UNIVERSIDADE FEDERAL FLUMINENSE
CÉSAR VARGAS DOS SANTOS
DANIEL VARGAS DOS SANTOS

WEB SCRAPING: UMA SOLUÇÃO PARA COLETA DE INFORMAÇÕES DE ARTIGOS CIENTÍFICOS.

Niterói
2020

CÉSAR VARGAS DOS SANTOS
DANIEL VARGAS DOS SANTOS

WEB SCRAPING: UMA SOLUÇÃO PARA COLETA DE INFORMAÇÕES DE ARTIGOS CIENTÍFICOS.

Trabalho de Conclusão de Curso submetido ao Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Computação.

Orientador(a):
ALTOBELLI DE BRITO MANTUAN

NITERÓI
2020

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S237w Santos, Daniel Vargas dos
Web Scraping: Uma Solução para Coleta de Informações de Artigos Científicos / Daniel Vargas dos Santos, César Vargas dos Santos ; Altobelli de Brito Mantuan, orientador. Niterói, 2020.
49 f. : il.

Trabalho de Conclusão de Curso (Graduação em Tecnologia de Sistemas de Computação)-Universidade Federal Fluminense, Instituto de Computação, Niterói, 2020.

1. Web scraping. 2. Scopus. 3. Automatização na captura de dados. 4. Pesquisa farmacêutica. 5. Produção intelectual. I. Santos, César Vargas dos. II. Mantuan, Altobelli de Brito, orientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecário responsável: Sandra Lopes Coelho - CRB7/3389

CÉSAR VARGAS DOS SANTOS
DANIEL VARGAS DOS SANTOS

WEB SCRAPING: UMA SOLUÇÃO PARA COLETA DE INFORMAÇÕES DE ARTIGOS CIENTÍFICOS.

Trabalho de Conclusão de Curso submetido ao Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Computação.

Niterói, ___ de _____ de 2020.

Banca Examinadora:

_____

Prof. Altobelli de Brito Mantuan, MSc. – Orientador
UFF – Universidade Federal Fluminense

_____

Prof a. Simone de Lima Martins, Doutora em Informática – Avaliador
UFF – Universidade Federal Fluminense

CÉSAR

Dedico este trabalho a Deus, que colocou pessoas fascinantes em minha caminhada de vida.

DANIEL

Dedico este trabalho a minha amada esposa que com tanta paciência, carinho e amor apoiou-me . Lilian, eu te amo!

AGRADECIMENTOS
CÉSAR

Agradeço a Deus, por me ajudar nas dificuldades, por me guiar e me iluminar em meus objetivos.

Agradeço aos meus pais, por todos os sacrifícios, e esforços para me conduzir nessa jornada.

Agradeço aos meus irmãos, pelo grande incentivo e apoio.

Agradeço aos meus amigos, pela força e colaboração durante essa caminhada.

Agradeço aos meus professores do colegial, que me ajudaram a ter uma base forte.

Agradeço ao meu orientador Altobelli, pela paciência e atenção dedicada.

AGRADECIMENTOS
DANIEL

A Deus, que me deu forças para concluir este projeto e que sempre iluminou a minha caminhada.

Aos meus pais, Madalena e Deoclides, os dois maiores incentivadores das realizações dos meus sonhos, pelo carinho, afeto, dedicação e cuidado.

Ao, meu colega e irmão, César Vargas, por ter aceitado enfrentar essa caminhada comigo.

Ao meu orientador Altobelli Mantuan por seu precioso direcionamento, que me manteve focado na trilha certa para a conclusão deste projeto.

“Eu quero saber como Deus pensa. O resto são detalhes.”
Albert Einstein

RESUMO

A Internet vem sendo cada vez mais essencial como fonte de informação. Essa fonte quase inesgotável, abrangente, carregada de informação, exige um olhar com novas perspectivas. Obter dados sem que se faça grandes esforços, vem sendo uma necessidade que pode determinar o sucesso em qualquer área de pesquisa, projetos e até negócios. Temos então ferramentas conhecidas por realizarem essa atividade conhecida como *web scraping*, que buscam realizar coletas de dados cruciais, de forma automatizada e com agilidade. Dados antes coletados de forma manuais, hoje são personalizados de acordo com os desejos de seus utilizadores por meio de *softwares*. Nesse trabalho, procurou-se apresentar a implementação da ferramenta de busca na base de dados da *Elsevier,* conhecida por seu vasto acervo de artigos, denominada de *Scopus,* de forma a realizar melhorias de desempenho e buscar novas fontes novas de dados.

Palavras-chaves: *web scraping*, *Scopus*, automatização na captura de dados, projeto de software e pesquisa farmacêutica

ABSTRACT

The internet has become increasingly essential as a source of information. This almost inexhaustible source, embracing, loaded with information, requires a look with new perspectives. Obtain data without much effort, has been a necessity that can determine success in any area of research, projects and even business. We then have tools known to perform this activity known as web scraping, seeking to collect crucial data, in an automated and agile way. Data previously collected manually, today they are customized according to the wishes of their users through software. In this work, sought present the implementation of the search tool in the Elsevier database, known for its vast collection of articles, called Scopus, in order to make performance improvements and seek new sources of data.

Key words: *web scraping*, *Scopus*, data capture automation, *software* design and pharmaceutical research.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

MD5 – *Message-Digest algorithm 5*

DNS – *Domain Name System*

IP – *Internet Protocol*

HTTP – *Hypertext Transfer Protocol*

HTML – *Hypertext Markup Language*

CSS – *Cascading Style Sheets*

IDE – *Integrated Development Environment*

JS – JavaScript

DOM – *Document Object Model*

RIA – *Rich Internet Application*

OOP – *Object-oriented Programming*

POO – Programação orientada a objetos

JSON – *JavaScript Object Notation*

CSV – *Comma-separated Values*

XML – *Extensible Markup Language*

SQL – *Structured Query Language*

URL – *Uniform Resource Locator*

RDF – *Resource Description Framework*

DOI – *Digital Object Identifier*

ODT - *Object Data Transfer*

SGBD - Sistema de Gestão de Banco de Dados

TCC – Trabalho de Conclusão de Curso

SUMÁRIO

1 INTRODUÇÃO

As universidades, a partir da metade do século XX, passaram a ter uma função importante na geração de inovações que vieram a influenciar as atividades humanas em diversos campos [1]. Partindo dessa conjuntura, pode-se observar que a pesquisa científica distanciou-se da busca desinteressada e passou a buscar conhecimento que pudesse contribuir para a geração de novas tecnologias.

Devido à globalização, se faz necessário que o setor produtivo seja mais competitivo e é somente com inovações que isso se tornará possível [2]. Impulsionadas por tal demanda crescente de tecnologia, um número cada vez maior de publicações científicas têm sido produzidas.

Esse afluxo de publicações resultou em uma crescente demanda por questões de informação. No entanto, o excesso de informação disponível dificulta o seu consumo, tornando a aquisição de referências um desafio [3]. Por essa razão é que *scripts* são desenvolvidos no sentido de automatizar a seleção e obtenção das mesmas. Esse *scripts* também são chamados de *scraping* [4].

Neste trabalho é implementado um *software* capaz de automatizar os processos de busca, extração e organização de informações básicas em publicações científicas cuja a fonte é a base de dados do Elsevier, conhecida como *Scopus* [34].

Essas informações coletadas servirão de apoio para a escolha das publicações que sejam mais relevantes à pesquisa. Dentre as contribuições deste trabalho pode-se pontuar:

- Profundo entendimento sobre a API do *Scopus*, incluindo suas restrições de funcionamento;
- Otimização do tempo de processamento quando comparado a algoritmos similares [5].
- Identificação e tratamento de erros, o que contribuiu para um melhor desempenho da aplicação;

- A descoberta de um novo fluxo de busca feito através do indexador DOI que, além de permitir a captura dos mesmo dados do fluxo anterior, traz consigo a possibilidade de fazer buscas em redes não habilitadas e de adquirir uma série de outros dados que não estavam disponíveis, incluindo os de referência bibliográfica.

Para mais detalhes sobre o conteúdo definido, o desenvolvimento e os testes relacionados a este trabalho, incluindo o código-fonte, basta visitar o repositório público no GitHub através do endereço eletrônico a seguir: https://github.com/altobellibm/CEDERJ_2020_DANIEL_VARGAS_CESAR_SANTOS.git

2 FUNDAMENTAÇÃO TEÓRICA

Um artigo da *Scientific American* divulgou que em 2016 produziu-se tantos dados quanto em toda a história da humanidade até 2015 [6]. Isso só é possível devido ao surgimento de novas tecnologias que além de facilitar que os seres humanos exponham todo tipo de informação, também vem conectando diversos tipos de objetos à *Internet*. Além disso, os instrumentos de medição têm evoluído, ao ponto de as métricas serem cada vez mais precisas e abrangentes.

Neste capítulo, serão apresentadas as categorias de dados mais relevantes, as suas formas de armazenamento, e as técnicas para a coleta e análise de dados.

2.1 COLETA DE DADOS

A coleta de dados é um processo que visa adquirir dados acerca de variáveis da qual se tem interesse. Os dados são utilizados não apenas para tarefas de pesquisa, mas, também, para planejamento, desenvolvimento, experimentações, teste de hipóteses, estudos de caso etc [7]. Ou seja, a coleta de dados é o núcleo de qualquer tipo de pesquisa, seja ele em ciências exatas, físicas, humanas e sociais, marketing, negócios e assim por diante.

Esse processo é feito de maneira estruturada utilizando técnicas específicas de pesquisa. Ainda que, em cada área, os métodos de coleta possuam variações, deve-se ressaltar que a garantia de precisão e veracidade da mesma permanece [7]. A coleta de dados pode se dar através de três fontes de dados: fonte primária, secundária e terciária [8].

As fontes primárias, também definida como fontes originais, são as adquiridas diretamente pelo autor da pesquisa e buscam atender as demandas próprias

[9]. Como por exemplo, a pesquisa de intenção de voto, realizada pelo DataFolha, no qual os dados de intenção de voto são coletados diretamente da população, utilizando técnicas de amostragem.

A fonte secundária é, neste ponto, onde os dados já sofreram algum tipo de manipulação, sendo agrupados e organizados a fim de facilitar o seu uso [9]. Como exemplos, tem-se os dicionários, enciclopédias, bases de dados etc.

As fontes terciárias são guias para encontrar fontes primárias e secundárias [8]. Por exemplo, guias de leitura, livrarias, catálogos etc.

Neste trabalho as fontes de dados são provenientes de um tipo de catálogo digital e, portanto, pertencem a uma fonte terciária.

2.2 TIPOS DE DADOS

Uma perspectiva pela qual pode-se entender os dados são as diversas formas pelas quais eles são categorizados. Cada esquema de categorização é um modelo relativo a algum aspecto dos dados. Dentre as diversas formas de categorização de dados, essa seção tratará de quatro grandes grupos: quantitativos, qualitativos [10], primários e secundários [11].

Os dados qualitativos são as características que não possuem valores, ou seja, são dados não numéricos que são definidos por várias categorias que representam uma classificação do indivíduo. Além disso, esses dados podem ser divididos em nominais e ordinais [10], como demonstrado exemplos na Tabela 1.

Tabela 1: Exemplos de dados nominais e ordinais.

Tipo	Descrição	Exemplos
Nominal	Não existe ordenação dentre as categorias	Sexo, cor dos olhos, fumante/não fumante, doente/sadio
Ordinal	Existe uma ordenação entre as categorias	Escolaridade, estágio da doença, mês de observação

Os dados quantitativos são dados de valores mensuráveis, ou seja, que podem ser expressados através de valores numéricos, podendo ser de caráter intervalar ou razão, tomados em uma escala métrica definida, podemos classificar as mesmas como discretas quando assumem valores finitos, e contínuas quando assumem valores infinitos [10].

Tabela 2: Tabela sobre variáveis quantitativas.

Variáveis	Tipo	Descrição	Exemplos
Quantitativas	**Discreta**	Conjunto com um valor enumerável.	Número de alunos, Número de filhos, Idade, etc.
	Continua	Conjunto correspondente a um intervalo real	Peso, Altura, Salário, etc.

Pode-se a partir da Tabela 2 definir a forma de como esses dados podem ser coletados, em seguida processados e apresentados em sua forma final.

Como mencionado na seção 2.1, os dados primários estão associados às fontes primárias de coleta. Sendo assim, eles são obtidos pelo pesquisador diretamente com o objetivo de abordar o fenômeno estudado. Além do mais, os pesquisadores ao partirem de uma ação planejada, tornam a extração dos dados mais precisa permitindo responder perguntas direcionadas o que concede maior confiabilidade aos dados coletados [11].

Em geral, como desvantagem, esse tipo de dado possui alto custo, pois exige a constituição de uma equipe para coleta e sua extração é de difícil acesso. Os dados primários são coletados por meio de diferentes métodos de pesquisa. Como por exemplo: entrevistas em profundidade, questionários, estudos de caso, grupos de discussão etc [11].

Também, como mencionado na seção 2.1, os dados secundários são associados às fontes secundárias de coleta. São obtidos através de bases de dados como IBGE, FGV, IPEA entre outros. São mais rápidos de obter e possuem um custo menor [11].

Sua desvantagem, em geral, está nos dados não serem atualizados e que nem todas as perguntas a qual são importantes estão respondidas. Por consequência o sucesso da pesquisa pode ser afetado [11]. Logo, é imprescindível a atenção quanto à metodização da coleta, cuidando desde a definição de critérios quanto à seleção de fontes, metodologia empregada para a sua coleta até sua manipulação.

2.3 ORGANIZAÇÃO E ARMAZENAMENTO

Com a coleta de dados houve a necessidade de armazená-los e dispô-los de maneira prática. Os dados podem estar organizados de três maneiras diferentes: estruturados, semiestruturados e não-estruturados [12]. Na Figura 1, representados as diferentes organizações de dados.

Os dados estruturados são os mais tradicionais estando presente em sistemas e relatórios [11]. Porém, atualmente, houve um rápido aumento na geração de dados semiestruturados e não-estruturados. Por isso a escolha da forma de armazenamento dependerá da necessidade e do problema.

Figura 1: Diferentes estruturas dos dados. [14]

Os dados estruturados são aqueles que obedecem regras rígidas previamente estabelecidas para armazenamento de novos registros. Um exemplo de da-

dos estruturados, são os banco de dados relacionais, que possuem tabelas composta de campos que são organizados em colunas que representam atributos chaves. Esses atributos podem ser do tipo:

- **Nome**, que representa um conjunto de caracteres sem espaços;
- **CPF**, que representa um conjunto de números não mutáveis;
- **Idade**, que representa um conjunto de números mutáveis;

Cada linha dessa tabela corresponde a um registro, ou seja, representa um conjunto de atributos de uma determinada instância e que, nesse caso, representaria uma pessoa [11].

Os dados semiestruturados são dados que possuem uma estrutura que também é flexível, ou seja, que possuem uma estrutura com um alto grau de alterabilidade, tornando possível qualquer alteração de seu escopo o que permite ter um melhor controle [12]. Um exemplo de normalmente utilizado é o arquivo semiestruturado XML (*Extensible Markup Language*) que possui rótulos de abertura e fechamento, possuindo uma forma amigável de visualização para diversos *softwares* como o Navegador, por exemplo.

Os dados não estruturados diferentes dos dados estruturados não possuem um formato pré-definido geralmente são dados que são apresentados de forma bruta, ou seja, são dados que não possuem uma forma ou tamanho máximo. Uma forma básica de armazenamento é a partir de um arquivo de texto que pode conter diferentes tipos de dados, outros exemplos de dados não estruturados são: Imagens, Vídeos, Musicas, TXT, etc [11].

Na Tabela 3, são apresentadas as características de cada estruturação de dados.

Tabela 3: Resumo de organização de dados.

Tópicos	Dados Estruturados	Dados Semiestruturados	Dados Não Estruturados
Organização	Os dados são formatados de forma organizada	Parcialmente organizados	Forma não organizada de dados
Esquema de dados	Esquema de dados fixo é usado.	Ele carece da estrutura formal de modelo de dados	Esquema desconhecido
Gestão de Transação	Amadurecida e são utilizadas várias técnicas de simultaneidade	Basicamente uma adaptação do SGBD (Sistema de Gestão de Banco de Dados)	Não há gerenciamento de transações nem simultaneidade.
Flexibilidade	São menos flexíveis, pois os dados estruturados são dependentes do esquema.	Mais flexível do que estruturado, no entanto, menos do que os dados não-estruturados	São muito flexíveis devido à ausência de esquema
Buscas	Permite junções complexas	Permite as consultas sobre o mesmo nó	Aqui, apenas as consultas textuais são possíveis
Escalabilidade	Complexibilidade alta ao dimensionar o esquema do banco de dados	Simples comparado aos dados estruturados	Muito escalável
Exemplos	Dados do sistema de gerenciamento de banco de dados relacional	XML e JSON	Arquivos multimídia

2.4 NAVEGADORES DE INTERNET

Com relação a estruturas de páginas, os navegadores utilizam páginas semi-estruturadas (HTML), de forma a apresentar para o usuário uma interface utilizando as *tags* para uma formatação mais amigável, conectando-as por através de links denominados de hipertexto [15]. O processo apresentado na Figura 2, representa as requisições do acesso ao endereço de um *website* por um usuário, através de seu navegador, que realiza uma sequência de eventos para a obtenção e o tratamento do HTML que será apresentado ao usuário.

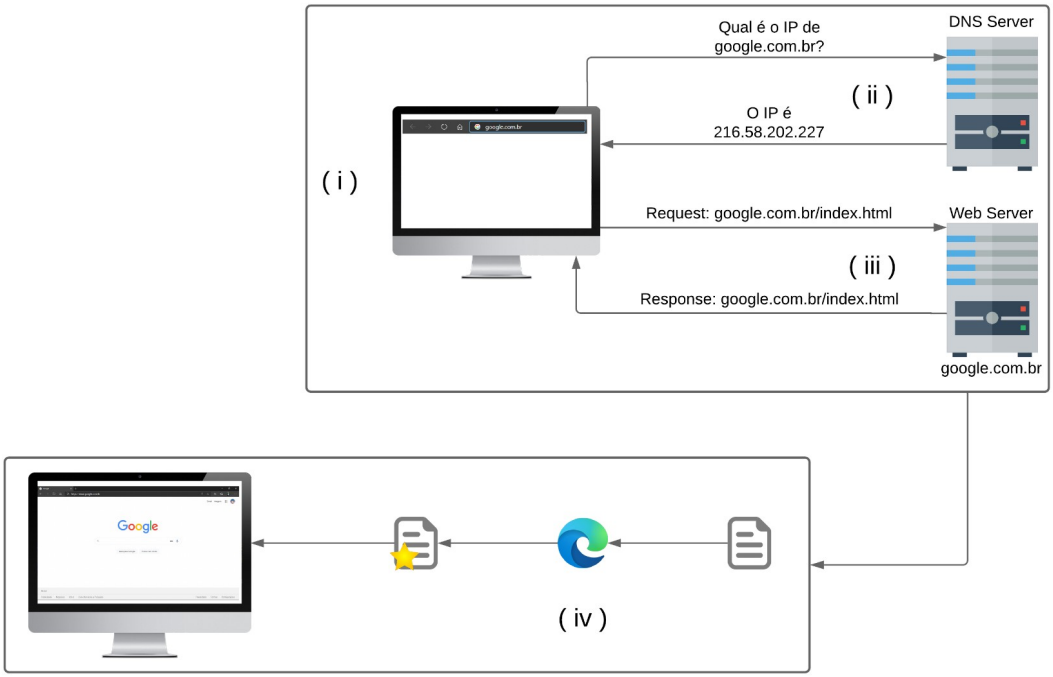

Figura 2: Processo requisição e simplificação de um HTML.

I. O navegador realiza a extração do nome do hospedeiro do endereço.
II. Através do DNS o navegador obtém o endereço de IP do servidor que hospeda a página.
III. Por meio do endereço o navegador realiza o download do arquivo HTML, e todos os arquivos correspondentes a referenciação nele presente.
IV. O navegador realiza operações de simplificação textual para a exibição do conteúdo ao usuário.

2.5 ESTRUTURA DE PÁGINAS

Todos os sites possuem estruturas HTML básica, compostas por uma ou diversas páginas que utilizam de links (hipertexto) [15] para vincular seus conteúdos. Cada página possui seus específicos conteúdos que variam entre gráficos, imagens, textos, entre outros formatos.

Embora possuam variações, todas elas dispõem de estruturas comuns denominadas de blocos que são estruturas bem definidas, delimitadas por um par de *tags*. que definem o seu cabeçalho que contém metainformação sobre seu conteúdo e um corpo que agrupa o restante dos elementos de conteúdos distintos [16].

Na Figura 3, tem-se a representação de uma página básica, composta da demarcação HTML (*Hypertext Markup Language*) [17], criada para a internet e tinha o intuito de facilitar a troca de documentos entre universidades [15], trazendo uma estrutura visual mais clara.

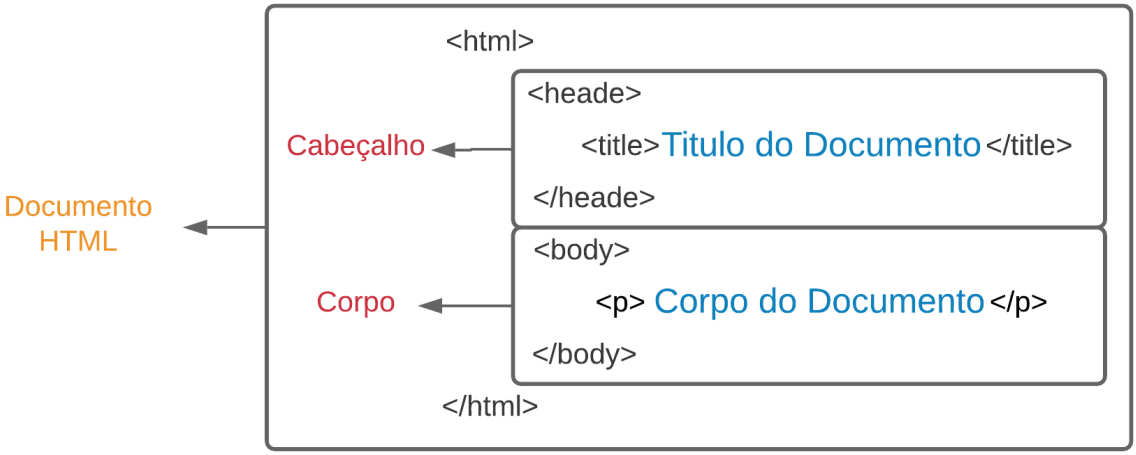

Figura 3: Estrutura básica de um documento HTML

A principal vantagem desse modelo é a personalização com a possibilidade da introdução de arquivos distintos, tornando um arquivo mais rico em detalhe e consequentemente mais atraente visualmente como apresentado na Figura 4. Sua

popularização ocorreu a partir do ano de 1990 [18] com programas capazes de entender e reproduzir o HTML conhecidos como navegadores.

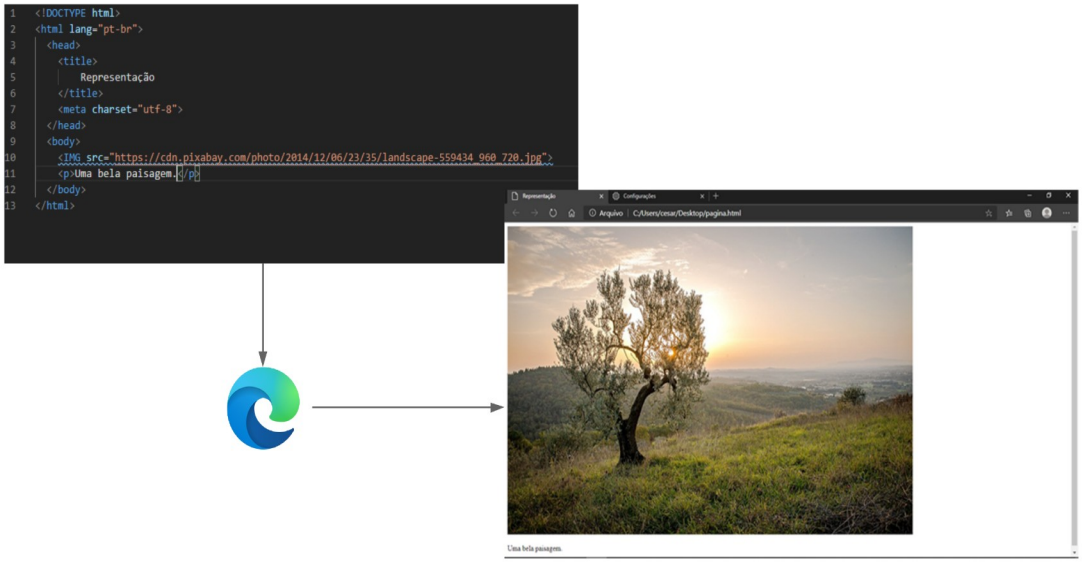

Figura 4: Código HTML apresentado em um navegador.

Em 1998 criou-se o DOM (*Document Object Model*) que permitiu manipular o HTML como um conjunto de objetos independentes como apresentado na Figura 5. Isso adicionou um grau de dinamismo as páginas, tornando-as em Aplicações de Internet Rica (RIA) [19] que foram suportadas com a evolução dos navegadores. Pode-se destacar nessa nova tecnologia, grupos que tiveram um grande destaque, tais como folhas de estilo(CSS) que flexibiliza a estilização por meio de grupos previamente classificados, JavaScript (JS) [20] que executam códigos de alto nível diretamente no navegador, permitindo alterações em uma página em tempo real.

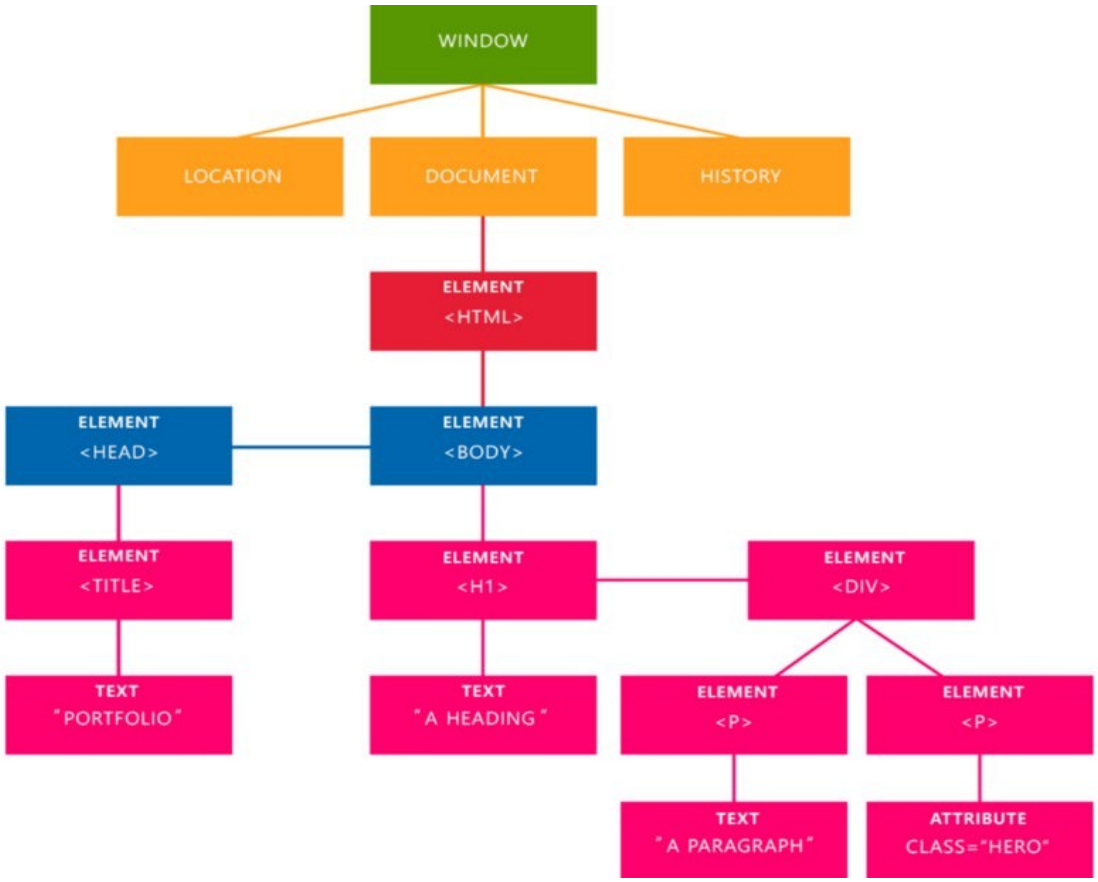

Figura 5: Estrutura do HTML lida através do DOM. [21]

Por mais que o acesso às páginas da *web* seja mais conhecido por meio de navegadores, há diversos programas de que podem interagir como clientes para os servidores, com o mesmo princípio apresentado nos navegadores, utilizando de linguagens de alto nível, por meio de algumas linhas de código para extrair o HTML de qualquer página, portanto que ele possua o acesso à mesma.

Temos então que uma página da *web* não se trata apenas de um padrão sólido consolidado, mas um conjunto de elementos definidos que fazem parte de toda internet, que possibilita uma avaliação do conteúdo de uma página da *web*, uma vez que seu conteúdo hipertexto seja obtido com sucesso e tenha seus objetos devidamente interpretados.

Considerando determinado conjunto de páginas de internet como um aplicativo e, também, que ao serem criadas foram empregadas boas práticas de desenvolvimento de *software,* seguindo as definições da linguagem, isso as tornará está-

veis e de simples manutenção. Partindo dessa ideia, para que seja possível a aplicação de filtros nessas páginas, será necessário que:

- Os dados de um mesmo tipo sejam identificados por uma mesma classe ou uma mesma *tag* HTML,
- O documento HTML possua código essencialmente adequado ao ser construído e que seja feita escolha de *tag* mais apropriada para determinada estruturação de conteúdo.
- Mesmo com o aumento de utilização de *frameworks* para a construção de páginas dinâmicas, a estruturação básica do código HTML ainda se mantém, permitindo que a escolha de *tags* específicas seja feita.

Portanto, tendo em vista que uma determinada página possua os atributos mencionados acima, a mesma poderá ser escolhida como fonte de extração de uma certa coleção de dados.

2.6 WEB SCRAPING

No ambiente *web*, as principais fontes de dados são os documentos HTML. Esses documentos são, a princípio, publicados para serem exibidos em navegadores (*Web Browsers*). No entanto, como esses dados são objetos de interesse para estudos, eles devem ser convertidos em estruturas mais apropriadas a tal. O processo de coleta pode ser feito de dua maneiras: manualmente ou automatizado.

- Na modalidade manual, navega-se até a página *web* desejada onde as informações de interesse serão identificadas visualmente e, por fim, o dado será coletado para processamento posterior.

- Já na modalidade automatizada, utiliza-se um aplicativo de computador para realizar as requisições HTTP em busca de uma determinada página web. Ao receber a página, o aplicativo fará a busca pelas informações de-

mandas e, ao final, retornará as partes do documento no qual o dado está presente.

Como em todo documento de texto, nas páginas *web* pode-se utilizar técnicas de busca desde as mais simples, como a busca por palavra-chave, até as mais complexas, como a busca por expressões regulares *regex*, do inglês *regular expression* [22].

No entanto, as técnicas de busca, da qual se tem interesse, são as que utilizam a estrutura do documento para alcançar uma determinada informação. A título de exemplo, tem-se a linguagem Xpath [23] que estabelece um caminho até a informação. Há, também, ferramentas como a biblioteca *Beautiful Soup* [24] que realizam o *parsing* do documento, ou seja, criam uma representação encadeada de objetos.

O termo *parsing,* nada mais é do que a análise sintática do texto [25] de qualquer linguagem de programação, inclusive dos documentos HTML. Sua função é verificar se a estrutura de uma página web está de acordo com as regras que a definem. Portanto, ao encontrar entradas que não estejam dentro das regras gramaticais da linguagem, o analisador sintático as recusará. Em outras palavras, é uma etapa onde ocorrerá a tradução de *strings* em uma estrutura compreensível pelo computador e que, posteriormente, poderá extrair informações de interesse.

3 SCRAPING SCOPUS

Neste capítulo, trataremos sobre os detalhes da implementação, ou seja, discorremos sobre as tecnologias que fazem parte desse projeto e a razão de sua escolha. Apresentaremos, também, um diagrama de classes representando uma visão geral do sistema, falaremos sobre a fonte que foi utilizada para o desenvolvimento da ferramenta e, por fim, de como se dá a busca pelas informações desejadas.

3.1 TECNOLOGIAS

Para que se possa extrair dados de páginas *Web é* necessário escolher uma ferramenta *Web scraping*. Por isso, as três principais ferramentas gratuitas serão analisadas levando em consideração os aspectos a seguir: se realiza buscas recursivamente em páginas *Web* (*WebSpider*), se possui *API* flexível a fim de que se possa programar as funcionalidades necessárias e, por fim qual o tipo de licença associada.

Tabela 4: Comparação entre as bibliotecas Python.

Biblioteca	Web Spider	API	Licença
Scrapy	Sim	Sim	BSD-3
Selenium Webdriver	Não	Sim	Apache-2
Ghost.py	Não	Sim	MIT

Como pode ser visto na Tabela 4, a biblioteca que representa a melhor escolha é a *Scrapy* [26], pois possui todos os quesitos e, além disso, é uma ferramenta robusta, escrita em *Python*, veloz, multitarefas e possui um motor assíncrono [27]. A

biblioteca *Scrapy* possui ferramental necessário para lidar tanto com o *back-end*, para gerenciar e realizar requisições HTTP quanto com o *front-end* das aplicações Web, para navegar via CSS *path* [28] e de *Xpath [23]*. Após a coleta dos dados a biblioteca Scrapy ainda fornece maneiras de convertê-los para os seguintes formatos: XML [29], JSON [30] e CSV, sendo esse último o que será utilizado para apresentar os dados neste trabalho.

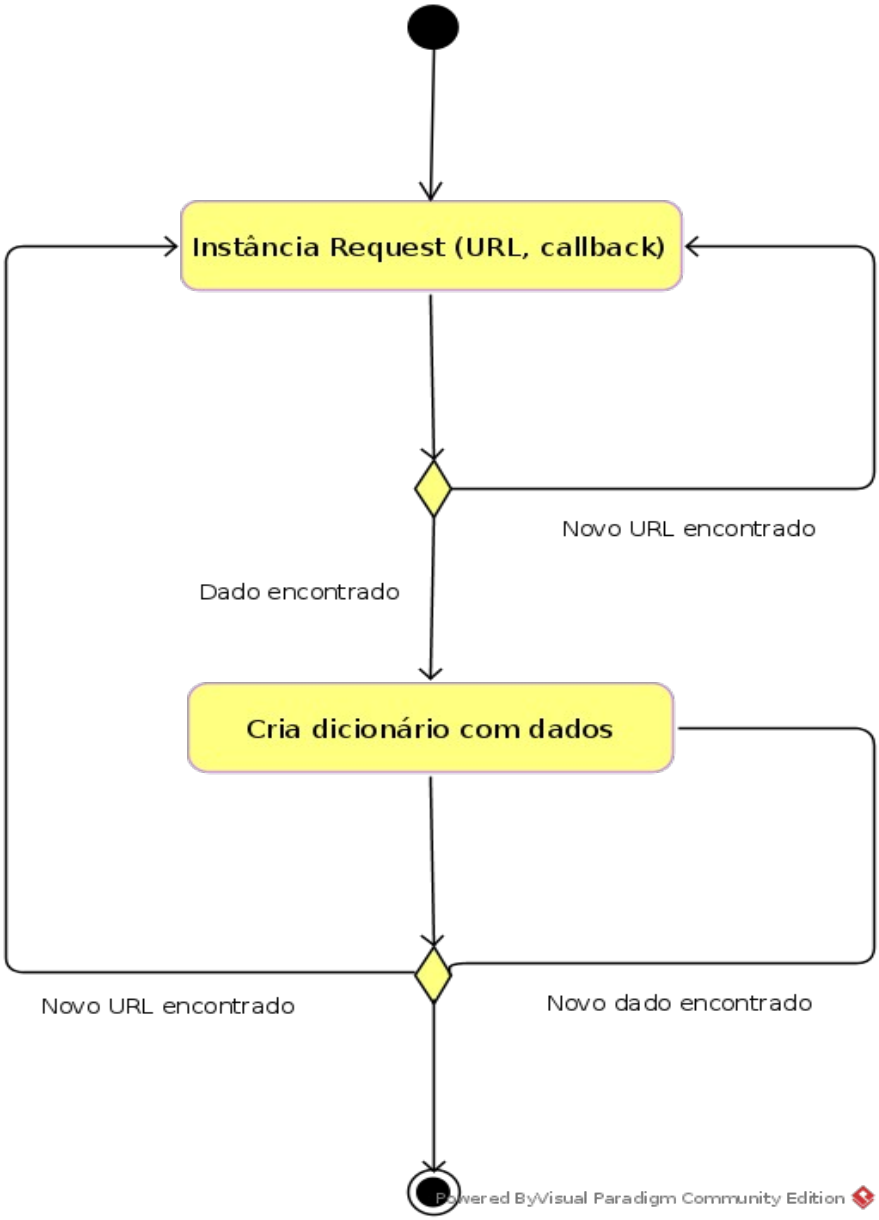

Figura 6: Fluxo de requisições mínimo.

Como pode ser visto na Figura 6, há um esquema de implementação mínima do *Scrapy* que consiste em sobrescrever o método *start_request* e estender a classe *Spider*. Portanto, o método *start_request* instanciará sucessivamente a classe Request que, antes de mais nada, necessitará do endereço URL cujo o gerenciamento é feito pelo *framework*.

Além disso, o *Scrapy* possui autonomia tanto para realizar a requisição quanto para realizar nova tentativa em caso de falha. Assim sendo, caso a requisição tenha êxito, é chamada a função responsável por criar objetos do tipo *Request* que, também, está incumbida de realizar a busca desejada dentro do documento. Nesse caso serão criados objetos do tipo dicionário que serão tratados pelo *Scrapy* como item final da coleta de dados e, então, armazenados.

3.2 DIAGRAMA DE CLASSE

Baseada no *framework Scrapy* [26], a Figura 7 reproduz a estrutura de forma geral e os relacionamentos entre as classes. Para tal, foi utilizado o modelo de programação orientada a objetos, POO, que faz uso dos conceitos de herança e de sobreposição, onde determinadas classes genéricas do *framework* foram estendidas a fim de criar um artifício para execução de tarefas ligadas a atividade de *web scraping*. Além disso, devido à necessidade de inclusão de novas fontes, foi necessária a especialização de uma classe responsável por essa lógica.

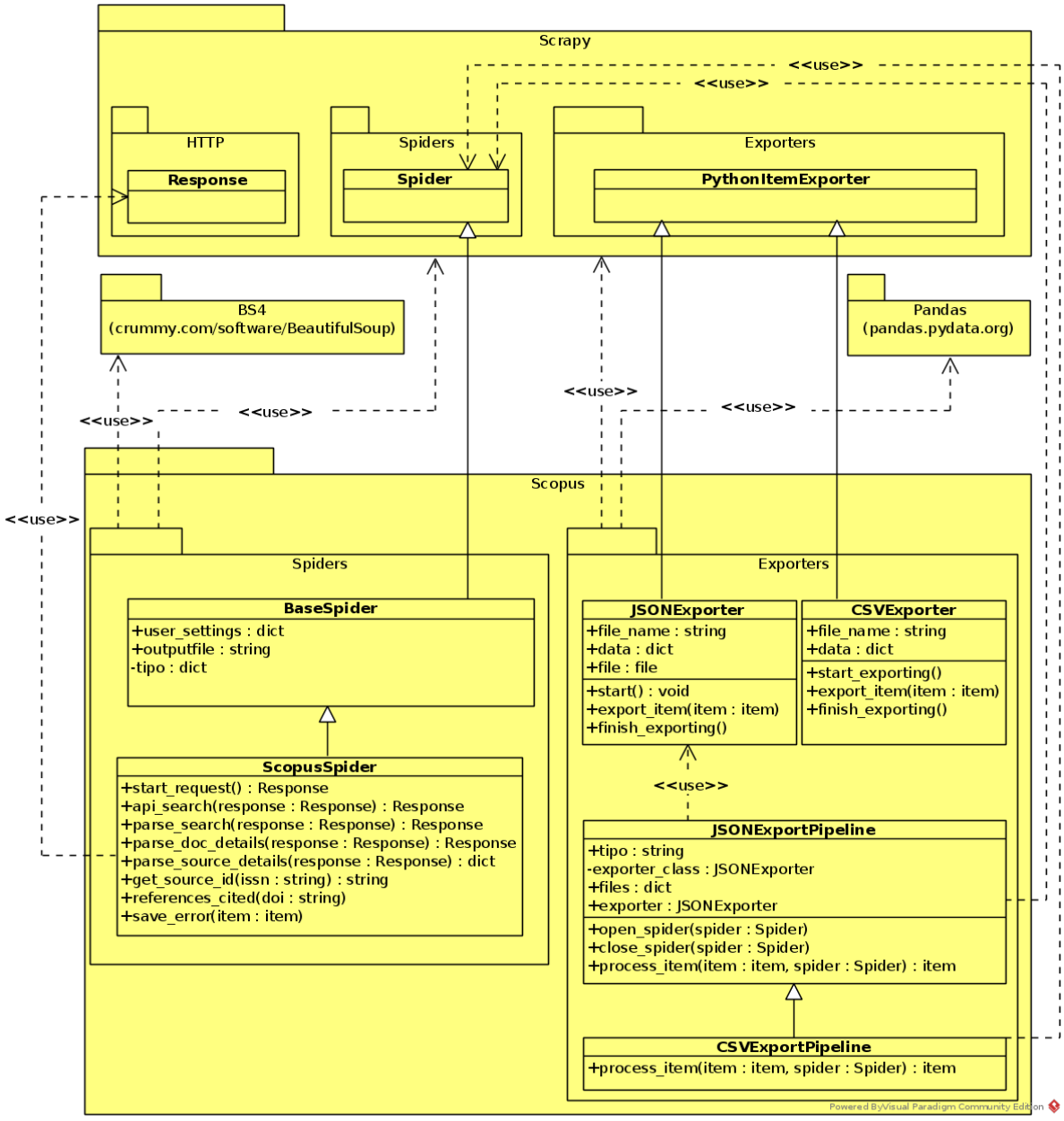

Figura 7: Diagrama de classes do sistema.

Tabela 5: Descrição das funcionalidades das classes da aplicação.

Classe	Descrição
CSVExporter	O objetivo dessa classe é realizar a escrita em disco dos dados coletados nas fontes. O formato de saída utilizado é o CSV com codificação ANSI. Isso é feito como auxílio da biblioteca Pandas [31]

JSONExportPipeline	O objetivo dessa classe é de servir como base para a implementação do CSVExportPipeline. Onde a estruturação do *framework* é obedecida e são implementados uma série de métodos que auxiliarão no processo de exportação de dados [32].
CSVExportPipeline	Essa classe possui o objetivo de exportar os dados em formato CSV e para isso ela sobrescreve alguns métodos e propriedades que pertencem a classe pai JSONExportPipeline.
BaseSpider	Esta classe é a especialização da classe *Spider* a qual está ligada a extração de dados. Embora essa classe devesse ser abstrata, a mesma precisou ser concreta devido as tecnologias escolhidas, porém não deve ser instanciada. Dela será criada, por especialização, a classe *Scopus Spider* o qual deverá sobrepor seus métodos[33]
ScopusSpider	A classe ScopusSpider é encarregada pela extração de dados da fonte "*Scopus*". Os dados são coletados através de requisições feitas a uma API que fornece os dados em JSON. Além disso, essa classe verifica se há duplicidade de artigos consultando o DOI de cada um.

3.3 SCOPUS

O *Scopus* é uma das maiores bases de dados do mundo referentes a resumos e citações revisadas de livros, revistas científicas e artigos publicados de praticamente todos os periódicos acadêmicos de menor ou maior importância no mundo [34]. Seu perfilamento de autores e instituições facilita a localização de novos artigos dos referidos autores nessas instituições. O modelo de dados do *Scopus* foi desen-

volvido baseado na ideia de que os artigos são escritos por autores afiliados a instituições. Na figura 8, pode ser visto, de maneira simplista o modelo relacional.

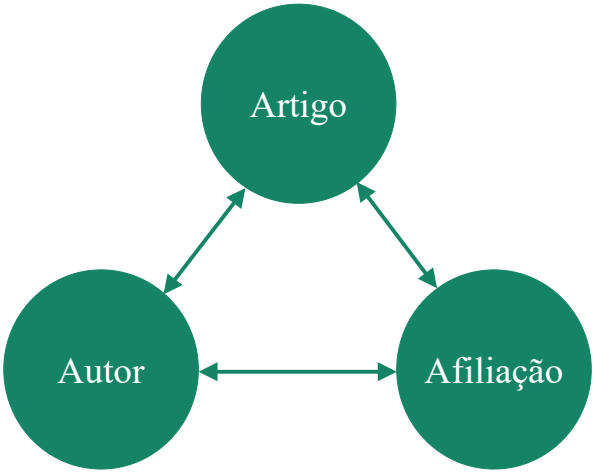

Figura 8: Modelo de dados relacional do Scopus.

3.3.1 DOCUMENTAÇÃO

Como forma de permitir que desenvolvedores escrevam programas para extrair dados automaticamente, o *Scopus* disponibiliza uma Interface de programação de aplicativos (API) [36] que retorna um documento em formato JSON. Embora o formato JSON seja um objeto de transferência de dados quase universal, o *framework Scrapy* só fornece ferramentas para lidar com páginas HTML. No entanto, o Python possui uma biblioteca nativa chamada JSON [37] para navegação nesses documentos e as trata como objetos onde os nomes dos atributos são como as chaves de um dicionário (*Map*). Na figura 9 é apresentado como a API do *Scopus* atua como um mediador entre a base de dados do *Scopus* e a aplicação criada.

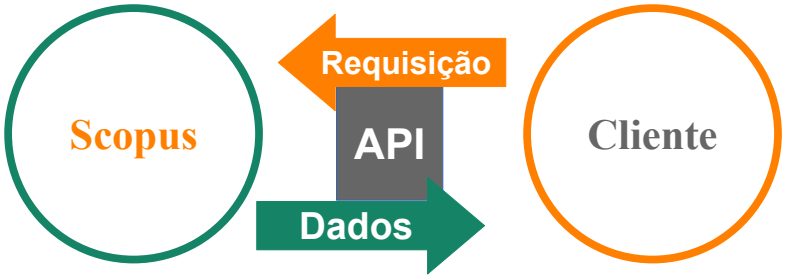

Figura 9: Esquema de atuação a API do Scopus.

Na tabela 6, conforme as instruções presentes da documentação da API, é apresentada a URL utilizada para a busca. Em sua estrutura o termo *QUERY_STRING* deverá ser substituído pelo parâmetro a ser buscado.

Tabela 6: Representação da URL. [38]

Base URL + Route	Query Param
https://api.elsevier.com/content/search/scopus	?query=QUERY_STRING

No entanto, antes de realizar qualquer requisição é necessário adquirir uma chave da API do *Scopus* que será fornecida após a realização de cadastro no site do mesmo. Essa chave habilita recursos da API, cotas e níveis de serviço [38]. A mesma deverá constar no cabeçalho da requisição HTTP, da seguinte forma:

X-ELS-APIKey: [chave da API] [38]

O processo de acedimento feito por assinantes e não assinantes são diferenciados pelo endereço IP. Como exemplo temos a rede interna da UFF que possui habilitação ao serviço e ao ser realizada requisição, a resposta se ajustará as da coluna “assinante” [38]. Do contrário, quando a requisição vem de um IP desconhecido, a resposta será proveniente da coluna “não assinante”.

Mesmo de posse de uma chave de acesso e possuindo o IP habilitado para buscas, o serviço ainda possui algumas limitações, tal como: quantidade de requisições que podem ser feitas semanalmente e quantidade de requisições por segundo [39]. Na tabela 7, são apresentadas as taxas de estrangulamento.

Tabela 7: Tabela das limitações de utilização da API do Scopus. [39]

Nº	Nome API	Habilitado	Não assinante	Assinante	Cota Semanal	Pedidos / segundo
1	Título de Série	Sim	PADRÃO, visualizações COVERI-MAGE Padrão 25 resultados Máximo 200 resultados	PADRÃO, COBERTURA, MELHORADO Padrão 25 resultados Máximo 200 resultados	20.000	6
2	Metadados de contagem de citações	Não	N / D	Visualização PADRÃO Padrão de 25 resultados Máximo de 200 resulta-dos	50.000	24
3	Visão geral das citações	Não	N / D	Visualização PADRÃO Padrão de 25 resultados Máximo de 200 resulta-dos	20.000	6
4	Classificações de assunto	Sim	Sem Restrições	Sem Restrições	N/D	N/D
5	Recuperação abstrata	Sim	Vista META	Todas as visualizações, visualização COMPLETA padrão	10.000	9
6	Recuperação de afiliação	Sim	N / D	Todas as visualizações, visualização padrão *STANDARD*	5.000	9
7	Recuperação do Autor	Sim	N / D	Todas as visualizações, visualização padrão *STANDARD*	5.000	6
8	Pesquisa de afiliação	Sim	N / D	Padrão de 25 resultados Máximo de 200 resulta-dos Limite de resultados de 5000 itens	5.000	6
9	Pesquisa de Autor	Sim	N / D	Padrão de 25 resultados Máximo de 200 resulta-dos Limite de resultados de 5000 itens	5.000	6
10	Scopus Search	Sim	Visualização PADRÃO / Padrão 25 resultados	Visualização PADRÃO Máx. 200 resultados Visualização COMPLETA Máx. 25 resultados Visualização COMPO-NENTE Máx. 25 resultados	20.000	9

				Limite total de resultados de 5000 itens sem paginação do ‘‘cursor’’.		
11	Feedback do autor	Não	N / D	N / D	N / D	N / D

Ao exceder os limites de fornecimento, a API do *Scopus* retornará o erro “429 *Too Many Requests*”, porém, após um período de 7 dias, as cotas são redefinidas [40]. Abaixo segue uma lista de respostas que a API pode retornar. Na tabela segue todas respostas que o servidor do *Scopus* pode fornecer.

Tabela 8: Respostas possíveis do servidor do Scopus. [40]

Status	**Resposta**	**Descrição**	**Formato de Retorno**
200	Scopus JSON	XML que representa o documento solicitado	JSON
200	Scopus Atom	XML que representa o documento solicitado	ATOM + XML
200	Scopus XML	XML que representa o documento solicitado	XML
400	Pedido Inválido	Erro que o ocorre quando informações inválidas são enviadas.	XML
401	Erro de Autenticação	Erro que ocorre quando um usuário não pode ser autenticado devido a credenciais ausentes ou inválidas.	XML
403	Erro de autorização	Erro que ocorre quando um usuário não pode ser autenticado ou os direitos não podem ser validados.	XML
405	Método HTTP inválido	Erro que ocorre quando o Método HTTP solicitado é inválido.	XML
406	Tipo MIME inválido	Erro que ocorre quando o tipo MIME solicitado é inválido.	XML
429	Cota excedida	Erro que ocorre quando um solicitante excede os limites de cota associados à sua chave de API.	XML
500	Erro Genérico	Erro de propósito geral, normalmente devido a erros de processamento de *back-end*.	XML

3.3.2 INFORMAÇÕES BUSCADAS

Na aplicação desenvolvida, o termo a ser buscado deve ser definido dentro de um Objeto de transferência de Dados, ODT, juntamente à chave e a definição da quantidade de artigos a ser retornada. Por definição, "*count*": 0 não significa ausência e busca, mas, sim, que serão retornados o máximo de itens permitidos pela API.

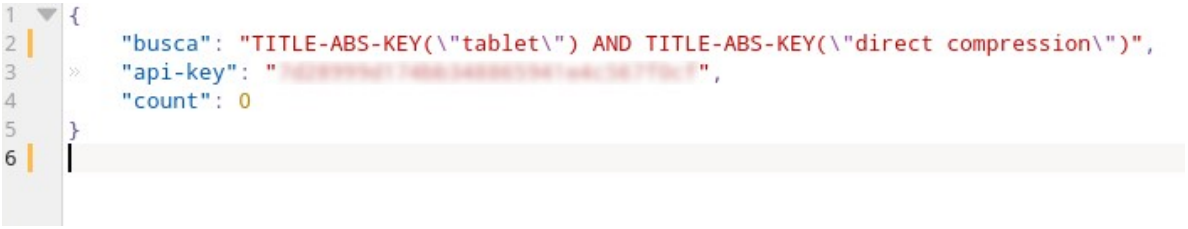

Figura 10: Modelo de entrada de dados na aplicação.

Como resultado da busca realizada, um arquivo XML é retornado e com ele as chaves "*entry*" e "*search-results*" que ao serem seguidas levam a outro XML contendo mais artigos desejados que, por fim, irá compor a lista de artigos desejados. A API do *Scopus* retorna uma série de informações relativas a cada artigo, entretanto algumas informações adicionais são necessárias, como:

- Pontuação recebida pela publicação nos três últimos anos;
- *Link* para o artigo referenciado pelo código DOI [41];
- *Abstract* completo.

  Como pode ser visto no Gráfico 1, são apresentadas as etapas de processamento ocorridos na API do *Scopus*.

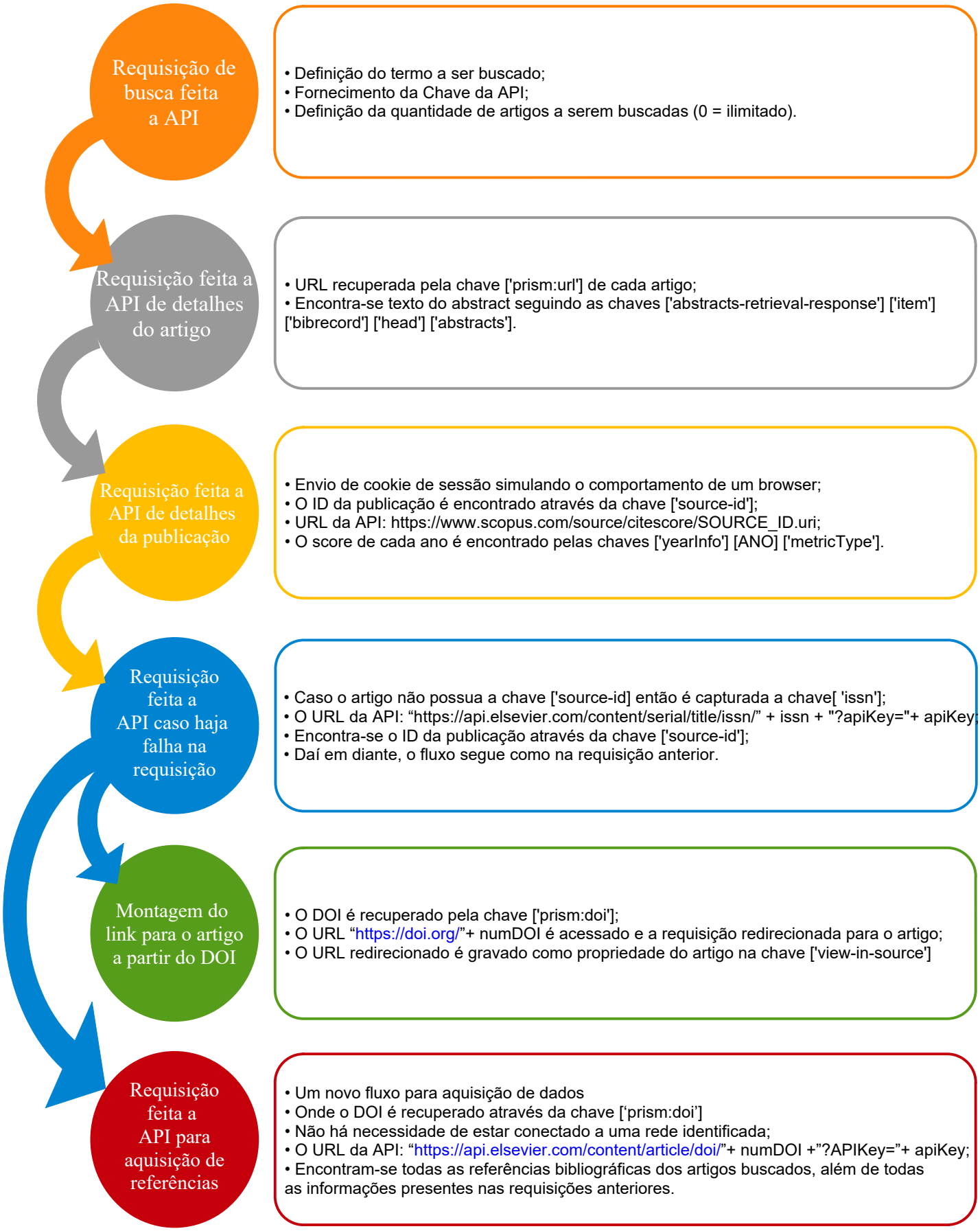

Gráfico 1: Processamento simplificado.

4 TESTES

Nesse capítulo serão apresentados detalhes sobre a execução do *Scopus* constatando o tempo de execução, melhorias de desempenho e eventuais falhas em comparação ao algorítimo apresentado em “Web Scraping: Uma Solução para Coleta de Informações na Área Farmacêutica” de Denier e Zanovelli [5] que possui um tempo elevado para retorno de buscas.

Os testes foram realizados em um computador com as seguintes configurações:

Tabela 9: Configuração do sistema usado nos testes.

Processador	Intel(R) Core(TM) i7-2600 3,40GHz
Memória física	10 Gigabytes
Sistema operacional	Windows 10 Pro
Arquitetura	x64
Python	3.7.4
Anaconda	2019.10

Os parâmetros iniciais necessários para obtenção dos dados que é capturado por meio de requisições diretamente na estrutura do *website scopus*, até a obtenção dos dados requisitados pelo especialista, O mesmo deve fornecer parâmetros contidos em um arquivo do tipo JSON, que é passado o como parâmetro de entrada para o programa, juntamente do padrão CSV de saída e o destino que o arquivo será gerado contendo os dados gerados.

```
{
  "busca": "( ( TITLE-ABS-KEY ( tenofovir )  AND  TITLE-ABS-KEY ( lamivudine )  AND  TITLE-ABS-KEY ( efavirenz ) )  OR  ( TITLE-ABS-KEY ( fixed-
  "api-key": "SUA APIKEY AQUI ",
  "count": 0
}
```

Figura 11: Estrutura do arquivo JSON de entrada.

O tempo foi capturada através da biblioteca ‘time’ chamada no arquivo _main_.py onde foi calculada a diferença do tempo final do tempo inicial.

A configuração do JSON, apresentada na Figura 11, foi utilizada com o intuito de obter o máximo possível de artigos. Como resultado, o algoritmo desenvolvido retornou 5.000 artigos. Entretanto, realizando a mesma busca no *website* do *Scopus,* obteve-se um total de 16.682 resultados. Essa diferença entre as quantidades de artigos encontrados deve-se a restrição imposta pela API do Scopus que impede as requisições de retornarem mais de 5000 artigos.

Além disso, com relação ao tempo de execução, essa configuração levou cerca de 1.534,63 segundos, ou seja, 25,58 minutos para retornar 4.999 artigos completos e 1 artigo incompleto, que foi capturado no tratamento de erros. Esse erro ocorreu devido à ausência de identificadores no artigo. Ainda assim, em um pós-processamento, foi possível extrair os dados incompletos de tal artigo que foi armazenado em um arquivo JSON especifico para erros desse tipo.

O desempenho desse *software* em comparação ao seu antecessor se mostrou consideravelmente mais rápido, de acordo com o gráfico de testes realizados com 3 buscas distintas, de 1.000 artigos cada, onde seus tempos são representados nos gráficos a seguir:

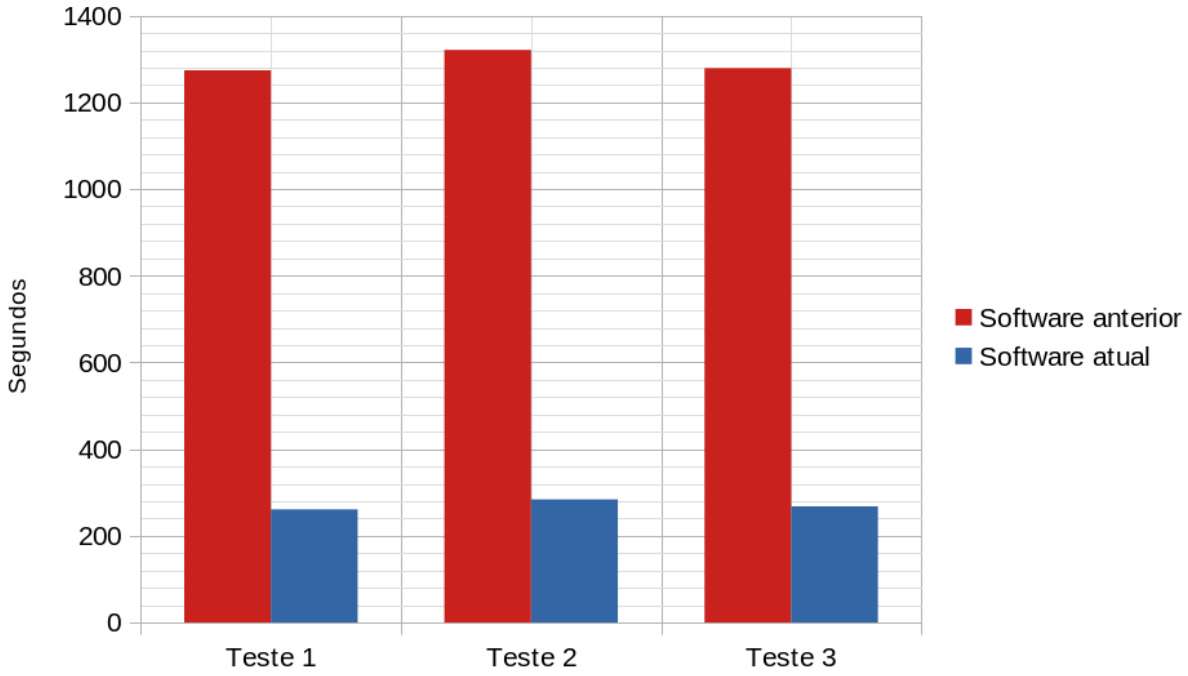

Gráfico 2: Comparação de tempo do software anterior e o atual.

De acordo com os testes, houve um ganho de cerca de 477% a mais de velocidade com relação ao *software* anterior [5]. Na Tabela 10, são apresentados parâmetros que foram utilizados para realização dos testes em ambos os *softwares*:

Tabela 10: Parâmetros utilizados nos testes.

Teste	Busca	*Count*
1	((TITLE-ABS-KEY (tenofovir) AND TITLE-ABS-KEY (lamivudine) AND TITLE-ABS-KEY (efavirenz)) OR (TITLE-ABS-KEY (fixed-dose AND combin*) OR TITLE-ABS-KEY (fdc)))	**1000**
2	(TITLE-ABS-KEY (hiv-1) TITLE-ABS-KEY (hiv) OR TITLE-ABS-KEY (\"Immunodeficiency Virus\") OR TITLE-ABS-KEY (aids) OR TITLE-ABS-KEY (hiv-2) OR TITLE-ABS-KEY (\"HIV Infections\") OR TITLE-ABS-KEY (htlv-iii) OR TITLE-ABS-KEY (\"Acquired Immune Deficiency Syndrome\") OR TITLE-ABS-KEY (\"Acquired Immuno-Deficiency Syndrome\") OR TITLE-ABS-KEY (\"treatment as prevention\") OR TITLE-ABS-KEY (prep) OR TITLE-ABS-KEY (\"pre-exposure prophylaxis\") OR TITLE-ABS-KEY (\"pre-exposure prophylaxi\"))	**1000**

3	((TITLE-ABS-KEY (tenofovir) AND TITLE-ABS-KEY (lamivudine) AND TITLE-ABS-KEY (efavirenz)) OR (TITLE-ABS-KEY (fixed-dose AND combin*) OR TITLE-ABS-KEY (fdc))) AND (TITLE-ABS-KEY (hiv-1) TITLE-ABS-KEY (hiv) OR TITLE-ABS-KEY (\"Immunodeficiency Virus\") OR TITLE-ABS-KEY (aids) OR TITLE-ABS-KEY (hiv-2) OR TITLE-ABS-KEY (\"HIV Infections\") OR TITLE-ABS-KEY (htlv-iii) OR TITLE-ABS-KEY (\"Acquired Immune Deficiency Syndrome\") OR TITLE-ABS-KEY (\"Acquired Immuno-Deficiency Syndrome\") OR TITLE-ABS-KEY (\"treatment as prevention\") OR TITLE-ABS-KEY (prep) OR TITLE-ABS-KEY (\"pre-exposure prophylaxis\") OR TITLE-ABS-KEY (\"pre-exposure prophylaxi\"))	**1000**

Através de pesquisas foi também identificado uma nova forma de obtenção de dados utilizando do DOI (*Digital Object Identifier*) [41], que poderiam ser obtidos de forma externa a rede com acesso ao *Scopus*.

https://api.elsevier.com/content/article/doi/10.1016/j.ibusrev.2010.09.002?APIKey=sua Apikey aqui

Através dessa URL conseguimos obter em rede externa, dados como o *Abstract, Publisher, Author, CoverDate*, ISSN, etc.

Através de uma rede com o acesso total ao *Scopus* conseguimos ir além e obter todas as referências de um determinado artigo e *Author keywords*, através de um teste na captura das referências, limitado a 10 artigos, exigiu um grande poder de processamento, devido a grande quantidade de dados apresentado em cada página, havendo erros de localização para alguns artigos, portanto retornando referências de 6 artigos em aproximadamente 30 segundos.

5 CONCLUSÕES E TRABALHOS FUTUROS

Nesse trabalho, foi apresentado um algoritmo de *web scraping* para automatização de acesso a dados de publicações científicas da base do *Scopus*. Durante o desenvolvimento desse algoritmo, uma das dificuldades encontradas foi a de obtenção de informações relativas as limitações de acesso e ao retorno de erros da API da base de dados. Essas dificuldades, em grande parte, foram ocasionadas pela disposição não intuitiva das informações no web site do *Scopus*. Outra dificuldade encontrada foi quanto ao suporte no Brasil [42], em que várias tentativas de contato com os especialistas foram feitas, tanto por e-mail quanto por telefone, em busca de informações sobre: detalhes de erros, assinaturas e suas limitações de acesso, entretanto, nenhum retorno fora obtido.

Em contrapartida, com o conhecimento mais aprofundado sobre a API do *Scopus* foi possível alcançar melhoria de 477% na velocidade do algoritmo desenvolvido em comparação ao algoritmo similar proposto no trabalho "*Web Scraping:* Uma Solução para Coleta de Informações na Área Farmacêutica" de Denier e Zanovelli. Além disso, foi descoberta uma fonte de dados alternativa, identificada através do DOI [41], que em um único XML concedeu os mesmos dados que foram adquiridos através da API, mais as referências bibliográficas em uma rede não habilitada, ou seja, sem a necessidade de assinatura do ao serviço do *Scopus*.

Por fim, segue a sugestão para trabalhos futuros. Poderá haver incrementos ao algorítmo, como a possibilidade de utilizar dados de forma resumida em qualquer fonte de acesso à internet, lidar com a limitação de 5000 artigos, a utilização do programa de forma offline, e possivelmente incrementando o programa para obter mais dados através do fluxo alternativo proporcionado pelo DOI.

REFERÊNCIAS BIBLIOGRÁFICAS

1 GARDNER, Eldon.; ***History of biology.*** *Edition: First, Publisher: Burgess, Minneapolis,* 1972.

2 SCHAFF, Adam; **A sociedade informática.** Edição: Primeira, Editora: UNESP; São Paulo, 1990.

3 HEWSON, Claire; STEWART, David. ***Internet Research Methods***, 2016. *Wiley StatsRef: Statistics Reference Online, 1–6*.

4 SCRAPINGHUB; et al. ***Framework Python*** **para extração de dados de websites.** <https://scrapy.org> Acesso em 22 set. 2020.

5 DERNIER, M.; ZANOVELLI, V. ***Web Scraping:*** **Uma Solução para Coleta de Informações na Área Farmacêutica.** 2019. 78 f. Trabalho de Conclusão de Curso (Tecnologia em Sistemas de Computação) – Universidade Federal Fluminense, Niterói, 2019.

6 HELBING, D. et al. ***Will Democracy Survive Big Data and Artificial Intelligence? - Scientific American.*** Disponível em: <https://www.scientificamerican.com/article/will-democracy-survive-big-data-and-artificial-intelligence/>. Acesso em: 19 set. 2020.

7 KABIR MUHAMMAD SAJJAD, Syed. ***Basic Guidelines for Research: An Introductory Approach for All Disciplines****, Edition: First, Chapter: 9, Publisher: Book Zone Publication, Chittagong-4203, Bangladesh, pp.201-275*.

8 **FACIMED – Metodologia Científica**. Disponível em: <https://moodle.ufsc.br/pluginfile.php/1255592/mod_resource/content/1/unidade_2_Fontes_de_Informacao/html/top1_1.html>. Acesso em: 20 set. 2020.

9 MATTAR, F. N.. **Pesquisa de Marketing: Metodologia e Planejamento**. 6. ed. São Paulo: Atlas, 2005.

10 MAGALHÃES, Marcos. PEDROSO DE LIMA, Antonio. **Noções de Probabilidade e Estatística**, Edição: sétima, Capítulo 1 – Introdução à Análise Exploratória de Dados, Editora: EDUSP, pág. 5-31.

11 ZOZUS NAHM, Meredith. ***The data book collection and management of research data-Chapman and Hall CRC Press***, 2017, p. 36.

12 ABITEBOUL, Serge; BUNEMAN, Peter; SUCIU, Dan. ***Data on the web – From relations to semistructured data and XML***, Morgan Kaufmann *Publishers*.

13 **Diferença entre dados estruturados, não estruturados e semiestruturados.** Disponível em: <https://www.astera.com/pt/type/blog/structured-semi-structured-and-unstructured-data/>. Acesso em: 01 out. 2020.

14 **Dados Estruturados e Não Estruturados • Universidade da Tecnologia.** Disponível em: <https://universidadedatecnologia.com.br/dados-estruturados-e-nao-estruturados/>. Acesso em: 02 nov. 2020.

15 LONGMAN WESLEY, Addison. ***A History of HTML***, 1998**.** <https://www.w3.org/People/Raggett/book4/ch02.html> Acesso em 20 set. 2020.

16 LAWSON, Bruce; SHARP, Remy. **Introducing HTML5**, 2nd Edition, 2011. Livro. New Riders Press, Berkeley, Canadá.

17 BERNERS-LEE, Tim. ***Information Management: A Proposal***, 1989. <https://www.w3.org/History/1989/proposal.html> Acesso em 30 set. 2020.

18 CAILLIAU, Robert; ASHMAN, Helen. ***Hypertext in the Web – a History***, 1999. Artigo científico – Laboratório Europeu de Partículas Físicas, Suíça e Universidade de Nottingham, Reino Unido.

19 FRATERNALI, Piero; ROSSI, Gustavo; SÁNCHEZ-FIGUEROA, Fernando. ***Rich Internet Applications***, 2010. Publicação acadêmica – *IEEE Internet Computing*, vol. 14, no. 3, pp. 9-12.

20 FLANAGAN, David. ***JavaScript: The Definitive Guide, 3rd edition***, 1998. Livro. *O'Reilly & Associates, Sebastopol*, Canadá.

21 **Entendendo o DOM** <https://tableless.com.br/entendendo-o-dom-document-object-model/> Acesso em 30 out. 2020.

22 AHO, Alfred; ULLMAN, Jeffrey. ***Foundations of Computer Science***, 1992, capítulo 10 – *Patterns, and Regular Expressions*.

23 **XML Path Language (XPath) 3.0** <https://www.w3.org/TR/xpath-30/> Acesso em 20 ago. 2020.

24 RICHARDSON, Leonard. **Biblioteca Python para "*scraping* de tela"**. <https://crummy.com/software/BeautifulSoup> Acesso em 06 set. 2020.

25 **PARSING | Significado, definição em Dicionário Inglês.** Disponível em: <https://dictionary.cambridge.org/pt/dicionario/ingles/parsing>. Acesso em: 10 out. 2020.

26 SCRAPINGHUB; *et al*. ***Framework* Python para extração de dados de web-sites**. <https://scrapy.org> Acesso em 06 set. 2020.

27 Kazil, J. & Jarmul, J. **Data Wrangling with Python,** 2016. Sebastopol, O'Reilly Media, Inc.

28 **Seletores CSS.** <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors> Acesso em 12 nov. 2020.

29 **XML – *Extensible Markup Language*.** <https://www.w3.org/XML/> Acesso em 10 out. 2020.

30 **JSON – *JavaScript Object Notation*.** < http://www.json.org/> Acesso em 12 out. 2020.

31 **PANDAS. Biblioteca Python para análise de dados**. <https://pandas.pydata.org/> Acesso em 12 out. 2020.

32 ***ITEM EXPORTERS*. Documentação do *framework* Scrapy**. <https://docs.scrapy.org/en/latest/topics/exporters.html> Acesso em 15 out. 2020.

33 ***SPIDERS*. Documentação do *framework* Scrapy**. <https://docs.scrapy.org/en/latest/topics/spiders.html> Acesso em 15 out. 2020.

34 **ELSEVIER Scopus | O maior banco de dados da literatura revisada por pares.** Disponível em: <https://www.elsevier.com/pt-br/solutions/scopus>. Acesso em: 02 nov. 2020.

35 ***Scrapy Autothrotle extension*.** <https://docs.scrapy.org/en/latest/topics/autothrottle.html> Acesso em 10 de nov. 2020.

36 **Scopus *Search* API.** <https://dev.elsevier.com/documentation/ScopusSearchAPI.wadl> Acesso em 10 nov. 2020.

37 **Biblioteca Python json.** <https://docs.python.org/3/library/json.html> Acesso em 12 nov. 2020.

38 **ELSEVIER Elsevier *Developer Portal***. Disponível em: <https://dev.elsevier.-com/tecdoc_api_authentication.html>. Acesso em: 02 out. 2020.

39 **ELSEVIER Elsevier *Developer Portal***. Disponível em: <https://dev.elsevier.-com/api_key_settings.html>. Acesso em: 02 out. 2020.

40 **ELSEVIER *Authenticate API***. Disponível em: <https://dev.elsevier.com/documentation/AuthenticationAPI.wadl>. Acesso em: 02 out. 2020.

41 **DOI – *Digital Object Identifier*.** <https://www.doi.org/> Acesso em 23 nov. 2020.

42 **ELSEVIER Elsevier Suporte e Contato**. Disponível em: <https://www.elsevier.com/pt-br/support>. Acesso em: 06 dez. 2020.