UNIVERSIDADE FEDERAL FLUMINENSE
MIQUÉIAS SANTOS DA SILVA DERNIER
VINICIUS GONÇALVES ZANOVELLI

WEB SCRAPING: UMA SOLUÇÃO PARA COLETA DE INFORMAÇÕES NA ÁREA FARMACÊUTICA

Niterói
2019

MIQUÉIAS SANTOS DA SILVA DERNIER
VINICIUS GONÇALVES ZANOVELLI

WEB SCRAPING: UMA SOLUÇÃO PARA COLETA DE INFORMAÇÕES NA ÁREA FARMACÊUTICA

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
2019

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

D436w Dernier, Miquéias Santos da Silva
WEB SCRAPING: UMA SOLUÇÃO PARA COLETA DE INFORMAÇÕES NA ÁREA FARMACÊUTICA / Miquéias Santos da Silva Dernier, Vinicius Gonçalves Zanovelli ; Altobelli Mantuan, orientador. Niterói, 2019.
78 f. : il.

Trabalho de Conclusão de Curso (Graduação em Tecnologia de Sistemas de Computação)-Universidade Federal Fluminense, Instituto de Computação, Niterói, 2019.

1. Web scraping. 2. Mineração de dados. 3. Pesquisa farmacêutica. 4. Produção intelectual. I. Zanovelli, Vinicius Gonçalves. II. Mantuan, Altobelli, orientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

MIQUÉIAS SANTOS DA SILVA DERNIER
VINICIUS GONÇALVES ZANOVELLI

WEB SCRAPING: UMA SOLUÇÃO PARA COLETA DE INFORMAÇÕES NA ÁREA FARMACÊUTICA

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 2019.

Banca Examinadora:

_____

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

_____

Prof. Eduardo Vera Sousa, MSc. – Avaliador
UFF – Universidade Federal Fluminense

_____

Profª. Marianne Grilo Rezende, MSc. – Avaliador
UFRJ – Universidade Federal do Rio de Janeiro

VINICIUS

Dedico este trabalho aos meus muitos pais, tios, avós, irmãos e a minha esposa.

AGRADECIMENTOS
VINICIUS

Ao meu Tutor Coordenador Gioliano Bertoni pelo estímulo, atenção e paciência.

Aos Tutores Camila Curcio Gomes e Jairo Godinho Dutra pelo apoio e atenção.

A Coordenação e a Secretaria do curso pela presteza e cordialidade.

Ao meu Orientador Altobelli de Brito Mantuan pelo estímulo, atenção e disponibilidade.

A especialista farmacêutica Marianne Grilo Rezende pela colaboração e disponibilidade.

Ao colega Miquéias Dernier pela oportunidade de aprendizado durante esta experiência.

A todos os meus familiares e amigos pelo apoio e compreensão.

Em especial ao meu pai, a minha mãe e a minha esposa pelo apoio e incentivo.

“A Escola é uma arena onde grupos sociais lutam por legitimidade e poder”.

Dinair Leal da Hora

RESUMO

A internet é uma fonte virtualmente inesgotável de informações e que desempenha um papel cada vez mais relevante no auxílio à tomada de decisões. Obter um dado preciso em um tempo razoável e mantê-lo atualizado, sem grande esforço, pode ser um fator determinante para o sucesso de um projeto, negócio ou pesquisa. Atualmente, existem tecnologias e ferramentas que possibilitam a extração automática e seletiva (mineração) de dados da internet, atividade conhecida como *web scraping*, de forma simples e rápida através de pequenos *softwares* criados com poucas linhas de código. Neste trabalho apresentamos a implementação de três robôs de mineração, para três respectivas fontes específicas de dados, relevantes à pesquisa farmacêutica. Para elaboração deste projeto, uma reunião foi feita com um pesquisador farmacêutico especialista que contribuiu para a definição dos requisitos desta demanda e das três fontes de dados das quais as informações pertinentes são extraídas. Ainda com a participação do especialista farmacêutico, foram realizados testes com o sistema desenvolvido para a validação dos resultados obtidos considerando a abrangência da coleta automática, a precisão dos dados coletados e o tempo de processamento necessário para cada uma das fontes de dados definidas.

Palavras-chave: *web scraping*, mineração de dados, extração automática de dados, projeto de *software* e pesquisa farmacêutica.

ABSTRACT

The Internet is a virtually-inexhaustible source of information that plays an increasingly important role in aiding decision-making. Obtaining accurate data within a reasonable time frame and effortless keeping these data up to date can be a determining factor in the success of a project, business or research. Nowadays, there are technologies and tools that make automatic and selective internet data extraction (data-mining or "web scraping") possible, simply and quickly, through developing small software using only a few lines of code. In this paper, we present the implementation of three data-mining robots for three respective and specific data sources, all relevant to pharmaceutical research. To develop this project, a meeting was held with a specialist pharmaceutical researcher who contributed to the definitions regarding this demand's requirements and the definition of the three data sources from which the information is extracted. Yet supported by the pharmaceutical specialist, the developed system was tested to validate the observed results taking into consideration the comprehensiveness of the automatic data gathering, the accuracy of the collected data and the processing time for each one of the defined data sources.

Keywords: web scraping, data-mining, automatic data extraction, software project and pharmaceutical research.

LISTA DE ILUSTRAÇÕES

Figura 1: Visão geral do mecanismo de acesso às páginas de internet.....26
Figura 2: Estrutura básica de um documento HTML em uma IDE de programação. 27
Figura 3: Representação visual do código HTML em um navegador de internet.……28
Figura 4: Estrutura de composição de um documento HTML contendo CSS e JS....29
Figura 5: Documento HTML representado como texto e como árvore de objetos.....32
Figura 6: Visualização de um documento HTML.....33
Figura 7: Diagrama do caso de uso *Scrap* DrugBank.....34
Figura 8: Diagrama do caso de uso *Scrap* MerckMillipore.....37
Figura 9: Diagrama do caso de uso *Scrap* Scopus.....39
Figura 10: Diagrama de atividades de um programa Scrapy mínimo.....44
Figura 11: Diagrama de Classes do sistema.....47
Figura 12: Página de detalhes do fármaco Tylenol no DrugBank.....48
Figura 13: Página de detalhes do fármaco Lactose Monohidratada no Merck.....49
Figura 14: Exemplo de *script* para execução dos testes.....53
Figura 15: Modelo de entrada para o *scraper* do *website* DrugBank.....54
Figura 16: Modelo de arquivo de saída para o *scraper* do *website* DrugBank.....55
Figura 17: Modelo de entrada para o *scraper* do *website* Merck Millipore.....55
Figura 18: Modelo de arquivo de saída para o *scraper* do *website* Merck Millipore.. 56
Figura 19: Modelo de entrada para o *scraper* do *website* Scopus.....56
Figura 20: Modelo de arquivo de saída para o *scraper* do *website* Scopus.....57

LISTA DE TABELAS

Tabela 1: Descrição do caso de uso *Scrap* DrugBank.....35
Tabela 2: Descrição do caso de uso *Scrap* MerckMillipore.....37
Tabela 3: Descrição do caso de uso *Scrap* Scopus.....39
Tabela 4: Configurações do sistema utilizado nos testes.....53

LISTA DE GRÁFICOS

Gráfico 1: Índice de chaves JSON retornadas pela API de busca do Scopus.....51
Gráfico 2: Etapas para o processamento da fonte Scopus.....52

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

ADR – *Adverse drug reaction*

FDA – *Food and Drug Administration (U.S. Federal agency)*

HPSG – *Head-driven phrase structure grammar*

MD5 – *Message-Digest algorithm 5*

ICMS – *Institute of Chinese Medical Sciences*

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*

SUMÁRIO

AGRADECIMENTOS VINICIUS....6
RESUMO....8
ABSTRACT....9
LISTA DE ILUSTRAÇÕES....10
LISTA DE TABELAS....11
LISTA DE GRÁFICOS....12
LISTA DE ABREVIATURAS E SIGLAS....13
1 INTRODUÇÃO....16
2 TRABALHOS RELACIONADOS....18
2.1 DISCUSSÃO....20
3 FUNDAMENTAÇÃO TEÓRICA....21
3.1 COLETA DE DADOS....21
3.2 TIPOS DE DADOS....22
3.2.1 QUALITATIVOS....22
3.2.2 QUANTITATIVOS....22
3.2.3 PRIMÁRIOS....22
3.2.4 SECUNDÁRIOS....23
3.3 ORGANIZAÇÃO E ARMAZENAMENTO....23
3.3.1 ESTRUTURADOS....23
3.3.2 NÃO ESTRUTURADOS....24
3.3.3 SEMIESTRUTURADOS....24
3.4 ESTRUTURA DE PÁGINAS E NAVEGADORES DE INTERNET....25
3.4.1 PADRÕES DE PÁGINAS DE INTERNET....30
3.5 WEB SCRAPING....31
3.5.1 BUSCANDO INFORMAÇÕES EM UM DOCUMENTO HTML....32
4 IMPLEMENTAÇÃO....34
4.1 CASOS DE USO....34

4.1.1 DIAGRAMA DO CASO DE USO SCRAP DRUGBANK.....34
4.1.2 DESCRIÇÃO DO CASO DE USO SCRAP DRUGBANK.....35
4.1.3 DIAGRAMA DO CASO DE USO SCRAP MERCKMILLIPORE.....37
4.1.4 DESCRIÇÃO DO CASO DE USO SCRAP MERCKMILLIPORE.....37
4.1.5 DIAGRAMA DO CASO DE USO SCRAP SCOPUS.....39
4.1.6 DESCRIÇÃO DO CASO DE USO SCRAP SCOPUS.....39
4.2 TECNOLOGIAS.....43
4.3 DIAGRAMA DE CLASSES.....45
4.3.1 DESCRIÇÃO DAS CLASSES.....45
4.4 FONTES.....47
4.4.1 DRUGBANK.....48
4.4.2 MERCK MILLIPORE.....49
4.4.3 SCOPUS.....50
5 TESTES.....53
5.1 DRUGBANK.....54
5.2 MERCK MILLIPORE.....55
5.3 SCOPUS.....56
6 CONCLUSÕES E TRABALHOS FUTUROS.....58
ANEXOS.....64
ANEXO A – CONTEÚDO DA REUNIÃO COM O ESPECIALISTA.....64
ANEXO B – DOCUMENTO DE ACEITAÇÃO ASSINADO PELO ESPECIALISTA 78

1 INTRODUÇÃO

A popularização da internet e o aumento da facilidade da publicação de conteúdos digitais (como, por exemplo, fóruns, *blogs*, *websites* e documentos virtuais) fizeram surgir um imenso acervo público, contendo as mais diversificadas informações sobre as mais variadas áreas existentes [54][55].

Enquanto o amplo acesso à informação é benéfico para o crescimento de qualquer segmento (contribuindo diretamente nos campos da pesquisa, do desenvolvimento e dos negócios, e apoiando à tomada de decisões de todo tipo), o excesso de informações pode se tornar um revés inferindo desafios para o consumo assertivo e otimizado de tanta informação indistinta disponível a todos [55].

A velocidade e a frequência com que informações são propagadas e atualizadas são extremamente altas quando comparadas a uma ou duas décadas passadas e podem significar um aumento da demanda do tempo de pesquisa quando o conhecimento sobre determinado assunto precisa ser mantido criticamente atualizado em relação a novas descobertas e avanços tecnológicos [56].

Um exemplo de obtenção rápida de dados precisos – como um processo crítico – pode ser observado na área da saúde, com ênfase na pesquisa e desenvolvimento de medicamentos, na qual o conhecimento exato e mais atualizado sobre certos dados específicos pode impactar diretamente na saúde de pacientes [57].

Ainda pensando nos desafios da pesquisa na área farmacêutica, a mineração (ou extração automática) de dados da internet, também conhecida como *web scraping*, parece ser uma solução para otimizar a busca e a extração de informações inerentes à formulação de fármacos e excipientes [61].

Este trabalho implementa um projeto de *software* que, a partir da definição prévia de fontes confiáveis para os dados de interesse, tem como objetivo extrair e organizar informações específicas, também previamente definidas, que apoiam, de maneira relevante, decisões tomadas por farmacêuticos. Dentre as contribuições deste projeto podemos destacar:

- Reunião com pesquisador da área de farmácia para definição dos requisitos e das atividades relacionadas ao desenvolvimento deste trabalho (e do sistema proposto);
- Definição e desenvolvimento de robôs de extração de dados para três fontes confiáveis para os dados de interesse: DrugBank, Merck Millipore e Scopus;
- Validação dos resultados obtidos junto a um pesquisador farmacêutico especialista.

Todo conteúdo definido e/ou desenvolvido relacionado a este trabalho, incluindo o código-fonte do sistema implementado, estão disponíveis em um repositório público no GitHub no seguinte endereço de internet: https://github.com/altobellibm/CEDERJ_2019_MIQUEIAS_DERNIER_E_VINICIUS_ZANOVELLI-

2 TRABALHOS RELACIONADOS

Com o objetivo de reunir, como *background*, conteúdo relevante a pesquisa proposta neste trabalho, foram analisadas e listadas abaixo publicações acadêmicas relacionadas ao domínio farmacêutico.

GLEZ-PEÑA *et al*. [1] descrevem a dependência de tecnologias, ferramentas e dados *online* da pesquisa biomédica em função do crescimento do volume de informação e da necessidade de otimizar os benefícios resultantes de novas pesquisas através da extração de conteúdos de interesse em um tempo razoavelmente baixo e sem consumir muitos recursos, identificando os elementos necessários para a aplicação das técnicas de *web scraping*, comparando diferentes *softwares* e *frameworks* disponíveis e voltados para este fim, e propondo um escopo para aplicações no domínio biomédico tendo em vista o extenso uso de APIs para disponibilizar informações nesta área. Dentre as diversas tecnologias consideradas, os autores citam a biblioteca jsoup [2] para a linguagem de programação Java [3] e os *frameworks* Scrapy [4] e Beautiful Soup [5], ambos para a linguagem Python [6].

FELDMAN *et al*. [7] apresentam uma metodologia para extração de dados relacionados a reações adversas a medicamentos (ADRs) de fóruns médicos da internet e demonstram a habilidade do método em prever reações não listadas em testes clínicos antes que elas sejam reportadas à agência federal Estadunidense *Food and Drug Administration* (FDA), responsável por assuntos relacionados a saúde pública no país. A metodologia se baseia na combinação do conceito *head-driven phrase structure grammar* (HPSG) com padrões relacionais conhecidos do domínio de interesse aplicados a grandes quantidades de textos avulsos pós-processando a informação obtida com algoritmos automatizados para refinar o resultado obtido. A implementação da solução proposta utiliza o *framework* Scrapy [4], a biblioteca (nativa) MD5 da linguagem Python [6], a plataforma NLTK [8], também para a linguagem Python, e o *framework* URE (descrito como uma ferramenta de aproximação entre o HPSG e o processo *Automatic acquisition of lexicon*).

LIU, LIANG e WISHART [9] oferecem uma segunda versão para um sistema *online* chamado PolySearch [10] que é apresentado como uma ferramenta de *text-mining* significantemente melhorada para a descoberta de associações entre doenças humanas, genes, medicamentos, toxinas e muitas outras entidades biomédicas através de consultas (*queries*) generalizadas do tipo “dada uma variável X, encontre todas as ocorrências associadas da variável Y” onde X e Y podem assumir qualquer uma das entidades do domínio suportadas pelo sistema, que foi desenvolvido utilizando Python e uma tecnologia de motor de busca chamada ElasticSearch [11]. A partir dos parâmetros informados, uma ampla busca na internet é realizada em múltiplas e variadas fontes como artigos acadêmicos e bancos de dados amplamente utilizados pelas comunidades biomédica e farmacêutica como DrugBank [12], UniProt [13] e Human Metabolome Database [14].

AMALIA, AFIFA e HERRIYANCE [15] abordam o hábito dos Indonésios de recorrer, como uma medida de primeiros socorros, a motores tradicionais de busca para encontrar informações sobre doenças e possíveis medicamentos ao passo que estes são por vezes incapazes ou ineficientes ao lidar com sinônimos populares para nomes científicos, além de retornar artigos não relacionados com o assunto desejado e propõem, como solução, a construção de um motor de busca baseado em *semantic web* [16] capaz de extrair vocabulário de *websites* populares que sabidamente ou potencialmente tenham o conteúdo desejado, criando um relacionamento entre termos científicos e populares, incluindo sinônimos para as doenças mais pesquisadas (o artigo foca em doenças tropicais, embora o contexto possa ser expandido). A solução utiliza técnicas de *web scraping* para obter dados de fontes como o banco de dados Drugs.com [17], o portal AloDokter [18] e o *website* da agência nacional de controle de alimentos e medicamentos da Indonésia [19] gerando, como resultado, serializações do modelo RDF [20] fazendo uso de *frameworks* do tipo CSV2RDF [21] para a linguagem Python.

GRÄSSER *et al*. [22] buscam conhecer a preferência e a experiência de usuários do domínio farmacêutico através da análise automatizada por *web scraping* de fóruns e *websites* que contenham opiniões e *reviews* destes usuários: primeiro uma análise de sentimentos [22, p.121] é feita e uma previsão é gerada quanto a satisfação geral, efeitos colaterais e efetividade de um determinado medicamento. Em seguida, é investigada a compatibilidade dos modelos de classificação obtidos atra-

vés da sua transferência entre diferentes domínios e fontes de dados, mostrando que esta abordagem pode ser utilizada para apontar, explorar e comparar similaridades e análises de sentimentos entre esses domínios. Os dados são obtidos de duas fontes independentes, distintas e amplamente utilizadas no campo farmacêutico por profissionais e consumidores – Drugs.com [17] e DrugLib.com [23] – através de um web *crawler* implementado na linguagem Python utilizando Beautiful Soup [5].

2.1 DISCUSSÃO

Como é possível observar no tópico anterior, não foram encontrados muitos artigos no campo farmacêutico, sendo a maior parte do conteúdo relacionado a *data-mining* e *web scraping* disponível na internet voltado para a generalização destas atividades (e suas metodologias) ou para aplicações em domínios próximos como o da biomedicina.

HAN *et al*. [24] trazem um estudo sobre a predição da desintegração de comprimidos com administração via oral utilizando técnicas de redes neurais, destacando a importância de dois fatores para o sucesso das predições: os dados avaliados e os algoritmos de processamento utilizados, percebendo a extração de dados como uma etapa crítica do processo e citando o uso da tradicional extração manual de dados em pesquisas atuais por cientistas farmacêuticos com experiência na área:

> "*In order to ensure a satisfied prediction accuracy, two key factors are to be considered: data and algorithm. The first issue is the reliable data in pharmaceutical research. Deep-learning attempts to learn these characteristics to make better representations and create models from reliable data. Thus, data extraction is a critical step. In current research, reliable formulation datasets were manually extracted and labeled from the research articles of Web of Science by experienced pharmaceutical scientists*." [24, p. 339]

Considerando os dados apresentados, é possível perceber que há espaço para soluções mais viáveis de coleta de dados no domínio farmacêutico através de uma metodologia simples, fundamentada na determinação de fontes confiáveis para os dados de interesse associada à aplicação das técnicas de *web scraping* para extrair, com velocidade e confiabilidade, as informações desejadas dos dados obtidos.

3 FUNDAMENTAÇÃO TEÓRICA

À medida que surgem novas tecnologias, são produzidos mais e mais dados. São aprimorados instrumentos de medição, que geram dados cada vez mais precisos e versam sobre temas cada vez mais amplos. Neste capítulo, serão abordadas as diversas categorias de dados, as suas formas de armazenamento, e apresentadas técnicas para a coleta e análise destes dados.

3.1 COLETA DE DADOS

A coleta de dados é a primeira etapa para a obtenção dos dados e informações sobre variáveis de interesse, de forma sistemática, que permite responder perguntas, testar hipóteses e avaliar os resultados [32]. Esta etapa da pesquisa é comum a todas as áreas de estudo, incluindo ciências físicas e sociais, humanas, negócios, entre outras. Embora os métodos variem por disciplina, a ênfase em garantir uma coleta precisa e honesta permanece a mesma.

Em muitos casos, a coleta de dados deve ser realizada diretamente da fonte primária, como por exemplo, o censo demográfico realizado pelo IBGE [34], que precisa coletar, ainda que por amostragem, os dados diretamente da população por meio de entrevistas que são realizadas diretamente com o objeto da pesquisa. Mas em outros casos, o trabalho de produção dos dados já foi realizado por fontes terceiras confiáveis e precisam apenas ser coletados para compilação e análise. A técnica a ser empregada na coleta de dados depende substancialmente da forma em que estes dados se apresentam.

3.2 TIPOS DE DADOS

Os dados são divididos em dois grandes grupos: quantitativos e qualitativos [33] e, quanto à origem dos dados, em primários e secundários [35].

3.2.1 QUALITATIVOS

São dados não numéricos, geralmente descritivos de características de um objeto de interesse [33]. São exemplos de dados qualitativos: o nome de uma pessoa, sexo, a classificação de uma espécie ou a cor de um produto [33].

3.2.2 QUANTITATIVOS

São dados numéricos que podem ser discretos, quando assumem valores enumeráveis, geralmente provenientes de contagem ou contínuos, quando assumem valores reais, geralmente provenientes de mensuração [33].

É de suma importância esta classificação, pois ela determina como estes dados podem ser obtidos, processados e finalmente apresentados. Exemplos de dados quantitativos são a idade de um aluno, a altura, o tamanho da população de uma espécie ou o peso de um produto [33].

3.2.3 PRIMÁRIOS

Dados que são coletados em primeira mão. Ainda não foram publicados e têm a vantagem de, por sua obtenção ter sido planejada especialmente para o estudo de interesse, serem mais confiáveis e adequados ao que se deseja deles. Têm a

desvantagem de, por vezes, o método de extração ser de difícil acesso e custoso. Servem como fontes de obtenção de dados primários: os experimentos que devem ser controlados de forma a eliminar fatores de influência externa e as entrevistas, normalmente utilizadas em pesquisas sociais [35].

3.2.4 SECUNDÁRIOS

Dados que são coletados a partir de medições e pesquisas já realizadas anteriormente. Tais dados nem sempre estão disponíveis com a integridade requerida, ou apresentados no formato mais adequado ao tipo de estudo que se deseja fazer deles. Por este motivo, ainda que de mais fácil obtenção, necessitam de igual atenção à sistemática de coleta, desde a definição de critérios para a seleção das fontes, até a metodologia empregada para a sua coleta e manejo [35].

3.3 ORGANIZAÇÃO E ARMAZENAMENTO

Os dados coletados precisam ser armazenados de forma a garantir acesso prático a eles. Os dados podem ser estruturados, não estruturados ou semiestruturados, e a escolha para cada tipo depende do problema e da necessidade do domínio onde se encontram os dados [52].

3.3.1 ESTRUTURADOS

São dados que possuem um formato rígido bem definido. Cada registro obedece a regras fixas onde existe um formato preestabelecido em relação a sua inclusão. São, por exemplo, as tabelas de banco de dados relacionais em que cada linha representa um registro, ou seja, uma pessoa, um objeto ou uma instância qual-

quer, e cada coluna possui registros específicos de dados. A mesma coisa ocorre em um formulário de cadastro que tenha campos fixados para preenchimento. Cada campo que deve ser preenchido foi predeterminado [35].

- Exemplo(s): bancos de dados relacionais, formulários de cadastro.
- Forma(s) de acesso: através de linguagens de consulta da tecnologia utilizada (na maioria dos casos, SQL para os bancos de dados relacionais), ou busca em ficheiros comumente organizados por ordem alfabética no caso de um formulário físico de cadastro.

3.3.2 NÃO ESTRUTURADOS

Não possuem estrutura fixa. Os principais exemplos são os arquivos de texto, que podem conter uma infinidade de informações, variando em idioma, tamanho, tema ou formatação [35].

- Exemplo(s): arquivo de texto.
- Forma(s) de acesso: interpretação para a busca da informação desejada.

3.3.3 SEMIESTRUTURADOS

Não têm uma predefinição com relação ao conteúdo, mas algumas regras que balizam o seu formato com a intenção de facilitar a organização e a apresentação das informações ali contidas. Ainda assim possuem alto grau de liberdade para a sua composição [52].

- Exemplo(s): XML, HTML, JSON.
- Forma(s) de acesso: através de programas de computador visualizadores que, pelo conhecimento prévio das regras que pautam a sua formatação, os apresentam de maneira legível e/ou amigável ao usuário, tal como fazem os *browsers* de internet com arquivos no formato HTML.

3.4 ESTRUTURA DE PÁGINAS E NAVEGADORES DE INTERNET

Explicar o surgimento e o funcionamento da internet não faz parte da proposta deste trabalho. Por outro lado, para entender as metodologias e técnicas de extração de dados é essencial elucidar dois de seus principais elementos, ambos tão antigos quanto a própria internet [25]: as páginas e os navegadores.

Todo *website*, independente do seu formato ou conteúdo (portal de notícias, aplicativo de mapas geográficos, rede social, motor de pesquisa, entre outros), é composto por uma ou mais páginas de internet organizadas e relacionadas entre si através de referências (*links*) que permitem uma navegação intuitiva por suas partes [60].

Cada uma dessas páginas de internet são arquivos comuns de computador essencialmente compostos de hipertexto [26] estruturado em blocos de elementos bem definidos principalmente por um par de *tags* que delimita o início e o fim de cada bloco, organizados a partir de duas divisões principais: um cabeçalho contendo metadados sobre o conteúdo da página e um corpo agrupando os diferentes elementos do conteúdo da página em si [60].

Segundo LONGMAN [26], o conceito de hipertexto existe desde a década de 1940 e começou a se tornar popular na década de 1980 com o surgimento dos computadores pessoais. Ao mesmo tempo, em meados dos anos 1980, é introduzido o sistema de nomes de domínios (DNS), que tem papel fundamental na forma com a qual acessamos páginas de internet até os dias de hoje.

Quando acessamos um endereço na internet através de um navegador (por exemplo: https://www.google.com), uma sequência bem definida de eventos ocorre de maneira transparente ao usuário, conforme pode ser observado na Figura

1: (i) o navegador de internet extrai, do endereço da página, o nome de seu hospedeiro; (ii) com o auxílio de um sistema de DNS, o navegador obtém o endereço IP do computador (servidor) que hospeda a página a ser acessada; (iii) através do endereço do hospedeiro, o navegador acessa (e baixa) a base do documento HTML e todos os arquivos nele referenciados; e (iv), por fim, uma representação visual do conteúdo HTML é exibida para o usuário.

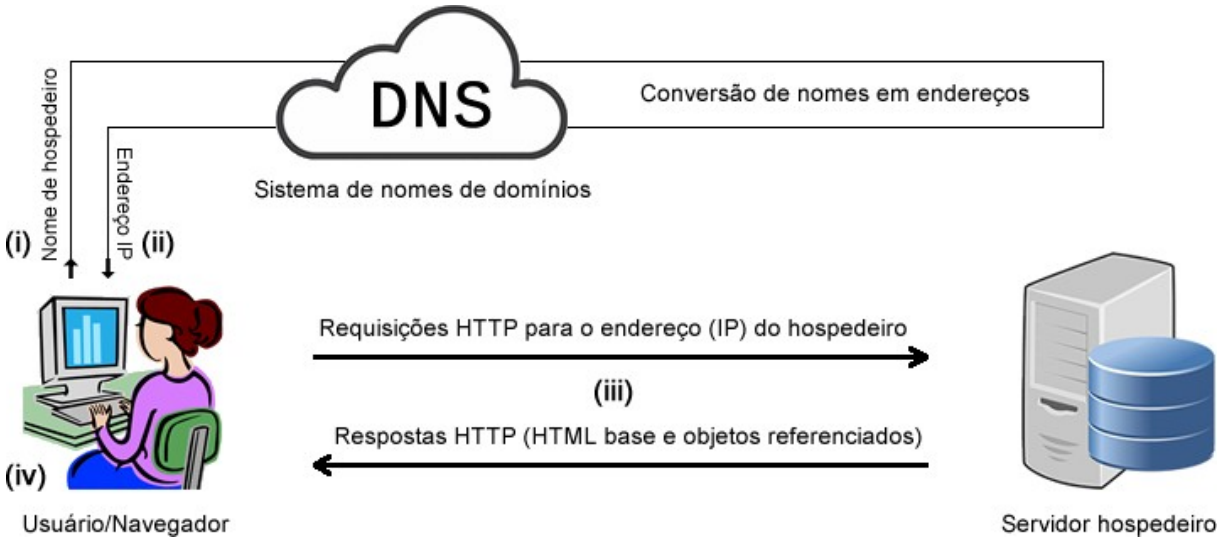

Figura 1: Visão geral do mecanismo de acesso às páginas de internet.

Embora formada por complexas redes de computadores interconectados através de diversos meios e protocolos físicos e lógicos, a internet foi inicialmente concebida simplesmente para compartilhar documentos entre universidades [26] e para que esses documentos pudessem oferecer uma estrutura clara, alguma formatação visual e, principalmente, vínculos diretos (*links*) entre partes de um mesmo documento ou entre diferentes documentos, o conceito de página de internet foi introduzido através da criação de uma linguagem de marcação de hipertexto chamada de HTML (*Hypertext Markup Language*) [27]. A Figura 2 mostra a estrutura básica de um documento HTML visualizado em uma IDE de programação.

```html
<html>
  <head>

    <!--
        CABEÇALHO DO DOCUMENTO:
        Metadados referentes ao documento como, por exemplo, referências a
        outros objetos (como folhas de estilos CSS) e informações como o título
        da página e parâmetros para navegadores de internet.
    -->

    <title>
        A tag "title" permite a declaração de um título (nome) para o documento
    </title>

  </head>
  <body>

    <!--
        CORPO DO DOCUMENTO:
        Conteúdo organizado através de elementos caracterizados por nomes
        e delimitados essencialmente por pares de tags (início e fim) que
        agrupam diferentes partes do documento em blocos bem definidos.
    -->

    <p>
        A tag "p", por exemplo, define um elemento do tipo "parágrafo" no HTML
    </p>

  </body>
</html>
```

Figura 2: Estrutura básica de um documento HTML em uma IDE de programação.

Uma das principais vantagens da utilização de hipertexto, além do mecanismo de referências (introduzido no HTML), é a possibilidade de mesclar textos com arquivos multimídia [28] (como imagens, por exemplo) produzindo um conteúdo não apenas mais rico, mas também potencialmente mais visual.

A partir da aceitação do HTML e com o início da popularização da internet, já no início da década de 1990 [29], começaram a surgir os navegadores de internet: programas de computador capazes de traduzir HTML em representações visuais (através, por exemplo, da renderização de imagens) tornando a visualização de documentos mais amigável para os usuários, conforme ilustrado na Figura 3.

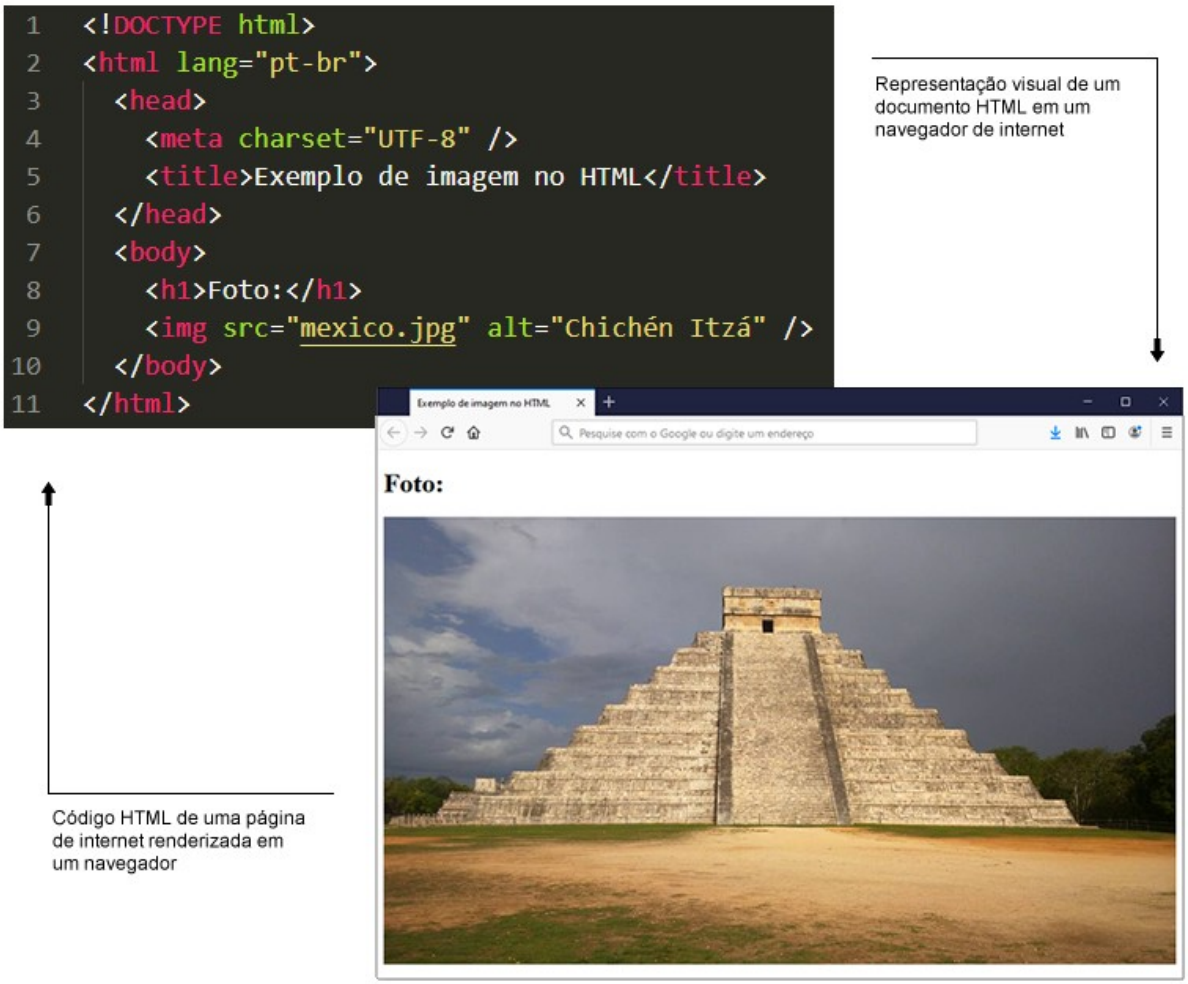

Figura 3: Representação visual do código HTML em um navegador de internet.

Com a introdução do *Document Object Model* (DOM) em 1998, que permite a manipulação de elementos HTML como uma coleção de objetos independentes, novas tecnologias surgiram adicionando bastante dinamismo às páginas de internet [30] que tiveram seu papel ampliado até se tornarem verdadeiras aplicações (RIA) [31] suportadas pelos avanços igualmente feitos nos navegadores de internet.

Dentre tais tecnologias então emergentes, destacam-se: (1) as folhas de estilos (CSS) [58], que possibilitam, por exemplo, classificar e estilizar grupos de elementos dentro de um mesmo documento HTML com muita flexibilidade; e (2) a linguagem de programação JavaScript [59], capaz de executar códigos de alto nível diretamente nos navegadores de internet permitindo, por exemplo, modificar o conteú-

do de uma página em tempo real (durante a navegação do usuário, reagindo a eventos como o clique de um botão).

A Figura 4 ilustra a composição de um documento HTML que faz uso dessas tecnologias citadas.

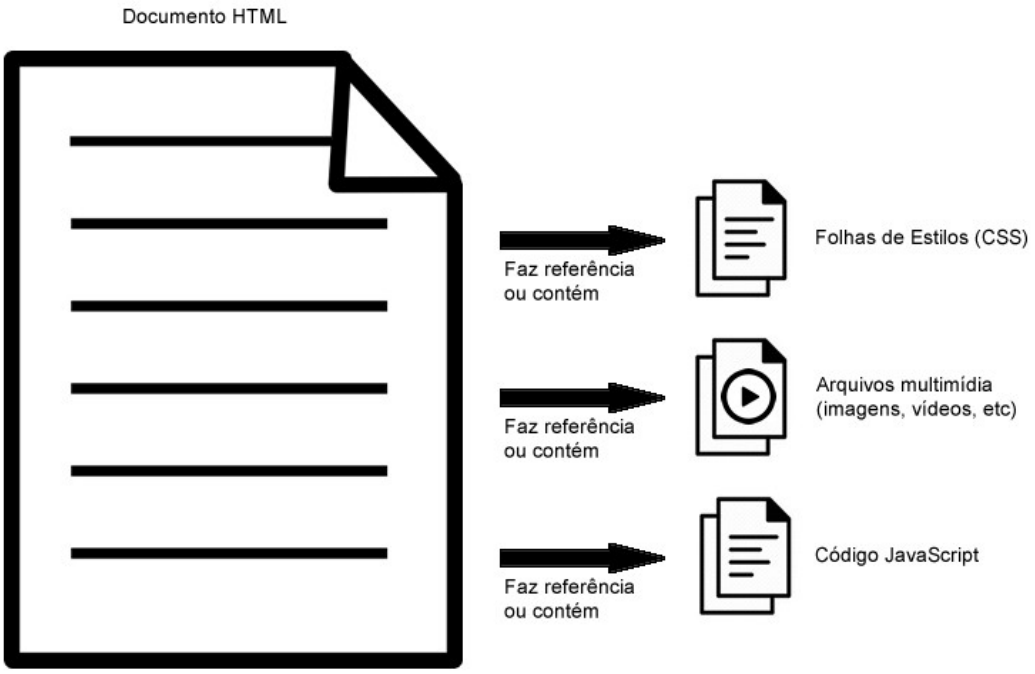

Figura 4: Estrutura de composição de um documento HTML contendo CSS e JS.

Embora a forma de acesso às páginas *web* mais conhecida e utilizada por usuários seja através dos navegadores, qualquer programa de computador pode agir como um cliente para servidores de páginas de internet através dos mesmos princípios pelos quais os navegadores são criados e, como vimos no Capítulo 2, as linguagens de programação de alto nível atualmente oferecem bibliotecas e *frameworks* que possibilitam implementações deste tipo de forma bastante direta e simplificada permitindo que o código HTML de praticamente qualquer página na internet (partindo do princípio de que se tenha acesso a ela) possa ser extraído com poucas linhas de código.

A partir destes conceitos, têm-se então não apenas um padrão sólido e consolidado para a estrutura de páginas na internet (e, consequentemente, uma de-

finição clara do papel dos navegadores), mas também um conjunto muito bem definido de elementos que representam a base de absolutamente todo conteúdo na internet fazendo com que a avaliação do conteúdo de uma “página de internet” (ou partes específicas de uma determinada página), uma vez que seu código hipertexto tenha sido obtido, seja possível e muito viável (como, por exemplo, dentre todo conteúdo de uma página, considerar apenas elementos do tipo “tabela” ou selecionar apenas elementos que contenham uma determinada classe CSS associada) sendo necessário observar apenas as possíveis alterações dinâmicas de conteúdo resultantes da execução de códigos JavaScript, quando houver.

3.4.1 PADRÕES DE PÁGINAS DE INTERNET

Dentre as muitas *tags* que hoje compõem o HTML, é possível destacar às específicas para divisões de áreas, parágrafos, listas, tabelas e formulários, além das estruturais, responsáveis por delimitar, no corpo do documento, seções para cabeçalho, rodapé, *menu* de navegação e conteúdo principal.

A partir da interpretação de um agrupamento de páginas de internet como um aplicativo *web*, temos o encontro do HTML (e tecnologias relacionadas) com os padrões de *software* e as boas práticas de desenvolvimento que, buscando motivar a criação de produtos estáveis e de fácil manutenção, permitem a esperança de que *websites* criados por equipes de profissionais buscarão respeitar as definições da linguagem permitindo diversos filtros e predições assertivas, desde que:

1. O código HTML da página seja minimamente válido;
2. Informações de um mesmo tipo, ainda que exibidas em páginas diferentes, sejam identificadas por uma mesma classe CSS ou elemento HTML (padronização);
3. A escolha de *tags* para determinada parte de um documento HTML seja lógica: embora, por exemplo, seja possível criar uma tabela sem o uso da *tag* “table”, a *tag* mais apropriada (neste caso, “table”) deverá ser eleita;

4. Por mais dinâmico que o conteúdo de uma página seja, com ou sem o uso de códigos JavaScript, a organização do código HTML (escolha de *tags*) tende a mudar com pouca frequência (a estrutura e o *layout* são pouco dinâmicos).

Partindo da ideia de que as afirmativas acima são verdadeiras para uma determinada página (ou *website*) eleita como fonte para extração de um determinado tipo ou conjunto de dados, podemos intuitivamente perceber que boa parte do trabalho de extração de dados fica simplificado pela possibilidade de se obter facilmente apenas as informações desejadas (relevantes) de qualquer documento HTML.

3.5 WEB SCRAPING

Dados são comumente publicados na *web* como documentos HTML. Tais documentos têm regras e definições próprias. Este formato tem o objetivo de permitir a apresentação do documento dentro de um *web browser*, como o Google Chrome [37], o Mozilla Firefox [38] ou o Microsoft Internet Explorer [39].

Alguém interessado em coletar dados publicados originalmente para serem apresentados em um *browser*, terá a tarefa inicial de converter estes dados em estruturas próprias adequadas ao seu estudo. Este processo pode ser feito manualmente ou automatizado por um programa de computador. Manualmente, utilizando um *web browser*, navega-se até a página pretendida; busca-se a informação visualmente e coleta-se o dado para futuro processamento. Automaticamente, o programa faz uma requisição HTTP e recebe um documento HTML; analisa o documento em busca da informação pretendida e retorna o trecho do documento que contém o dado.

3.5.1 BUSCANDO INFORMAÇÕES EM UM DOCUMENTO HTML

O documento HTML, é essencialmente um documento de texto e, portanto, técnicas de busca de uso geral em documentos de texto podem ser aplicadas. Dentre elas estão a busca por palavras-chave simples e a busca por padrões de texto mais complexos, como os padrões definidos com a linguagem de busca por padrões “regex” (*regular expression*) [40]. Há ainda as técnicas que utilizam a estrutura do documento para alcançar o dado pretendido. Exemplos de técnicas deste tipo são a linguagem XPath [41], que define um caminho dentro do documento até a informação, e a utilização de ferramentas que realizam o *parsing* do documento, criando uma representação de objetos encadeados, como a biblioteca Beautiful Soup [5].

Figura 5: Documento HTML representado como texto e como árvore de objetos

O *parsing* de um documento HTML é a análise sintática do texto. Nesta análise é identificada a estrutura da página e verificado se está de acordo com as regras de composição de um arquivo HTML. É uma etapa importante para a tradução do arquivo-texto em uma estrutura que pode ser compreendida e manipulada pelo computador. No exemplo da Figura 5, o título da página pode ser extraído seguindo-se o caminho: html; head; title. A Figura 6 imprime a página em formato gráfico exibida por um *web browser.*

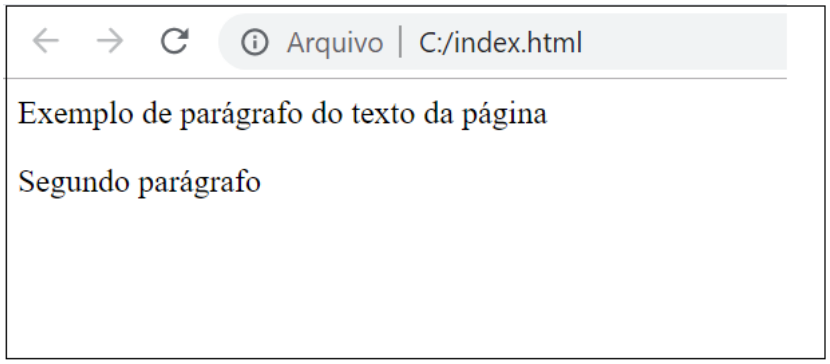

Figura 6: Visualização de um documento HTML.

4 IMPLEMENTAÇÃO

Com base nas definições feitas em reunião junto a um pesquisador farmacêutico especialista no domínio de interesse deste trabalho (conforme detalhado no Anexo A), foi possível identificar todas as demandas necessárias para a implementação de uma ferramenta de *web scraping* que tem como objetivo automatizar a busca e a coleta de dados específicos sobre as propriedades e os excipientes de determinados fármacos e suas respectivas composições.

Neste capítulo abordaremos os detalhes técnicos desta implementação: apresentaremos os casos de uso disponíveis, falaremos sobre as tecnologias eleitas, forneceremos uma visão geral do sistema através de um diagrama de classes e definiremos as fontes a serem utilizadas pela ferramenta desenvolvida, conforme a orientação do especialista farmacêutico consultado.

4.1 CASOS DE USO

4.1.1 DIAGRAMA DO CASO DE USO SCRAP DRUGBANK

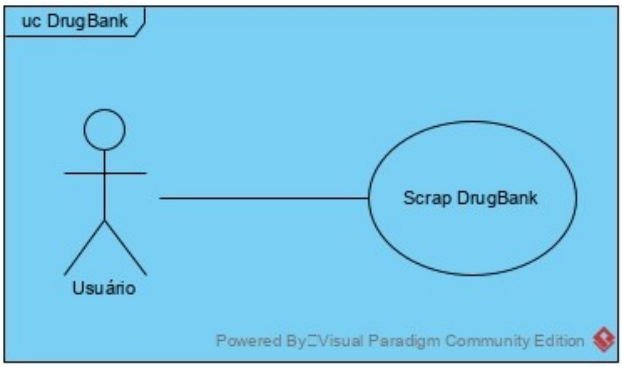

Figura 7: Diagrama do caso de uso *Scrap* DrugBank.

4.1.2 DESCRIÇÃO DO CASO DE USO SCRAP DRUGBANK

Tabela 1: Descrição do caso de uso *Scrap* DrugBank.

ID:	**UC01**
Objetivo:	Extrair dados de interesse da fonte “DrugBank”.
Requisitos:	Estar conectado à internet.
Atores:	Usuário.
Pré-condições:	Os pacotes, linguagens, *frameworks* e bibliotecas usados pela ferramenta precisam estar instalados no ambiente. Os dados de interesse precisam estar definidos no arquivo (JSON) de configurações/entradas da fonte.
Pós-condições:	As informações obtidas deverão estar salvas em arquivo(s) no disco.
Fluxo Principal:	***Scrap* DrugBank** 1. O ator executa a ferramenta, através de linha de comando, com os parâmetros que definem a fonte “DrugBank”; 2. A ferramenta busca os dados de interesse (definidos em um arquivo configurável do tipo JSON); 3. A ferramenta gera arquivo(s) de saída (conforme os parâmetros informados) no disco com as informações encontradas.
Fluxos Alternativos:	Não há.
Erros/Exceções:	O tratamento de erros (como requisições inválidas e *timeouts*) é feito pelo *framework* utilizado (mensagens de erro

	podem ser exibidas para o usuário). Em caso de falha(s) na execução, as pós-condições não serão verdadeiras (os dados de interesse não serão obtidos).
Mensagens:	*Logs* de saída do *framework* utilizado.
Regras de negócio:	**[RN01]** O endereço de internet “https://www.drugbank.ca/unearth/q" deve aceitar requisições do tipo GET com os parâmetros “utf8”, “searcher” e “query” e os respectivos valores “✓”, “drugs” e “{variável_de_interesse}” (este último contendo a variável desejada) e deve ser utilizado para buscar os dados de interesse. **[RN02]** O resultado da requisição anterior (**RN01**) deverá ser uma página HTML válida contendo um elemento do tipo “h1” e outro do tipo “table” com ID “drug-moldb-properties”, de onde os dados de interesse devem ser extraídos. **[RN03]** O resultado da requisição feita pela **RN01** poderá ser uma página HTML válida contendo um elemento “a” dentro de um elemento “h2” que, por sua vez, deverá estar dentro de um elemento “div” com a classe CSS “search-result” atribuída. Neste caso, uma nova requisição do tipo GET deverá ser feita ao endereço contido no atributo “href” da referida *tag* “a” e o resultado obtido desta nova requisição deverá ser o descrito na **RN02**.

4.1.3 DIAGRAMA DO CASO DE USO SCRAP MERCKMILLIPORE

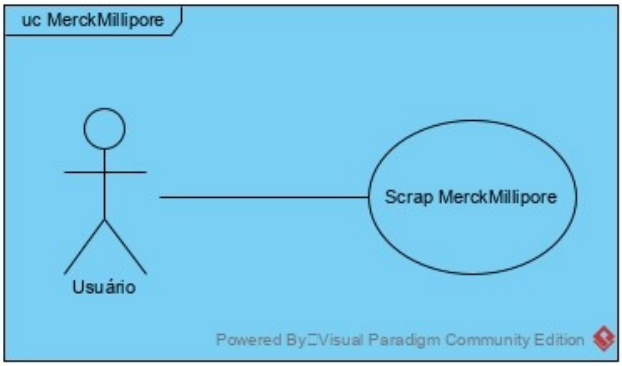

Figura 8: Diagrama do caso de uso *Scrap* MerckMillipore.

4.1.4 DESCRIÇÃO DO CASO DE USO SCRAP MERCKMILLIPORE

Tabela 2: Descrição do caso de uso *Scrap* MerckMillipore.

ID:	**UC02**
Objetivo:	Extrair dados de interesse da fonte “Merck Millipore”.
Requisitos:	Estar conectado à internet.
Atores:	Usuário.
Pré-condições:	Os pacotes, linguagens, *frameworks* e bibliotecas usados pela ferramenta precisam estar instalados no ambiente. Os dados de interesse precisam estar definidos no arquivo (JSON) de configurações/entradas da fonte.
Pós-condi-ções:	As informações obtidas deverão estar salvas em arquivo(s) no disco.
Fluxo Princi-pal:	***Scrap* MerckMillipore**

	1. O ator executa a ferramenta, através de linha de comando, com os parâmetros que definem a fonte “Merck Millipore”; 2. A ferramenta busca os dados de interesse (definidos em um arquivo configurável do tipo JSON); 3. A ferramenta gera arquivo(s) de saída (conforme os parâmetros informados) no disco com as informações encontradas.
Fluxos Alternativos:	Não há.
Erros/Exceções:	O tratamento de erros (como requisições inválidas e *timeouts*) é feito pelo *framework* utilizado (mensagens de erro podem ser exibidas para o usuário). Em caso de falha(s) na execução, as pós-condições não serão verdadeiras (os dados de interesse não serão obtidos).
Mensagens:	*Logs* de saída do *framework* utilizado.
Regras de negócio:	**[RN04]** O endereço dinâmico de internet “http://www.merckmillipore.com/BR/pt/search/{variável_de_interesse}” (onde “{variável_de_interesse}” deve conter a variável desejada) deve aceitar requisições do tipo GET e deve ser utilizado para buscar as informações desejadas. **[RN05]** O resultado da requisição anterior (**RN04**) deverá ser uma página HTML válida contendo um elemento do tipo “h1” que, por sua vez, deve conter um elemento do tipo “span” e, também, um elemento do tipo “table” com a classe CSS “attribute-group-table” atribuída, de onde os dados de interesse devem ser extraídos. **[RN06]** O resultado da requisição feita pela **RN04** poderá ser

	uma página HTML válida contendo um elemento “a” dentro de um elemento “h2” que, por sua vez, deverá estar dentro de um elemento “div” com a classe CSS “container-serp” atribuída. Neste caso, uma nova requisição do tipo GET deverá ser feita ao endereço contido no atributo “href” da referida *tag* “a” e o resultado obtido desta nova requisição deverá ser o descrito na **RN05**.

4.1.5 DIAGRAMA DO CASO DE USO SCRAP SCOPUS

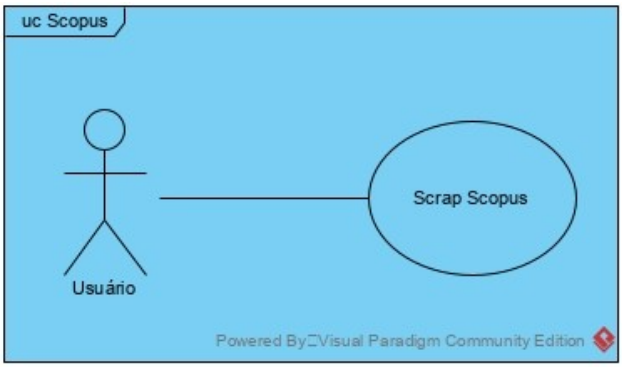

Figura 9: Diagrama do caso de uso *Scrap* Scopus.

4.1.6 DESCRIÇÃO DO CASO DE USO SCRAP SCOPUS

Tabela 3: Descrição do caso de uso *Scrap* Scopus.

ID:	**UC03**
Objetivo:	Extrair dados de interesse da fonte “Scopus”.
Requisitos:	Estar conectado à internet na rede da Universidade Federal Fluminense (ou em outra rede que forneça acesso ao site scopus.com e à sua API).

	Ter uma chave de acesso válida à API do “Scopus”.
Atores:	Usuário.
Pré-condições:	Os pacotes, linguagens, *frameworks* e bibliotecas usados pela ferramenta precisam estar instalados no ambiente. A busca a ser realizada precisa estar definida no arquivo (JSON) de configurações/entradas da fonte. A chave de acesso à API do “Scopus” precisa estar definida no arquivo (JSON) de configurações/entradas da fonte.
Pós-condições:	As informações obtidas deverão estar salvas em arquivo(s) no disco.
Fluxo Principal:	***Scrap* Scopus** 1. O ator executa a ferramenta, através de linha de comando, com os parâmetros que definem a fonte “Scopus”; 2. A ferramenta busca os dados de interesse (definidos em um arquivo configurável do tipo JSON); 3. A ferramenta gera arquivo(s) de saída (conforme os parâmetros informados) no disco com as informações encontradas.
Fluxos Alternativos:	Não há.
Erros/Exceções:	O tratamento de erros (como requisições inválidas e *time-outs*) é feito pelo *framework* utilizado (mensagens de erro podem ser exibidas para o usuário). Em caso de falha(s) na execução, as pós-condições não serão verdadeiras (os dados de interesse não serão obti-

	dos).
Mensagens:	*Logs* de saída do *framework* utilizado.
Regras de negócio:	**[RN07]** Apenas artigos publicados nos últimos 10 (dez) anos devem ser considerados. **[RN08]** As palavras-chave buscadas devem aparecer no título e/ou no *abstract* do artigo para que o mesmo seja considerado. **[RN09]** Cada artigo deve ter uma fonte associada e cada fonte deve ter uma pontuação (*score*) que determina seu fator de impacto – apenas fontes que tenham a média dos últimos 3 (três) anos da sua respectiva pontuação maior que 1 (um) devem ser consideradas. **[RN10]** Cada artigo deve ter um código DOI único associado – este código deve ser utilizado para evitar a coleta de artigos duplicados. **[RN11]** O endereço de internet “https://api.elsevier.com/content/search/scopus" deve aceitar requisições do tipo GET com os parâmetros “query” e “count” com os valores das variáveis “termo buscado” (ou consulta – *query* – definida conforme os padrões da fonte “Scopus”) e “número desejado de resultados”, respectivamente, e deve ser utilizado para buscar os dados de interesse. **[RN12]** A chave de acesso à API do “Scopus” deve estar presente no cabeçalho das requisições de busca (**RN11**), no campo de nome “X-ELS-APIKey”. **[RN13]** O número de resultados obtidos pode ser limitado

	pelo usuário através das definições no arquivo de configurações/entradas da fonte. **[RN14]** Os resultados das requisições de busca (**RN11**) devem estar no formato JSON e os registros encontrados devem estar na chave de nome “search-results” e conter, para cada artigo retornado pela busca, os campos “source-id”, “prism:doi” e “prism:url” com o ID da fonte, o código DOI e o endereço de internet associados, respectivamente. **[RN15]** Os endereços de internet referentes aos artigos encontrados (obtidos pelas requisições definidas na **RN11** e conforme descrito na **RN14**) devem aceitar requisições do tipo GET com os parâmetros “httpAccept” e “fields” com os respectivos valores “application/json” e “description” e devem ser utilizados para coleta dos dados de interesse. **[RN16]** Os resultados das requisições de acesso aos artigos (**RN15**) devem estar no formato JSON contendo, dentro da chave aninhada “abstracts-retrieval-response → item → bibrecord → head”, o *abstract* e o endereço da fonte do artigo nas chaves “abstracts” e “source → website”, respectivamente. **[RN17]** O endereço dinâmico de internet “https://doi.org/{doi}” (onde “{doi}” deve conter a código DOI único de cada artigo obtido pelas requisições definidas na **RN11** e conforme descrito na **RN14**) deve aceitar requisições do tipo GET e deve ser utilizado para verificar a validade de cada artigo e, também, evitar coletas duplicadas. **[RN18]** O endereço dinâmico de internet “https://www.sco-

	pus.com/source/citescore/{id_da_fonte}.uri” (onde “{id_da_fonte}” deve conter o valor do campo “source-id” obtido pelas requisições definidas na **RN11** e conforme descrito na **RN14**) deve aceitar requisições do tipo GET e deve ser utilizado para obter os dados de pontuação (*score*) das fontes dos artigos encontrados. **[RN19]** Os resultados das requisições definidas na **RN18** devem estar no formato JSON e conter o campo “rp”, definido na chave aninhada “yearInfo → {ano} → metricType”, onde {ano} assume o valor do ano desejado. **[RN20]** As requisições definidas na **RN19** só devem ser consideradas caso a resposta contenha o campo “documentType” com o valor “all”.

4.2 TECNOLOGIAS

A biblioteca Python Scrapy [4] foi construída com o objetivo de ser um *framework* completo para a tarefa de extrair dados de *websites* a partir do documento HTML. Para isto, ela gerencia as requisições HTTP ao *website*, expõe uma ferramenta para navegação pelo conteúdo do documento HTML via XPath [41] ou seletores CSS [49], e ainda, exporta os dados coletados para arquivos com diferentes opções de formato: CSV, JSON [45], XML [50].

A Figura 10 representa um diagrama de atividades para uma implementação mínima, mas ainda extremamente flexível de um programa baseado no Scrapy:

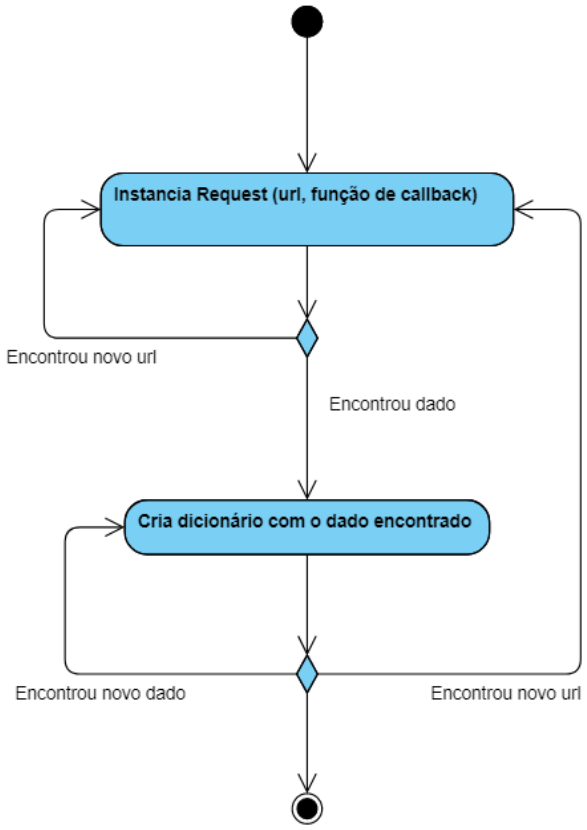

Figura 10: Diagrama de atividades de um programa Scrapy mínimo.

Esta implementação consiste em estender a classe Spider e sobrescrever o método start_request. O método start_request deve realizar criações sucessivas de instâncias da classe Request. A requisição para o endereço indicado na criação do objeto é gerenciada pelo *framework*. O *framework* decide, também, o momento adequado de realizar a requisição e se será necessário tentar novamente em caso de falha. Após uma requisição bem-sucedida, a função que é passada como parâmetro na criação do objeto de Request é chamada (*callback*) e é responsável por realizar a procura desejada dentro do documento. A função deve criar, então, mais instâncias da classe Request ou objetos do tipo dicionário, que serão tratados pelo *framework* como um item do resultado final da coleta de dados.

4.3 DIAGRAMA DE CLASSES

O diagrama de classes do sistema (Figura 11) representa a estrutura geral, a organização e os relacionamentos entre as classes da ferramenta implementada com base no *framework* Scrapy [4].

Recorrendo aos conceitos de herança e sobreposição de métodos do paradigma de programação orientada a objetos (POO), as classes do sistema estendem determinadas classes generalizadas do *framework* utilizado obtendo assim um mecanismo de execução das tarefas relacionadas a atividade de *web scraping*, e especializam, também, uma classe específica responsável pela lógica individual de cada fonte de dados (se beneficiando dos recursos oferecidos pelo *framework* e permitindo fácil expansão através da inclusão de novas fontes de dados, se necessário).

4.3.1 DESCRIÇÃO DAS CLASSES

- **JSONExporter –** Esta classe é responsável pela escrita no disco do(s) arquivo(s) de saída no formato JSON, utilizando a codificação UTF-8, com os dados obtidos de cada fonte.
- **CSVExporter –** Com o auxílio da biblioteca Pandas [42], esta classe realiza a escrita no disco do(s) arquivo(s) de saída no formato CSV, utilizando a codificação ANSI, com os dados obtidos de cada fonte.
- **JSONExportPipeline –** Respeitando a estrutura do *framework* utilizado, esta classe implementa um conjunto pré-definido de métodos de auxílio ao processo de exportação de dados [43]. No sistema proposto, esta classe apoia a geração de arquivos no formato JSON.
- **CSVExportPipeline –** Esta classe sobrescreve um pequeno conjunto de propriedades e métodos da classe JSONExportPipeline para apoiar a exportação de dados no formato CSV.

- **BaseSpider –** Esta é uma classe intermediária que estende a classe especial “Spider” do *framework* utilizado (responsável pelas atividades específicas de extração de dados), agregando as propriedades comuns às outras classes do sistema deste mesmo tipo (descritas a seguir) e garantindo que a classe genérica (provida pelo *framework*) seja inicializada corretamente com os parâmetros necessários. Idealmente, esta classe deveria ser abstrata; em função das particularidades das tecnologias eleitas (e para simplificar a implementação), trata-se de uma classe concreta que não deve ser instanciada ou utilizada diretamente, mas especializada para sobrepor seus principais métodos [44] com a lógica relacionada a cada diferente fonte de dados.
- **DrugBankSpider –** Esta classe é responsável pela extração de dados da fonte “DrugBank”, contendo toda lógica relacionada às regras de negócio desta fonte. A classe realiza a busca pelos dados de interesse através de requisições do tipo GET ao *website* da fonte em questão e extrai as informações relevantes encontradas com base na estrutura HTML já conhecida da página acessada pelo sistema.
- **MerckSpider –** Análoga a classe DrugBankSpider, a classe MerckSpider desempenha o mesmo papel descrito acima, porém para a fonte “Merck Millipore”: requisições do tipo GET são feitas à página de busca do *website* desta fonte e as variáveis de interesse encontradas são extraídas do conteúdo da página HTML retornada com estrutura também já conhecida.
- **ScopusSpider –** A classe ScopusSpider desempenha o mesmo papel das duas últimas classes descritas acima, porém para a fonte “Scopus”. Seu funcionamento difere das duas classes anteriores na forma com que os dados são obtidos: as requisições são feitas a uma API que disponibiliza os dados no formato JSON, com estrutura conhecida, de onde as informações são extraídas; a classe também faz requisições ao *website* da fonte para atender as regras de negócio referentes a pontuação (*score*) da origem de cada artigo encontrado e consulta o *website* https://doi.org para validar os artigos considerados relevantes pelo sistema e evitar a coleta de artigos duplicados.

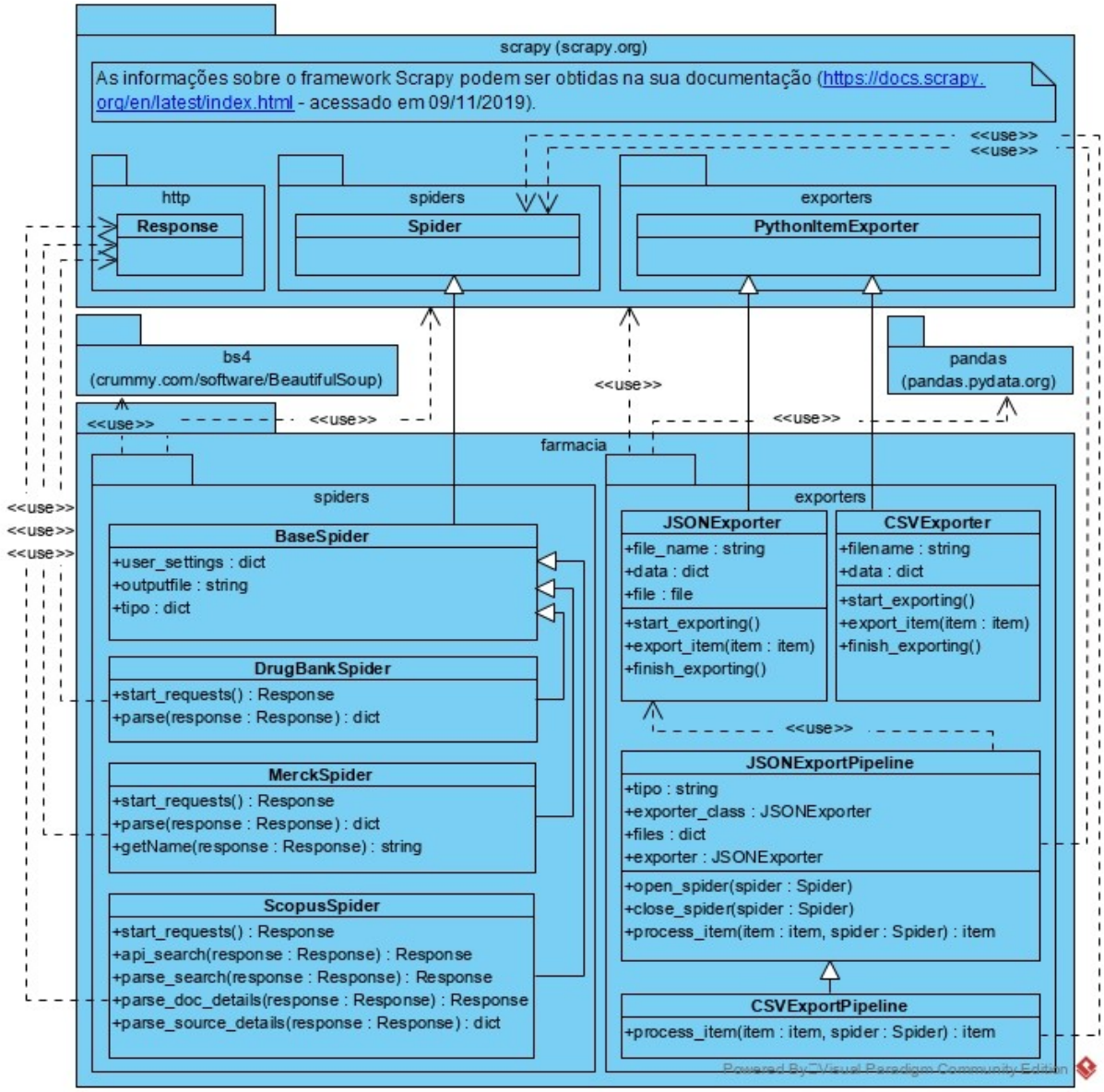

Figura 11: Diagrama de Classes do sistema.

4.4 FONTES

Esta seção detalha, para cada fonte utilizada, a sequência de atividades desenvolvidas para alcançar os objetivos de extração de dados.

4.4.1 DRUGBANK

O processamento é iniciado com a busca pelo nome do fármaco. O primeiro objeto do tipo Request é criado para o endereço https://www.drugbank.ca/unearth/q?utf8=%E2%9C%93&**query**=nome_do_farmaco&searcher=drugs, onde o parâmetro **query** indica qual é o fármaco a ser pesquisado. Esta requisição pode retornar uma lista de resultados, um resultado único ou nenhum resultado.

Caso o retorno seja uma lista de resultados, o primeiro item da lista é seguido, abrindo uma página de detalhes para o item. Caso seja um resultado único, a lista de resultados não é apresentada e o *website* redireciona automaticamente o acesso para a página de detalhes do único fármaco encontrado. A Figura 12 mostra, como exemplo, a página de detalhes resultado da busca para o fármaco Tylenol.

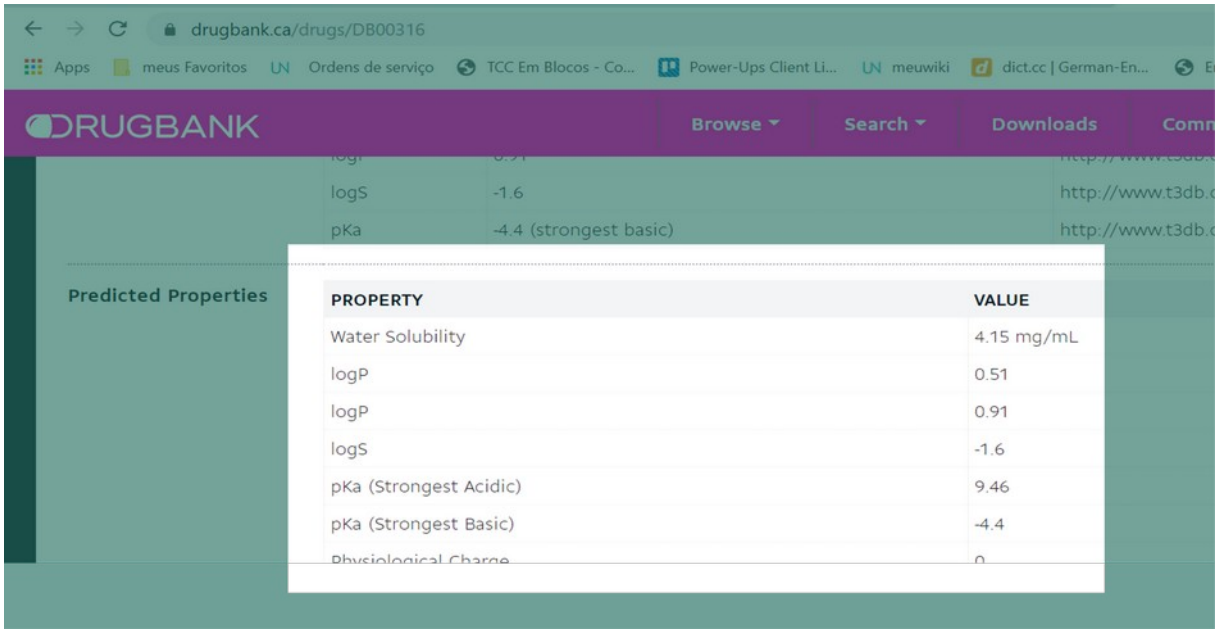

Figura 12: Página de detalhes do fármaco Tylenol no DrugBank.

A tabela em evidência é identificada pelo id da *tag* HTML igual a "drugmoldb-properties". A primeira célula de cada linha da tabela indica o nome da propriedade, e a segunda célula indica o valor.

4.4.2 MERCK MILLIPORE

O processamento é iniciado com a busca pelo nome do fármaco. O primeiro objeto do tipo Request é criado para o endereço http://www.merckmillipore.-com/BR/pt/search/nome_da_substancia, onde o termo **nome_da_substancia** indica qual é o fármaco a ser pesquisado. Esta requisição pode retornar uma lista de resultados, um resultado único ou nenhum resultado.

Caso o retorno seja uma lista de resultados, o primeiro item da lista é seguido, abrindo uma página de detalhes para o item. Caso seja um resultado único, a lista de resultados não é apresentada e o *website* redireciona automaticamente o acesso para a página de detalhes do fármaco. A Figura 13 mostra, como exemplo, a página de detalhes do resultado da busca para a substância Lactose Monohidratada.

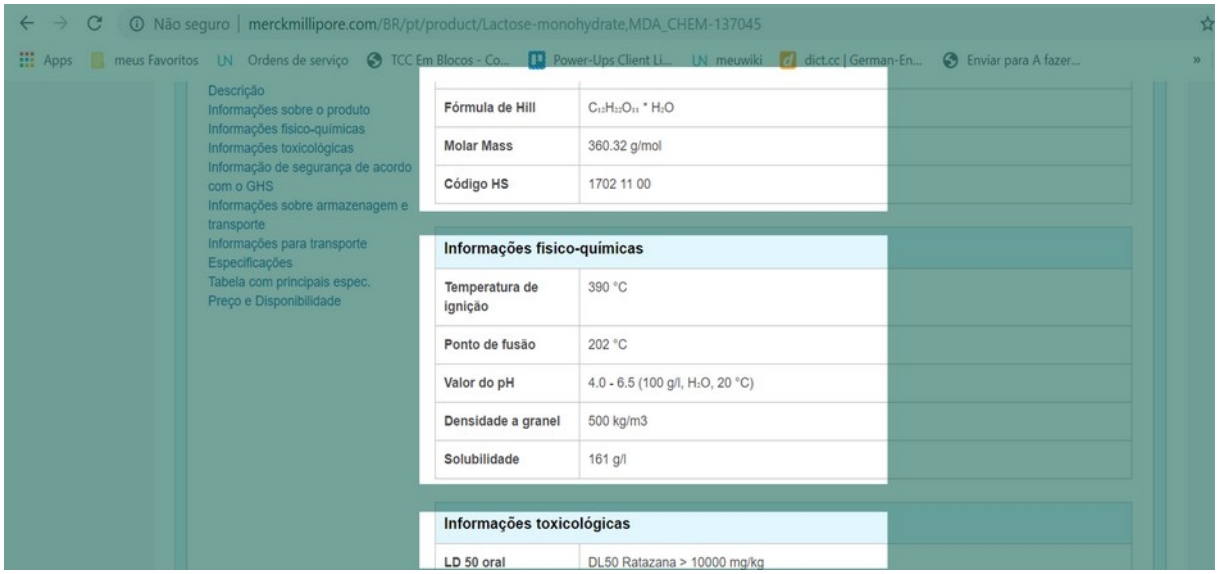

Figura 13: Página de detalhes do fármaco Lactose Monohidratada no Merck.

O grupo de tabelas em evidência é selecionado como todos os elementos de *tag* “table” que contém a classe “attribute-group-table”. A primeira coluna de cada linha indica o nome da propriedade, e a segunda o valor.

4.4.3 SCOPUS

O Scopus disponibiliza uma API [51] para facilitar o trabalho de busca de dados em seus serviços, que retorna um documento em formato JSON. Apesar de o *framework* Scrapy fornecer ferramentas específicas para trabalhar apenas com documentos HTML, esta deficiência foi superada utilizando a biblioteca “json” [53], nativa da linguagem Python, para navegação pelo documento retornado. Esta biblioteca transforma o texto em formato JSON em um objeto ou lista de objetos e, desta forma, permite acesso às informações diretamente através de suas propriedades.

Seguindo as instruções da documentação da API do Scopus, utilizamos a URL **https://api.elsevier.com/content/search/scopus?query=QUERY_STRING**, onde o termo **QUERY_STRING** define os parâmetros de busca. Para que a requisição seja processada pelo Scopus, o *header* da requisição deve conter a chave X-ELS-APIKey com um valor de chave de API habilitada para utilizar o serviço. Obtivemos uma chave de API do Scopus após realizar um cadastro no próprio *website* desta fonte. O Scopus restringe, ainda, o acesso ao serviço somente a endereços IP previamente habilitados, como é o caso da rede interna da Universidade Federal Fluminense, utilizada neste projeto para acessar este serviço. O termo de busca fica definido pelo usuário no momento da execução. O Gráfico 1 mostra um índice de chaves do arquivo JSON retornado pela API.

```
search-results
search-results -> opensearch:totalResults
search-results -> opensearch:startIndex
search-results -> opensearch:itemsPerPage
search-results -> opensearch:Query
search-results -> opensearch:Query -> @role
search-results -> opensearch:Query -> @searchTerms
search-results -> opensearch:Query -> @startPage
search-results -> link
search-results -> link -> [] -> @_fa
search-results -> link -> [] -> @ref
search-results -> link -> [] -> @href
search-results -> link -> [] -> @type
search-results -> entry
search-results -> entry -> [] -> @_fa
search-results -> entry -> [] -> link
search-results -> entry -> [] -> link -> [] -> @_fa
search-results -> entry -> [] -> link -> [] -> @ref
search-results -> entry -> [] -> link -> [] -> @href
search-results -> entry -> [] -> prism:url
search-results -> entry -> [] -> dc:identifier
search-results -> entry -> [] -> eid
search-results -> entry -> [] -> dc:title
search-results -> entry -> [] -> dc:creator
search-results -> entry -> [] -> prism:publicationName
search-results -> entry -> [] -> prism:issn
search-results -> entry -> [] -> prism:volume
search-results -> entry -> [] -> prism:pageRange
search-results -> entry -> [] -> prism:coverDate
search-results -> entry -> [] -> prism:coverDisplayDate
search-results -> entry -> [] -> prism:doi
search-results -> entry -> [] -> pii
search-results -> entry -> [] -> citedby-count
search-results -> entry -> [] -> affiliation
search-results -> entry -> [] -> affiliation -> [] -> @_fa
search-results -> entry -> [] -> affiliation -> [] -> affilname
search-results -> entry -> [] -> affiliation -> [] -> affiliation-
city
search-results -> entry -> [] -> affiliation -> [] -> affiliation-
country
search-results -> entry -> [] -> prism:aggregationType
search-results -> entry -> [] -> subtype
search-results -> entry -> [] -> subtypeDescription
search-results -> entry -> [] -> article-number
search-results -> entry -> [] -> source-id
search-results -> entry -> [] -> openaccess
search-results -> entry -> [] -> openaccessFlag
```

Gráfico 1: Índice de chaves JSON retornadas pela API de busca do Scopus.

A lista de artigos desejados é encontrada seguindo as chaves “search-results” e “entry”. Além das propriedades já retornadas pela API para cada artigo, adicionamos: (i) o texto completo do *abstract;* (ii) um link para o artigo gerado a partir do DOI [36] do mesmo; e (iii) a pontuação dos últimos três anos dada pelo Scopus para a publicação. O Gráfico 2 resume as etapas do processamento realizado para a fonte Scopus.

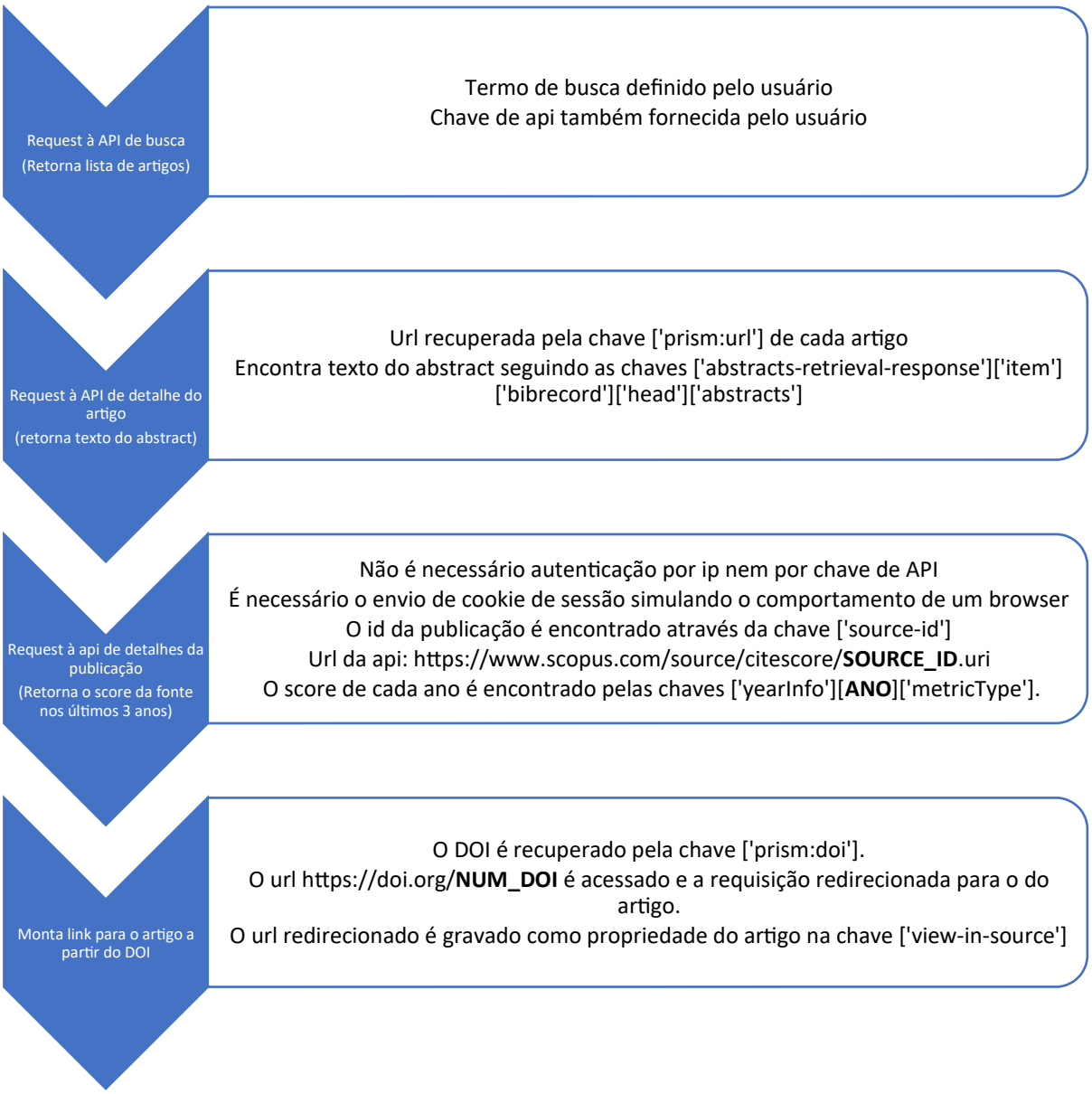

Gráfico 2: Etapas para o processamento da fonte Scopus.

5 TESTES

Neste capítulo será descrito em detalhes a execução de um caso concreto para cada um dos *scrapers* desenvolvidos. Foram avaliados o tempo de execução e a ocorrência de eventuais falhas.

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

Tabela 4: Configurações do sistema utilizado nos testes.

Nome do sistema operacional	Microsoft Windows 8.1 Enterprise
Tipo de sistema	x64-based PC
Processador(es)	[01]: Intel64 Family 6 Model 60 Stepping 3 GenuineIntel ~1900 Mhz
Memória física total	16.213 MB
Versão do Python instalada	Python 3.6.8 :: Anaconda 4.4.0 (64-bit)

Para cada fonte, foi criado um programa com objetivo específico de navegar pela estrutura do *website* (ou documento retornado) até a obtenção dos dados de interesse. Para isto, o usuário especialista deve fornecer uma série de parâmetros encapsulados em um arquivo do tipo JSON que é fornecido como entrada para o programa. O usuário, na chamada do programa, define ainda o tipo de arquivo de saída (CSV e/ou JSON) e o caminho do arquivo a ser gerado contendo o resultado do processamento.

A medição do tempo de execução de cada programa foi realizada através um arquivo de *script* do Windows, demonstrado na Figura 14.

```
1 set startTime=%time%
2 python farmacia {drugbank,merck,scopus} configfile [--tipo {csv,json}] [--saida SAIDA]
3 echo Start Time: %startTime%
4 echo Finish Time: %time%
```

Figura 14: Exemplo de *script* para execução dos testes.

O *script* registra os tempos antes e após a execução do programa. O comando genérico exibido na linha 2 da Figura 14, é composto pela chamada do programa; por um parâmetro para especificar qual(is) fonte(s) será(ão) executada(s) – DrugBank [12], Merck Millipore [46] ou Scopus [47]; pelo nome do arquivo de configuração que define os termos de busca e saída de acordo com a especificação de cada fonte; pelo formato de saída; e pelo nome do arquivo de saída.

Todos os resultados foram submetidos à análise da equipe de especialistas que deu o seu parecer, disponível no Anexo B deste trabalho.

5.1 DRUGBANK

O objetivo do programa referente ao *website* DrugBank [12] é encontrar uma lista de fármacos pelo nome. Encontrada a página específica de cada fármaco, buscar por uma lista de propriedades. Ambas as entradas, lista de fármacos e de propriedades, são de responsabilidade do usuário especialista. O usuário edita um arquivo no formato JSON [45] seguindo o modelo demonstrado na Figura 15 e o referencia como entrada para o programa.

```
1  {
2      "inputs": [
3          "Physostigmine glycolate",
4          "Famotidine",
5          "Metronidazole",
6          "Prednisolone",
7          "Meloxicam",
8          "Carvedilol",
9          "Itopride hydrochloridre",
10         "Betahistin dihydrochloride",
11         "Chlorzoxazone",
12         "Rizatriptan benzoate",
13         "Candersatan cilexetil",
14         "Metoprolol",
15         "Domperidone"
16     ],
17     "outputs": [
18         "solubility",
19         "density",
20         "particle size distribution",
21         "porosity"
22     ]
23 }
```

Figura 15: Modelo de entrada para o *scraper* do *website* DrugBank.

A lista de fármacos a serem buscados se encontra na chave “inputs” e a lista de propriedades, na chave “outputs”. O programa foi executado com o comando ilustrado na Figura 14 e processado em 2,92 segundos produzindo como saída um arquivo em formato CSV conforme o modelo apresentado na Figura 16.

	A	B	C	D
1	buscado	encontrado	url	propriedades__solubility__water solubility
2	Meloxicam	Meloxicam	https://www.drugbank.ca/drugs/DB00814	0.154 mg/ml
3	Carvedilol	Carvedilol	https://www.drugbank.ca/drugs/DB01136	0.00444 mg/ml

Figura 16: Modelo de arquivo de saída para o *scraper* do *website* DrugBank.

5.2 MERCK MILLIPORE

O objetivo do programa referente ao *website* Merck Millipore [46] é encontrar uma lista de substâncias pelo nome. Encontrada a página específica de cada substância, buscar por uma lista de propriedades. Ambas as entradas, lista de substâncias e de propriedades, são de responsabilidade do usuário especialista. O usuário edita um arquivo no formato JSON [45] seguindo o modelo demonstrado na Figura 17 e o referencia como entrada para o programa.

```
1  {
2      "inputs": [
3          "Celulose microcristalina",
4          "Polivinilpirrolidona",
5          "Croscarmelose sódica",
6          "Aerosil",
7          "Estearato de magnésio",
8          "Fosfato de cálcio",
9          "Lactose monohidratada"
10     ],
11     "outputs": [
12         "Solubilidade",
13         "Densidade"
14     ]
15 }
```

Figura 17: Modelo de entrada para o *scraper* do *website* Merck Millipore.

A lista de substâncias a serem buscadas se encontra na chave “inputs” e a lista de propriedades, na chave “outputs”. O programa foi executado em 9,72 segundos produzindo como saída um arquivo em formato CSV conforme o modelo apresentado na Figura 18.

	A	B	C	D	E
1	buscado	encontrado	url	propriedades__Solubilidade__solubilidade	propriedades__Densidade__densidade a granel
2	Lactose monohidratada	Lactose monohidratada	http://w	161 g/l	500 kg/m3

Figura 18: Modelo de arquivo de saída para o *scraper* do *website* Merck Millipore.

5.3 SCOPUS

O objetivo do programa referente ao *website* Scopus [47] (e sua API [51]) é encontrar uma lista de artigos por um critério de busca definido pelo usuário especialista. O usuário edita um arquivo no formato JSON [45] seguindo o modelo demonstrado na Figura 19 e o referencia como entrada para o programa. O Scopus realiza autenticação por endereço IP. Para que esta execução fosse possível, o computador utilizado foi conectado à rede da Universidade Federal Fluminense.

```
1 {
2     "busca": "TITLE-ABS-KEY(\"tablet\") AND TITLE-ABS-KEY(\"direct compression\") AND TITLE-ABS-KEY(\"hardness\") AND PUBYEAR > 2009",
3     "api-key": "7d28999d174bb348865941e4c567f0cf"
4 }
```

Figura 19: Modelo de entrada para o *scraper* do *website* Scopus.

O critério de busca a ser utilizado encontra-se na chave “busca”. Inicialmente, o resultado esperado utilizando a chave descrita na Figura 19 era um arquivo com 1257 artigos visto que este era o número de resultados que o sistema do Scopus indicava tanto através de sua API como em sua interface *web*. No entanto, após diversas tentativas, foi identificado um comportamento inesperado no *scraper* que não conseguia processar 100% dos resultados. O arquivo de saída continha entre 1210 e 1250 resultados. Por este motivo, foi acionada uma extensão ao *framework*

Scrapy [4] chamada *autothrotle* [48] que tem o objetivo de evitar a superutilização dos recursos do *website*, aumentando o tempo de espera entre as requisições. Após o ajuste, o programa foi executado em 21 minutos e 18 segundos produzindo como saída um arquivo no formato CSV contendo todos os 1257 resultados esperados conforme o modelo apresentado na Figura 20 (indicando os principais campos).

	A	B	C	D	E	F	G	H	I
1	dc:title	dc:creator	prism:publicationName	prism:elssn	prism:coverDate	link:scopus	abstract	view-in-source	source-score
2	Lignin and cellulc	Domínguez-	Biomolecules	2218273X	01/09/2019	https://www.scopus.com/inv	© 2019 by the auth	https://www.mdpi.com/2218-273X/9/9/423	5.72

Figura 20: Modelo de arquivo de saída para o *scraper* do *website* Scopus.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi possível observar como as informações disponíveis na *web* sob formas diversas de apresentação podem ser coletadas sistemicamente através de *softwares* especialmente desenvolvidos para este fim, chamados *web scrapers*. Foi demostrado na prática a utilização de algumas técnicas para *web scraping* através da coleta de dados específicos para três fontes distintas: DrugBank [12], Merck Millipore [46] e Scopus [47]. Além disso, foram discutidos os resultados junto ao especialista farmacêutico no qual ficou evidente o ganho de produção e a acurácia na execução da coleta de dados pelo *web scraper* em comparação à sua execução manual.

Por fim, nossas sugestões de complementação futura a este trabalho: o programa criado para a busca de artigos científicos na base Scopus poderá ser incrementado com a funcionalidade de *download* da versão integral de cada artigo e, ainda, com uma busca textual de informações contidas nestes artigos. Sempre que exista a necessidade da busca por fontes alternativas de dados, é possível a criação de um novo programa de *scraping* que, ainda que mantenha uma estrutura similar aos programas desenvolvidos neste projeto, deverá ser adaptado à estrutura específica da nova fonte de dados. Ademais a possibilidade de criação de *web scrapers* providos de inteligência artificial e treinados para adaptar-se à diferentes *websites* e/ou APIs para a busca das informações convenientemente parametrizadas pelo pesquisador especialista.

REFERÊNCIAS BIBLIOGRÁFICAS

1. GLEZ-PEÑA, Daniel; *et al.* ***Web scraping technologies in an API world***, 2014. *Briefings in Bioinformatics, Volume 15, Issue 5*.
2. HEDLEY, Jonathan. **Biblioteca Java para trabalhar com HTML.** <https://jsoup.org> Acesso em 06 set. 2019.
3. ORACLE. **Linguagem de programação**. <https://java.com> Acesso em 06 set. 2019.
4. SCRAPINGHUB; *et al*. ***Framework*** **Python para extração de dados de websites**. <https://scrapy.org> Acesso em 06 set. 2019.
5. RICHARDSON, Leonard. **Biblioteca Python para "*scraping* de tela"**. <https://crummy.com/software/BeautifulSoup> Acesso em 06 set. 2019.
6. PYTHON SOFTWARE FOUNDATION. **Linguagem de programação**. <https://python.org> Acesso em 06 set. 2019.
7. FELDMAN, Ronen; *et al*. ***Utilizing Text Mining on Online Medical Forums to Predict Label Change due to Adverse Drug Reactions***. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 10-13, 2015, Sydney, NSW, Australia*.
8. NLTK PROJECT. **Plataforma para criar programas Python para trabalhar com dados de linguagem humana**. <https://www.nltk.org> Acesso em 09 set. 2019.
9. LIU, Yifeng; LIANG, Yongjie; WISHART, David. ***PolySearch2: a significantly improved text-mining system for discovering associations between human diseases, genes, drugs, metabolites, toxins and more.*** *Nucleic Acids Research, Volume 43, Issue W1, 1 July 2015, Pages W535–W542*.
10. CANADIAN INSTITUTES OF HEALTH RESEARCH; *et al*. **Ferramenta *online* para *text-mining***. <http://polysearch.ca> Acesso em 09 set. 2019.
11. ELASTICSEARCH B. V.. **Motor de busca**. <https://elastic.co> Acesso em 09 set. 2019.

12. CANADIAN INSTITUTES OF HEALTH RESEARCH; *et al*. **Banco de dados de medicamentos e substâncias bioquímicas**. <https://drugbank.ca> Acesso em 09 set. 2019.
13. NATIONAL INSTITUTES OF HEALTH, *U.S. Department of Health and Human Services*; *et al*. **Banco de dados de informações funcionais e sequências de proteínas**. <https://uniprot.org> Acesso em 09 set. 2019.
14. *CANADIAN INSTITUTES OF HEALTH RESEARCH*; *et al*. **Banco de dados de informações sobre o metabolismo de pequenas moléculas encontradas no corpo humano**. <http://hmdb.ca> Acesso em 09 set. 2019.
15. AMALIA, Amalia; AFIFA, Rizky Maulidya; HERRIYANCE, Herriyance. ***Resource Description Framework Generation for Tropical Disease Using Web Scraping***, *2018 IEEE International Conference on Communication, Networks and Satellite (Comnetsat), Medan, Indonesia, 2018, pp. 44-48*.
16. *WORLD WIDE WEB CONSORTIUM*, W3C. **Padrões da "*web* semântica"**. <https://w3.org/standards/semanticweb> Acesso em 09 set. 2019.
17. DRUGS.COM. **Banco de dados independente com informações sobre medicamentos**. <https://drugs.com> Acesso em 09 set. 2019.
18. ALODOKTER. **Portal *web* de informações médicas e sobre saúde**. <https://alodokter.com> Acesso em 09 set. 2019.
19. AGÊNCIA NACIONAL DE CONTROLE DE ALIMENTOS E MEDICAMENTOS DA INDONÉSIA. ***Website* oficial da instituição**. <http://pom.go.id> Acesso em 09 set. 2019.
20. *WORLD WIDE WEB CONSORTIUM*, W3C. **Padrões do modelo *Resource Description Framework***. <https://w3.org/RDF/> Acesso em 09 set. 2019.
21. *WORLD WIDE WEB CONSORTIUM*, W3C. **Procedimentos e regras para conversão de dados tabulares em RDF**. <https://w3.org/TR/csv2rdf/> Acesso em 09 set. 2019.
22. GRÄSSER, Felix; *et al*. ***Aspect-Based Sentiment Analysis of Drug Reviews Applying Cross-Domain and Cross-Data Learning***, 2018. *In DH'18:2018 International Digital Health Conference, April 23–26, 2018, Lyon, France. ACM, New York, NY, USA, 5 pages*.
23. DRUGLIB.COM. **Banco de dados sobre medicamentos e pesquisas farmacêuticas**. <http://druglib.com> Acesso em 09 set. 2019.

24. HAN, Run. ***Predicting oral disintegrating tablet formulations by neural network techniques***, 2018. Artigo Científico (publicado no *Asian Journal of Pharmaceutical Sciences 13*) – *Institute of Chinese Medical Sciences* (ICMS), *University of Macau*, Macau, China.
25. BERNERS-LEE, Tim. ***WWW: past, present, and future***, 1996. Publicação acadêmica – *Computer*, vol. 29, no. 10, pp. 69-77.
26. LONGMAN WESLEY, Addison. ***A History of HTML***, 1998**.** <https://www.w3.org/People/Raggett/book4/ch02.html> Acesso em 27 set. 2019.
27. BERNERS-LEE, Tim. ***Information Management: A Proposal***, 1989. <https://www.w3.org/History/1989/proposal.html> Acesso em 27 set. 2019.
28. *WORLD WIDE WEB CONSORTIUM*, W3C. **Definições e recomendações do HTML 5.2 para conteúdos embutidos**. <https://www.w3.org/TR/html52/semantics-embedded-content.html> Acesso em 27 set. 2019.
29. 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.
30. CHOUDHURY, Nupur. ***World Wide Web and Its Journey from Web 1.0 to Web 4.0***, 2014. Artigo científico – Instituto de Tecnologia da Universidade Sikkim Manipal, Índia.
31. FRATERNALI, Piero; ROSSI, Gustavo; SÁNCHEZ-FIGUEROA, Fernando. ***Rich Internet Applications***, 2010. Publicação acadêmica – *IEEE Internet Computing*, vol. 14, no. 3, pp. 9-12.
32. 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*.
33. 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.
34. **Instituto Brasileiro de Geografia** <https://censo2010.ibge.gov.br/materiais/guia-do-censo/questionarios.html> Acesso em 17 de outubro de 2019.
35. ZOZUS NAHM, Meredith. ***The data book collection and management of research data-Chapman and Hall CRC Press***, 2017, p. 36.

36. **DOI – *Digital Object Identifier*.** <https://www.doi.org/> Acesso em 21 de novembro de 2019.
37. **Google Chrome** <https://www.google.com/chrome> Acesso em 17 de outubro de 2019.
38. **Mozilla Firefox** <https://www.mozilla.org/en-US/firefox/new> Acesso em 17 de outubro de 2019.
39. **Microsoft Intenet Explorer** <https://www.microsoft.com/pt-br/download/internet-explorer.aspx> Acesso em 17 de outubro de 2019.
40. AHO, Alfred; ULLMAN, Jeffrey. ***Foundations of Computer Science***, 1992, capítulo 10 – *Patterns, and Regular Expressions*.
41. **XML Path Language (XPath) 3.0** <https://www.w3.org/TR/xpath-30/>
42. PANDAS. **Biblioteca Python para análise de dados**. <https://pandas.pydata.org/> Acesso em 27 nov. 2019.
43. *ITEM EXPORTERS*. **Documentação do *framework* Scrapy**. <https://docs.scrapy.org/en/latest/topics/exporters.html> Acesso em 27 nov. 2019.
44. *SPIDERS*. **Documentação do *framework* Scrapy**. <https://docs.scrapy.org/en/latest/topics/spiders.html> Acesso em 28 nov. 2019.
45. **JSON – *JavaScript Object Notation*.** < http://www.json.org/> Acesso em 21 de novembro de 2019.
46. **MERCK MILLIPORE**. <http://merckmillipore.com/> Acesso em 21 de novembro de 2019.
47. SCOPUS.COM. ***Abstract and citation database of peer-reviewed literature: scientific journals, books and conference proceedings*.** <https://www.scopus.com/> Acesso em 21 de novembro de 2019.
48. **Scrapy Autothrotle *extension*.** <https://docs.scrapy.org/en/latest/topics/autothrottle.html> Acesso em 21 de novembro de 2019.
49. **Seletores CSS.** <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors> Acesso em 21 de novembro de 2019.
50. **XML – *Extensible Markup Language*.** <https://www.w3.org/XML/> Acesso em 21 de novembro de 2019.
51. **Scopus *Search* API.** <https://dev.elsevier.com/documentation/ScopusSearchAPI.wadl> Acesso em 21 de novembro de 2019.

52. ABITEBOUL, Serge; BUNEMAN, Peter; SUCIU, Dan. ***Data on the web – From relations to semistructured data and XML***, Morgan Kaufmann *Publishers*.
53. **Biblioteca Python json.** <https://docs.python.org/3/library/json.html> Acesso em 21 de novembro de 2019.
54. KUIPER, Els; VOLMAN, Monique. ***The Web as a source of information for students in K–12 education***, 2008. *In J. Coiro, M. Knobel, C. Lankshear, & D. Leu (Eds.), Handbook of research on new literacies (pp. 241–266). New York: Lawrence Erlbaum*.
55. HEWSON, Claire; STEWART, David. ***Internet Research Methods***, 2016. *Wiley StatsRef: Statistics Reference Online, 1–6*.
56. SAGIROGLU, Seref; SINANC, Duygu. ***Big data: A review***, 2013. *International Conference on Collaboration Technologies and Systems (CTS)*.
57. YOO, Illhoi; *et al*. ***Data Mining in Healthcare and Biomedicine: A Survey of the Literature***, 2011. *Journal of Medical Systems, 36(4), 2431–2448*.
58. BADROS, Greg; *et al*. ***Constraint cascading style sheets for the Web***, 1999. *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology – UIST '99*.
59. FLANAGAN, David. ***JavaScript: The Definitive Guide, 3rd edition***, 1998. Livro. *O'Reilly & Associates, Sebastopol*, Canadá.
60. LAWSON, Bruce; SHARP, Remy. ***Introducing HTML5, 2nd Edition***, 2011. Livro. *New Riders Press, Berkeley,* Canadá.
61. FELKEY, Bill; LIANG, Huigang; KRUEGER, Kem. ***Data Mining for the Health System Pharmacist***, 2003. *Hospital Pharmacy, 38(9), 845–850*.

ANEXOS

ANEXO A – CONTEÚDO DA REUNIÃO COM O ESPECIALISTA

O documento abaixo apresenta o conteúdo da reunião feita em 15/08/2019 da qual participaram Altobelli de Brito (Orientador), Marianne Rezende (Especialista Farmacêutico), Miquéias Dernier (aluno) e Vinicius Zanovelli (aluno):

Manual da Busca de Parâmetros de entrada e saída

Devido ao interesse deste projeto em buscar informações sobre as propriedades reológicas dos fármacos e correlacioná-las com as propriedades formulativas de comprimidos obtidos por compressão direta, inicialmente se fez necessária à pesquisa por artigos que contenham as variáveis de interesse. Primeiramente é necessário começar pelas propriedades de saída do fármaco (ex: friabilidade, dureza, tempo de desintegração e tempo de dissolução). Os artigos selecionados precisam ter no mínimo uma das variáveis de saída. Foi observado ao longo do estudo, que as variáveis de entrada não se encontram no mesmo artigo que contem as propriedades de saída. Dessa forma, para cada artigo selecionado, foi necessário identificar o fármaco contido nele e com esta informação buscar as variáveis de entrada em outras fontes (Ex: Drugbank (solubilidade), SciFinder (densidade)).

A seguir são demonstrados os passos para obtenção dos parâmetros definidos previamente.

Fontes de Informação:

Scopus - https://www.scopus.com/search/form.uri?display=basic
Requisito mínimo – Ter acesso em redes de universidades

A seguir é demonstrado em etapas, de maneira a contextualizar, um exemplo de busca da variável dureza.

6 Acessar o site do Scopus;

7 Pesquisar no Scopus no campo “Search” as palavras “tablet”; “direct compression”; “hardness”;

A **Figura 1** contextualiza o início da busca dos artigos científicos.

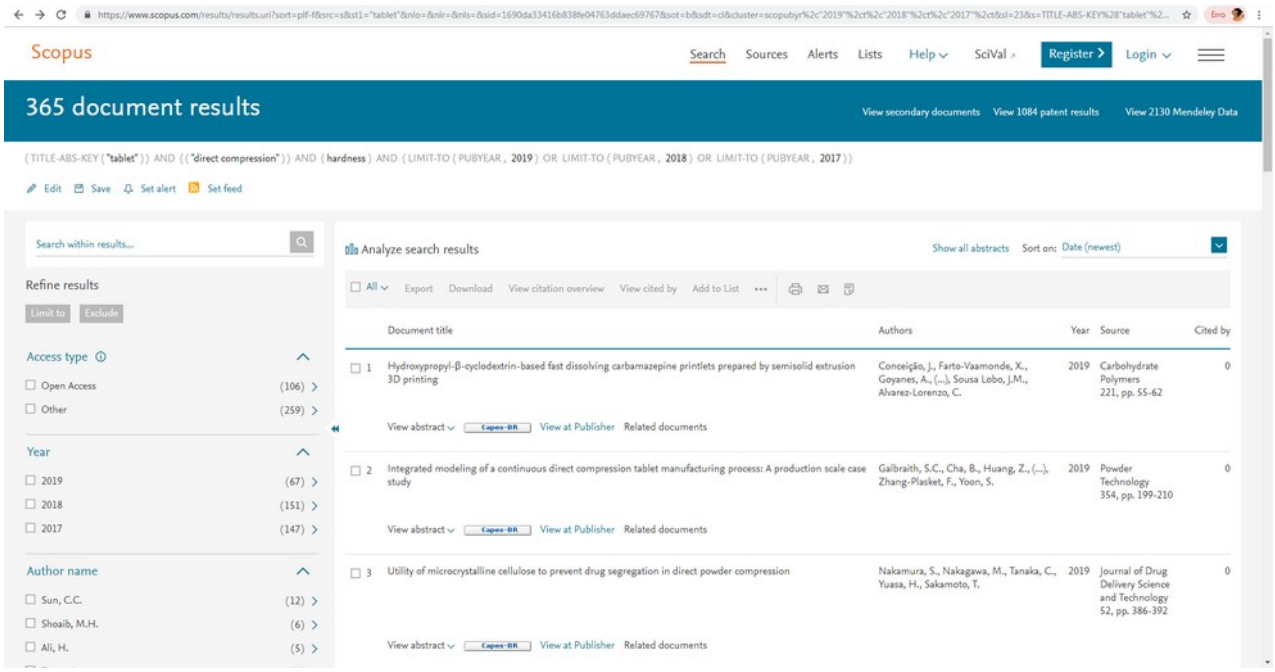

Figura 1. Resultados obtidos da busca realizada com as palavras-chave descritas acima.

8 Buscar artigos dos dez últimos anos, de maneira a obter dados mais atualizados. A **Figura 2** mostra um exemplo de seleção dos anos 2019, 2018 e 2017.

Figura 2. Demonstração de como é realizada a etapa de inclusão/exclusão dos anos.

9 Após filtrar os artigos constantes nos anos definidos, a pesquisa avança para a etapa de escolha dos artigos. Para tal, é feita uma leitura prévia do título e do conteúdo do *abstract* (**Figura 3**).

Figura 3. Artigo científico após ser selecionado para que seja avaliado se apresenta as palavras-chave no título ou *abstract*.

Se o artigo científico apresentar as informações necessárias (palavras-chave), o mesmo é selecionado para que seja avaliado o fator de impacto, como pode ser observado na **Figura 4**. Para isso, é necessário clicar no nome da revista (demonstrado na **Figura 3**), sendo logo após apresentado o valor de fator de impacto.

Figura 4. Revista selecionada informando o fator de impacto.

Caso o artigo esteja dentro dos requisitos estabelecidos (Fator de impacto >1 e nos últimos 3 anos), é realizada a leitura mais aprofundada do mesmo, de maneira a verificar se de fato estão contidas as informações desejadas (*inputs* e *outputs*).

SciFinder- https://scifinder.cas.org/
Requisito mínimo - login e senha para acesso

A seguir é demonstrado em etapas, de maneira a contextualizar, a busca pelas propriedades farmacêuticas (Solubilidade e Densidade).

A **Figura 5** contextualiza o início da busca das propriedades de densidade e solubilidade farmacêuticas.

Entrar no site do SciFinder através da Plataforma de periódicos da Capes

1. Entrar na plataforma de periódicos Capes (**Figura 5**);

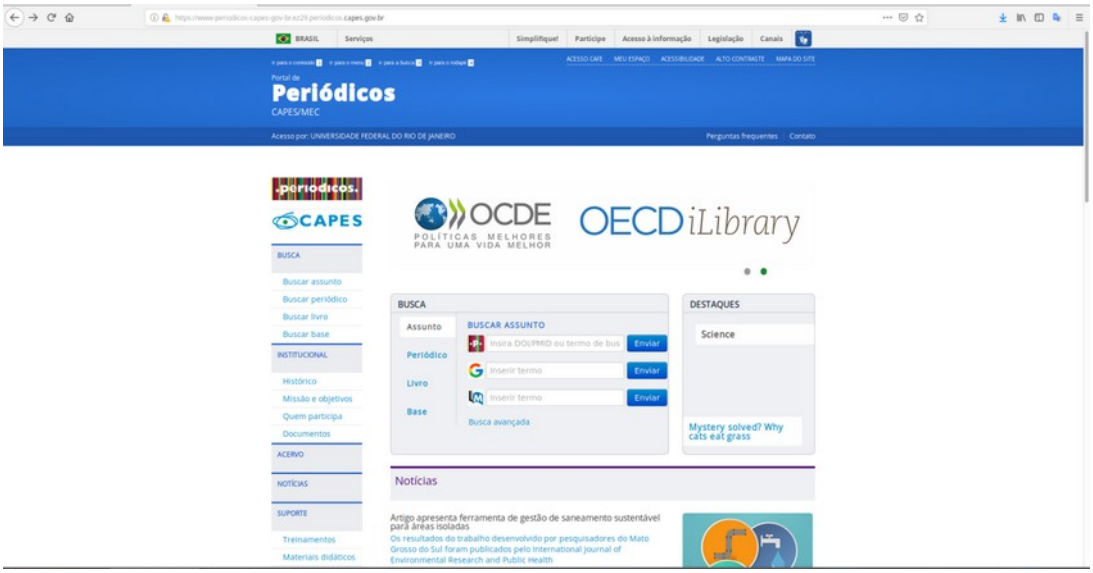

Figura 5. Site da plataforma de periódicos da CAPES.

2. Selecionar, nos itens de busca, o termo “Base” e digitar SciFinder (**Figura 6**);

Figura 6. Demonstração da busca da base de dados do SciFinder no portal de periódicos da CAPES.

3. Acessar o SciFinder (**Figuras 7**, **8** e **9**).

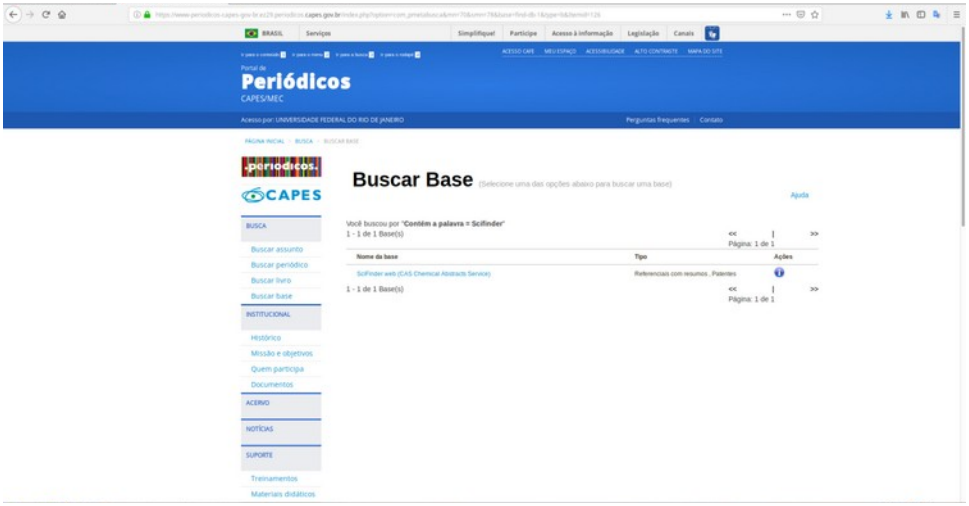

Figura 7. Acesso ao SciFinder.

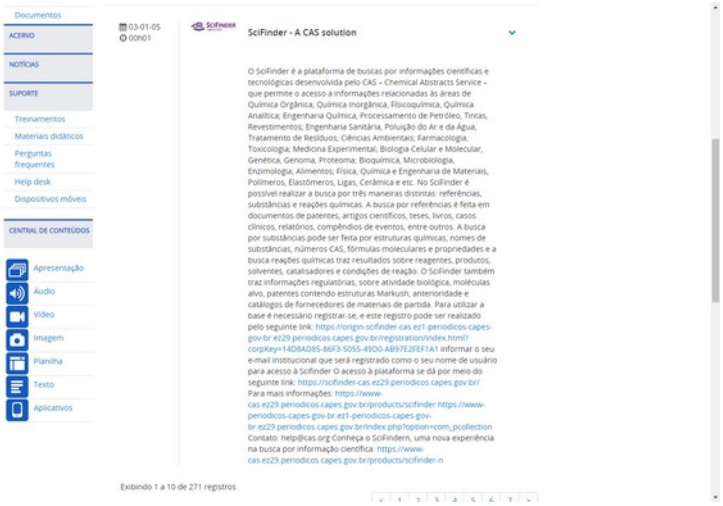

Figura 8. Descrição do SciFinder e link de acesso à plataforma.

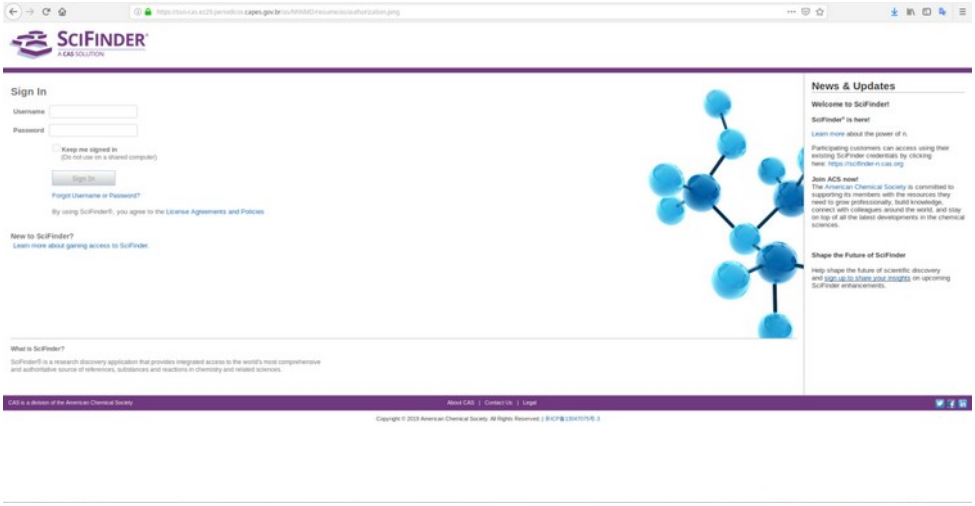

Figura 9. Página de acesso do SciFinder.

4. Acessar a plataforma utilizando um login e senha pré-definidos.

Ter login e senha disponíveis

10 Para a busca das variáveis de entrada dos fármacos, selecionar o item “Substance Identifier” e escrever o código da molécula (CAS) (Figura **10**);

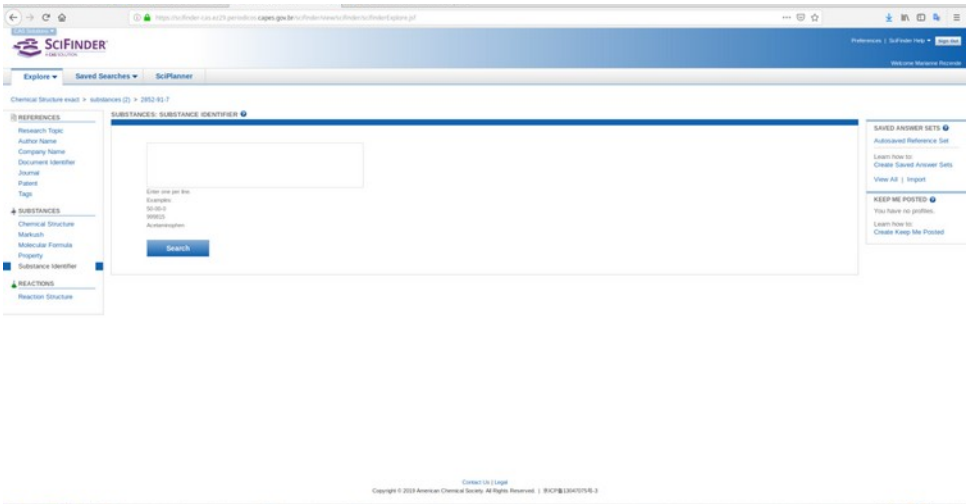

Figura 10. Página para busca das substâncias desejadas.

11 Para encontrar o código da substância desejada, realizar uma busca no site de compras da Sigma-Aldrich (https://www.sigmaaldrich.com/brazil.html) com o nome do fármaco (em português) em questão, de maneira a se obter o CAS (**Figura 11**).

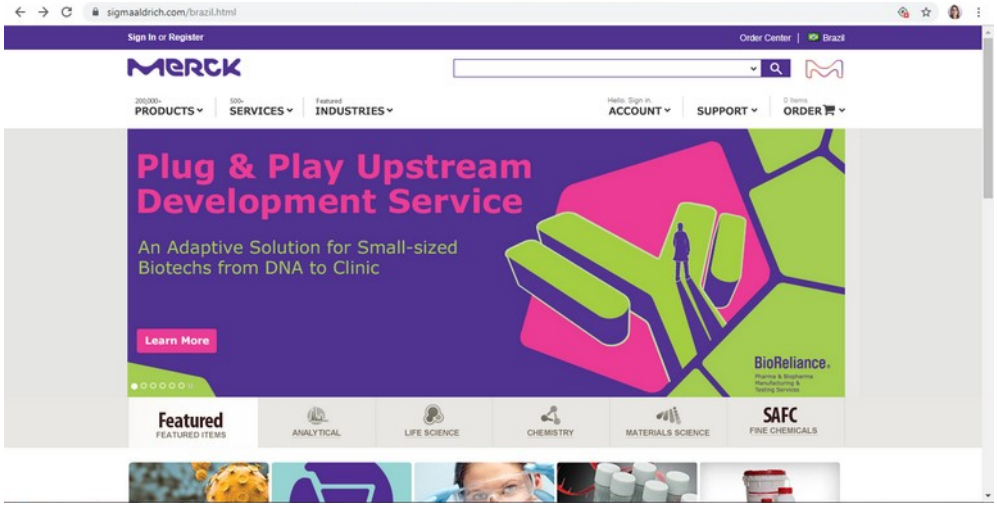

Figura 11. Site de vendas da Sigma-Aldrich.

12 No espaço destinado a busca, digitar o nome da substância desejada (**Figura 12**);

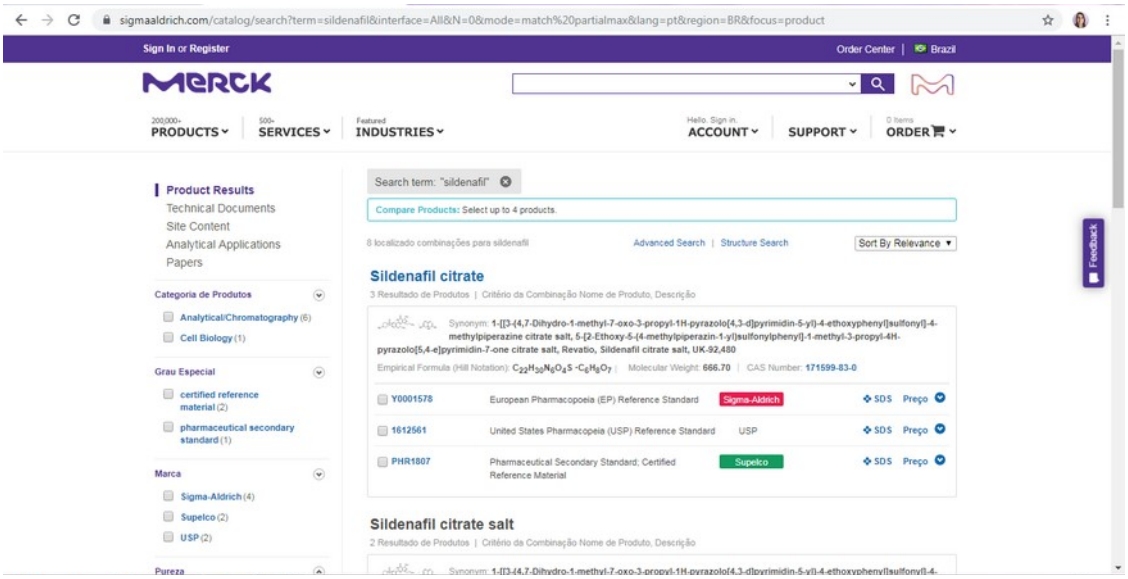

Figura 12. Exemplo de busca da substância Sildenafila e os resultados obtidos.

13 Nos resultados obtidos, selecionar a primeira opção e coletar o número CAS da substância (**Figura 13**);

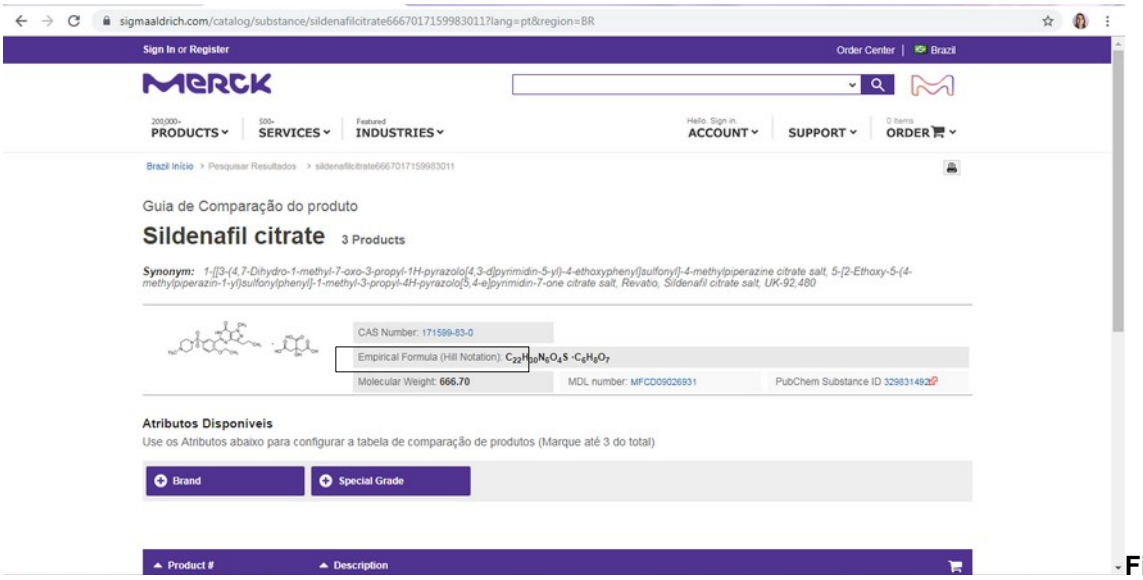

Figura 13. Apresentação do produto contendo o número CAS.

14 Digitar o número CAS coletado no SciFinder (**Figura 10**);

15 Selecionar a estrutura que surgir da busca (**Figura 14**);

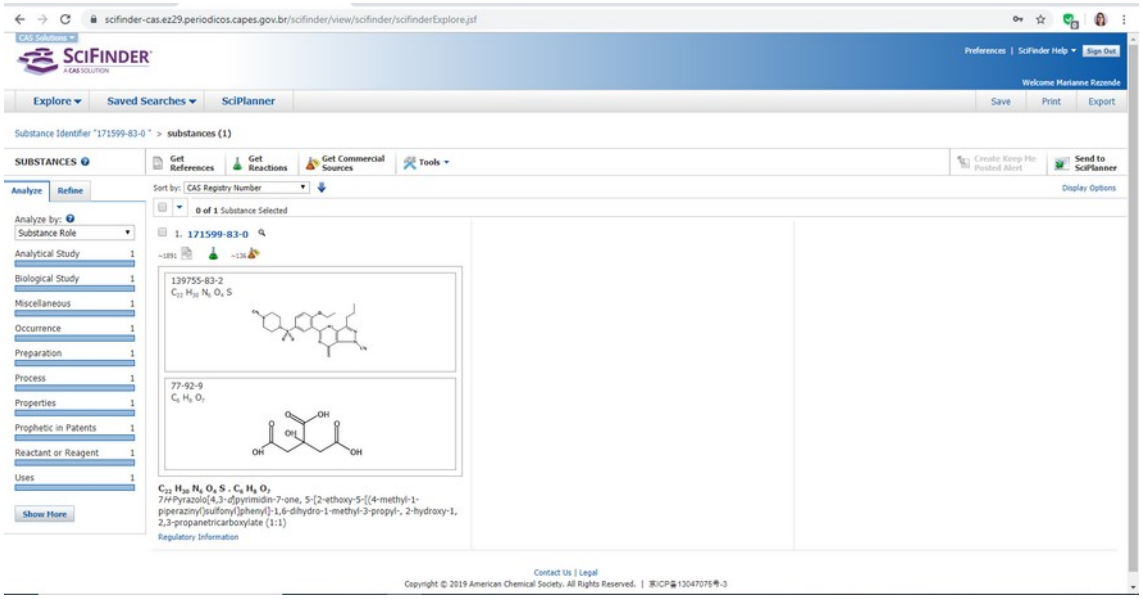

Figura 14. Resultado da busca pelo CAS da substância Sildenafila.

16 Acessar o item “Experimental Properties” e coletar os parâmetros de entrada desejados (**Figura 15**).

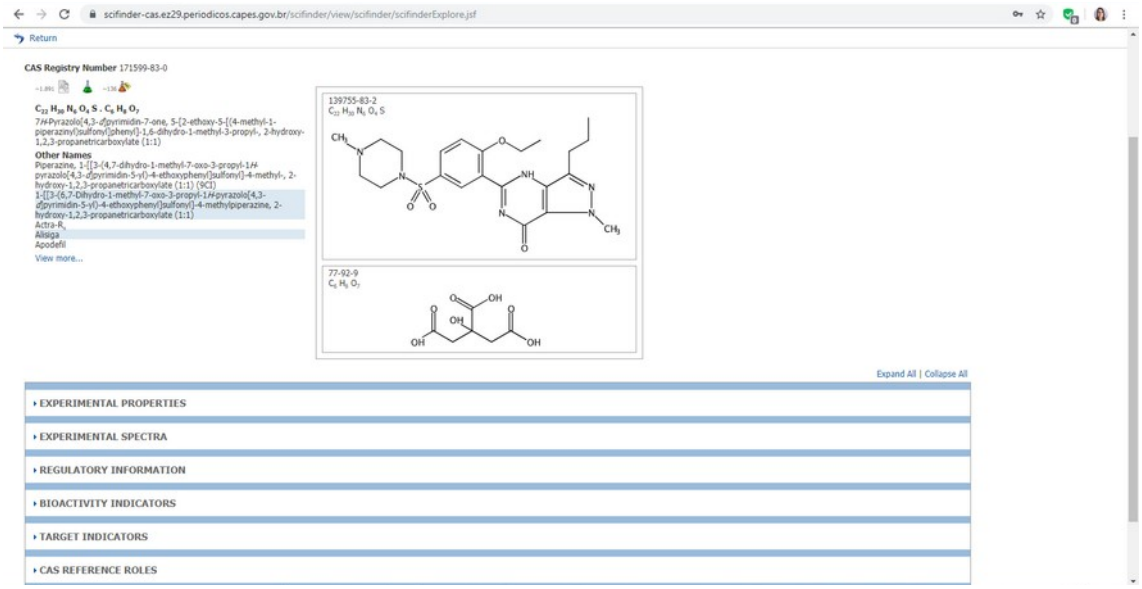

Figura 15. Acesso aos dados experimentais da substância Sildenafila.

DrugBank -https://www.drugbank.ca/

Sem requisito mínimo para acesso

A seguir é demonstrado em etapas de maneira a contextualizar a busca pelas propri-edades farmacêuticas (Solubilidade e Densidade).

1. Entrar no site DrugBank (**Figura 16**);

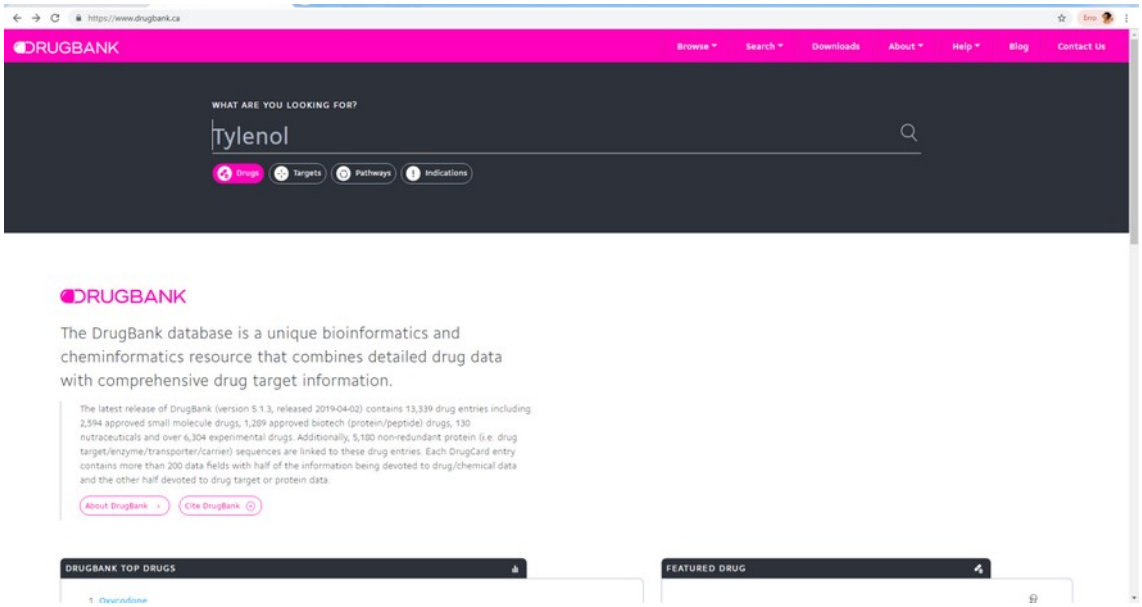

Figura 16. Site do Drugbank.

2. Digitar o nome do fármaco no campo: “ What are you looking for?”, que pode ser observado na **Figura 16** acima;

3. O valor de solubilidade e/ou densidade se encontra na coluna “Predicted Properties”. Nessa coluna foi utilizado o parâmetro “water solubility” e/ou “density”. Esse valor se encontra no campo “Value”.

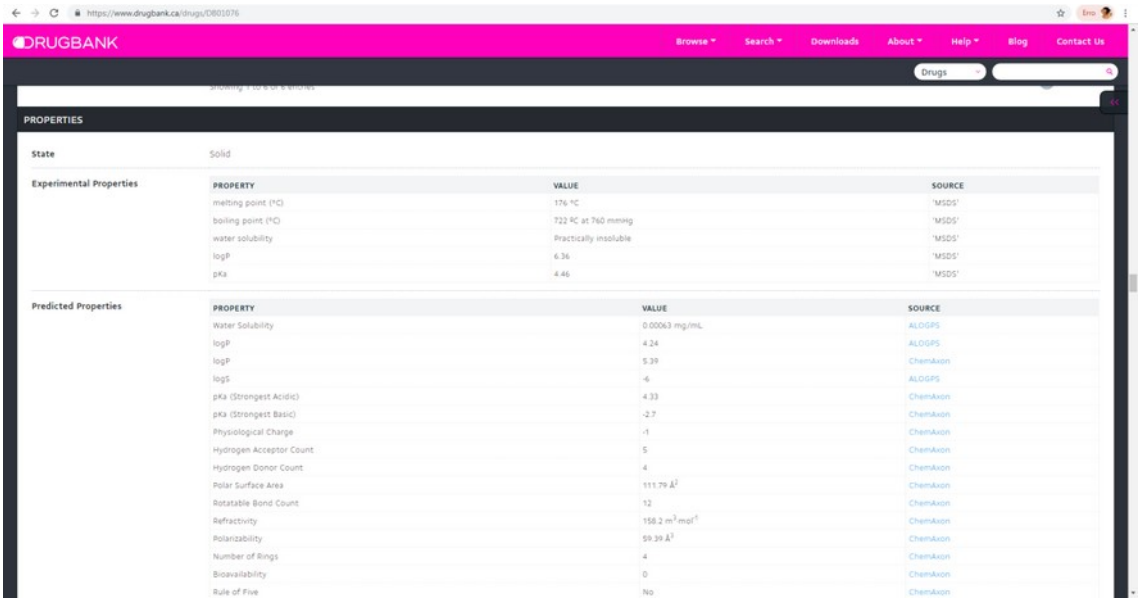

Figura 17. Demonstrativo de onde encontrar as propriedades de solubilidade e densidade dos fármacos.

Merck - http://www.merckmillipore.com/BR/pt?ReferrerURL=https%3A%2F%2Fwww.google.com%2F

Sem requisito mínimo para acesso

Excipientes definidos previamente mais comumente utilizados em comprimidos obtidos por compressão direta: **Celulose microcristalina, Polivinilpirrolidona, Croscarmelose sódica, Aerosil, estearato de magnésio, fosfato de cálcio e lactose monohidratada**.

A seguir é demonstrado em etapas de maneira a contextualizar a busca pelas propriedades referente aos excipientes (Solubilidade e Densidade).

1. Acessar o site de vendas da empresa Merck (**Figura 18**);

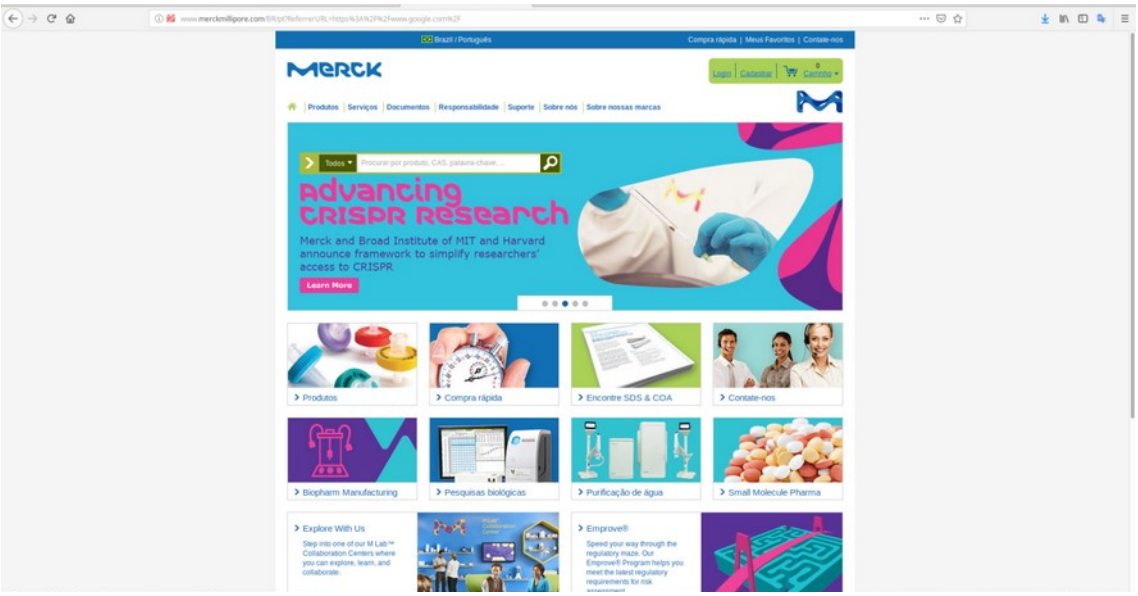

Figura 18. Site de vendas da empresa Merck.

2. Digitar o nome do excipiente em português no espaço destinado “Procurar por produto, CAS, palavra-chave, ...” (que pode ser observado na **Figura 18**);

Caso apareça mais de um resultado, escolher o primeiro.

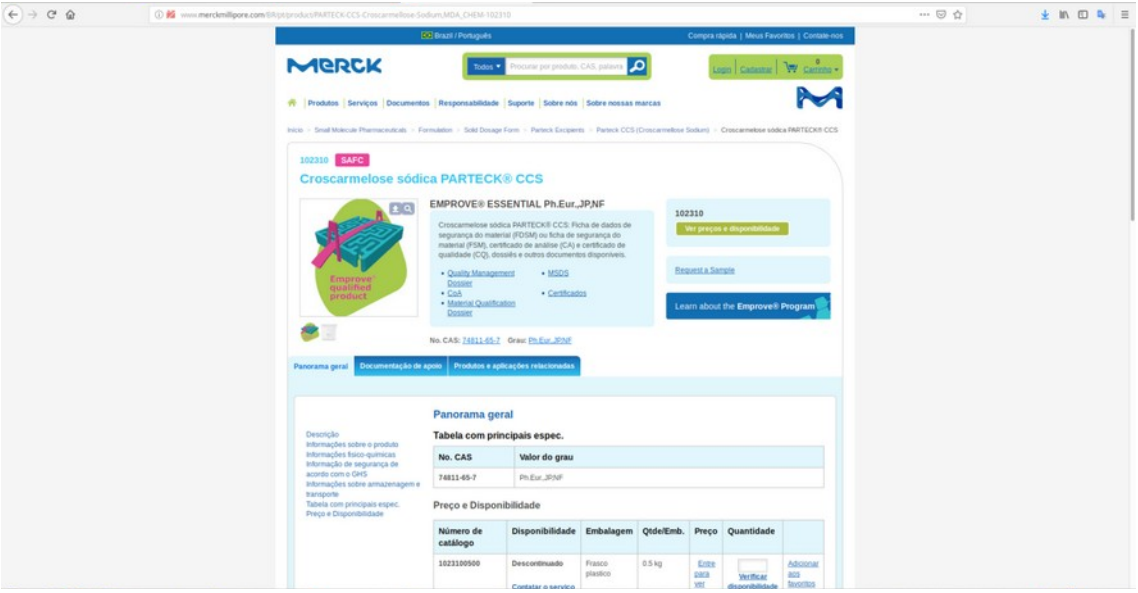

Figura 19. Exemplo de busca do excipiente Croscarmelose sódica.

3. O valor de solubilidade e/ou densidade se encontra na coluna “Informações físico-químicas”;

4. Da mesma maneira que foi utilizado para os fármacos selecionados, os parâmetros de entrada dos excipientes também foram explorados no SciFinder, utilizando a mesmo ordem descrita anteriormente, porém não sendo necessária a estrutura química, somente o código de identificação da substância (CAS), código este que pode ser encontrado no próprio site da Merck mencionado anteriormente (**Figura 19**);

5. Para a busca das variáveis de entrada dos excipientes, selecionar o item “Substance Identifier” e digitar o CAS da molécula desejada (**Figura 10**);

6. Selecionar a substância (**Figura 20**);

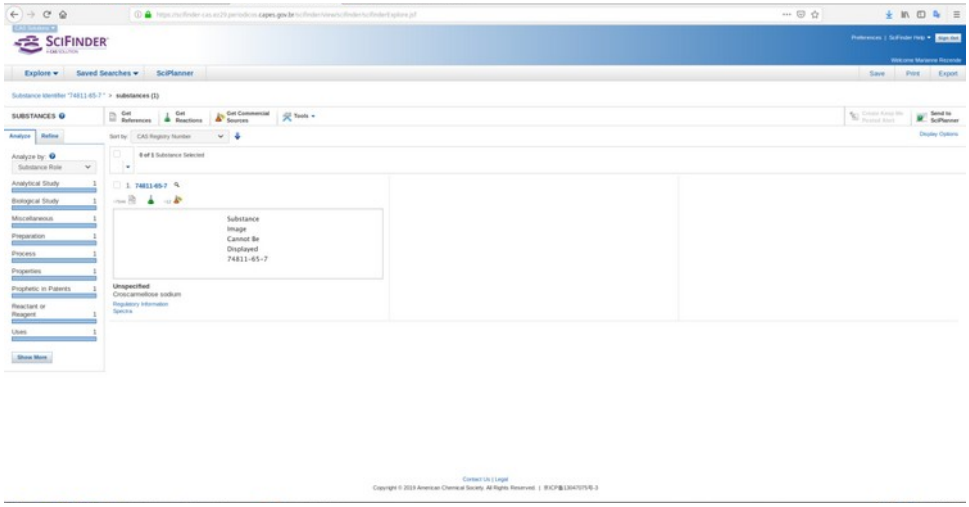

Figura 20. Resultado da busca pelo CAS da substância desejada.

7. Coletar as informações necessárias (**Figura 21**);

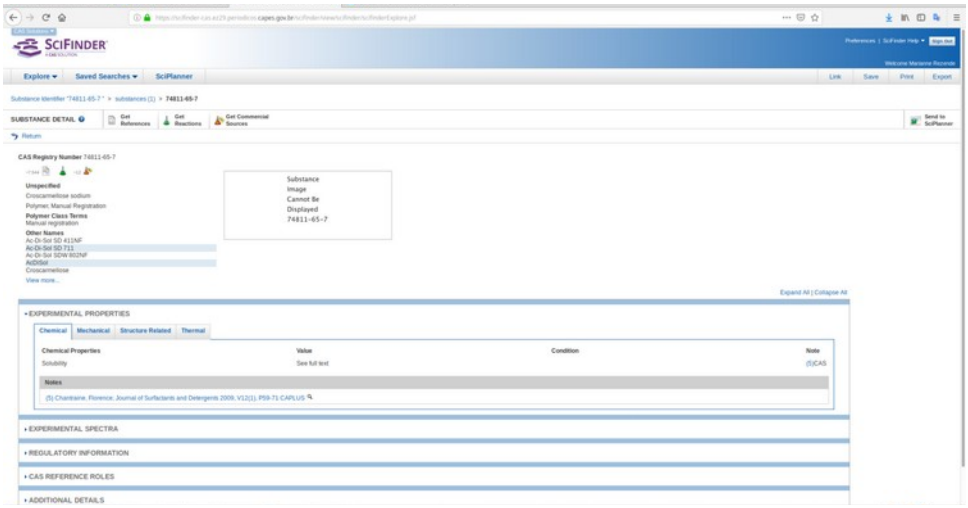

Figura 21. Demonstrativo de onde encontrar as propriedades de solubilidade e densidade dos fármacos.

Observações: Em alguns casos o Scifinder não disponibiliza do valor desejado, e recomenda a base bibliográfica de onde o valor foi retirado, para casos assim, o artigo é acessado de maneira a coletar as informações faltantes (**Figuras 21** e **22**).

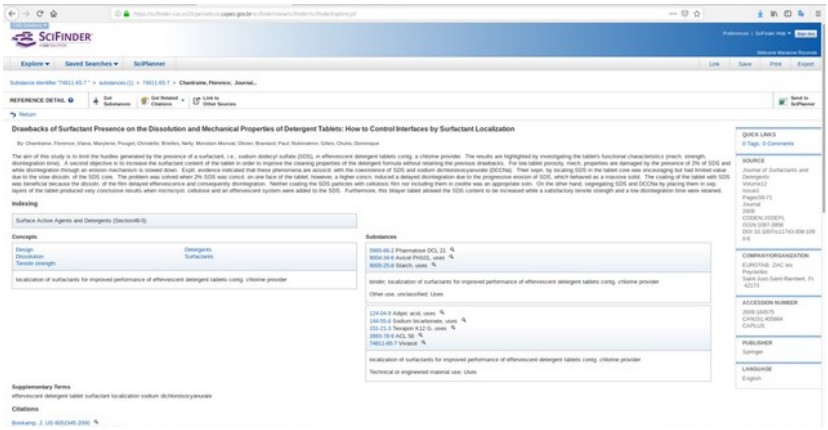

Figura 22. Acesso ao artigo que consta as informações desejadas.

O nome do artigo é então levado para o site Sci-Hub (http://sci-hub.tw/), no qual o mesmo é disponibilizado gratuitamente para download (**Figura 23**);

Figura 23. Demonstração de busca de artigo no Sci-Hub.

Com o artigo, realiza-se a busca dos parâmetros desejados (**Figura 24**).

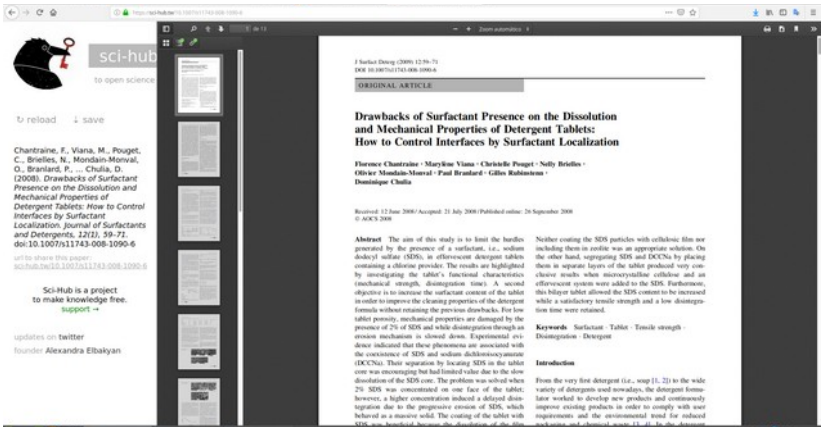

Figura 24. Artigo disponível para a busca dos parâmetros desejados.

ANEXO B – DOCUMENTO DE ACEITAÇÃO ASSINADO PELO ESPECIALISTA

UNIVERSIDADE DO BRASIL
UFRJ

FACULDADE DE FARMÁCIA

DOCUMENTO DE VALIDAÇÃO DE RESULTADO

Eu, Marianne Grilo Rezende, aluna de Doutorado do Programa de Pós-Graduação em Ciências Farmacêuticas (PPGCF) da Universidade Federal do Rio de Janeiro (UFRJ), no Centro de Ciências da Saúde (CCS), venho por meio deste validar os resultados obtidos no teste realizado com a ferramenta de *web scrapping* utilizando as fontes DrugBank, Merck e Scopus. Para isso, foram utilizados como parâmetros de entrada o nome dos fármacos e excipientes definidos para o estudo, e como parâmetros de saída "solubility", "density", "particle size distribution" e "porosity" para as fontes DrugBank e Merck, e para a fonte Scopus os parâmetros de entrada utilizados foram "tablet", "direct compression" e "hardness".

Por ser verdade, firmo o presente para que surte seus efeitos legais.

Rio de Janeiro, 28 de novembro de 2019.

Marianne Grilo Rezende

MSc. Marianne Grilo Rezende
Faculdade de Farmácia – FF – UFRJ
Programa de Pós-Graduação em Ciências Farmacêuticas - PPGCF

Universidade Federal do Rio de Janeiro – Departamento de Fármacos e Medicamentos - Faculdade de Farmácia, Ilha do Fundão, Prédio do CCS, Laboratório de Tecnologia Industrial Farmacêutica (LabTIF), Bloco L, Subsolo, sala 20. CEP: 21.941-590.

Figura 1 – Documento de aceitação por parte do especialista.