UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS APARECIDO DA SILVA ROBERTO

UMA SOLUÇÃO DE EXTRAÇÃO E GEORREFERENCIAMENTO DE ANÚNCIOS IMOBILIÁRIOS DA INTERNET

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2019

MATHEUS APARECIDO DA SILVA ROBERTO

UMA SOLUÇÃO DE EXTRAÇÃO E GEORREFERENCIAMENTO DE ANÚNCIOS IMOBILIÁRIOS DA INTERNET

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. André Koscianski

PONTA GROSSA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO

UMA SOLUÇÃO DE EXTRAÇÃO E GEORREFERENCIAMENTO DE ANÚNCIOS IMOBILIÁRIOS DA INTERNET

por

MATHEUS APARECIDO DA SILVA ROBERTO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 20 de novembro de 2019 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. André Koscianski
Orientador

_____
Prof. Dr. Diego Roberto Antunes
Membro titular

_____
Prof. Dr. Tarcizio Alexandre Bini
Membro titular

_____
Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof.ª. Dr.ª. Mauren Louise Sguario
Coordenadora do curso

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

AGRADECIMENTOS

Gostaria de expressar todo o meu agradecimento as pessoas que auxiliaram a realizar este trabalho.

Agradeço ao meu orientador Prof. Dr. André Koscianski, pela sabedoria, apoio e conhecimento que me transmitiu desde do primeiro dia de aula da faculdade à finalização deste trabalho.

Quero deixar o meu reconhecimento a toda minha família, tios, tias, primos e primas que me auxiliaram a chegar até aqui.

Um agradecimento especial ao meus pais Jairo Aparecido Roberto e Valdeli Aparecida da Silva Roberto e minha irmã Esther Aparecida da Silva Roberto, pois sem o apoio deles este trabalho e este sonho não poderiam ser realizados.

Gostaria deixar um agradecimento especial para minhas avós Benedita da Conceição Roberto e Sebastiana dos Santos Silva, que sempre me incentivaram nos estudos e demonstraram que a educação é peça chave para conquistar o seu lugar no mundo.

Gostaria de lembrar de duas pessoas que me ajudaram a iniciar está jornada mas não puderam me ver concluir: minha tia-avó Elisabeth Maria da Conceição Rogoski que me abriu as portas de sua casa e me auxiliou no início a faculdade e meu avô Luiz Carlos Roberto que sempre demonstrou que é preciso lutar para conquistar seus sonhos e estar junto à família é o mais importante.

Agradeço aos meus amigos de casa que tornaram está jornada mais divertida, à minha namorada pelo apoio e auxílio na escrita deste trabalho, ao pessoal da atlética Javas que foram como uma segunda família para mim, e aos meus amigos que mesmo longe, me deram força para finalizar está jornada, pois mesmo nos dias mais conturbados eles queriam me ver vencer.

Finalmente agradeço a todos que me ajudaram direta e indiretamente neste trabalho, na faculdade e na minha vida, peço desculpas se eu esquecer de alguém, mas saiba que agradeço de todo o coração aos que me auxiliaram.

RESUMO

ROBERTO, Matheus Aparecido da Silva. **Uma solução de extração e georreferenciamento de anúncios imobiliários da internet.** 2019. 66 f. Trabalho de Conclusão de Curso Bacharelado, em Ciência da - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2019.

O crescimento do território urbano é algo que não se pode controlar, tornando-se uma dificuldade para as organizações municipais. Com a utilização de sistemas de informação geográfica e simulação urbana surge uma nova maneira para tomar decisões no planejamento da cidade. Mas existe uma defasagem grande para isto acontecer de forma natural pelos municípios, isso se deve à falta de bases de dados georreferenciadas para estudo da malha urbana. Como a Internet é um local repleto de informações, se torna uma ótima fonte para extrair o conteúdo desejado. A partir do estudo de vendas de imóveis é possível identificar alterações na cidade pela modificação do preço de venda e expansão do território pelo surgimento de novos imóveis em uma determinada região. Este trabalho propõe criar uma solução que possibilita a extração de informações de sites imobiliários, tais como o endereço e valor de venda e transformar o endereço em coordenadas geográficas, latitude e longitude, disponibilizando uma base de dados para ser utilizada em estudos da cidade. Neste trabalho fez-se uso de sites imobiliários da cidade de Ponta Grossa – Paraná com intuito de extrair informações não estruturadas, endereços e valores de venda, para que este conteúdo pudesse ser transformado em informações aplicáveis à sistemas de simulação urbana e informação geográfica. Para transformar endereço em coordenadas geográficas se tornou necessário utilizar serviço de geolocalização como *Google Maps* e *Open Street Map*. Como estas informações não seguem um padrão foi utilizado o formato orientado a documentos para para que se pudesse armazenar os dados encontrados. A aplicação se mostrou eficiente pela quantidade de anúncios extraídos e pela independência do usuário para extrair e georreferenciar a informação.

Palavras-chave: Extração de Dados. *Web Scraping*. Georreferenciamento.

ABSTRACT

ROBERTO, Matheus Aparecido da Silva. **A solution for extracting and georeferencing real estate advertisements from the Internet**. 2019. 66 p. Work of Conclusion Course Graduation in Computer Science - Federal Technology University - Paraná. Ponta Grossa, 2019.

The growth of urban territory is something that cannot be controlled, becoming a difficulty for municipal organizations. With the usage of geographic information systems and urban simulation, a new way to make decisions in city planning emerges. Although there is a big gap for it happen naturally by municipalities, this is due to the lack of georeferenced databases for the study of the urban network. As the Internet is a place full of information, it becomes a great source to extract the desired content. From the study of real estate sales, it is possible to identify changes in the city by modifying the sale price and expanding the territory by the emergence of new real estate in a given region. This work proposes to create a solution that enables the extraction of information from real estate sites, such as the address and sale value and turn the address into geographic coordinates, latitude, and longitude, providing a database to be used in studies of the city. In this work it was made use of real estate sites in the city of Ponta Grossa - Paraná to extract unstructured information, addresses, and sales values so that this content could be turned into information applicable to urban simulation systems and geographic information. To transform address into geographic coordinates it became necessary to use geolocation services such as Google Maps and Open Street Map. As this information does not follow a pattern the document-oriented format was chosen so it could store the found data. The application proved to be efficient due to the number of ads extracted and the user's independence to extract and georeferenced the information.

Keywords: Data Extraction. Web Scraping. Georeferencing.

LISTA DE ILUSTRAÇÕES

LISTA DE ACRÔNIMOS

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
FTP	File Transfer Protocol
GIL	Global Interpreter Lock
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol Address
IPTU	Imposto Predial e Territorial Urbano
JSON	JavaScript Object Notation
OSMF	OpenStreetMap Foundation
PDF	Portable Document Format
PPGCC	Programa de Pós-Graduação em Ciência da Computação
SIG	Sistema de Informação Geográfica
URL	Uniform Resource Locator
UTFPR	Universidade Tecnológica Federal do Paraná
WWW	World Wide Web
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O registro de informações independentemente da forma que estejam escritas ou identificadas é muito importante para a sociedade. O ser humano guarda informações desde da invenção da escrita, em diferentes meios e maneiras, como em rochas, papeis, livros e computadores. A escrita mais antiga que se conhece é a sumeriana, com registros contábeis por volta de 3000 a.C. (HIGOUNET, 2003).

A maior parte dos registros de dados da humanidade foram gerados sem a utilização dos computadores e muitos continuam assim. Dados guardados em papel causam uma série de dificuldades: armazenamento e recuperação, o que leva à ideia de digitalização de documentos, visando aumentar vida útil dos arquivos, além de facilitar o acesso a ele posteriormente. A adoção de processos tecnológicos para digitalização dos dados, traz vantagens como maior transparência em registros públicos, diminui a chance de extravio de documentos e traz rapidez para acesso seja via Internet ou Intranet, reduzindo o tempo em filas e desburocratizando processos (E-GESTÃO PÚBLICA, 2018; GREEN DOC, 2010; TUPÃ ESTÂNCIA TURÍSTICA, 2013). Um exemplo prático disso foi a emissão do Cartão Nacional da Pessoa Idosa e Pessoa com Mobilidade Reduzida que diminuíram o tempo gasto para obter este cartão de 40 dias para 10 minutos pela utilização do serviço de forma online (COMPUTERWORLD, 2019). Outro serviço que é implantando em muitas cidades é o Zona Azul Digital, que visa diminuir o número de fraudes com cartões de estacionamento em zona azul das cidades, diminuir o tempo para preenchimento de talões e tornar o serviço mais fácil para o usuário (CIDADE DE SÃO PAULO, 2019).

Textos contendo informações são usados em inúmeros casos. Pode-se citar contratos comerciais, que contém dados de identificação pessoal; documentos de caráter legal, que com frequência citam outros documentos e que assim acabam se aproximando da ideia de hipertexto; artigos científicos, que contem gráficos e tabelas; prontuários médicos; e anúncios comerciais. Todos os exemplos citados tem uma característica comum: embora os registros possam conter muita informação útil, ela se apresenta dispersa, misturada e desorganizada. Para resolver estes problemas pode ser utilizado a digitalização destes registros, permitindo também o processamento por computadores e armazenamento em bases de dados.

As informações são classificadas e separadas, por exemplo em colunas de tabelas, tornando-as facilmente identificáveis.

Existem ainda certos cadastros ou bases de dados que estão em computadores, porém, não estão preparados para ser manipulados pelo computador. Exemplo disso são sites comerciais de anúncios de produtos, tais como Bondfaro, Buscapé e Zoom (BUSCAPÉ COMPANY INFORMAÇÃO E TECNOLOGIA LTDA., 2019). Estes sites utilizam algoritmos denominados robôs que vasculham sites de varejistas para obter informações sobre produtos ofertados, valor, forma de pagamento, parcelamento e comentários sobre o produto e apresentam para os usuários os dados atualizados. Estes sites criaram uma tecnologia para substituir ação manual que os consumidores faziam: visitar lojas físicas para verificar valores, forma de pagamentos, características dos produtos, para tomar a decisão de onde adquirir. A tecnologia relevante apresentada pelos sites é a recuperação de informações por outros agentes, em diferentes locais, agrupando em um único local para comparação e uso do consumidor.

O segundo tipo de dado relevante para este trabalho, está relacionado a Sistemas de Informação Geográfica - SIG. Diversas situações de processamento de dados envolvem mapas. Exemplo disso são dados geográficos para uso público, em que a prefeitura disponibiliza plantas de imóveis, tipos de solo, dados de recursos naturais para busca e processamento de dados, também revisão de IPTU (Imposto sobre a propriedade predial e territorial urbana) e alvará para construção. Existe uma particularidade neste tipo dado que é a localização geográfica. Essa informação permite apresentar dados usando um formato gráfico, ou seja, mapas que permitem localizar, por exemplo, todas as unidades de saúde de uma cidade de uma maneira intuitiva para pessoas. Além disso, a partir da localização espacial podem ser derivadas outras informações, tais como distância entre pontos, ou área de cobertura. No exemplo citado, seria possível estimar a população que é atendida por um dado hospital. As aplicações de SIG são muitas variadas e incluem áreas como: Agricultura, Engenharia Civil, Geologia, Hidrografia, Logística, Planejamento, Segurança Pública, Preservação de Recursos Naturais, Transportes Urbano, Trânsito.

A ideia de utilizar SIGs auxilia na tomada de decisões e no planejamento, porém ela ainda tem um problema a ser resolvido principalmente no Brasil, que é a falta de base de dados. “Os bancos de dados, por outro lado, são elementos que

exigem grande atenção em qualquer implantação de SIG. De uma maneira geral, o processo de aquisição de dados é caro e complexo" (SILVA, 1998).

Uma aplicação que reúne os dois tipos de dados mencionados, ou seja, bases de dados não estruturadas e informações de natureza geográfica, são anúncios imobiliários. Esse tipo de aplicação é utilizado por um grande número de pessoas: corretores de imóveis, usuários em busca de uma residência, empresários que planejam instalar um novo negócio. Com frequência, esse tipo de base de dados é composto por textos com informações misturadas, ou usa uma organização para destacar dados exemplo: número de quartos, vagas na garagem, quantidade de banheiros. Porém sem um padrão estabelecido entre os sites imobiliários. Além disso, a informação geográfica se limita a um endereço, muitas vezes incompleto ou grafado erroneamente, e sem apresentar coordenadas geográficas.

Um exemplo de solução comercial para criar uma base de dados com dados imobiliários é a parceria entre a Fundação Instituto de Pesquisas Econômicas (FIPE) e portal ZAP, para o desenvolvimento do FIPE-ZAP para acompanhamento de valores de imóveis nas capitais dos estados brasileiros, apresentado pelo trabalho de NEDER et al., (2017). Outro exemplo é o uso da *Google Maps API* (GOOGLE MAPS, 2019) para o georreferenciamento de bairros na cidade do Rio de Janeiro para obter informações sobre mortalidade no município (SILVEIRA; OLIVEIRA; JUNGER, 2017). O georreferenciamento também é importante pela sua precisão em determinar locais ao menor nível de localização possível, exemplo portas de residência na China, para alimentar bases de dados de uso policial, sem precisar qualquer interferência de agentes públicos para confirmação do dado apresentado (AIJUN; LICHUAN, 2010).

Um banco de dados de anúncios georreferenciados permite realizar análises sobre mercado de imóveis de uma cidade. Uma base de dados com anúncios de imóveis de uma cidade pode auxiliar o poder público com simulações urbanas. Criar um histórico de valorização imobiliária, transporte coletivo, saúde pública, entre outros. Por exemplo, a comparação entre valores de anúncios de imóveis e os valores de venda declarados ao governo de Taiwan são apresentados em CHEN et al., (2015), utilizando dados de sites do Governo de Taiwan e uma rede de anúncios de imóveis, se utilizando de algoritmos para popular está base de dados.

Este trabalho apresenta uma solução que extrai informações de sites imobiliários da cidade de Ponta Grossa - Paraná, obtendo assim logradouro e valor

de venda utilizando a técnica de *web scraping*, que permite extrair informações de sites registradas na linguagem HTML. A partir da extração são tratados endereços com grafia errada (exemplo: Rua Coronel Cláudio, Rua Cel Cláudio), em seguida são georreferenciados (latitude e longitude) e registrados em uma base de dados. O processo de extração de dados se confronta com uma grande variedade de formatos de sites, estrutura de páginas HTML e nomes de atributos. Está fora do escopo deste trabalho tratar automaticamente essas diferenças, de forma que para incluir novas imobiliárias é necessário ajustar o código.

1.1 ORGANIZAÇÃO DO TRABALHO

Este trabalho é composto por cinco capítulos, os quais abrangem conceitos importantes para o desenvolvimento desta pesquisa. O capítulo 2 descreve a revisão da literatura sobre extração de dados da Internet e Georreferenciação de endereços, apresentando uma breve introdução sobre extração de dados de *tags HTML* e alguns métodos utilizados para georreferenciação. O capítulo 3 propõe a abordagem para realizar extração de dados e georreferenciação de endereços aplicando técnicas da literatura. O capítulo 4 descreve os resultados que foram obtidos com a solução apresentada nesta pesquisa. Por fim, o capítulo 5 faz as considerações finais sobre o trabalho e relata alguns trabalhos futuros que podem dar continuidade a esta pesquisa.

2 FERRAMENTAS E TECNOLOGIAS

Nesta seção será apresentado uma visão geral sobre os problemas e ferramentas tratados no trabalho; isto inclui tópicos sobre extração de dados a partir de páginas HTML e o georreferenciamento de endereços.

2.1 EXTRAÇÃO DE DADOS DA WEB: VISÃO GERAL

O acesso à informação vem se tornando mais prático e rápido. Computadores que cabem na palma da mão, trazem informações do mundo em questões de segundos, graças ao desenvolvimento da *WWW* (*World Wide Web)*. Mas tais informações são criadas na maior parte das vezes para uso por pessoas e não para processamento por computadores (FERRARA et al., 2014; PARVEZ et al., 2018); exemplo disso são textos livres e figuras, em lugar de tabelas de dados estruturados. Isto se apresenta como desafio, extrair dados que se encontram na Internet, mas não estão em formato adequado para tratamento computacional.

Web Mining consiste em adquirir informações de páginas da Internet, e extrair dados (BHARANIPRIYA; PRASAD, 2011). A recuperação destes registros tem por finalidade obter informações uteis para serem processadas posteriormente por pessoas ou empresas para uma análise ou pesquisa de um determinado assunto (FERRARA et al., 2014; PARVEZ et al., 2018).

Ao aplicar *data mining* deve-se levar em consideração alguns detalhes para obter o melhor resultado da técnica, pois nem todas soluções são idênticas. Alguns desafios são cruciais para obter um bom resultado da técnica de extração, como grau de automação, escalabilidade em processar um grande volume de dados em tempo relativamente curto e privacidade para manter informações e não afetar as pessoas com seus dados pessoais seja por dano, extravio ou divulgação de suas informações.

Pode-se classificar a extração de dados em duas possibilidades: *Web Crawler* e *Web Scraping*. *Web Crawler* é o método usado em motores de busca como *Google*, *Bing*, *DuckDuckGo*, entre outros (DESAI et al., 2017; KAUSAR; DHAKA; SINGH, 2013). Aplicações *Web Scraping* visam baixar automaticamente os

dados de uma página *web* e extrair informações especificas delas, seja para replicar em outro site ou para análise de dados (MITCHELL, 2015).

2.2 *WEB CRAWLER*: VISÃO GERAL

A técnica de *Web crawler* é conhecida por nomes como *robots, spiders e worms.* Um algoritmo utiliza as técnicas de *web crawler* para vasculhar a *Web* seguindo estrutura de hiperlinks para atingir diferentes sites, indexando e armazenando as páginas num repositório para consultas futuras de usuários (PARVEZ et al., 2018). Em síntese *web crawler* é um programa que abre uma determinada página de um site e vai vasculhando todo e qualquer *link* presente nesta página, criando um índice, permitindo posteriormente recuperar tais dados (KAUSAR; DHAKA; SINGH, 2013). O uso de *web crawlers*, permite simplificar a tarefa de pesquisar informações, sejam páginas ou arquivos, conforme às necessidades do usuário, pois estas informações são mantidas para serem processadas por um motor de busca, assim reduzindo o tempo de resposta da pesquisa do usuário (KAUSAR; DHAKA; SINGH, 2013).

Um exemplo de um trabalho com um volume relativamente grande de informações é apresentado por FETTERLY, MANASSE e NAJORK (2004). Nesse trabalho, mais de 150 milhões de páginas foram monitoradas durante onze semanas para encontrar alterações e assim determinar um grau de mudança de páginas da *web.*

A técnica de *web crawler* necessita atender alguns pré-requisitos:

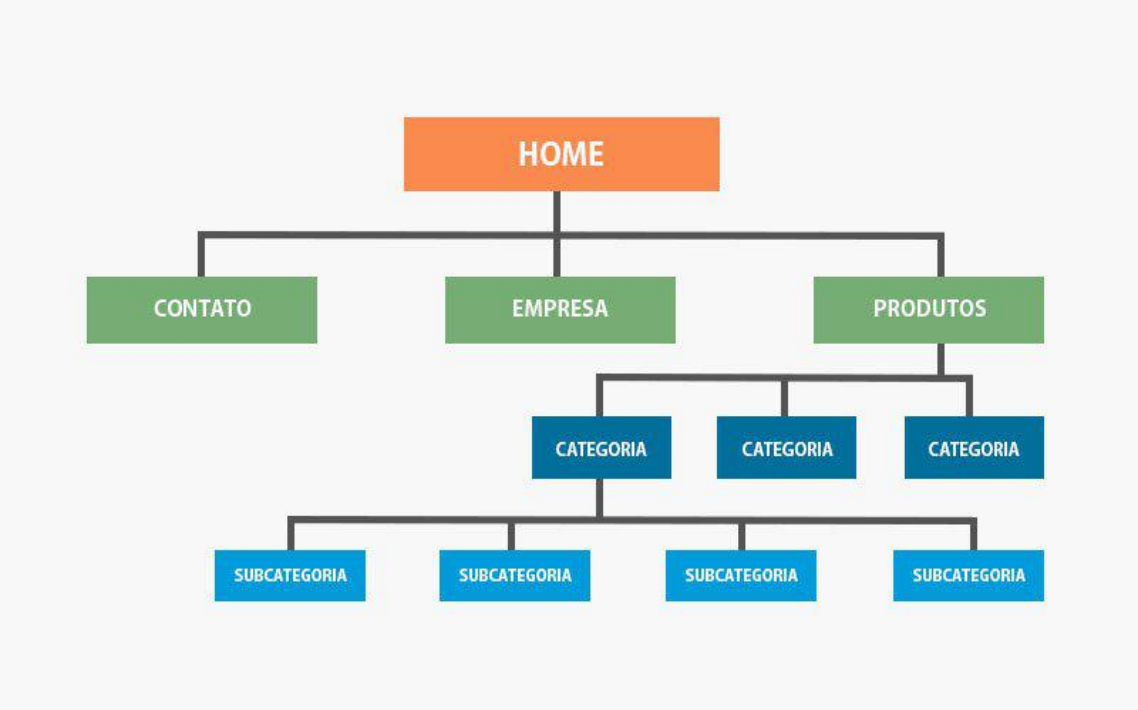
- Robustez: suportar sites com erros de sintaxe HTML, apresentando links perdidos ou links formando ciclos.
- Polidez (*Politeness*): velocidade de acesso a um site, e só fazer a indexação de sites que permitam isso.
- Distribuição: o *crawler* deve ser distribuído dentre várias máquinas. Com finalidade de adquirir vantagens na implantação, programação e depuração.
- Escalabilidade: a arquitetura do *crawler* deve permitir balanceamento da taxa de indexação, inserindo mais máquina e largura de banda.

- Eficiência: uso inteligente dos recursos, como memória, largura de banda e processadores.
- Qualidade: encontrar às melhores páginas para indexação, sendo assim páginas de alta qualidade ou *page rank* deve ser indexado por primeiro.
- Atualizado: sempre deve estar buscando novas páginas. E aceitar novos tipos de dados, como XML (*Extensible Markup Language*), protocolos como FTP (*File Transfer Protocol*), etc.

Cada um desses requisitos traz um impacto diferente sobre o funcionamento de um *crawler* (DESAI et al. 2017). Existe rastreadores robustos preparados para grandes massas de dados e um grande número de páginas web de diferentes sites e assunto. Rastreadores focados se destacam por armazenarem páginas pertencentes a um determinado assunto.

Para uso dos *crawlers* é preciso definir qual modo de busca a técnica irá utilizar para encontrar novas páginas. Um conjunto de páginas web forma um grafo, como ilustrado na Figura 1. Isso leva a utilizar algoritmos de busca em grafos para resolver o problema. Existem opções de algoritmo como o melhor primeiro (*Best First*), algoritmos de busca em largura (BFS) e profundidade (DFS). Os mais utilizados são os algoritmos de busca em largura (BFS) e profundidade (DFS) (SHARMA e GUPTA 2015).

Figura 1 - Exemplo de *Site Map*



Fonte: IDEAL MARKETING (2019)

2.3 *WEB SCRAPING*: VISÃO GERAL

A expansão da Internet trouxe consigo inúmeras fontes de informação, textos, imagens e outros tipos de mídia. Com esta grande massa de dados, obter alguns conteúdos que estão presentes na *web*, se tornou valioso para pessoas e empresas. Para obter estas informações presente na Internet, a técnica de *web scraping* surgiu. Ela recebe nomes como *web harvesting* ou *web data*. A técnica tem como finalidade tornar informações especificas de sites disponíveis para processamento (SCHRENK, 2012).

> *Web scraping* vem se tornando uma importante ferramenta para obter o conteúdo da *Web* usando um *softbot* adotado por muitos mecanismos de pesquisa. Mas principal função dele se concentra na transformação de conteúdo da *Web* não estruturado em dados estruturados que podem ser armazenados e analisados em um banco de dados ou planilha. (MALIK; RIZVI, 2011).

A técnica *web scraping* tenta simular um sistema genérico que extrai informação de sites, capaz de simular o uso humano, seja via protocolo *HTTP*

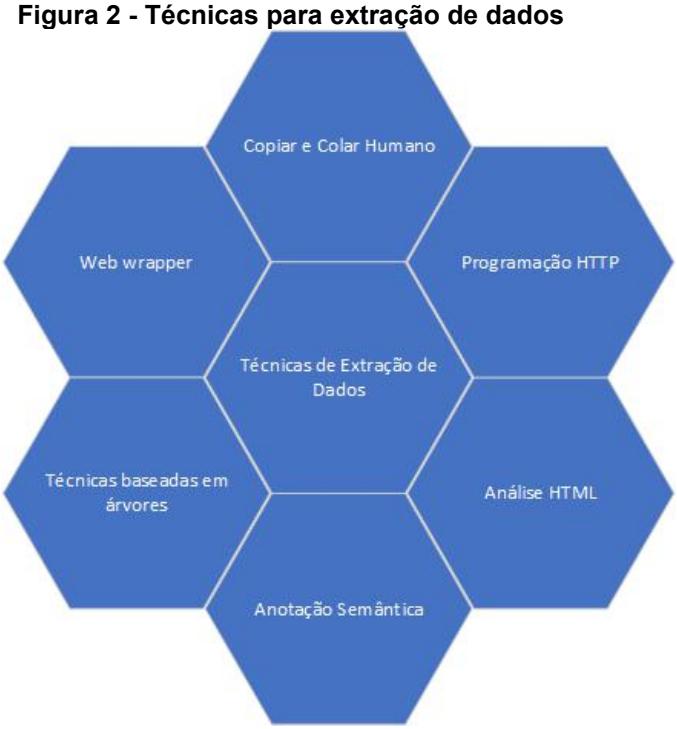
(*Hypertext Transfer Protocol*) ou ações automatizadas de navegadores (NEIL, 2016). Esta técnica vem sendo muito utilizada em inteligência para negócios, *marketing* digital e mineração de dados (CHEN et al., 2015; NEIL, 2016).

> As razões pelas quais uma empresa adota o WS (*Web Scraping*) como ferramenta são, construir um motor de busca vertical específico, buscar produtos e preços para comparação, recrutamento de talentos, monitoramento de marca, verificação de anúncios (*marketing*), recolha de anúncios imobiliários, para fins de pesquisa, coleta de grande volume de dados em sites de mídia, *scraping* para criar novos sites, geração de leads em sites de vendas (GUIMARÃES, 2018).

A técnica *web scraping* pode substituir trabalho braçal de acessar a inúmeros sites de empresas de venda para verificar o preço do produto ou comentários de outros utilizadores do serviço ou produto pesquisado para reunir informações. Atende também empresas, para extração de sugestões, elogios e críticas aos seus produtos, até mesmo obter informações para de se comunicar com o cliente, seja por *e-mails*, telefones, endereços para um *marketing* direto.

Na maioria das vezes, o protocolo *HTTP* é utilizando para a técnica de *web scraping,* simulando a navegação humana, realizando acessos automáticos a um ou vários sites para se obter informações desorganizadas do conteúdo *HTML* (*Hypertext Markup Language*), extraindo estes dados para serem processados por pessoas ou empresas (GLEZ-PEÑA et al., 2013). Os algoritmos que utilizam a técnica de *web scraping* permitem trazer informações de várias páginas que o usuário deseja. Para obter estes dados eles utilizam técnicas ilustradas na Figura 2.

Figura 2 - Técnicas para extração de dados



Fonte: Adaptado de PARVEZ et al. (2018)

A técnica copiar e colar (*Human copy and paste*) é considerada a primeira técnica para extração de dados (PARVEZ et al., 2018). Mas ela não é adequada para trabalhos massivos, pois está sujeita a um grande índice de erro humano, além de requisitar uma quantidade de pessoas e tempo para realizar a ação de extrair informações, tornando impraticável a técnica para um grande volume de dados (MALIK; RIZVI, 2011; PARVEZ et al., 2018).

Para páginas *web,* existem técnicas como *HTTP programming*, *HTML parsing* e *Tree-Based Techniques*, para tornar mais fácil a obtenção do conteúdo de formas simples e até genérica, possibilitando o funcionamento para um ou mais sites ao mesmo tempo (MALIK; RIZVI, 2011; PARVEZ et al., 2018). As técnicas serão detalhadas nas seções a seguir.

2.3.1 *HTTP Programming*

O protocolo *HTTP* permite tratamento de respostas cliente-servidor. Para extrair informações de uma página *web* o protocolo se tornou peça fundamental, pois ele permite prever ações e exceções para cada página visitada conforme a resposta do protocolo.

O *scraping* necessita trabalhar com os *status code* do protocolo *HTTP* para ter um bom funcionamento. Pois ele permite verificar quando deve-se tentar novamente a requisição para uma página (código 408), acesso a páginas inexistentes (código 404), erro de resposta (código 403), e requisições bem-sucedidas (código 200). Com isso pode-se ter certeza que será obtida a informação desejada.

2.3.2 *HTML Parsing*

A técnica *HTML Parsing* é um processo que se inicia em solicitar uma página *web*, receber o *status code* do protocolo *HTTP* para a requisição feita, obter o *HTML* e extrair informações necessárias (GLEZ-PEÑA et al., 2013). O *HTML* recebido pelo servidor pode ser comparado a um texto, alguns podem estar estruturados, conter palavras chaves que irão auxiliar na extração de dados, tornando uma palavra ou frase facilmente encontrável e identificável. A técnica *HTML parsing* realiza a transformação do *HTML* retornado pelo servidor, em um objeto estruturado e devidamente marcado, podendo assim encontrar a informação necessária. A Figura 3 monstra como seria difícil encontrar algum texto ou identificação única no meio do *HTML*. A técnica de *parsing* permite o algoritmo transformar o *HTML* em objetos, tornando-se fácil buscar qualquer informação em meio ao *HTML* de uma página.

Figura 3 - Exemplo *body response HTML*

```
<div id="divPrincipal" style="{width: 95%}" class="noprint">
<fieldset id="fsPrincipal" class="fsportal"><legend class="fsportal">Seja Bem-vindo ao Portal do Aluno</legend>
<div class="fsbutton" id="fsbutton" onclick="fshideshow();" onmouseout="className='fsbutton';" onmouseover="className='fsbutton_on';">[-]</div>
<div id="div_fsInterno"><table width="100%" border="0px" cellpadding="0px" cellspacing="0px" style="{vertical-align: middle;}" >
<tr><td rowspan="2" width="180px">
</td><td align="center"><font size="3"><b>Aluno: </b>1700243 - MATHEUS APARECIDO DA SILVA ROBERTO</font></td>
<td class="nav2off" onmouseover="className='nav2on';" onmouseout="className='nav2off';" onclick="window.location.href='mpmenu.pcLogout'" >
<table><tr><td><img class="imagelink" height="22" width="22" alt="sair do sistema" src="/imagem/sair.png"/></td>
<td><font size="1">Desconectar-se</font></td></tr></table></td></tr>
<tr><td align="center"><form name="cursos" id="cursos" action=""><center>
<font size="3"><strong>Curso:</strong> Bacharelado Em Ciência Da Computação - <strong>Situação:</strong> Regular</font>
</center>
<SCRIPT>jsCursos = [ { linha:1, curscodnr: 39, tpcurcodnr: 2, alcuordemnr: 1}];</SCRIPT>
</form></td><td class="nav2off" onmouseover="className='nav2on';" onmouseout="className='nav2off';"
onclick="window.location.href='mpAlteraSenhaAluno.inicio'">
<table><tr><td><img class="imagelink" height="22" width="22" alt="alterar senha" src="/imagem/password.png"/></td>
<td><font size="1">Alterar Senha</font></td></tr></table></td></tr></table></div></fieldset>
<div id="div_CarregaAjaxMenu">
<SCRIPT>AjaxSelecionaCurso(1);</SCRIPT>
</div></div>
```

Fonte: Autoria Própria

A técnica de *HTML parsing*, além de ser rápida, traz grande facilidade aos desenvolvedores para encontrar informações, *links* e imagens. A técnica tem grande

aprovação pelos programadores, pelo seu *design* simples, velocidade de processamento e capacidade de resolver *HTML* (PARVEZ et al., 2018). Existem várias bibliotecas que auxiliam o programador a obter estes resultados. Projetadas conforme a necessidade dos desenvolvedores, no qual cada programador utiliza e melhora conforme a dificuldade particular.

Algumas bibliotecas são bem difundas no universo dos desenvolvedores, algumas linguagens de programação apresentam mais de uma biblioteca para sua utilização, e cada uma com sua vantagem. São apresentadas a seguir algumas bibliotecas.

2.3.2.1 Biblioteca *Beautiful Soup*

A biblioteca *Beautiful Soup* (RICHARDSON, 2015) desenvolvida em *Python* (PYTHON SOFTWARE FOUNDATION, 2019) foi projetada para extrair *tags* de páginas *web* que utilizam a linguagem de marcação *HTML*. A biblioteca se torna uma ferramenta poderosa pois dispõem de recursos como (MITCHELL, 2015; RICHARDSON, 2015):

- Facilidade para navegar, modificar e pesquisar numa árvore de comandos, podendo utilizar ferramentas já programadas para dissecar um documento e extrair a informação necessária sem muito esforço.
- Conversão de documentos automática para UTF-8, então não a necessidade de se preocupar qual idioma está escrito a página, a biblioteca consegue identificar sem grandes problemas.

A biblioteca *Beautiful Soup* tenta simplificar o *HTML*, pois possibilita formatar e organizar páginas *webs* desorganizadas, corrigindo *HTML* inválido e apresentado objetos em *Python* (MITCHELL, 2015). Além disso existem algumas ferramentas já programas na biblioteca que torna mais rápido o desenvolvimento, que possibilitam extrair *links*, isolar títulos e textos do *HTML*, facilitando a extração do dado (PARVEZ et al., 2018).

2.3.2.2 Biblioteca *Jsoup*

A biblioteca *Jsoup*, foi desenvolvida na linguagem *Java*, para facilitar a extração de informações contidas em páginas *HTML*. A ferramenta apresenta recursos para se obter e manipular informações, utilizando os melhores métodos de seleção de *tags HTML*. O destaque da biblioteca se dá pela facilidade de se extrair e analisar dados em *HTML* a partir de uma *URL* (*Uniform Resource Locator*), arquivo ou *string* (WANG et al., 2016).

2.3.3 *Tree-based Techniques*

A extração de dados da *Internet* explora a semiestruturarão de páginas *web*. As páginas podem ser representas como árvores ordenadas e rotuladas, nas quais as *tags* de sintaxe da linguagem *HTML* seriam os rótulos, e a hierarquia da árvore seria definida pelos diferentes níveis de alinhamento dos elementos que formam a página *web* (FERRARA et al., 2014).

A representação de *tags* e elementos é denominada de *DOM* (*Document Object Model*) (FERRARA et al., 2014; PARVEZ et al., 2018). A *DOM* se utiliza de palavras chaves especificas definidas pela linguagem de marcação *HTML*. O navegador interpreta os elementos específicos da página *WEB* (por exemplo, hiperlinks, botões, imagens e assim por diante), assim como texto livre (FERRARA et al., 2014). Com essa estrutura definida, pode-se explorar a DOM para extração de dados da web com algumas técnicas que serão abordadas a seguir.

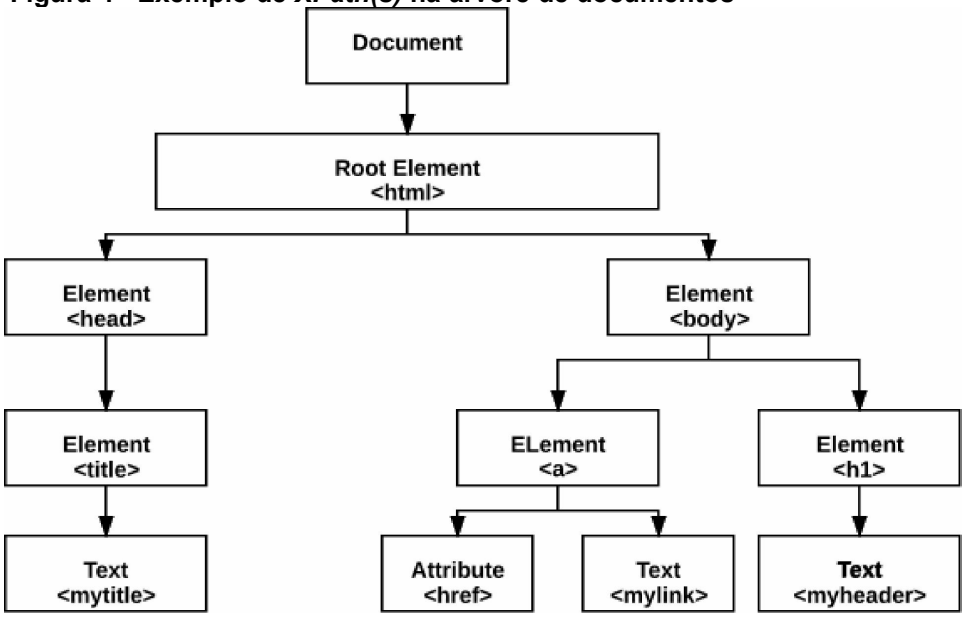
2.3.3.1 Endereçamento de elementos na árvore do documento (*XPath*)

A grande vantagem de se utilizar a *DOM* (*Document Object Model)*, é poder transformar o conteúdo em texto, e utilizar a estrutura do documento da página *web* e *tags* para extrair a informação. O *XPath* é uma ferramenta que identifica *tags HTML*, organiza a estrutura da página de forma parecida a linguagem *XML* e tem uma grande aceitação pela literatura. A ferramenta, por outro lado, tem um problema que é a falta de flexibilidade, a sintaxe precisa estar exatamente relacionada a

página *web*, então qualquer mudança na estrutura da página pode causar falha na execução da ferramenta (FERRARA et al., 2014).

São explorados recursos da árvore *DOM* utilizando os recursos da linguagem *XML* possibilitando utilizar inúmeros critérios. A Figura 4 serve de base para apresentar duas abordagens para o *XPath*:

Figura 4 - Exemplo de *XPath(s)* na árvore de documentos



Fonte: PARVEZ et al. (2018)

1. Seleção de um item único: o *XPath* é utilizado para poder identificar um único item na Figura 4 com o código de exemplo presente na Figura 5 tem-se um único item no documento *HTML*.

Figura 5 - Código de exemplo *XPath* para obter um único elemento

```
/document[1]/html[1]/body[1]/h1[1]/my header
```

Fonte: Autoria Própria

2. Seleção de vários itens: para encontrar várias ocorrências do mesmo elemento *tag HTML* é utilizado o código da Figura 6. Identificam-se múltiplas instâncias do mesmo tipo de elemento compartilhado naquele nível hierárquico da página *web*, baseado na estrutura da Figura 4.

Figura 6 - Código de exemplo para *XPath* para obter vários elementos

```
/document[1]/html[1]/body[1]/a[2]
```

Fonte: Autoria Própria

2.3.3.2 Algoritmo de distância para edição de árvore de correspondência

O algoritmo de distância para edição de árvore de correspondência é utilizado para converter uma árvore em uma nova árvore com às operações de edição que fornecem uma sequência de custo mínimo. Deve-se levar em consideração três operações para edição da árvore (PARVEZ et al, 2018):

- Remover um nó existente e conectar seus filhos ao pai obedecendo à hierarquia.
- Inserir um novo nó entre um nó existente e uma continuação de filhos consecutivos deste nó.
- Rotular novamente um nó.

Deve-se atribuir um custo em cada operação de edição para poder editar uma árvore. A distância da árvore se dá pela sequência de custo mínimo para sua edição. Para obter uma resolução computacional eficiente, o algoritmo de distância de edição de árvore de correspondência se dá pelo algoritmo de árvore de correspondência direta ou simples. O custo computacional para algoritmo de árvore de correspondência simples é O(nós(A) * nós(B)) (PARVEZ et al, 2018).

O algoritmo tem um baixo custo computacional, quando aplicado a árvores de *HTML*. Porém, existem problemas no uso deste algoritmo, um deles é que ele não permite à permutação de nós e nem é possível ter correspondência em diferentes níveis hierárquicos.

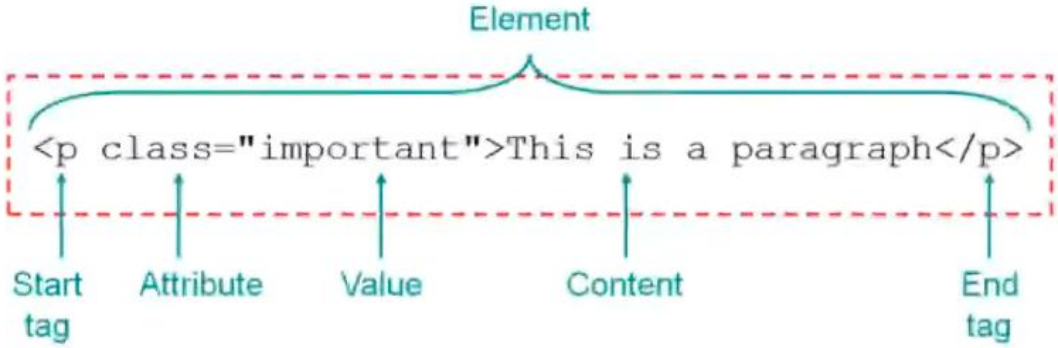
2.3.3.3 *CSS Selector*

Outra técnica para seleção de elementos é conhecida como *CSS Selector.* Ele torna mais prática a seleção de elementos *HTML* sem a preocupação existente no *XPath*, que seriam dependências da estrutura do documento. Neste caso a

identificação de informações se dá a partir de atributos do documento, como “*class*”, “*type*” ou “*id*”.

A Figura 7 demonstra a composição de um elemento, a *tag,* atributos, valores, conteúdo, o que permite selecionar o mesmo elemento de diferentes maneiras. Tornam-se práticas futuras alterações da sintaxe de seleção, e podendo ser mais rápido que a técnica *XPath* variando conforme estrutura e sintaxe (FAROOQ; HUSAIN; SUAIB, 2018; YIN; HE; LIU, 2018).

Figura 7 - Estrutura de um elemento HTML



Fonte: BATISTA (2017)

A Figura 8 apresenta a identificação de formas para selecionar os elementos, seja por atributos *id* ou *class*, a estrutura de *tags* “*div* > *div* > *span*”, tornando-se uma ferramenta poderosa para diferentes sites e totalmente legível para outros programadores (YIN; HE; LIU, 2018).

Figura 8 - Exemplo de uma estrutura HTML

```
1  <div>
2      <div>
3          <span>UTFPR</span>
4          <p id="texto" class="center">Hello World!</p>
5      </div>
6  </div>
```

Fonte: Autoria própria

2.4 GEORREFERENCIAMENTO: VISÃO GERAL

Existem casos que informações atreladas a localização espacial são necessárias, seja para situações cotidianas como traçar uma rota, verificar condições de trânsito e clima da região, ou situações de estudo científicos seja para saúde, ciências sociais ou ciências ambientais e naturais (OZIMEK; MILES, 2011).

> Qualquer tipo de documento que seja relevante para um local geográfico em particular é um exemplo de um recurso georreferenciado. Georreferenciamento foi inicialmente preocupado principalmente com recursos como mapas, fotografia aérea e imagens de satélite, cujo conteúdo é distribuído em toda a Terra. (SOLINA; RAVNIK, 2010).

Georreferenciar uma informação é o termo utilizado para designar a localização geográfica de alguma entidade de uma base de dados geográfica.

> Geocodificação é o processo de converter endereços ou locais (como "3600 Market Street, Filadélfia, PA" ou simplesmente "Filadélfia, PA") em coordenadas geográficas (como [39.95581, -75.19466] ou [39.95324, -75.16739]). Uma vez conhecidas as coordenadas geográficas, os dados podem ser mapeados e as relações espaciais entre os pontos de dados podem ser incorporadas às análises. (OZIMEK; MILES, 2011).

É possível identificar no cotidiano o uso de informações geográficas, localização espacial, como encontrar, restaurantes, hospitais, postos de gasolina nas proximidades do usuário. Mas também existem aplicações cientificas que utilizam a localização espacial. Para isso são utilizados programas especializados para manipular informação espacial, para obter resultados que podem auxiliar a projetar uma cidade, a prevenir casos de doenças, entre outras situações. Estes programas são denominados SIG (Sistema de Informações Geográficas).

Um problema que se apresenta para uso de SIG é falta de bases de dados de fácil acesso (OLIVEIRA FILHO; SILVA, 2010; SILVA, 1998). A alternativa encontrada para contornar essa questão são serviços em que se pode consultar localidades e obter informações espaciais (FLORCZYK et al., 2010; OZIMEK; MILES, 2011; SOLINA; RAVNIK, 2010; WILSON; SWIFT; GOLDBERG, 2008). Em razão do interesse em serviços de geocodificação neste trabalho, as seções a seguir abordam alguns tópicos a respeito.

2.4.1 SIG

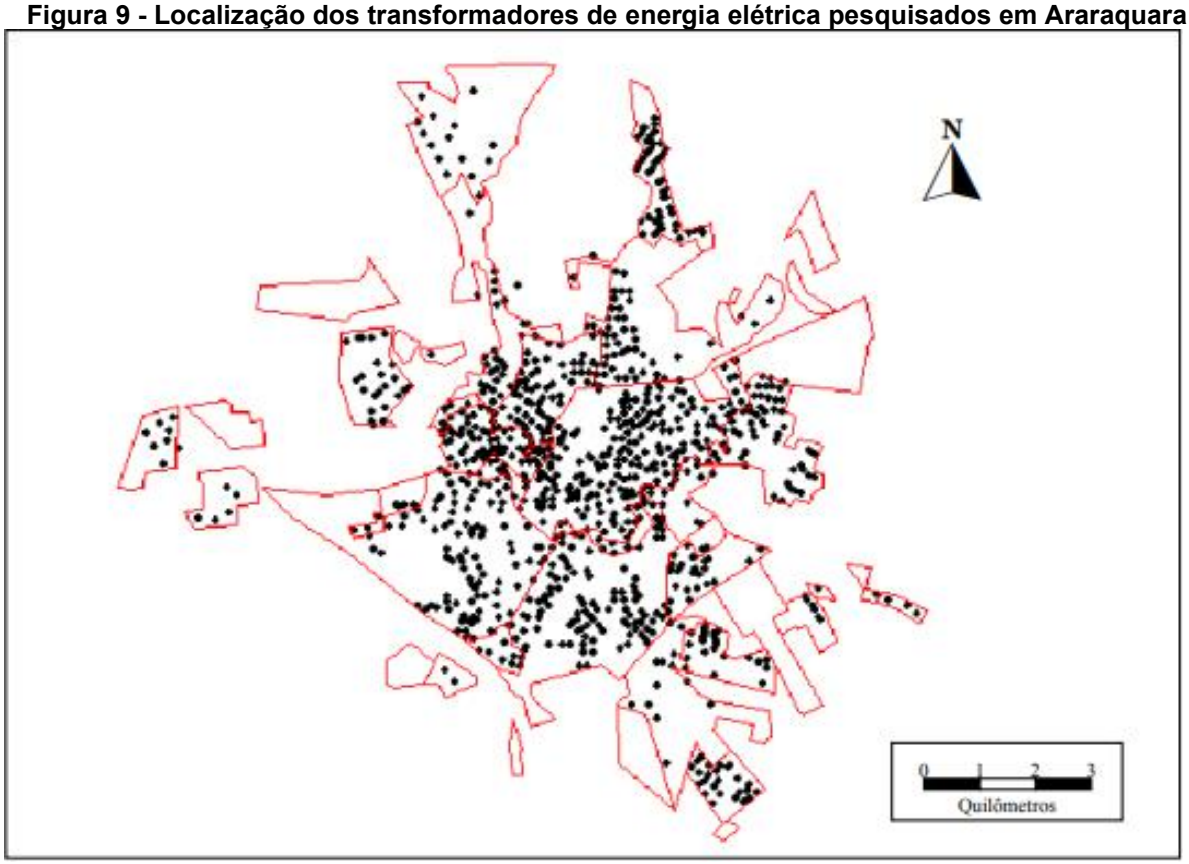
Um sistema de informação geográfica (SIG) é composto por um conjunto de programas computacionais, que integram, equipamentos e pessoas com objetivo de coletar, armazenar, recuperar, manipular, visualizar e analisar dados espacialmente referenciados a um sistema de coordenadas conhecido (FITZ, 2008). SIG é definido como um sistema de apoio a decisões, ele engloba uma lista de funções automatizadas com eficácia em armazenar, comandar, transformar e visualizar dados espacialmente referenciados.

> [...] pois ainda que seja possível o processamento de dados espaciais em meio analógico, a computação eletrônica redefiniu os SIG, caracterizando-o como uma tecnologia desenvolvida em softwares e hardwares de processamento de dados de informação geográfica. O potencial dessas tecnologias reside na estrutura de suporte à decisão e análise espacial, traduzindo o raciocínio do especialista quando esse faz referências aos fatos e elementos localizados no espaço. (FERREIRA; RAFFO, 2012)

SIGs encontram aplicações em problemas que envolvem tomadas de decisões e apoio para melhoramento de uma determinada localidade. Um exemplo da utilização de SIG é a montagem de um plano de transporte na cidade de Araraquara, São Paulo (SILVA, 1998). Informações são obtidas da prefeitura da cidade, para determinar localidades residenciais, centro comerciais e industriais. Também são utilizados dados da companhia responsável pelo fornecimento de energia da cidade, para determinar a classe social conforme o uso de energia da residência.

Com a base de dados da quantidade de energia utilizada por cada residência, utilizou-se um SIG para determinar a localização dos transformadores de energia com base na utilização das residências. A Figura 9 apresenta um mapa com estes pontos. O SIG ajuda na tomada de decisão para determinar ruas, pontos de ônibus e quantidades de linhas conforme a necessidade da localização espacial da cidade (SILVA, 1998).

Figura 9 - Localização dos transformadores de energia elétrica pesquisados em Araraquara



Fonte: SILVA (1998)

SIG também é utilizado para planejamento urbano. A ferramenta permite identificar locais para construção residencial, industrial e comercial. Além de localidades que se encontram solo para plantio, mananciais de água e reservas florestais, possibilitando utilizar SIG para apoio de decisões na estruturação e crescimento da cidade (COELHO, 2009).

> As vantagens do uso do SIG como instrumento de apoio (e não de decisão) na elaboração de Planos Diretores Municipais, como também na gestão territorial são inúmeras, a começar, pela eficiência, precisão e qualidade da informação especializada; por possuir uma Base de Dados Espaciais que possibilita armazenar, consultar, exibir, alterar e excluir informações georreferenciadas; permitir a criação de cadastros; gerar relatórios, gráficos; processar informações; calcular áreas; realizar estudos temporais, simulações. (COELHO, 2009)

Outra aplicação é utilizar SIG para criar planos de arborização para cidade, criando um modelo, no qual é apresentado quais árvores correm um menor risco de

pragas. Não afetam serviços urbanos e embelezam passeios ao ar livres e ruas da cidade (OLIVEIRA FILHO; SILVA, 2010).

No sistema de saúde, SIG vem ganhando força como ferramenta para auxiliar a controlar doenças, encontrar fatores para epidemias, determinar quais doenças são mais propensas por localidade. A ferramenta pode auxiliar quais fatores podem ser determinísticos e quais fatores podem influenciar uma região sofrer um determinado surto ou epidemia de uma doença, podendo assim criar planos para controle destas doenças junto aos órgãos governamentais (SILVA, 2017).

> O uso de SIG vem crescendo e tornando-se muito importante na área de saúde devido ao surgimento de grandes epidemias como a do ebola, em 2015, na África. Em um futuro próximo, seu uso será indispensável em qualquer estudo sobre doenças e epidemias, ajudando no combate e erradicação das doenças e servindo como material de apoio para os governos na elaboração de planos e medidas no controle a doenças. (SILVA, 2017)

Serviços para georreferenciação são utilizados para montar um mapa de taxa de mortalidade da cidade Rio de Janeiro (SILVEIRA; OLIVEIRA; JUNGER, 2017). Com este serviço é possível georreferenciar moradias de pacientes que deram entrada em hospitais e faleceram, podendo assim criar planos para acompanhamento e campanhas para melhorar a saúde dos moradores baseado na residência de pacientes que faleceram (SILVEIRA; OLIVEIRA; JUNGER, 2017).

As aplicações apresentadas demonstram várias formas de utilização dos SIG. Para planejamento, apoio em decisões, criação de planos para melhoramento da cidade. Em todos os trabalhos a necessidade de uma base de dados georreferenciadas é fundamental para que a ferramenta SIG seja utilizada de forma correta.

2.4.2 Serviços de georreferenciamento

Algumas informações e objetos necessitam de sua localização espacial para serem analisados e pesquisados. Serviços de georreferenciamentos se apresentam para sanar a problemática. O uso destes serviços é uma forma de transformar endereços em posições espaciais, em sua maioria de casos latitude e longitude,

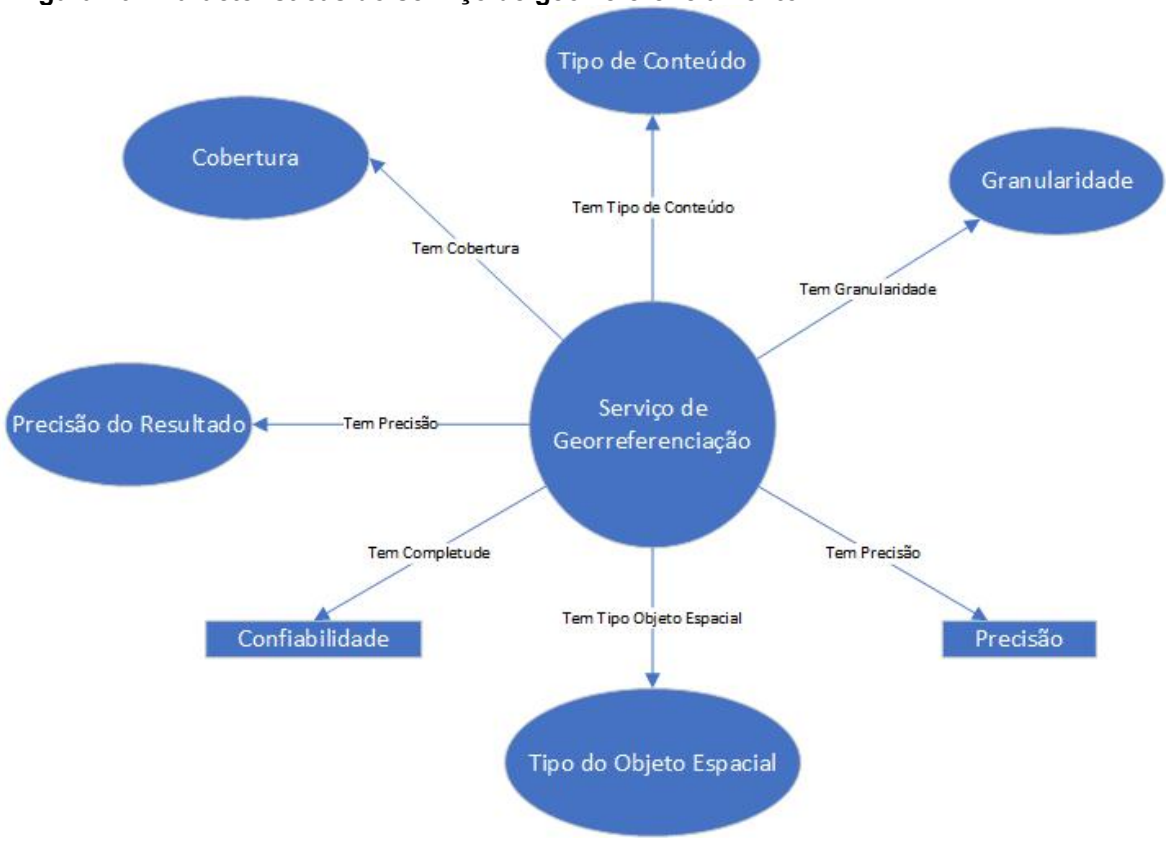
para serem utilizados por sistemas GPS. Em algumas aplicações, estes serviços de localização espacial são utilizados de forma robusta para melhorar a interação do usuário com a plataforma, como por exemplo:

- *Foursquare* (FOURSQUARE, 2019): recomenda locais próximos aos usuários conforme sua localização e mostra onde estão seus contatos pessoais.
- *TripAdvisor* (TRIPADVISOR LLC, 2019): mostra hotéis e restaurantes nas proximidades em um mapa e classifica-os baseado em notas dos clientes que utilizaram.
- *Google Maps* (GOOGLE MAPS, 2019): com o resultado da posição do usuário e/ou rota, informa sobre engarrafamentos, bloqueios de estradas ou perigos.
- *Wallapop* (WALLAPOP, 2019): leva em consideração a localização do usuário e dos fornecedores de produtos para mostrar os produtos mais relevantes nas proximidades para o usuário.
- *PokemonGo* (NIANTIC, 2019): o jogo de realidade aumentada é baseado especificamente em tecnologias de geolocalização.

Além de aplicativos e de *marketing*, a georreferenciação vem sido usada para revolucionar o campo da cartografia, podendo assim reconstruir planos cartográficos que estão em museus e bibliotecas, para auxiliar em estudos (CASCÓN-KATCHADOURIAN; RUIZ-RODRÍGUEZ; ALBERICH-PASCUAL, 2018).

A Figura 10 ilustra situações que devem ser levadas em consideração para escolher um provedor para georreferenciar informações. A primeira é cobertura que a ferramenta oferece, alguns se limitam as pequenas regiões ou cidades. A segunda é o tipo do conteúdo que o serviço oferece. Este conteúdo fica estritamente dependente de qual recurso geográfico que o serviço fornece. O terceiro é o resultado espacial da informação, alguns serviços de georreferenciação podem resultar objetos espaciais, como pontos, polígonos ou entidade 3D (FLORCZYK et al., 2010).

Figura 10 - Características do serviço de georreferenciamento



Fonte: Adaptado de FLORCZYK et al. (2010)

A comparação entre serviços de georreferenciamento se torna necessário, para que o serviço mais adequado ao problema seja escolhido. A cada projeto surge uma necessidade, como precisão, valor, limitações entre outros pontos (CIEPŁUCH et al., 2010; FLORCZYK et al., 2010; WILSON; SWIFT; GOLDBERG, 2008).

Existem duas ferramentas que servem como base para georreferenciação. A primeira é o *Geocoding API* (GOOGLE MAPS, 2019) uma ferramenta mantida pela empresa *Google* (GOOGLE INC, 2019). A segunda é a ferramenta do *Open Street Map (Nominatin)* (FUNDAÇÃO OPENSTREETMAP, 2019), mantida por voluntários e pela fundação *OpenStreetMap (OSMF)*. A forma de utilização das ferramentas acontece pelo o envio de endereços, textos e obtenção de dados geográficos, em maior parte latitude e longitude. Também é possível enviar coordenadas geográficas e receber como resposta endereços, facilitando o mapeamento de localidades graças às bases de dados destas ferramentas.

O serviço *Geocoding* (GOOGLE MAPS, 2019) oferecido pela empresa Google (GOOGLE INC, 2019), apresenta formas de utilizar mapas e serviços

cartográficos para vários sites ou empresas, a fim de manter uma base atualizada e de abrangência global, tornando assim o único meio de se ter informações geográficas em algumas localidades (CIEPŁUCH et al., 2010; SILVEIRA; OLIVEIRA; JUNGER, 2017; SZTUTMAN, 2014). O Google (GOOGLE INC, 2019) se esforça em deixar suas bases de dados atualizada, seja por imagens espaciais ou por intermédio de usuário que utilizam suas plataformas, *Google Maps*, rotas, locais, entre outros, tornando assim umas das bases espaciais mais bem atualizada do mundo (CIEPŁUCH et al., 2010; SZTUTMAN, 2014).

A ferramenta *Nominatim*, também tem como alvo apresentar coordenadas geográficas provenientes de endereços. A base de dados é mantida por voluntários, o que pode torna-la ela um pouco defasada em alguns locais (CIEPŁUCH et al., 2010). Mas se destaca pelo uso gratuito, que atrai alguns usuários a contribuir mais com a ferramenta.

2.5 CONSIDERAÇÕES DO CAPÍTULO

Neste capítulo foram apresentados conceitos sobre extração de dados da Internet, detalhando a utilização da técnica de *Web Scraping*, bibliotecas para auxiliar na extração, formas para selecionar e obter *tags HTML*. Foi abordado a importância de informações georreferenciadas, a utilização em SIG, quais são os métodos e serviços existentes para georreferenciar uma informação. Os conceitos trabalhados neste capítulo são necessários para a compreensão de como a ferramenta foi desenvolvida. No próximo capítulo será descrito cada fase e dificuldade encontrada para desenvolver a solução proposta nesta pesquisa.

3 DESENVOLVIMENTO

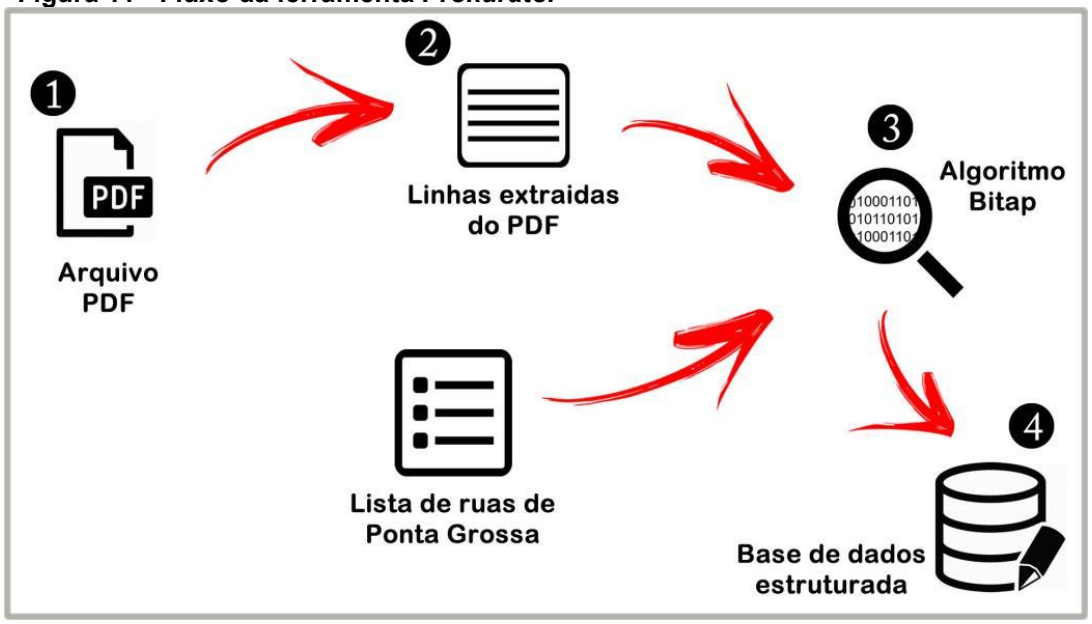
Este trabalho iniciou como uma nova versão de uma ferramenta para extração de dados de anúncios imobiliários a partir de textos de arquivos PDF (*Portable Document Format*) denominada *Prokurator* (BATISTA; XAVIER, 2018).

3.1 CONTEXTO DO PROBLEMA

O PPGCC (Programa de Pós-Graduação em Ciência da Computação) da UTFPR (Universidade Tecnológica Federal do Paraná), campus Ponta Grossa, desenvolve trabalhos relacionados à simulação urbana, e que necessitavam de bases de dados de anúncios imobiliários georreferenciados da cidade Ponta Grossa. Uma abordagem inicial para o problema procurou extrair dados de arquivos em formato PDF, por meio de uma ferramenta semiautomática (BATISTA; XAVIER, 2018).

O funcionamento do processo é resumido na Figura 11.

Figura 11 - Fluxo da ferramenta *Prokurator*



Fonte: BATISTA e XAVIER (2018)

Com a ferramenta elaborada, os autores tornaram mais rápido a extração da informação desejada, em comparação com o mesmo processo realizado manualmente. A saída da ferramenta utilizou o formato *JSON* (*JavaScript Object Notation*) e um exemplo é apresentado na Figura 12.

Figura 12 - Dados de saída da ferramenta *Prokurator*

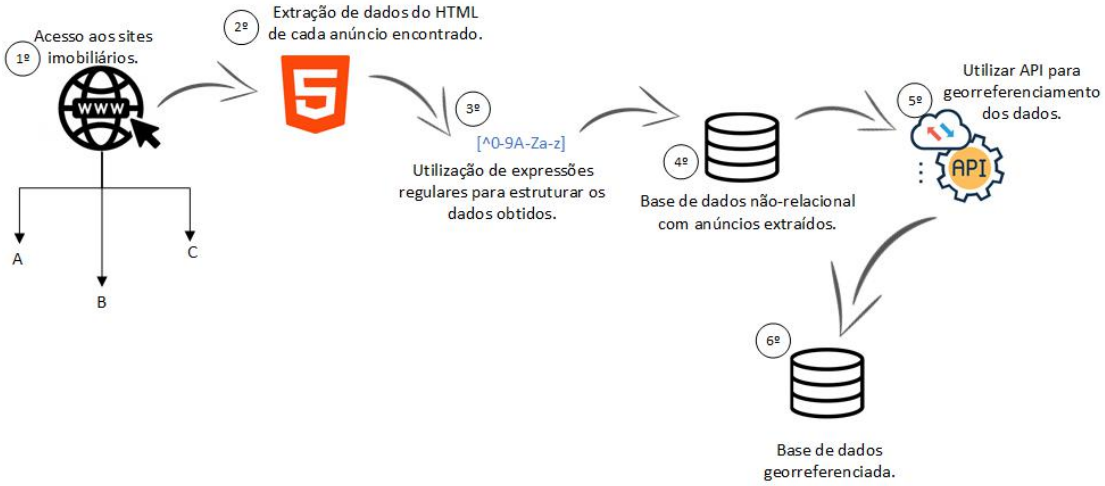
```
{
  "imoveis":[
    {
      "endereco":"Rua Comendador Miró",
      "preco":" 630.000,00",
      "data":"5/5/2018"
    },
    {
      "endereco":"Rua Arnaldo José de Moraes",
      "preco":" 100.000,00",
      "data":"5/5/2018"
    },
```


Fonte: BATISTA e XAVIER (2018)

A abordagem inicial apresentou algumas limitações. A base de anúncios ainda não era georreferenciada. Além disso, depois da realização do trabalho inicial as bases de dados existentes em *PDF* se tornaram menos relevantes que os anúncios criados na Internet, pois era difícil encontrar novos *PDFs* com anúncios imobiliários atualizados. Com o intuito de resolver os problemas, buscou-se a extração de informações de sites imobiliários, permitindo a obtenção de anúncios atualizados e em grande quantidade. Para georreferenciar os dados, partiu-se da ideia de utilizar ferramentas já existentes como, por exemplo: *Google Maps* e *OpenStreetMap*.

O funcionamento geral da solução proposta consiste em uma série de etapas ilustradas pela Figura 13.

Figura 13 – Fluxo do funcionamento da ferramenta



Fonte: Autoria própria

A primeira etapa consiste em acessar a Internet à procura de sites imobiliários para obter os anúncios. A segunda etapa, consiste em obter pedaços do *HTML* com os dados dos anúncios. A terceira consiste em transformar pedaços da estrutura *HTML* em texto e utilizar expressões regulares para minerar e estruturar os dados necessários. Na quarta etapa será o resultado estruturado da extração de anúncios em um arquivo *JSON* para utilização na próxima etapa. A quinta fase irá utilizar o resultado da extração e estruturar tentativas para georreferenciação utilizando uma *API (Application Programming Interface*). Para que na sexta etapa o resultado seja um arquivo de saída no formato *JSON* com todos anúncios georreferenciados. Cada etapa é descrita a seguir na implementação da ferramenta.

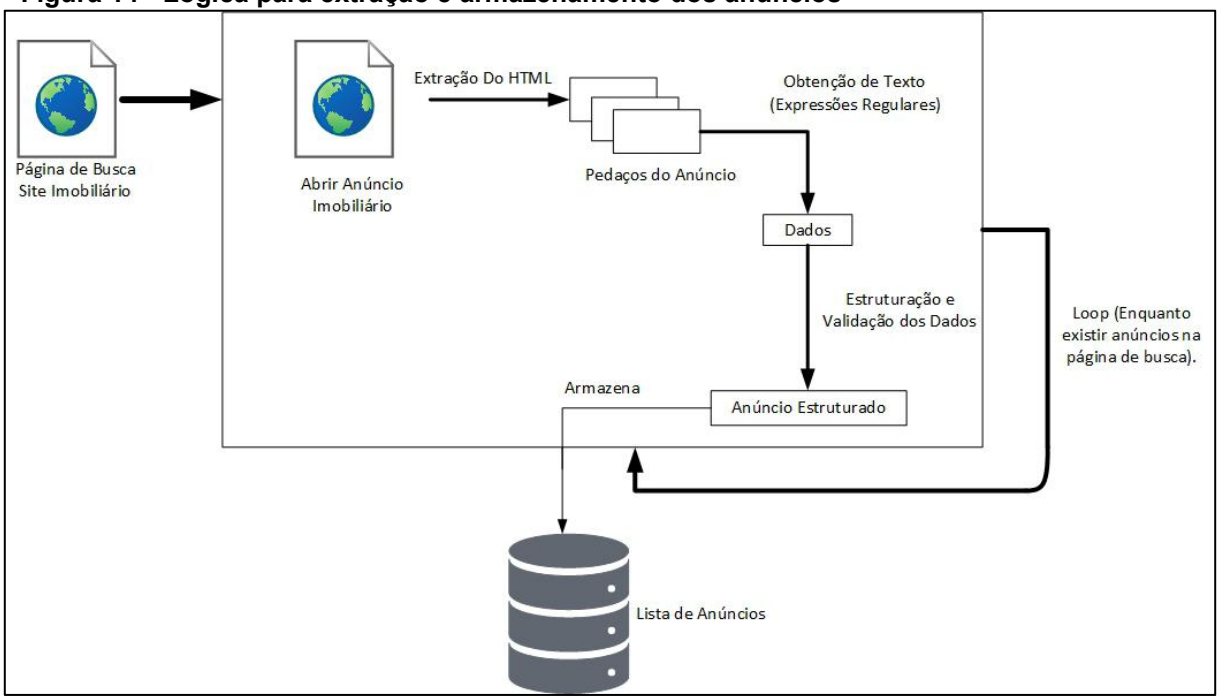
3.2 IMPLEMENTAÇÃO

O sistema todo foi escrito na linguagem *Python*. As razões para escolha desta linguagem foi pelo conhecimento prévio da linguagem e, principalmente, pela disponibilidade de bibliotecas que se encaixavam com o objetivo deste trabalho, tais como a requisição de páginas, extração de informações do *HTML* e estruturação de texto (MITCHELL, 2015), tornando prático e rápido a codificação deste trabalho.

3.3 OBTENÇÃO DE DADOS

Para obter os anúncios, é necessário fazer uma requisição à página de busca do site da imobiliária, e verificar anúncio por anúncio para obter os dados necessários. A Figura 14 apresenta o funcionamento desde a requisição da página de busca até o armazenamento do anúncio encontrado. Será detalhado cada parte da Figura 14, nas seções seguintes.

Figura 14 - Lógica para extração e armazenamento dos anúncios



Fonte: Autoria própria

3.3.1 Criando conexão com o site base

Para realizar a requisição da página de busca do site imobiliário, a biblioteca *requests-html* (REITZ, 2018) tornou-se necessária. A mesma permite criar uma conexão com a *URL* requisitada tendo como resposta os *status code* do protocolo *HTTP*, se a conexão com a *URL* for bem-sucedida, pode-se se obter a página *HTML* desejada. A biblioteca oferece ferramentas em potencial para auxiliar na programação, como a extração de *links* da página.

A Figura 15 mostra o uso da biblioteca *requests_html*. A chamada da função *session.get()*, recebe como parâmetro a *URL* que se deseja obter conexão.

Geralmente as imobiliárias tem mais de uma página de anúncios, por isso o parâmetro {i} é utilizado para fazer alternância entre as páginas. Com a resposta de sucesso da requisição da página, todos os *links* pertencentes naquela *URL* são extraídos, estes endereços de página remetem aos anúncios imobiliários, para que isso ocorra novamente a biblioteca *requests_html* é utilizada para obter todos os *links* disponíveis.

Figura 15 - Requisição de *URL* usando biblioteca *requests_html*

```
1   from requests_html import HTMLSession
2
3   session = HTMLSession()
4
5   while (i <= j):
6       print(f'{i} de {j}')
7       r = session.get(
8           f"https://www.conceitoimoveispg.com.br/busca/venda/cidade_ponta-grossa/pag_{i}")
9
10
11      data = r.html.absolute_links
```

Fonte: Autoria própria

3.3.2 Tratamento de erros e otimização

Sites podem apresentar erros como links quebrados, links causando laços (apontando para páginas já visitadas) e erros de sintaxe. Para evitar laços foi mantida uma lista de links já visitados pelo código. Antes de disparar novas requisições essa lista é verificada.

A estrutura de alguns sites continha informações não relacionadas ao problema tratado, como anúncios diversos, ou links para números de telefone. Para evitar a navegação para esses enlaces e a consequente perda de tempo, foram incluídas no código informações que filtravam links sem importância. Isso foi implementado usando expressões regulares.

A função responsável pela extração dos *links* utilizáveis é invocada após a biblioteca *requests_html* retornar todos as *URLs* disponíveis na página desejada.

3.3.3 Paralelização para obtenção de dados

Para otimizar o processo de extração de anúncios, foi utilizado a biblioteca *threading,* para dividir o processo de extração em partes. Cada *thread* teria um

pedaço da lista para percorrer, encontrar anúncios e extrair os dados necessários do *HTML* da página.

A função *imovel*, responsável pela requisição das páginas, é invocada logo após o processo de divisão do total de páginas que cada *thread* deve percorrer. Foram divididas em quatros *threads*, pois era o maior número de requisições simultâneas que alguns sites permitiam sem apresentar erro de resposta ou exceder o limite de requisições simultâneas por *IP* (*Internet Protocol address*). Portando foi mantido o padrão de quatro *threads* para todos os sites.

No caso da ferramenta desenvolvida nesta pesquisa, for utilizada em computador com um número menor de quatro núcleos, a própria linguagem *Python* á gerencia cada *thread* da ferramenta, utilizando da *Global Interpreter Lock (GIL)*. Responsável por distribuir tempo igualitário de quinze milissegundos para cada processo, a *GIL* define quando e qual processo deve pausar ou continuar. Exemplificando, a função *imovel* será dividida em quatros processos distintos, como serão feitas requisições a Internet e isso pode demorar mais que 15 milissegundos, a *GIL*, irá pausar a execução de um processo e iniciar outro, não deixando nenhum processo parado, otimizando o processo em situações que quatro núcleos não estejam disponíveis para a ferramenta.

3.4 EXTRAÇÃO DE DADOS

Para extração de dados é necessário abordar, a obtenção do *HTML* e a extração das informações contidas nele. Utilizando a técnica de *web scraping* será obtido às informações da página *web*, com as expressões regulares será estruturado o dado obtido, assim finalizando as fases, dois, três e quatro da Figura 13.

Cada imobiliária tem a estruturação de seus sites e anúncios de maneira diferente, então é necessário encontrar padrões de *tags*, atributos de cada elemento, seja *CSS,* identificadores de cada site. Quais funções e padrões de expressões regulares se adequam a cada site. Para que a etapa de extração de dados seja iniciada. Por meio da Figura 16 é possível analisar que cada site mantém endereço e valor de venda em locais diferentes.

Figura 16 - Exemplo de anúncios imobiliários de cada site

Fonte: Autoria própria

A Figura 17 ilustra a estrutura *HTML*, com suas *tags*, identificadores e atributos, peça chave para escolha dos seletores para extrair a informação necessária.

Figura 17 - Estrutura HTML dos anúncios imobiliários

```
<div class="">
        <input type="hidden" id="idStats" value="770466"/><input type="hidden" id="idStats" value="770466"/>              <style>
                .formWhatsAppButton a{background-color: #006d19 !important;color: #FFF !important; font-size: 15px !important; transition: 0.15s all ease-in;float: righ
8px;font-weight: 600;width: 100%;text-align: center;padding: 10px;border-radius:4px}
                    .formWhatsAppButton a:hover {background-color: #25D366 !important;}
                    .formWhatsAppButton .fa {margin-right: 3px !important;}
                </style>
                <div class="imovelv2 imovelToSell">
                    <div class="content">
                        <div class="imovelInfoMain">
                            <div class="imovelInfoHeader">
                                <div class="imovelOpening">
                                <div class="imovelTitle">
                                <h2><span>Apartamento à venda na Rua Octávio de Carvalho, Jardim Carvalho - Ponta Grossa/PR - Cód. 391692.002</span></h2>
                                <h1 itemprop="name"><strong>Edifício Maison 700</strong></h1>
                                <i class="fa fa-map-marker"></i> Rua Octávio de Carvalho, 700 - Apartamento 12 - Jardim Carvalho/Jardim Carvalho - Ponta Grossa/PR
                                </div>
                                <div class="price">
                                    <span>Valor de Venda</span><small>R$</small> 335.273<small>,29</small>                                </div>
                            </div>
                        </div>
```

Fonte: Autoria própria

É necessário identificar o número de páginas totais de cada site, assim é possível determinar o intervalo que cada *thread* irá percorrer. A biblioteca *Beautiful Soup* foi utilizada neste projeto para tornar mais rápida a extração de dados e pelas suas funções disponibilizadas (MITCHELL, 2015). Pela facilidade e fácil manutenção, o *CSS Selector* foi utilizado para extrair o pedaço *HTML* desejado.

3.4.1 Extraindo informações de cada site

Conforme a Figura 14 ilustra é necessário requisitar a página de busca de cada site, obter-se o total de páginas e iniciar o processo de extração. Cada site segue um padrão de paginação diferente para visualização de seus anúncios. O site da imobiliária A apresenta a quantidade total de páginas, tornando fácil a obtenção

de seu valor. A imobiliária B tem uma questão peculiar, que em casos de visualização do site com uma resolução menor há um *tablet* (768 *pixels)* é possível visualizar a informação com o total de páginas, em resoluções maiores está informação é omitida, como é um site responsivo a informação permanece no *HTML* da página, podendo obter a quantidade total de páginas, com o seletor de *tag* correto. O site C não apresentava está informação, em nenhuma resolução possível, então foi necessário obter o valor total de anúncios. Para calcular o total de páginas, foi extraído o total de anúncios e dividido pela quantidade média de anúncios de cada página, que no caso são trinta anúncios por página.

Por meio da Figura 18 é possível destacar o funcionamento da solução proposta nesta pesquisa, desde da requisição da página de busca, com a biblioteca *requests_html*. A extração da informação com a quantidade de páginas, com a biblioteca *Beautiful Soup*. Podendo assim dividir o processo de extração dos anúncios em quatro *threads*, assim inicializando a busca de anúncios em cada página das imobiliárias para obter a informação necessária.

Figura 18 - Função *main* do arquivo *find*

```
def main():
    r = session.get(
        'https://www.conceitoimoveispg.com.br/busca/venda/cidade_ponta-grossa')
    soup = BeautifulSoup(r.text, 'html.parser')

    elemeSelect = soup.select(
        'div.paginacao > ul > li:nth-child(6) > a')
    npag = elemeSelect[0].text
    npag = npag.replace('[', '')
    npag = npag.replace(']', '')
    n = int(int(npag) / 4)

    thread1 = threading.Thread(target=imovel, args=(1, n))
    thread2 = threading.Thread(target=imovel, args=(n + 1, 2 * n))
    thread3 = threading.Thread(target=imovel, args=(2 * n + 1, 3 * n))
    thread4 = threading.Thread(target=imovel, args=(3 * n + 1, int(npag)))

    threads = []
    threads.append(thread1)
    threads.append(thread2)
    threads.append(thread3)
    threads.append(thread4)

    for t in threads:
        t.start()

    for t in threads:
        t.join()
```

Fonte: Autoria Própria

3.4.1.1 Extração de dados do anúncio endereço e valor

Com a separação dos processos por *threads*, é necessário abrir cada *link* de anúncio e extrair a informação desejada, endereço e valor de venda. A Figura 19 demonstra o processo de *loop* da Figura 14, com a requisição da página de busca, a extração de todas *URLs* da página, a seleção de *links* relevantes, e a requisição das páginas contendo o anúncio imobiliário, para extração de endereço e valor conforme a seleção do *HTML* desejado.

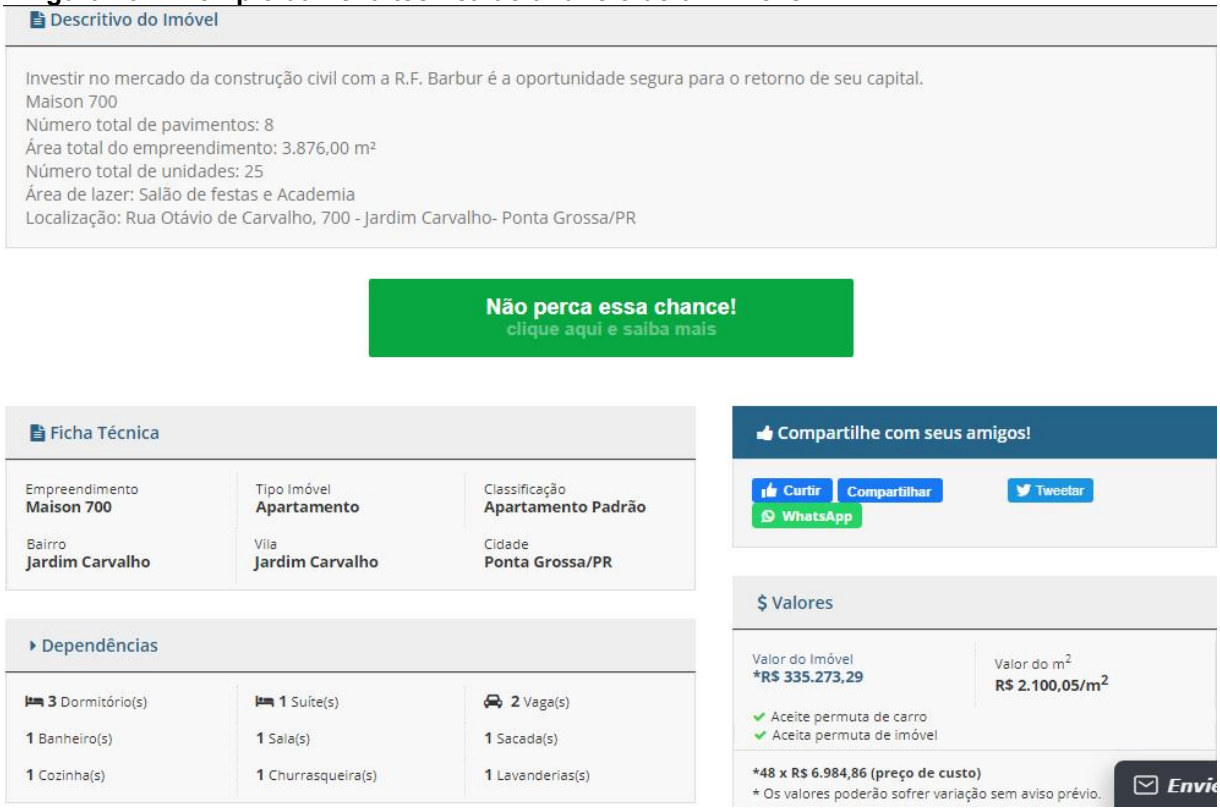
Figura 19 - Função *imovel* do arquivo *find*

```
1   def imovel(i, j):
2       while (i <= j):
3           print(f'{i} de {j}')
4           r = session.get(
5               f"https://www.tavarnaroconsultoria.com.br/imoveis/a-venda?pagina={i}")
6
7           data = r.html.absolute_links
8
9           links = extractLinks(data)
10
11          for d in links:
12              try:
13                  re = session.get(d)
14                  end = re.html.find('.header-title .sub', first=True)
15                  preco = re.html.find('.price', first=True)
16              except Exception as e:
17                  print(f"Link: {d}\n Error: {e}")
```

Fonte: Autoria própria

Em alguns sites podemos encontrar outros elementos que podem auxiliar na georreferenciação do anúncio, a Figura 20, exemplifica um desses casos, onde podemos encontrar, informações como bairro, vila e cidade.

Figura 20 - Exemplo da ficha técnica do anúncio de um imóvel



Fonte: Autoria própria

3.4.2 Estruturação do anúncio

Com a informação obtida é necessário realizar tratamento no dado extraído. A estruturação pode auxiliar no momento da georreferenciação, pois aumenta as chances de buscar o local desejado.

Além do complemento da função *imovel*, são apresentadas funções utilizadas para estruturar a informação e a maneira como a mesma é armazenada. A Figura 21 mostra algumas funções responsáveis por remover texto ou valores desnecessários dos elementos *HTML*, facilitando o uso da informação encontrada.

Figura 21 - Função *imovel* completa do arquivo *find*

```python
def imovel(i, j):
    while (i <= j):
        print(f'{i} de {j}')
        r = session.get(
            f"https://www.tavarnaroconsultoria.com.br/imoveis/a-venda?pagina={i}")

        data = r.html.absolute_links

        links = extractLinks(data)

        for d in links:
            try:
                re = session.get(d)
                end = re.html.find('.header-title .sub', first=True)
                preco = re.html.find('.price', first=True)
            except Exception as e:
                print(f"Link: {d}\n Error: {e}")

            try:
                if end and preco:
                    precoAnuncio = valorAnuncio(preco.text)
                    (pont, endMatch) = buscaRuaPG(end.text)
                    if pont >= 40 and precoAnuncio > 0:
                        print(r.status_code)
                        (rua, numero, bairro, cidade) = estruturandoEndereco(end.text)
                        anuncio = Anuncio(d.split(
                            '/')[5], end.text, rua, numero, bairro, cidade, precoAnuncio, endMatch, d)
                        anuncios.append(json.loads(anuncio.toJSON()))
            except Exception as e:
                print(f"Link: {d}\n Error: {e}")
        if r.status_code == 200:
            i += 1
```

Fonte: Autoria própria

Na Figura 22 o uso de expressões regulares está presente para estruturar endereços e valor de venda. Utilizando a biblioteca *fuzzywuzzy* (COHEN, 2011) é possível validar os dados, pois a biblioteca permite encontrar semelhanças entre *strings*. A comparação é feita entre os textos extraídos das páginas *HTML*, e de logradouros e bairros obtidos no site *Wikimapia* (WIKIMAPIA, 2019). A biblioteca fornece uma pontuação para semelhança entre *strings* facilitando a validação dos dados e alternativas para georreferenciação pois o endereço extraído pode conter erros de ortografia que poderia impossibilitar a georreferenciação.

Figura 22 – Funções complementares para estruturação do anúncio

```python
def buscaRuaPG(endEncontrado):

    if "Ponta Grossa/PR" not in endEncontrado or "Ponta Grossa, PR" not in endEncontrado:
        return (0, "")

    ruaEncontrada = rmBairroCidade(endEncontrado)
    if not ruaEncontrada:
        return (0, "")

    pontuacao = []
    fuzz.SequenceMatcher = difflib.SequenceMatcher
    for rua in logradouros:
        pontuacao.append(fuzz.ratio(ruaEncontrada, str(rua)))

    n_max = max(pontuacao, key=int)
    n_pos = pontuacao.index(n_max)

def valorAnuncio(preco):
    resultado = regex.findall(
        r"(?:[1-9](?:[\d]{0,2}(?:\.[\d]{3})*|[\d]+)|0)(?:,[\d]{0,2})?", preco)
    x = []
    if not resultado:
        return float(0.00)
    for r in resultado:
        value = r.replace('.', '')
        value = value.replace(',', '.')
        x.append(float(value))
    return max(x, key=float)

def rmBairroCidade(endereco):
    return regex.sub(r"([A-Za-záàâãéèêíïóôõöúçñÁÀÂÃÉÈÍÏÓÔÕÖÚÇÑ ]*?)(\s?-?)
        ([A-Za-záàâãéèêíïóôõöúçñÁÀÂÃÉÈÍÏÓÔÕÖÚÇÑ ]*)( - )(Ponta Grossa/PR|Ponta Grossa,PR)", '', endereco)
```

Fonte: Autoria própria

Na Figura 23 são utilizadas expressões regulares para quebrar em partes menores o endereço para facilitar a georreferenciação posteriormente. Pela variação de escrita é necessário verificar a possibilidade de extração do número e logradouro do anúncio.

Figura 23 - Funções complementares para estruturação do endereço encontrado

```python
def estruturandoEndereco(endereco):

    result = regex.search(
        r"([A-Za-záàâãéèêíïóôõöúçñÁÀÂÃÉÈÍÏÓÔÕÖÚÇÑ ]*?)(\s?-?)([A-Za-záàâãéèêíïóôõöúçñÁÀÂÃÉÈÍÏÓÔÕÖÚÇÑ ]*)( - )(Ponta Grossa/PR)", endereco)
        # Ou r"([A-Za-záàâãéèêíïóôõöúçñÁÀÂÃÉÈÍÏÓÔÕÖÚÇÑ ]*?)(\s?-?)([A-Za-záàâãéèêíïóôõöúçñÁÀÂÃÉÈÍÏÓÔÕÖÚÇÑ ]*)(, )(Ponta Grossa, PR)"

    bairro = result.group(0)
    bairro = bairro.replace(' - Ponta Grossa/PR', '') # Ou " , Ponta Grossa, PR', '' "
    bairro = bairro.replace(' - ', '')

    result_number = regex.search(r"\d+", endereco)
    if result_number is not None:
        numero = result_number.group(0)
        rua = endereco.split(',')
        return (rua[0], numero, bairro, "Ponta Grossa/PR")
    else:
        rua = endereco.split('-')
        return (rua[0], 0, bairro, "Ponta Grossa/PR")

    # Caso Conceito
    (rua, numero) = reconheceEndereco(endereco)

    return (rua, numero, bairro, bairro, "Ponta Grossa/PR")

def reconheceEndereco(endereco):
    if "s/n" or "S/N" or "s/c" in endereco:
        rua = endereco.split(',')
        return (rua[0], 0)
    else:
        rua = numero = ''
        try:
            result_number = regex.search(r"(\d+)", endereco)
            numero = result_number.group(0)
            numero = numero.replace(', ', '')
            rua = endereco.split(',')
        except Exception as e:
            print(f"Endereco: {endereco}\n Error: {e}")
    return (rua[0], numero)
```

Fonte: Autoria própria

Para transformar o anúncio estruturado em um objeto JSON, é utilizando a classe lógica da Figura 24 para que seja possível obter posteriormente as informações em texto do anúncio encontrado. A classe *anuncio* contém informações obtidas pela extração, qual URL foi feita a extração, a data da extração, informações adicionais como endereço de semelhança e o endereço de forma estruturada, variando alguns campos conforme cada anúncio. A classe *anuncio* contém uma função denominada *toJSON* responsável em converter a classe lógica em um objeto JSON.

Figura 24 - Classe lógica *anuncio* e *endereco*

```
class Endereco:
    def __init__(self, logradouro, bairro, vila, cidade, numero):
        self.logradouro = logradouro
        if(numero != 0):
            self.numero = numero
        self.bairro = bairro
        self.cidade = cidade
        self.vila = vila


class Anuncio:
    def __init__(self, ref, end, rua, numero, bairro, vila, cidade, valor, endMath, link):
        self.ref = ref
        self.enderecoAnuncio = end
        self.valor = valor
        now = datetime.now()
        self.data = '{:%d/%m/%Y}'.format(now)
        self.enderecoMatch = endMath
        self.endereco = Endereco(rua, bairro, vila, cidade, numero)
        self.link = link

    def toJSON(self):
        return json.dumps(self, ensure_ascii=False, default=lambda o: o.__dict__, sort_keys=True, indent=4)
```

Fonte: Autoria própria

Por fim é adicionado a uma lista, o objeto JSON, seguindo a estrutura da classe lógica *anuncio*. Ao término de extração de todos anúncios da página de busca visitada, o laço *while* da continuidade no processo para realizar novas extrações de outra página da lista de anúncios imobiliários. Desta maneira, finaliza-se o processo representado na Figura 14, desde da requisição da página de busca ao armazenamento do anúncio encontrado.

Para finalizar os processos de 1 a 3 da Figura 13, e obter o primeiro arquivo de saída, no caso, o quarto processo ilustrado na Figura 13, é necessário verificar os anúncios encontrados. É realizada uma busca entre todos os anúncios obtidos pela ferramenta desta pesquisa, visando encontrar anúncios com a mesma referência, um código que é utilizado para identificar cada anúncio imobiliário. Se não for encontrado nenhum anúncio anterior com este mesmo código, ele é adicionado à

lista final de anúncios. Em casos de anúncios de mesmo código de referência, é feita uma comparação entre endereços para afirmar que são os mesmos anúncios para armazenar *data* e *valor* encontrados na extração, podendo assim criar uma linha temporal do anúncio. É utilizada a estrutura de uma classe chamada *valor* para armazenar valor e data, conforme a Figura 25.

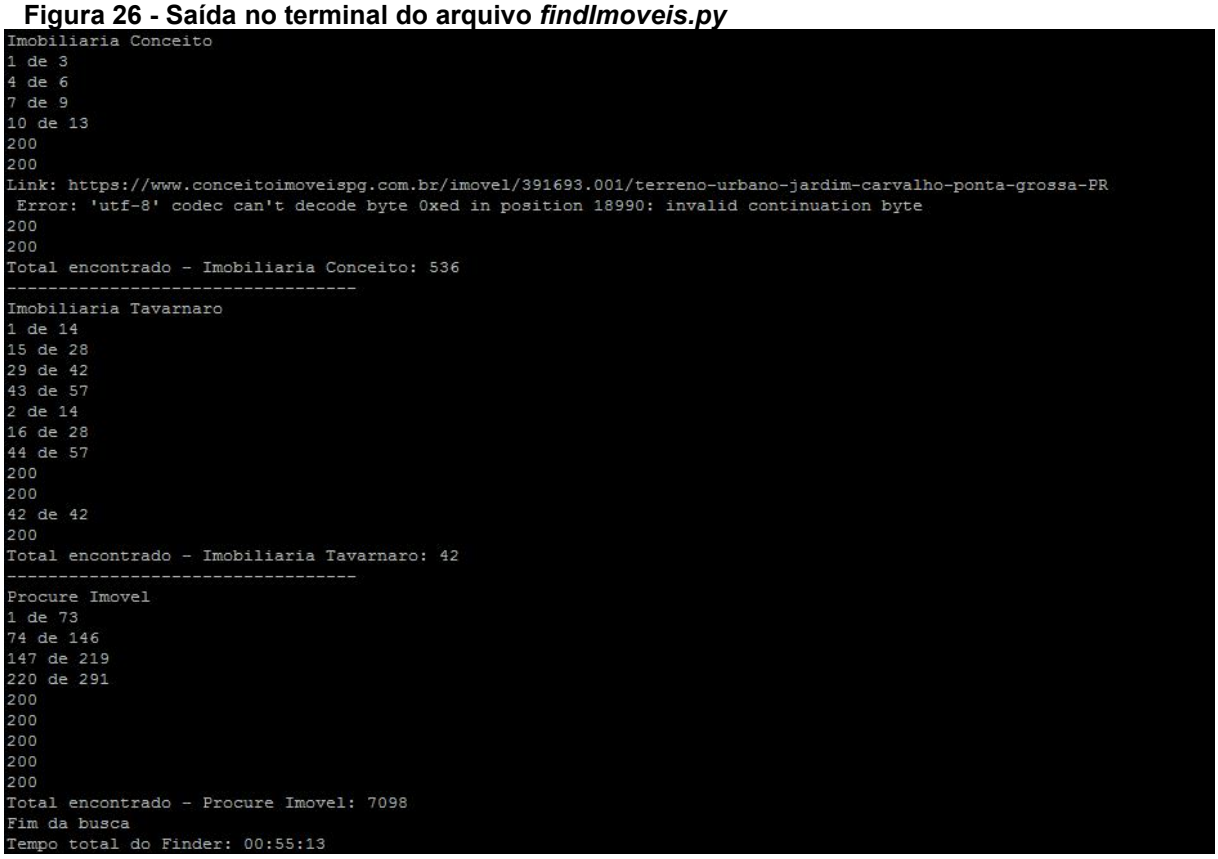
Figura 25 - Classe lógica valor

```
1  class Valor:
2      def __init__(self, data, valor):
3          self.data = data
4          self.valor = valor
5      def toJSON(self):
6          return json.dumps(self, ensure_ascii=False, default=lambda o: o.__dict__, sort_keys=True, indent=4)
```

Fonte: Autoria própria

Com a validação de data e valores, o processo de extração de dados é finalizado. Assim o primeiro arquivo de saída com os dados extraídos é criado. Tendo como exemplo na Figura 26, o funcionamento do *script* de extração, o código 200 significa extrações bem-sucedidas, também são apresentados as causas e os erros durante a execução, total de anúncios extraídos em cada site, a transição de páginas de busca das imobiliárias e a finalização com o total de anúncios extraídos.

Figura 26 - Saída no terminal do arquivo *findImoveis.py*



Fonte: Autoria própria

A Figura 27 apresenta a estruturação do anúncio e as validações executadas. É possível visualizar as informações obtidas pela técnica de *Web Scraping* e as informações estruturadas pela ferramenta como valor e data, dando a finalização na escrita do arquivo de saída.

Figura 27 - Exemplo do arquivo de saída com anúncios extraídos


```
[
    {
        "data": "19/09/2019",
        "endereco": {
            "bairro": "Orfãs",
            "cidade": "Ponta Grossa/PR",
            "logradouro": "Rua Saldanha da Gama",
            "vila": "Jardim Carvalho"
        },
        "enderecoAnuncio": "Rua Saldanha da Gama, 249 - ap 103 - Orfãs/Jardim Carvalho - Ponta Grossa/PR",
        "enderecoMatch": "Rua Saldanha da Gama\n",
        "link": "https://www.conceitoimoveispg.com.br/imovel/391123.037/apartamento-padrao-edifcio-palazzo-ducale-3-dormitorios-orfas-ponta-grossa-PR",
        "ref": "391123.037",
        "valor": 668000.0
    },
    {
        "data": "16/07/2019",
        "endereco": {
            "bairro": "Uvaranas",
            "cidade": "Ponta Grosa/PR",
            "logradouro": "Rua Afonso Celso",
            "numero": "000"
        },
        "enderecoAnuncio": "Rua Afonso Celso, 000 - Uvaranas, Ponta Grossa, PR",
        "enderecoMatch": "Rua Afonso Celso\n",
        "link": "https://procureimovel.com.br/imovel/sobrado-venda-ponta-grossa-uvaranas-ref-127823-4",
        "ref": "127823-4",
        "valor": [
            {
                "data": "16/07/2019",
                "valor": 190000.0
            },
            {
                "data": "18/09/2019",
                "valor": 190000.0
            }
        ]
    },
```


Fonte: Autoria própria

3.5 GEORREFERENCIAÇÃO DOS DADOS

Com os anúncios extraídos conforme a Figura 27, é dado início a etapa 5 da Figura 13, na qual é utilizada a informação e estruturação do anúncio para obter-se a coordenada geográfica (latitude e longitude) do anúncio. Dessa forma finaliza-se a etapa 6 do processo apresentado na Figura 13 com o arquivo de saída final, contendo a geolocalização dos anúncios.

3.5.1 Utilização de API para georreferenciamento

Para dar início ao processo de georreferenciamento, é preciso entender como funciona a API de georreferenciamento. A API irá fornecer a localização espacial de um endereço. A Figura 28 e Figura 29 demonstra a operação das ferramentas *Geocoding* (*Google*) e *Nominatin* (*OpenStreetMap*). Uma requisição é efetuada com uma *URL* base em conjunto com os parâmetros de busca, neste caso um endereço. Assim se obtém como resultado a latitude e longitude, endereço formatado, entre outras informações do endereço que foi passado como parâmetro.

Figura 28 - Exemplo de uso de API para georreferenciação *Nominatin*

https://nominatim.openstreetmap.org/search/Unter%20den%20Linden%201%20Berlin?format=json&addressdetails=1&limit=1


```
{
    "address": {
        "city": "Berlin",
        "city_district": "Mitte",
        "construction": "Unter den Linden",
        "continent": "European Union",
        "country": "Deutschland",
        "country_code": "de",
        "house_number": "1",
        "neighbourhood": "Scheunenviertel",
        "postcode": "10117",
        "public_building": "Kommandantenhaus",
        "state": "Berlin",
        "suburb": "Mitte"
    },
    "boundingbox": [
        "52.5170783996582",
        "52.5173187255859",
        "13.3975105285645",
        "13.3981599807739"
    ],
    "class": "amenity",
    "display_name": "Kommandantenhaus, 1, Unter den Linden, Scheunenviertel, Mitte, Berlin, 10117, Deutschland, European Union",
    "importance": 0.73606775332943,
    "lat": "52.51719785",
    "licence": "Data \u00a9 OpenStreetMap contributors, ODbL 1.0. https://www.openstreetmap.org/copyright",
    "lon": "13.3978352028938",
    "osm_id": "15976890",
    "osm_type": "way",
    "place_id": "30848715",
    "type": "public_building"
}
```


Fonte: Autoria própria

Na Figura 28, é necessário destacar alguns pontos como a *URL* de requisição da *API* da ferramenta *Geocoding*: o final da *URL* existe um parâmetro *key* com o valor *“YOUR_API_KEY”*, que deve ser substituído pelo código de identificação do projeto. Esta *key* permite que se faça a requisição para a base de dados da ferramenta do *Google*. Como é uma ferramenta paga ela se torna obrigatória para fazer requisições e obter respostas válidas.

Figura 29 - Exemplo de uso de API para georreferenciação *Geocoding*

https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY


```
{
   "results" : [
      {
         "formatted_address" : "1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA",
         "geometry" : {
            "location" : {
               "lat" : 37.4267861,
               "lng" : -122.0806032
            },
            "location_type" : "ROOFTOP",
            "viewport" : {
               "northeast" : {
                  "lat" : 37.4281350802915,
                  "lng" : -122.0792542197085
               },
               "southwest" : {
                  "lat" : 37.4254371197085,
                  "lng" : -122.0819521802915
               }
            }
         },
         "place_id" : "ChIJtYuu0V25j4ARwu5e4wwRYgE",
         "plus_code" : {
            "compound_code" : "CWC8+R3 Mountain View, California, United States",
            "global_code" : "849VCWC8+R3"
         },
         "types" : [ "street_address" ]
      }
   ],
   "status" : "OK"
}
```


Fonte: Autoria própria

Na resposta da requisição são apresentadas outras informações além da localização espacial do endereço. Uma destas informações é a localização espacial por ponto central. Ela é definida por um único ponto espacial no mapa, onde indica a posição mediana da informação. Existe a localização por polígono, são quatro coordenadas espaciais, que possibilita delimita os pontos extremos de um quadrado da área buscada, esta informação está contida nas propriedades do arquivo de retorno "*viewport*" ou "*boundingbox*", além de outras informações que podem auxiliar a confirmação da localidade.

Neste trabalho foi utilizado a localização por ponto, pois não é necessário medir o tamanho do anúncio, a localização pelo ponto central já supre a necessidade deste trabalho. A API *Nominatin* foi utilizada por ser uma ferramenta gratuita e de fácil utilização, mesmo com sua limitação de base de dados (CIEPŁUCH et al., 2010).

3.5.2 Georreferenciamento dos anúncios

Para que a georreferenciação dos anúncios fosse realizada, foi necessário ler os anúncios obtidos na extração, estruturar os endereços para busca e processar o resultado positivo ou negativo da georreferenciação. Após percorrer a lista de anúncios, e tentar georreferenciar todos os anúncios, é finalizando com escrita de um arquivo de saída com os anúncios georreferenciados.

Para realizar requisição e tratamentos de respostas da API de geolocalização foi utilizado a biblioteca *geopy* (GEOPY, 2018), conforme a Figura 30.

Figura 30 - Bibliotecas do arquivo *mapAnuncio.py*

```
1   # coding: utf-8
2   import json
3   import time
4   from geopy.geocoders import Nominatim
5   from geopy.extra.rate_limiter import RateLimiter
6   from datetime import datetime
7
8   anuncios = []
9   geoAnuncio = []
10  notGeoAnuncio = []
11  final = []
12  anunciosToday = []
13
14  geocoder = Nominatim(user_agent="GeomappingPontaGrossa")
15  localizador = RateLimiter(geocoder.geocode, min_delay_seconds=3)
```

Fonte: Autoria própria

Um problema encontrado na utilização da ferramenta *Nominatim* (FUNDAÇÃO OPENSTREETMAP, 2019) foi o número de requisições simultâneas para a API. O limite era uma requisição por segundo. Por conta desta restrição, em muitos casos era recebido o código de erro, apresentado que a ferramenta desenvolvida nesta pesquisa havia extrapolado o número máximo de requisições. Por isso foi utilizada outra ferramenta da biblioteca *geopy* (GEOPY, 2018) denominada *RateLimiter* (GEOPY, 2018), para definir que a cada três segundos, após a última requisição, será feita uma nova chamada. Estes três segundos é uma recomendação da própria biblioteca *geopy*.

A função *finder* está dividida nas duas figuras, Figura 31 e Figura 32. Na Figura 31, a função é iniciada pelo laço de repetição da linha 3 para verificar todos os anúncios extraído. Como existem anúncios que já foram encontrados em outras execuções, descartou-se a georreferenciação dos mesmos.

Figura 31 - Função *finder* do arquivo *mapAnuncio.py*

```
def finder():
    global final
    for k in range(0, len(anuncios)):
        if 'localizacao' in anuncios[k]:
            final.append(anuncios[k])
            geoAnuncio.append(anuncios[k])
            continue

        enderecoFinder = ""

        enderecoAnuncio = anuncios[k]["enderecoAnuncio"]
        enderecoMatch = anuncios[k]["enderecoMatch"]
        logradouro = anuncios[k]['endereco']['logradouro']
        bairro = anuncios[k]['endereco']['bairro']
        cidade = anuncios[k]['endereco']['cidade']
        numero = None
        vila = None
        if 'numero' in anuncios[k]['endereco']:
            numero = anuncios[k]['endereco']['numero']

        if 'vila' in anuncios[k]['endereco']:
            vila = anuncios[k]['endereco']['vila']

        enderecoFinder = enderecoAnuncio
        tries = 0
        coordinate = []
```

Fonte: Autoria própria

Para iniciar o processo de requisição para API com o endereço desejado, extraiu-se as informações necessárias tais como: logradouro, bairro, número e vila se existirem no anuncio, cidade, endereço encontrado por semelhança de *strings* para poder usar toda e qualquer informação possível para encontrar o anúncio na base de dados da API. Como condição de parada, foi utilizado um total de quatro tentativas, a primeira com endereço encontrado no próprio anúncio, a segunda com a estruturação deste endereço encontrado, logradouro, número se existir, bairro e cidade. Na terceira tentativa utilizou-se a vila se existir no anúncio e na última tentativa utilizou-se o endereço encontrado pela semelhança de *strings*. A Figura 32 mostra a utilização da biblioteca *geopy* denominada pela variável *localizador* que recebe como parâmetro a *string* de endereço baseado no número de tentativa. Com isso é reduzido a edição e o uso de *URLs* diretamente e tratar problemas com o resultado da busca.

Figura 32 - Complementação da função *finder* do arquivo *maAnuncio.py*

```
            finder = localizador(enderecoFinder)

            if finder is not None:
                coordinate.append(finder.latitude)
                coordinate.append(finder.longitude)

            tries += 1  # adiciono mais um no número de tentativas para controle

        if coordinate:
            anuncioGeolocalizado = Anuncio(
                anuncios[k], coordinate[0], coordinate[1])
            geoAnuncio.append(json.loads(anuncioGeolocalizado.toJSON()))
            anunciosToday.append(json.loads(anuncioGeolocalizado.toJSON()))
            print(f'{k} de {len(anuncios)} - Codigo: {anuncioGeolocalizado.ref} encontrado')
        else:
            notGeoAnuncio.append(anuncios[k])
            print(f'{k} de {len(anuncios)} - Codigo: {anuncios[k]["ref"]} não encontrado')
```

Fonte: Autoria própria

Utilizando-se da classe lógica da Figura 33 para estruturar o anúncio novamente, adicionando o atributo localidade com sua coordenada. A estrutura da classe dos anúncios extraídos é mantida, acrescentando a propriedade localidade com a coordenada geográfica encontrada.

Figura 33 - Classes lógicas do arquivo *mapAnuncio.py*

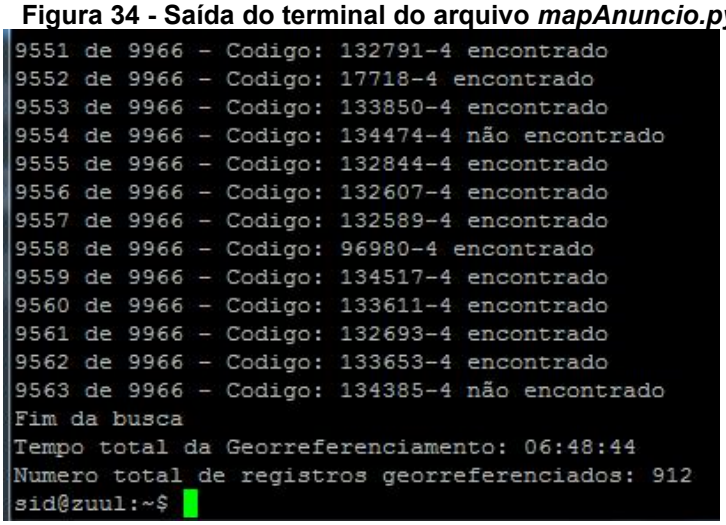
```
class Localizacao:
    def __init__(self, latitude, longitude):
        self.type = "Point"
        self.coordinates = []
        self.coordinates.append(latitude)
        self.coordinates.append(longitude)

```

Fonte: Autoria própria

Após tentar o georreferenciamento de todos os anúncios é finalizado o processo unindo todos os anúncios, mesmo que o anúncio não tenha sido georreferenciado. Isso ocorre para que nas próximas execuções, a ferramenta desenvolvida possa adicionar novas datas e valores para o anúncio. São mantidos dois arquivos de saída, um com todos os anúncios georreferenciados e outro com todos os anúncios extraídos, georreferenciados ou não para serem utilizados posteriormente.

A Figura 34 mostra a execução do *script*, exemplificando como o usuário é notificado se o anúncio foi georreferenciado ou não, outras informações são apresentadas tais como: Erros de requisição a API, tempo para finalizar o processo de georreferenciamento e a quantidade de anúncios georreferenciados.

Figura 34 - Saída do terminal do arquivo *mapAnuncio.py*



Fonte: Autoria própria

Assim é finalizado todo o processo da ferramenta conforme a Figura 13 demonstra.

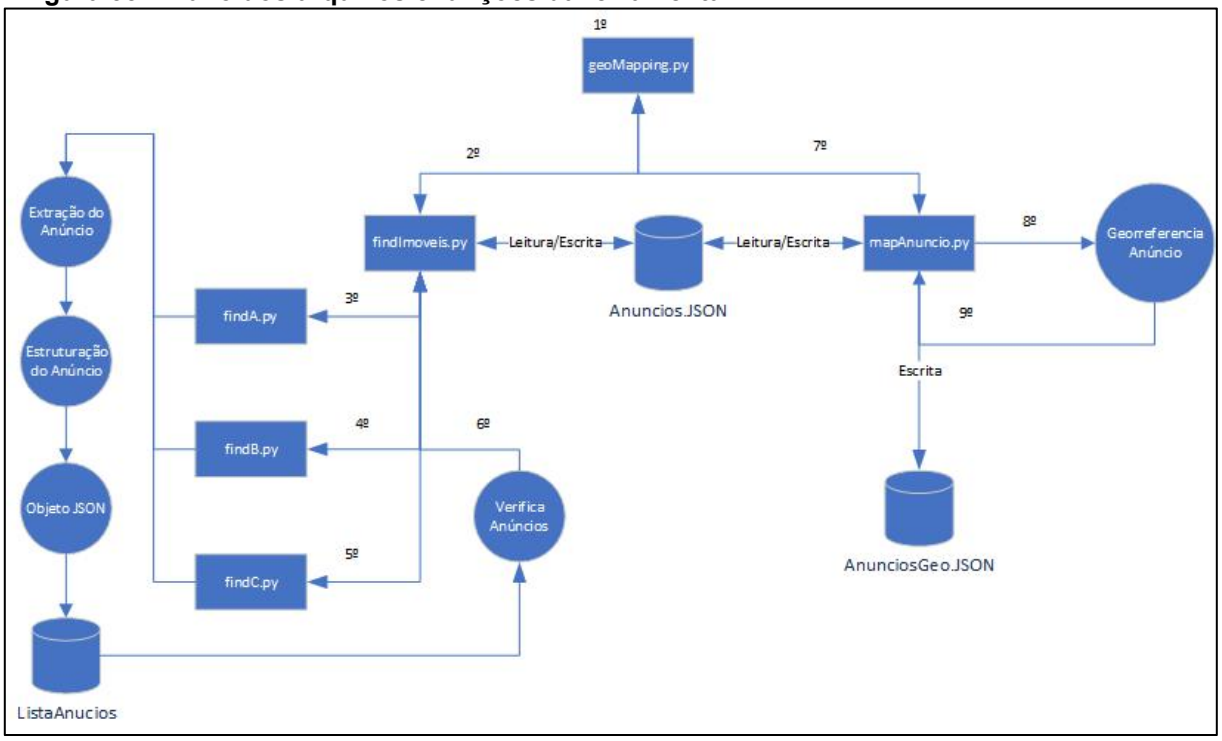
3.6 ESTRUTURA GERAL

O diagrama a seguir representado pela Figura 35 apresenta a estrutura geral da solução desenviolada nesta pesquisa, ela possibilita visualizar como está separado os *scripts* de execução, arquivos de saída e fluxo de funcionamento. Os retângulos representam cada arquivo da ferramenta, os círculos funções apresentada nesta pesquisa e as bases de dados resultado final de cada etapa da ferramenta.

Se houver necessidade de acrescentar um novo site para extração, deve-se utilizar como exemplo os arquivos *finders* presente no diretório *finders*, substituindo a *URL* de página de busca para obter o número de páginas, os seletores, expressões regulares e funções para estruturação do anúncio e seleção de *URLs* relevantes. Por fim adicionar a chamada deste novo *finder* no arquivo *findImoveis.py*, para unir o resultado a todos os *finders.*

Para se alterar a API de georreferenciação, alterasse no arquivo *mapAnuncio.py* onde deve se alterar a API conforme a biblioteca *geopy* (GEOPY, 2018) disponibiliza. Na Figura 30 podemos ver que a ferramenta utiliza a *Nominatin* deve se alterar tanto a importação, quanto a instanciação da ferramenta para a nova API que deseja.

Figura 35 - Fluxo dos arquivos e funções da ferramenta



Fonte: Autoria própria

4 RESULTADOS DA FERRAMENTA

No trabalho de Batista e Xavier (2018) foram utilizados testes para mostrar a eficiência da ferramenta desenvolvida, nos quais pessoas extraiam informações manualmente. Uma vez que tal ferramenta tinha objetivo de ser semiautomática, a comparação era um indicador direto de seu desempenho.

No presente caso optou-se por um programa que pudesse funcionar sem supervisão humana. Neste caso as medições de desempenho serão dependentes principalmente da conexão da internet. Apesar disso, é interessante comparar os resultados do primeiro trabalho, para observar o ganho de velocidade obtido.

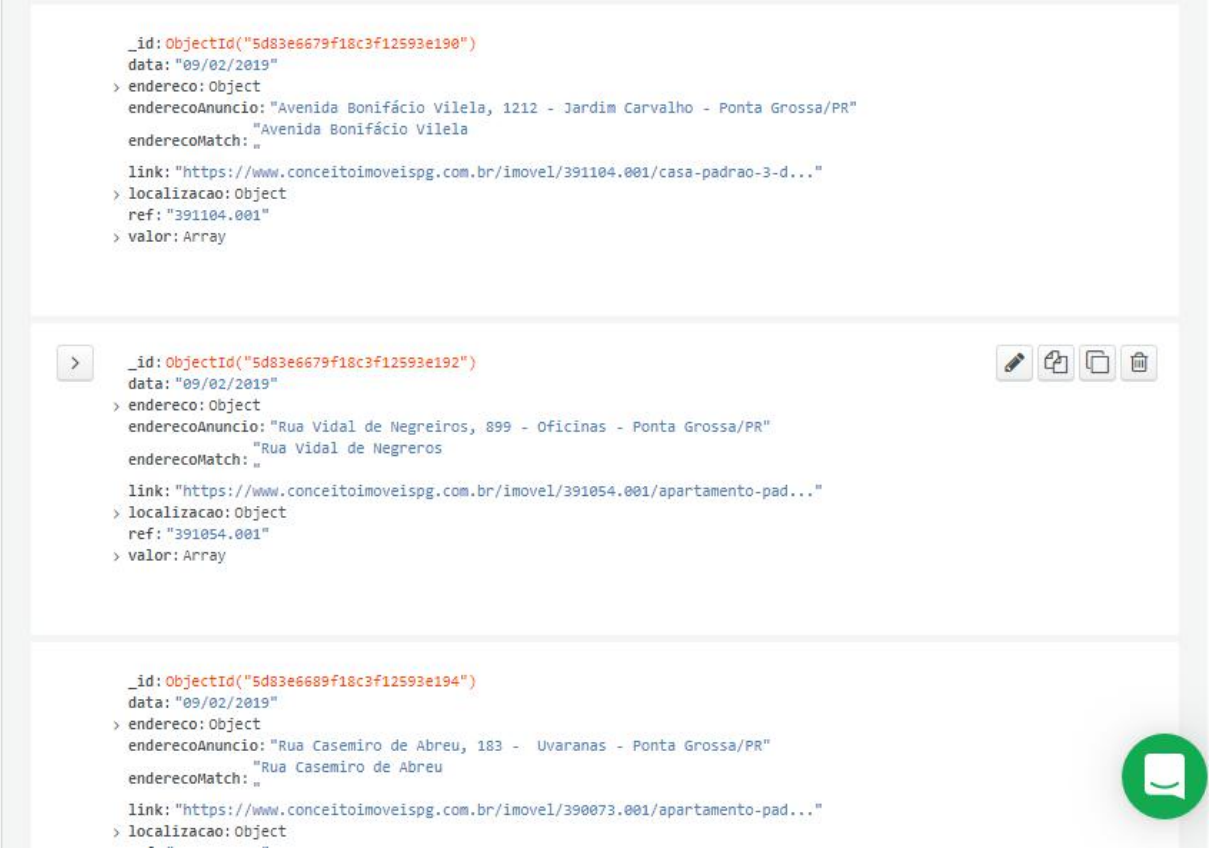
Foram realizados testes experimentais, em que algumas pessoas realizavam a ação de abrir sites de imobiliárias e extrair as informações necessárias, endereço e valor, contra a ferramenta desenvolvida neste trabalho, durante um prazo de dois minutos. A produtividade com a ferramenta nova é quase 30 vezes maior que a manual. A ferramenta *Prokurator* de Batista e Xavier (2018) levava em média 38,66 segundos para extrair 10 anúncios; o software desenvolvido neste trabalho apresentou valores em torno de 1,43 segundos para extrair 10 anúncios. Uma diferença relevante, nesta comparação, é que pelo fato do novo software funcionar sem revisão humana dos dados, se os algoritmos não conseguirem reconhecer um endereço devido a abreviaturas ou erros de ortografia, o respectivo anúncio será descartado. Em uma situação em que seja importante manter essa informação, basta acrescentar um arquivo de saída adicional. Neste trabalho foi descartada a ideia de tratar manualmente anúncios e o volume de informações obtido foi julgado satisfatório pelo usuário a quem se destinava o software (o orientador do trabalho).

Não foi possível realizar um teste com diferentes ferramentas para georreferenciação dos anúncios, pois a ferramenta disponibilizada pela empresa *Google* que contém uma base mais abrangente de dados espaciais, limitava a uma requisição gratuita diariamente, contra um número de requisições ilimitadas da ferramenta *Nominatin*, com a restrição de não poder realizar requisições simultâneas. Mesmo com a restrição da API *Nominatin* a ferramenta continuou eficiente, porém com tempo de execução alto.

Utilizando apenas três sites para extração de anúncios sendo executado a cada 40 dias num intervalo de 270 dias, foi possível criar uma base com 9500 anúncios extraídos e 8900 anúncios georreferenciados. O que mostra que a

ferramenta foi eficiente na extração de anúncios, pois em média 1407 anúncios foram extraídos e 1259 anúncios georreferenciados por execução. Todos os arquivos de saída da ferramenta desenvolvida estão em formato JSON, o que possibilita realizar a importação dos dados para um banco de dados não relacional, sendo de fácil acesso para outras ferramentas utilizarem conforme a Figura 36.

Figura 36 - Dados armazenado no banco de dados *MongoDB*



Fonte: Autoria própria

Foi utilizado o banco de dados NoSQL, *MongoDB* (MONGODB, 2019) por disponibilizar ferramentas para trabalhar com dados geográficos, podendo visualizar e criar buscas baseado em pontos geográficos.

5 CONCLUSÃO

Conforme apresentado na introdução, existe muita informação armazenada em diferentes meios. Algumas prontas para serem usadas por computadores, outras apenas para visualização de seus usuários. Este trabalho visou extrair informações não estruturadas armazenadas em sites imobiliários, possibilitando a visualização por usuários e a flexibilidade para o processamento computacional em cenários de simulação urbana.

Com algumas modificações no código da ferramenta, é possível aumentar a quantidade de informação extraída do anúncio, entre algumas, pode-se citar como exemplo: área útil do imóvel, dependências do imóvel. Utilizando o trabalho do autor (NEDER et al., 2017), para entregar uma ferramenta de apoio a prefeitura da cidade para obtenção de valor de IPTU baseado nas áreas construídas dos anúncios imobiliários.

No estudo de caso utilizado para este trabalho, a ferramenta se tornou muito eficaz, em um prazo de oito meses, foram extraídos aproximadamente 9500 (nove mil e quinhentos) anúncios e georreferenciados quase 8900 (oito mil e novecentos) anúncios. Criou-se assim um grande volume de dados para ser utilizado posteriormente por outros trabalhos.

A ferramenta necessita de conexão com a Internet para extrair e georreferenciar os anúncios, o que se apresenta como uma limitação: a velocidade de operação depende da conexão. Outra limitação é a não adaptabilidade a novos sites que, conforme comentado na introdução, é um problema complexo. As estruturas e layouts utilizados por um site podem sofrer alterações a qualquer momento, podendo assim criar um colapso na ferramenta pois não será encontrado com os mesmos seletores as informações desejadas. Isto é contornado no código pela modificação das *queries selectors*, do código e expressões regulares, tornando mais fácil a manutenção.

Como apresentado a API *Nominatin* se mostrou eficaz para o trabalho, porém não conseguia encontrar todos os anúncios extraídos. Uma melhoria futura para este trabalho seria utilizar API do *Google Maps*, pois tem uma base maior de endereços e uma precisão mais eficaz. Ela não pode ser utilizada neste trabalho pois é paga.

5.1 TRABALHOS FUTUROS

Como trabalhos futuros, pode-se trabalhar aplicando técnicas de interpolação para conseguir a localização de um anúncio baseado na base de dados já utilizada. Assim pode-se criar uma base de dados da cidade de Ponta Grossa com os endereços e coordenadas geográficas, podendo ser utilizado por este trabalho e outras ferramentas a serem desenvolvidas.

Para tornar a extração de dados mais eficaz e abrangente, utilizar os dados já extraídos para criar uma ferramenta que pode extrair anúncio de qualquer site, baseado em *machine learning* criar uma ferramenta *web crawler* que consegue identificar e extrair informações desejadas.

Realizar técnicas de refatoração e aplicação de padrões de projeto a ferramenta, a fim de melhorar a inclusão de novos sites, funções e expressões regulares. Também possibilitar a troca da API de georreferenciação, para facilitar ao programador ajustar a ferramenta conforme sua necessidade. A estruturação da ferramenta para que o usuário possa ter uma interface para controlar quais sites e informações deseja extrair.

REFERÊNCIAS

AIJUN, X.; LICHUAN, G. **Encoding & decoding of Chinese address and development of algorithms for intelligent address search**. 2010 International Conference on Computer Application and System Modeling. **Anais**...IEEE, out. 2010Disponível em: <http://ieeexplore.ieee.org/document/5620171/>. Acesso em: 24 jun. 2019

BATISTA, B. **Aprenda por definitivo a usar CSS Selector(Adeus Xpath).** Disponível em: <https://medium.com/automação-com-batista/aprenda-por-definitivo-a-usar-css-selector-adeus-xpath-1f3956763c2>. Acesso em: 23 set. 2019.

BATISTA, J. DA S.; XAVIER, E. S. **Criação de um banco de dados não relacional a partir de informação extraída de textos**Ponta GrossaUniversidade Tecnológica Federal do Paraná, , 29 maio 2018. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/9729>. Acesso em: 16 jun. 2019

BHARANIPRIYA, V.; PRASAD, V. K. **WEB CONTENT MINING TOOLS: A COMPARATIVE STUDYInternational Journal of Information Technology**. [s.l: s.n.]. Disponível em: <http://csjournals.com/IJITKM/PDF 4-1/43.V. Bharanipriya1 %26 V. Kamakshi Prasad2.pdf>. Acesso em: 24 jun. 2019.

BUSCAPÉ COMPANY INFORMAÇÃO E TECNOLOGIA LTDA. **Buscapé - Conheça o Buscapé**. Disponível em: <https://www.buscape.com.br/conheca-o-buscape>. Acesso em: 24 jun. 2019.

CASCÓN-KATCHADOURIAN, J.; RUIZ-RODRÍGUEZ, A.-Á.; ALBERICH-PASCUAL, J. Uses and applications of georeferencing and geolocation in old cartographic and photographic document management. **El Profesional de la Información**, v. 27, n. 1, p. 202, 2018.

CHEN, Z.-H. et al. **Big data: Open data and realty website analysis**. 2015 8th International Conference on Ubi-Media Computing (UMEDIA). **Anais**...IEEE, ago. 2015Disponível em: <http://ieeexplore.ieee.org/document/7297433/>. Acesso em: 24 jun. 2019

CIDADE DE SÃO PAULO. **CopiCola**. Disponível em: <https://copicola.prefeitura.sp.gov.br/#sobre>. Acesso em: 28 nov. 2019.

CIEPŁUCH, B. et al. Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps. **Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resuorces and Enviromental Sciences 20-23rd July 2010**, p. 337–341, 20 jul. 2010.

COELHO, A. L. N. SISTEMA DE INFORMAÇÕES GEOGRÁFICAS (SIG) COMO SUPORTE NA ELABORAÇÃO DE PLANOS DIRETORES MUNICIPAIS. **CAMINHOS DE GEOGRAFIA**, v. 10, n. 30, p. 93–110, 2009.

COHEN, A. **FuzzyWuzzy: Fuzzy String Matching in Python - ChairNerd**. Disponível em: <https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/>. Acesso em: 2 dez. 2019.

COMPUTERWORLD. **Prefeitura de São Bernardo reduz burocracia com transformação digital | Computerworld**. Disponível em: <https://computerworld.com.br/2019/05/13/prefeitura-de-sao-bernardo-reduz-burocracia-com-transformacao-digital/>. Acesso em: 28 nov. 2019.

DESAI, K. et al. Web Crawler : Review of Different Types of Web Crawler, Its Issues, Applications and Research Opportunities. **International Journal of Advanced Research in Computer Science**, v. 8, n. 3, p. 1199–1202, 2017.

E-GESTÃO PÚBLICA. **O que é preciso para digitalizar a gestão de prefeitura?** Disponível em: <https://www.e-gestaopublica.com.br/gestao-de-prefeitura/>. Acesso em: 28 nov. 2019.

FAROOQ, B.; HUSAIN, M. S.; SUAIB, M. **CRAWLING OF JAPANESE REAL-ESTATE WEBSITES USING SCRAPY**. International Journal of Advanced Research in Computer Science. **Anais**...2018Disponível em: <www.ijarcs.info>. Acesso em: 23 set. 2019

FERRARA, E. et al. Web data extraction, applications and techniques: A survey. **Knowledge-Based Systems**, v. 70, p. 301–323, 2014.

FERREIRA, R. V.; RAFFO, J. DA G. O USO DOS SISTEMAS DE INFORMAÇÃO GEOGRÁFICA (SIG) NO ESTUDO DA ACESSIBILIDADE FÍSICA AOS SERVIÇOS DE SAÚDE PELA POPULAÇÃO RURAL: REVISÃO DA LITERATURA. **Revista Brasileira de Geografia Médica e da Saúde**, v. 8, n. 15, p. 178–189, 2012.

FETTERLY, D.; MANASSE, M.; NAJORK, M. A Large-Scale Study of the Evolution of

Web Pages. **Software: Practice & Experience**, v. 34, n. 2, p. 213–237, 2004.

FITZ, P. R. **Geoprocessamento Sem Complicação**. Oficina de Textos ed. São Paulo: Oficina de Textos, 2008.

FLORCZYK, A. J. et al. Semantic selection of georeferencing services for urban management. **Electronic Journal of Information Technology in Construction**, v. 15, p. 111–121, 2010.

FOURSQUARE. **Foursquare - A empresa confiável de inteligência de dados de localização**. Disponível em: <https://pt.foursquare.com/>. Acesso em: 30 nov. 2019.

FUNDAÇÃO OPENSTREETMAP. **OpenStreetMap**. Disponível em: <https://www.openstreetmap.org/copyright>. Acesso em: 30 nov. 2019.

GEOPY, C. **Welcome to GeoPy's documentation!** Disponível em: <https://geopy.readthedocs.io/en/stable/>. Acesso em: 3 dez. 2019.

GLEZ-PEÑA, D. et al. Web scraping technologies in an API world. **Briefings in Bioinformatics**, v. 15, n. 5, p. 788–797, 30 abr. 2013.

GOOGLE INC. **Google**. Disponível em: <https://www.google.com/webhp?hl=pt-BR&sa=X&ved=0ahUKEwj6i5-1-ZLmAhUcDrkGHb18DLMQPAgH>. Acesso em: 30 nov. 2019.

GOOGLE MAPS, P. **Plataforma do Google Maps | Google Developers**. Disponível em: <https://developers.google.com/maps/documentation?hl=pt-br>. Acesso em: 4 nov. 2019.

GREEN DOC. **Prefeitura de São Cristóvão é Pioneira na Digitalização dos Documentos**. Disponível em: <https://suporte.greendoc.com.br/noticia/2/prefeitura-de-sao-cristovao-e-pioneira-na-digitalizacao-dos-documentos>. Acesso em: 28 nov. 2019.

GUIMARÃES, J. W. Elaboração e construção de um protótipo mínimo viável para o Tingoram : um sistema de mineração de dados web baseado em georreferenciamento para sugestão semi automatizada de doação de alimentos. 2018.

HIGOUNET, C. **HISTORIA CONCISA DA ESCRITA** . [s.l: s.n.].

IDEAL MARKETING. **O que é sitemap XML e por que usar um mapa no seu site?** Disponível em: <https://www.idealmarketing.com.br/blog/o-que-e-sitemap/>. Acesso em: 29 nov. 2019.

KAUSAR, M. A.; DHAKA, V. S.; SINGH, S. K. Web Crawler: A Review. **International Journal of Computer Applications**, v. 63, n. 2, p. 31–36, 15 fev. 2013.

MALIK, S. K.; RIZVI, S. **Information extraction using web usage mining, web scrapping and semantic annotation**. Proceedings - 2011 International Conference on Computational Intelligence and Communication Systems, CICN 2011. **Anais**...IEEE, out. 2011Disponível em: <http://ieeexplore.ieee.org/document/6112910/>. Acesso em: 23 jun. 2019

MITCHELL, R. E. **Web Scraping with Python Collecting Data from the Modern Web**. First ed. Sebastopol: O'Reilly Media, 2015.

MONGODB, I. **The most popular database for modern apps | MongoDB**. Disponível em: <https://www.mongodb.com/>. Acesso em: 3 dez. 2019.

NEDER, H. D. et al. Índice de defasagem do Imposto Predial e Territorial Urbano (IPTU) dos Municípios de Minas Gerais : um estudo de caso para Uberlândia (MG). Brasil. **Revista ESPACIOS**, v. 38, n. 46, p. 25–39, 23 jun. 2017.

NEIL, Y. Web Scraping the Easy Way. **University Honors Program Theses**, 1 jan. 2016.

NIANTIC, I. **Pokémon GO**. Disponível em: <https://pokemongolive.com/pt_br/>. Acesso em: 30 nov. 2019.

OLIVEIRA FILHO, P. C. DE; SILVA, S. V. K. DA K. DA. Um sistema de informações para suporte espacial e de decisões à gestão da arborização urbana no município de Guarapuava, Paraná. **Revista da sociedade brasileira de arborização urbana**, v. 5, n. 3, p. 82–96, 2010.

OZIMEK, A.; MILES, D. Stata utilities for geocoding and generating travel time and travel distance information. **The Stata Journal: Promoting communications on**

statistics and Stata, v. 11, n. 1, p. 106–119, 19 mar. 2011.

PARVEZ, M. S. et al. **Analysis of Different Web Data Extraction Techniques**. 2018 International Conference on Smart City and Emerging Technology, ICSCET 2018. **Anais**...Mumbai, India: IEEE, jan. 2018Disponível em: <https://ieeexplore.ieee.org/document/8537333/>. Acesso em: 24 jun. 2019

PYTHON SOFTWARE FOUNDATION. **Welcome to Python.org**. Disponível em: <https://www.python.org/>. Acesso em: 30 nov. 2019.

REITZ, K. **Requests-HTML: HTML Parsing for Humans (writing Python 3)!** . Disponível em: <https://requests-html.kennethreitz.org/>. Acesso em: 2 dez. 2019.

RICHARDSON, L. **Documentação Beautiful Soup**. Disponível em: <https://www.crummy.com/software/BeautifulSoup/bs4/doc.ptbr/>. Acesso em: 30 nov. 2019.

SCHRENK, M. **Webbots, spiders, and screen scrapers : a guide to developing Internet agents with PHP/CURL**. [s.l.] No Starch Press, 2012.

SILVA, A. N. R. DA. **Sistemas de Informações geográficas para planejamento de transportes**. [s.l.] Universidade de São Paulo, 1998.

SILVA, M. C. Sistemas De Informações Geográficas Na Identificação De Doenças E Epidemias. **Tekhne e Logos**, v. 8, n. 4, p. 94–106, 2017.

SILVEIRA, I. H. DA; OLIVEIRA, B. F. A. DE; JUNGER, W. L. Utilização do Google Maps para o georreferenciamento de dados do Sistema de Informações sobre Mortalidade no município do Rio de Janeiro, 2010-2012*. **Epidemiologia e Serviços de Saúde**, v. 26, n. 4, p. 881–886, nov. 2017.

SOLINA, F.; RAVNIK, R. **Georeferencing works of literature**. Proceedings of the ITI 2010, 32nd International Conference on Information Technology Interfaces. **Anais**...Cavtat, Croatia: IEEE, 2010Disponível em: <https://ieeexplore.ieee.org/document/5546410>

SZTUTMAN, P. **Análise da qualidade posicional das bases do Google Maps, Bing Maps e da Esri para referência espacial em projetos em SIG: aplicação**

para o município de São Paulo. São Paulo: Biblioteca Digital de Teses e Dissertações da Universidade de São Paulo, 9 dez. 2014.

TRIPADVISOR LLC. **TripAdvisor**. Disponível em: <https://www.tripadvisor.com.br/?fid=721dacfd-5ffb-4513-b1b3-2e53e84f53b7>. Acesso em: 30 nov. 2019.

TUPÃ ESTÂNCIA TURÍSTICA. **Digitalização possibilita preservação de documentos do município - Prefeitura de Tupã**. Disponível em: <https://www.tupa.sp.gov.br/noticia/225/digitalizacao-possibilita-preservacao-de-documentos-do-municipio.html>. Acesso em: 28 nov. 2019.

WALLAPOP. **wallapop, Local Free Classified Ads**. Disponível em: <https://www.wallapop.com/>. Acesso em: 30 nov. 2019.

WANG, J. et al. **The crawling and analysis of agricultural products big data based on Jsoup**. 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015. **Anais**...IEEE, ago. 2016Disponível em: <http://ieeexplore.ieee.org/document/7382112/>. Acesso em: 30 jun. 2019

WIKIMAPIA. **WikiMapia - Vamos descrever o mundo todo!** Disponível em: <http://wikimapia.org/#lang=pt&lat=-25.090885&lon=-50.157394&z=13&m=w>. Acesso em: 25 set. 2019.

WILSON, J. P.; SWIFT, J. N.; GOLDBERG, D. W. **Geocoding best practices: Review of eight commonly used geocoding systems**. Los Angeles, CA: [s.n.].

YIN, F.; HE, X.; LIU, Z. **Research on Scrapy-Based Distributed Crawler System for Crawling Semi-structure Information at High Speed**. 2018 IEEE 4th International Conference on Computer and Communications (ICCC). **Anais**...IEEE, dez. 2018Disponível em: <https://ieeexplore.ieee.org/document/8781062/>. Acesso em: 23 set. 2019