmed, 1998.

HEWITT, E. **Cassandra: The Definitive Guide**. Sebastopol: O'Reilly Media, 2011.

HURWITZ, J.; NUGENT, A.; HALPER, F.; KAUFMAN, M. **Big Data for Dummies**: A Wiley Brand. Hoboken: John Wiley & Sons, 2013.

ISLAM, R. **PHP and MongoDB Web Development**: Begginer's Guide. Birmingham: Packt Publisinhg, 2011.

JSON. **Introducing JSON**. Disponível em: http://json.org/ Acesso em: 02 nov. 2013.

LAM, C. **Hadoop In Action**. Stamford: Manning Publications, 2011.

LIU, B. **Web Data Mining**: Exploring Hyperlinks, Contents and Usage Data. 2.ed. Chicago: Springer, 2011.

MILANI, A. **PostgreSQL**: Guia do Programador. São Paulo: Novatec Editora, 2008.

REDMOND, E.; WILSON, Jim R. **Seven Databases in Seven Weeks**: A Guide to Modern Databases and the NoSQL Movement. Dallas: Pragmatic Programmers, 2012.

SENCHA. **Sencha ExtJS**: Javascript Framework for Rich Desktop Apps. Disponível em: http://www.sencha.com/products/extjs/ Acesso em: 02 nov. 2013.

TIWARI, S. **Professional NoSQL**. Indianapolis: John Wiley & Sons, 2011.

VIEIRA, M. R.; FIGUEIREDO, J. M.; LIBERATTI, G.; VIEBRANTZ, A. F. M. Bancos de dados NoSQL: Conceitos, ferramentas, linguagens e estudos de caso no contexto de Big Data. **Simpósio Brasileiro de Bancos de Dados.** 2012.