UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO POLITÉCNICO
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

Guilherme Alan Mohr

WEB SCRAPING DE DADOS REFERENTES A INDICADORES RELACIONADOS À PRODUÇÃO RURAL

Santa Maria, RS, Brasil
2023

Guilherme Alan Mohr

WEB SCRAPING DE DADOS REFERENTES A INDICADORES RELACIONADOS À PRODUÇÃO RURAL

Trabalho de Conclusão de Curso - TCC apresentado ao Curso Superior de Tecnologia em Sistemas para Internet da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de Tecnólogo em Sistemas para Internet

Orientador: Prof. Dr. Daniel Lichtnow

Santa Maria, RS, Brasil
2023

Guilherme Alan Mohr

WEB SCRAPING DE DADOS REFERENTES A INDICADORES RELACIONADOS À PRODUÇÃO RURAL

Trabalho de Conclusão de Curso - TCC apresentado ao Curso Superior de Tecnologia em Sistemas para Internet da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Tecnólogo em Sistemas para Internet**

Aprovada em 30 de janeiro de 2023

_____

Daniel Lichtnow, Dr.
(Presidente/Orientador)

_____

Juçara Salete Gubiani, Dra. (UFSM)

_____

Gustavo Pinto da Silva, Dr. (UFSM)

Santa Maria, RS
2023

RESUMO

WEB SCRAPING DE DADOS REFERENTES A INDICADORES RELACIONADOS À PRODUÇÃO RURAL

AUTOR: Guilherme Alan Mohr
ORIENTADOR: Prof.º Dr. Daniel Lichtnow

Este trabalho propõe-se a desenvolver uma Base de Dados representada por um ou mais arquivos CSV, contendo dados relativos ao contexto agrícola, de modo que seja possível permitir a realização de uma análise, e se possível auxiliar na tomada de decisão acerca da produção agrícola e a realização de pesquisas futuras. Além disto, pretende-se disponibilizar esta Base de Dados criada na página do GIPAG (Grupo Interdisciplinar de Pesquisas Agroalimentares Georreferenciadas). Estes dados serão obtidos através do processo de Web Scraping, que é o processo de Extração de Informação aplicado na Web. Para tanto, foram realizadas revisão teórica sobre o processo de Web Scraping e sobre ferramentas que podem ser utilizadas para realizar este processo e que sejam compatíveis com a Linguagem de Programação Python. A linguagem foi escolhida para a realização do processo de Web Scraping, pois é versátil e possui diversas bibliotecas que facilitam a realização deste processo. Com base nesta revisão sobre as ferramentas de busca, elencou-se três com maior destaque, sendo elas, as seguintes ferramentas: Scrapy, Beautiful Soup e Selenium. Sobre cada ferramenta serão apresentadas as principais características, juntamente com dois exemplos de extração de dados com cada ferramenta. Em seguida, serão apresentados os estudos dos portais e os sistemas que implementam o processo de Web Scraping neles. Posteriormente serão descritos alguns dos principais dados presentes na base desenvolvida, detalhando sua fonte e relevância. Por fim, serão apresentadas as considerações finais e as ideias para a continuidade deste trabalho.

Palavras-chave: Base de Dados. Produção Agrícola. *Web Scraping*.

ABSTRACT

WEB SCRAPING OF DATA REGARDING INDICATORS RELATED TO RURAL PRODUCTION

AUTHOR: Guilherme Alan Mohr
ADVISOR: Prof.º Dr. Daniel Lichtnow

This work proposes to develop a Database represented by one or more CSV files, containing data related to the agricultural context, so that it is possible to allow an analysis to be carried out and, if possible, to assist in decision-making about agricultural production. and carrying out future research. In addition, it is intended to make this Database created available on the GIPAG (Interdisciplinary Group of Georeferenced Agro-Food Research) page. This data will be obtained through the Web Scraping process, which is the Information Extraction process applied on the Web. Therefore, a theoretical review was carried out on the Web Scraping process and on tools that can be used to carry out this process and that are compatible with the Python Programming Language. The language was chosen to perform the Web Scraping process, as it is versatile and has several libraries that facilitate this process. Based on this review of search tools, the three most prominent were listed, namely, the following tools: Scrapy, Beautiful Soup and Selenium. About each tool, the main characteristics will be presented, along with two examples of data extraction with each tool. Next, the studies of the portals and the systems that implement the Web Scraping process in them will be presented. Later, some of the main data present in the developed database will be described, detailing their source and relevance. Finally, final considerations and ideas for the continuation of this work will be presented.

Keywords: Database. Agricultural Production. *Web Scraping*.

LISTA DE FIGURAS

FIGURA 1 - Estrutura padrão do Scrapy ……..17
FIGURA 2 - Site do Grêmio antes da execução da extração dos dados ..……..18
FIGURA 3 - Código que extrai as notícias em destaque no site do Grêmio ...……18
FIGURA 4 - Resultado da consulta realizada dia 03 de agosto de 2022 ……19
FIGURA 5 - Imagem do site antes do processo de extração ……20
FIGURA 6 - Código que extrai as principais citações ……..20
FIGURA 7 - Resultado da extração das citações ……21
FIGURA 8 - Imagem da página da Wikipédia ……22
FIGURA 9 - Código que extrai os estados do Brasil ……...22
FIGURA 10 - Resultado da extração dos estados brasileiros ……...23
FIGURA 11 - Site do G1 ……..24
FIGURA 12 - Código que extrai o nome dos cantores ……...24
FIGURA 13 - Resultado da extração dos nomes dos artistas ……..…25
FIGURA 14 - Código que insere texto no elemento HTML *input* ……26
FIGURA 15 - Navegador aberto com *Web Driver* na página indicada ……27
FIGURA 16 - Resultado da inserção de valor no campo input ……28
FIGURA 17 - Site do Clima Tempo ……28
FIGURA 18 - Código que extrai informações climáticas - Parte 1 ……29
FIGURA 19 - Código que extrai informações climáticas - Parte 2 ……30
FIGURA 20 - Código que extrai informações climáticas - Parte 3 ……31
FIGURA 21 - Código que inicializa a classe que extrai os dados ……31
FIGURA 22 - Resultado da extração dos dados climáticos ……..31
FIGURA 23 - Execução da digitação caracter por caracter ……..32
FIGURA 24 - Tabela 6954 - Parte 1……...35
FIGURA 25 - Tabela 6954 - Parte 2……..……..36
FIGURA 26 - Tabela 6954 - Parte 3……36
FIGURA 27 - Tabela 6954 - Parte 4……37
FIGURA 28 - Página Completa……..……..38
FIGURA 29 - Quadro de layout das tabelas de retorno……..39
FIGURA 30 - Variáveis e Tipologias selecionadas ……..39
FIGURA 31 - Produtos e Grupos de área total selecionados……39
FIGURA 32 - Retorno da Consulta no SIDRA……..40

FIGURA 33 - Página inicial do sistema da Conab……...41
FIGURA 34 - Tela de acesso aos dados do Conab…….42
FIGURA 35 - Organização da tabela de retorno…….....42
FIGURA 36 - Exemplo de consulta e retorno no site da Conab……...43
FIGURA 37 - Censo Demográfico 2010 - Página Inicial…….43
FIGURA 38 - Dados exibidos após a seleção do Estado e da Informação…...…….44
FIGURA 39 - Sistema de extração SIDRA Tabela 6954 - Parte 1……..…46
FIGURA 40 - Sistema de extração SIDRA Tabela 6954 - Parte 2……..…47
FIGURA 41 - Sistema de extração SIDRA Tabela 6954 - Parte 3……..…47
FIGURA 42 - Interface Web mostrando as tabelas ……..……48
FIGURA 43 - Sistema de extração SIDRA Tabela 6954 - Parte 4……..……48
FIGURA 44 - Sistema de extração SIDRA Tabela 6954 - Parte 5……..……49
FIGURA 45 - Arquivo CSV SIDRA Tabela 6954……..…49
FIGURA 46 - Sistema de extração Conab - Parte 1……50
FIGURA 47 - Sistema de extração Conab - Parte 2……51
FIGURA 48 - Sistema de extração Conab - Parte 3……52
FIGURA 49 - Arquivo CSV Conab……..52
FIGURA 50 - Sistema de extração de dados da População……..53
FIGURA 51 - Cálculo da população rural……. ……53
FIGURA 52 - Produtos e Grupos de área total selecionados ……...54
FIGURA 53 - Esquema de junção dos dados……..……55
FIGURA 54 - Cidades@ IBGE e outros dados dos municípios ……65
FIGURA 55 - SIDRA Tabela 6956……. ……65

SUMÁRIO

1 INTRODUÇÃO……..9
2 REFERENCIAL TEÓRICO……...12
3 FERRAMENTAS DE WEB SCRAPING……16
3.1 SCRAPY..... 16
3.1.1 Exemplo 1 de Algoritmo com Scrapy…….. 18
3.1.2 Exemplo 2 de Algoritmo com Scrapy…….. 19
3.2 BEAUTIFUL SOUP 21
3.2.1 Exemplo 1 de Algoritmo com Beautiful Soup……21
3.2.2 Exemplo 2 de Algoritmo com Beautiful Soup……24
3.3 SELENIUM……25
3.3.1 Exemplo 1 de Algoritmo com Selenium……..26
3.3.2 Exemplo 2 de Algoritmo com Selenium……..28
3.4 COMPARATIVO ENTRE AS FERRAMENTAS…… 32
4 ESTUDO DOS PORTAIS……..34
4.1 SISTEMA IBGE DE RECUPERAÇÃO AUTOMÁTICA (SIDRA)..... 34
4.2 CONAB……...40
4.3 IBGE CENSO DEMOGRÁFICO 2010……...43
5 SISTEMA DE EXTRAÇÃO DOS DADOS…… 45
5.1 SISTEMA IBGE DE RECUPERAÇÃO AUTOMÁTICA TABELA 6954……45
5.2 CONAB……...50
5.3 IBGE CENSO DEMOGRÁFICO 2010……...52
6 BASE DE DADOS MONTADA…… 55
7 CONSIDERAÇÕES FINAIS…… 64
7.1 CONTINUIDADE DO TRABALHO…… 65
REFERÊNCIAS…… 68

1 INTRODUÇÃO

O *Web Scraping,* que é um tipo de Extração de Informação focado no ambiente *Web*, apesar de não ser algo tão recente, tem ganhado destaque, pois conforme salienta Mitchell (2018, p. 11) “[...] engloba uma grande variedade de técnicas de programação e de tecnologias, por exemplo, análise de dados, parsing de idiomas naturais e segurança de informação”. Além disso, o *Web Scraping* de acordo com Mitchell (2018, p. 12) “Independentemente de sua área de atuação, [...] quase sempre oferece uma forma de orientar práticas de negócios com mais eficiência, melhorar a produtividade ou até mesmo dar origem a uma área totalmente nova.”

Na *Web* existem diversos dados públicos importantes relacionados e de interesse relacionados ao contexto rural, que estão dispersos em múltiplos sites. Mesmo existindo mecanismos que permitem inclusive baixar os dados em arquivos, porém é necessário que o usuário efetue múltiplas consultas para obter dados que lhe interessam ou realizar ajustes nos arquivos que é possível fazer download.

Um exemplo de dados que estão disponíveis publicamente na *Web*, são as informações relativas à comercialização e à produção, presentes no sistema SIDRA e no Conab, respectivamente. Estas informações a serem extraídas e os sistemas alvos de extração foram definidos conjuntamente com membros do GIPAG (Grupo Interdisciplinar de Pesquisas Agroalimentares Georreferenciadas)[1], que possuem conhecimento relacionado à área Agrícola.

Para minimizar este problema a proposta é desenvolver um sistema que automatize a construção das consultas e a extração de dados dos sistemas *Web*, posteriormente serão unidos os dados públicos que estão presentes nos sites da *Web*.

Objetivo geral: Extrair dados da *Web* de diferentes sites e portais para construir e disponibilizar acesso a uma base de dados com dados relevantes relacionados ao agronegócio e produzir um ou mais arquivos CSV contendo estes dados, de modo a permitir a realização de uma análise destes dados em pesquisas futuras.

Para tanto, este trabalho estabeleceu os seguintes objetivos específicos:

- Identificar Web sites com dados públicos e relevantes;

[1] https://www.ufsm.br/grupos/gipag

- Desenvolver um sistema para realizar a extração dos dados públicos sobre a produção agrícola;
- Desenvolver um sistema para gerar os arquivos CSV com os dados extraídos;
- Descrever o processo de extração dos dados dos sistemas *Web* pelo sistema de *Web Scraping*.

Este projeto visa, por meio da base de dados, permitir uma visão geral a respeito da Comercialização, Produção e População de produtos ligados à horticultura, e futuramente a fruticultura, unindo dados dispersos em diversos sites da *Web*, de modo que facilite o processo de análise de dados ou ainda servirem como alicerce para realização de pesquisas futuras, principalmente ligadas ao GIPAG.

Neste trabalho optou-se por utilizar a Linguagem de Programação Python para realizar o processo de *Web Scraping*, uma vez que esta possui "[...] uma ampla variedade de construções sintáticas, funções de biblioteca-padrão e recursos de ambiente interativo de desenvolvimento" (SWEIGART, 2015, p. 44). Além disso, o Python possui compatibilidade com ferramentas utilizadas para realizar *Web Scraping*, em especial com as que foram estudadas para o desenvolvimento deste trabalho e que serão apresentadas no capítulo 3.

Pretende-se disponibilizar o arquivo CSV criado com os dados agrícolas na página do GIPAG. Além disto, "Mesmo os produtores, que não dispõem de meios para colher, armazenar e transportar a produção até os mercados e feiras de abastecimento, necessita de informações gerenciais sobre a matriz produtiva na qual atuam." (BITTENCOURT *et al.*, 2016, p. 18), uma vez que estas informações podem auxiliá-los na tomada de decisões quanto ao que produzir.

Corroborando com esta ideia, Luiz e Maia (2014, p. 174) afirmam que os dados disponibilizados pelo IBGE e por outros portais:

> [...] podem e devem servir de fonte para trabalhos e estudos pela comunidade científica das áreas da estatística, em especial da biometria. O calendário de atividades, em especial de plantio e de colheita, é uma informação importante para diversos atores das cadeias produtivas agrícolas. O mercado de insumos e produtos, a logística de transporte e armazenamento, a flutuação dos preços, a demanda por mão de obra, a previsão de safras, entre muitas outras atividades, sofrem influência da sazonalidade das atividades agrícolas.

Visando cumprir estes requisitos, inicialmente, realizou-se a revisão teórica sobre Web Scraping e sua relação com a Linguagem de Programação Python, e a partir desta revisão foram feitos estudos sobre as ferramentas que realizam *Web Scraping* e que são compatíveis com o Python, cabe salientar que tais ferramentas serão descritas no capítulo 3 deste trabalho.

Em seguida, com auxílio de pesquisadores da área do Agronegócio, foram identificados e desenvolveu-se o estudo dos portais dos quais se extraiu os dados para a geração da base de dados (a descrição detalhada do estudo está presente no capítulo 4). O objetivo foi analisar os sistemas, para verificar a melhor forma de extrair os dados relacionados à produção agrícola que fossem realmente relevantes. Estes dados extraídos, são importantes para a construção de uma base de dados, que contenha dados relevantes sobre a produção e comercialização agrícola e o contexto ligado a ela, sendo que com tais dados visa-se possibilitar futuras análises, sobretudo para o GIPAG, que é um grupo de pesquisa vinculado ao Colégio Politécnico da Universidade Federal de Santa Maria (UFSM).

Após o estudo dos portais, realizou-se a construção dos sistemas de extração de dados para cada site elencado inicialmente. A descrição detalhada de cada sistema de extração de dados produzido, está presente no capítulo 5, no qual também é apresentado detalhes relativos ao desempenho dos sistemas e as principais dificuldades.

Em seguida, foi definida a estrutura da base de dados desenvolvida que consiste em um conjunto de arquivos CSV.

Posteriormente, desenvolveu-se as considerações finais deste trabalho, nas quais tratou-se de abordar os conhecimentos obtidos com o desenvolvimento deste trabalho, além de expor as metas atingidas com este trabalho e as ideias para a continuidade deste trabalho

Sendo assim, o presente trabalho está estruturado conforme descrito a seguir. No Capítulo 1, é apresentada a introdução do trabalho, que contém a motivação e os objetivos do mesmo. No Capítulo 2, é tratado o referencial teórico que fundamentou o desenvolvimento deste trabalho, sobretudo no que refere-se à revisão sobre Extração da Informação e *Web Scraping*. No Capítulo 3, é explicado sobre as ferramentas estudadas que possibilitam a realização do *Web Scraping* e que são compatíveis com Python. No Capítulo 4, é retratada a análise da estrutura de busca e de retorno de informações dos sites do Sidra, Conab e do IBGE, sendo possível

assim entender o desafio de automatizar a construção das consultas e de extração dos dados. Em seguida, no Capítulo 5, é apresentado e explicado partes importantes dos sistemas de extração de cada um dos sistemas citados anteriormente. No Capítulo 6 é apresentado uma descrição detalhada das principais colunas da base de dados criada, para que se possa entender a origem destas informações e sua relação com outras colunas. Por fim, no Capítulo 7 são apresentadas as considerações finais do projeto, dando destaque aos tópicos propostos que já foram alcançados e os próximos passos que pretende-se alcançar.

2 REFERENCIAL TEÓRICO

A Extração de Informação ou *Information Extraction*, segundo Moens (2006) é um processo que consiste em coletar ou recuperar diferentes tipos de dados, sejam explicitamente declarados (como tabelas ou listas) ou implícitos (como textos corridos), estruturá-los e combiná-los de forma que estes possam formar uma estrutura convencional, que conforme Lage *et al.* (2004) pode ser um arquivo XML ou tabelas relacionais. As fontes destes dados são variáveis, as quais podem estar mal organizadas ou completamente desestruturadas. Cabe salientar que o resultado do processo de Extração de Informação é sempre um par de atributo-valor, Nome: Fulano de Tal, por exemplo. Na extração de dados na web atributo pode ser o elemento HTML e o valor pode ser o valor do elemento. Estes dados podem ser armazenados em bancos de dados de forma estruturada para uso posterior.

Com o uso destes processos, é possível processar e refinar dados obtidos para que possam ser armazenados em um local centralizado. Esses locais de armazenamento podem ser na máquina local ou baseados em nuvem ou ainda um híbrido destas duas formas.

Para realizar o processo de extração dos dados usava-se antigamente as técnicas de aprendizagem e técnicas de engenharia do conhecimento – também chamadas de abordagens baseadas em aprendizagem e baseadas em regras, respectivamente. Estas técnicas foram pensadas para desenvolver sistemas que exigem experiência humana para definir regras (por exemplo, expressões regulares) para realizar com sucesso a extração de dados, além elas requerem conhecimentos específicos de domínio por parte do usuário, pois este deve ter um grau elevado de conhecimento de programação e um domínio considerável sobre a temática em que o sistema de extração de dados irá operar.

Atualmente existem muitas outras técnicas que são implementadas, dentre as quais destacam-se, conforme Laender *et al* (2002) as Linguagens para Desenvolvimento de *Wrapper*, as técnicas com reconhecimento de HTML (*HTML-aware Tools*), técnicas baseadas em Processamento de Linguagem Natural (NLP), as técnicas de indução de *wrapper*, as técnicas baseadas em modelagem e técnicas baseadas em ontologia.

As Linguagens para Desenvolvimento de *Wrapper*, como salienta Laender *et al* (2002) foram uma das primeiras iniciativas para tentar resolver o problema de

geração de *wrappers*. Elas surgiram para ser um meio alternativo, em relação às linguagens de propósito geral da época, para o desenvolvimento dos *wrappers*

As técnicas com reconhecimento de HTML ou *HTML-aware Tools*, de acordo com Laender *et al.* (2002), são técnicas que contam com recursos estruturais inerentes dos documentos HTML para realizar a extração de dados, sendo portanto uma ferramenta que utiliza técnicas baseadas em árvores. Estas ferramentas realizam a transformação do documento HTML em uma árvore de análise sintática, que é uma representação que reflete a hierarquia de tags HTML presentes no documento, essa transformação ocorre antes do processo de extração de dados.

Já, as técnicas baseadas em Processamento de Linguagem Natural ou *Natural Language Processing* (NLP) têm sido utilizadas por diversas ferramentas para aprender regras de extração para assim poder extrair dados relevantes existentes em documentos de linguagem natural. Essas técnicas utilizam "[...] técnicas de filtragem, marcação de parte da fala e marcação semântica lexical para construir relacionamentos entre frases e elementos de frases" (LAENDER *et al*., 2002, p. 85, tradução nossa), de modo que as regras de extração possam ser derivadas. Estas regras, conforme Laender *et al.* (2002) são baseadas em restrições sintáticas e semânticas que auxiliam na identificação de informações relevantes dentro do documento. As ferramentas baseadas em NLP são geralmente mais adequadas para páginas da *Web* que consistem em texto livre, como listas de empregos, anúncios de aluguel de apartamentos, anúncios de seminários, etc. Ferramentas representativas baseadas em tal abordagem são RAPIER, SRV e WHISK.

As técnicas de indução de wrappers, segundo Laender *et al.* (2002, p. 85, tradução nossa) "[...] geram regras de extração baseadas em delimitadores derivadas de um determinado conjunto de exemplos [...]". A principal diferença destas técnicas com as técnicas baseadas em NPL, como salienta Laender *et al.* (2002) é que elas não dependem de restrições linguísticas, mas sim de recursos de formatação que delineiam implicitamente a estrutura dos dados encontrados, sendo assim, estas técnicas são mais adequadas para documentos HTML do que as técnicas anteriormente citadas. "Ferramentas como W1EN, SoftMealy e STALKER são representativas dessa abordagem" (LAENDER *et al.*, p. 85, tradução nossa).

As técnicas baseadas em modelagem funcionam, de acordo com Laender *et al.* (2002), da seguinte maneira: dada uma estrutura de destino para objetos, estas

ferramentas tentam localizar em páginas da *Web* porções de dados que estão implicitamente em conformidade com essa estrutura fornecida inicialmente. Esta estrutura é fornecida de acordo com um conjunto de primitivas de modelagem (por exemplo, tuplas, listas, etc.) que estão em conformidade com um modelo de dados subjacente. Posteriormente, algoritmos similares aos utilizados nas técnicas de indução do *wrapper* identificam objetos com a estrutura fornecida nas páginas de destino. Exemplos de implementações que adotam essa abordagem são NoDoSE e DEByE.

Por fim, as técnicas baseadas em ontologia, conforme argumenta Laender *et al.* (2002), diferentemente das técnicas anteriores não necessitam que exista uma estrutura de recursos de apresentação dos dados dentro de um documento para gerar regras ou padrões para realizar a extração, pois a extração é realizada confiando diretamente nos dados, sendo assim, dada uma aplicação de domínio específica, uma ontologia pode ser usada para localizar constantes presentes na página e construir objetos com elas. “A ferramenta baseada em ontologia mais representativa é a desenvolvida pelo *Brigham Young University Data Extraction Group*” (LAENDER *et al*., 2002, p. 85-86, tradução nossa).

2.1 *Web Scraping*

Quando o processo de Extração de Informação é focado em sistemas *Web*, este passa a ser conhecido como *Web Scraping*, mas conforme Mitchell (2018, p. 10) também pode ser conhecido como: “*Screen Scraping*, *Web Harvesting* ou variações similares”, além disso, ele pode ser conhecido como Raspagem da *Web* ou Coleta de Dados da *Web*.

Conforme Mitchell (2018), o *Web Scraping* consiste em coletar dados por qualquer meio que não envolva a interação entre um programa e a uma API ou a utilização do navegador sendo feita manualmente por um humano. Sendo assim, para que possa extrair os dados e, se necessário, processá-los e refina-los, é feita a criação de um programa que automatize todo processo de solicitar o acesso à página, requisitar dados, formular consultas, para que por fim seja possível realizar a extração dos dados da página *Web*. Um exemplo de uso de *Web Scraping* é o Google, que “[...] executa diversos programas de web scraping para indexar páginas web em sua ferramenta de pesquisa.” (SWEIGART, 2015, p. 289).

Web Scraping, como salienta Mitchell (2018, p. 11) “[...] engloba uma grande variedade de técnicas de programação e de tecnologias, por exemplo, análise de dados, parsing de idiomas naturais e segurança de informação”.

Para realizar o processo de extração de dados da *Web* são desenvolvidos programas escritos em alguma linguagem de programação. Uma destas linguagens é o Python, que “[...] tem uma ampla variedade de construções sintáticas, funções de biblioteca-padrão e recursos de ambiente interativo de desenvolvimento.” (SWEIGART, 2015, p. 44).

Além disto, Mitchell (2018, p. 13) afirma que:

> Com poucas exceções, se você puder visualizar os dados em seu navegador, será possível acessá-los por meio de um script Python. Se for possível acessá-los com um script, você poderá armazená-los em um banco de dados. E, se puder armazená-los em um banco de dados, poderá fazer praticamente de tudo com esses dados.

Com base nesta afirmação, pode-se perceber que o Python pode facilmente ser introduzido nos processos de coleta dos dados e com isto implementar a coleta dos dados desejados. Ainda conforme Mitchell (2018, p. 13), o *Web Scraping* quase sempre “[...] oferece uma forma de orientar práticas de negócios com mais eficiência, melhorar a produtividade ou até mesmo dar origem a uma área totalmente nova.”

3 FERRAMENTAS DE WEB SCRAPING

Para o desenvolvimento deste trabalho, buscou-se ferramentas que facilitassem a realização do *Web Scraping* e que fosse possível ser programada por meio da Linguagem de Programação Python. Desta busca, três ferramentas foram identificadas e serão explicadas no decorrer do trabalho: o *Scrapy*, o *Beautiful Soup* e o *Selenium*.

Após a seleção das ferramentas, iniciou-se uma busca acerca das principais vantagens e a forma de utilização de cada uma das ferramentas, em seguida, realizou-se uma comparação entre as ferramentas para que fosse possível escolher a ferramenta que seria utilizada no decorrer deste trabalho.

O resultado da pesquisa sobre as ferramentas e da comparação entre elas será apresentado nesta seção, sendo que para cada ferramenta será destinado uma subseção contendo uma breve explicação sobre a ferramenta, suas principais utilidades e alguns exemplos que foram desenvolvidos para a aprendizagem dos conceitos fundamentais das ferramentas. Estes exemplos terão sua lógica e objetivo explicados, além de terem uma imagem do código.

Por fim, será apresentada uma seção destinada à comparação entre as vantagens e desvantagens das ferramentas analisadas e trabalhadas nesta monografia.

3.1 *SCRAPY*

A primeira ferramenta que foi estudada e usada foi o *Scrapy*[2] que é uma *framework* de código aberto e de fácil portabilidade para os diferentes sistemas operacionais. Ele foi desenvolvido originalmente para auxiliar no *Web Scraping*, mas também pode ser usado para outras tarefas como extrair dados usando APIs ou rastrear o uso geral na *Web*.

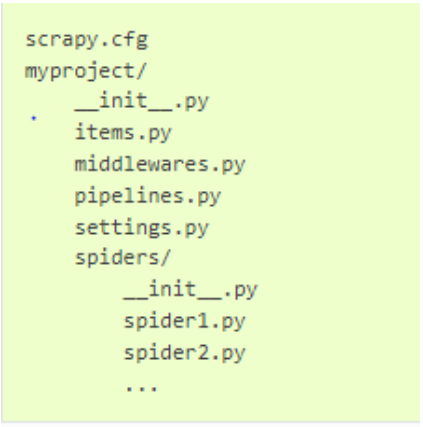
Com o *Scrapy* é possível realizar diferentes requisições, pois o *Scrapy* realiza o processamento da requisição de maneira assíncrona, isto significa que o *Scrapy* consegue iniciar outras requisições, mesmo que a requisição atual não tenha sido finalizada.

[2] https://docs.scrapy.org/en/latest/

O *Scrapy* possui uma variedade de suportes, tais como: suportes a extensão (o que permite ao desenvolver conectar outras funcionalidades), suporte para a geração da exportação de feeds em diversos formatos e suporte integrado para seleção e extração dos dados da *Web*, por meio do uso de seletores CSS estendidos e expressões XPath.

Outra característica do *Scrapy* é a estrutura de projeto padrão do *Scrapy* mostrada na Figura 1.

Figura 1: Estrutura padrão do *Scrapy*



Fonte: https://docs.scrapy.org/en/latest/topics/commands.html

Cada arquivo desta estrutura tem uma função específica, como por exemplo, o arquivo scrapy.cfg, que possui as configurações que serão carregadas no momento do *deploy*. Os demais arquivos listados anteriormente e suas funções estão listados a seguir:

- Items.py possui os modelos dos dados que deseja-se extrair da página Web;
- pipelines.py possui as classes executadas em sequência;
- settings.py possui as configurações gerais do projeto, tais como, o nome dos spiders;
- Cada arquivo spider.py possui o código que realiza a extração dos dados.

Com base nesta estrutura, a seguir serão apresentados dois códigos exemplos, nos quais se utilizou o *Scrapy* para realizar a extração dos dados. Nestes exemplos será mostrado somente o arquivo spider.py pois é neste arquivo está a parte responsável por extrair os dados.

3.1.1 Exemplo 1 de Algoritmo com *Scrapy*

O primeiro exemplo tem como objetivo acessar o site oficial do Grêmio (mostrado na Figura 2) e extrair o título e a descrição de notícias colocadas em destaque na página principal do site. A Figura 3, mostra o código do arquivo spider.py que é responsável por realizar esta extração.

Figura 2: Site do Grêmio antes da execução da extração dos dados

Fonte: O autor

Figura 3: Código que extrai as notícias em destaque no site do Grêmio

```
1   import scrapy
2
3   class GremioSpider(scrapy.Spider):
4       name = 'gremio'
5       start_urls = ['https://gremio.net/']
6
7       def parse(self, response):
8
9           for noticia in response.css('div.caption-text'):
10              yield {
11                  'Titulo': noticia.xpath('h1/text()').extract(),
12                  'Descricao': noticia.xpath('h2/text()').extract()
13
14              }
```

Fonte: O autor

Neste arquivo, primeiramente é realizado a importação da biblioteca *Scrapy*, em seguida é criada a classe que recebe como parâmetro um *Spider* que é uma sub biblioteca de *Scrapy*. Dentro da classe são criados dois atributos: o primeiro é o name, que é o nome do *spider* e o segundo é o *starts_urls*, que é o atributo que contém um *array* com as urls de onde serão extraídos os dados.

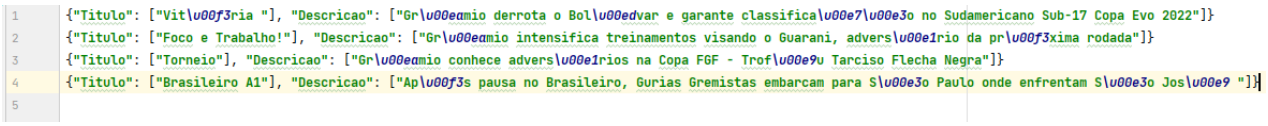
Posteriormente, é criado uma função obrigatória derivada da sub biblioteca *Spider*, que recebe como parâmetros o *self* e o *response*. O primeiro atributo é um atributo que é recebido automaticamente é o que aponta para a própria classe,

enquanto que o segundo atributo é a resposta da página *Web* colocada no *starts_urls*.

Neste exemplo, para extrair os dados, cria-se inicialmente uma estrutura de repetição (linhas 09 até 14, na Figura 3) que busca e estrutura pares atributo-valor entre os elementos HTML que são divs e que tenham a classe *caption-text*, juntamente dos elementos HTML presentes dentro destas divs.

Em seguida, dentro da estrutura de repetição, serão extraídos os elementos correspondentes às informações desejadas dos pares atributo-valor montados e montar o retorno que será armazenado em um arquivo com extensão txt.

Figura 4: Resultado da consulta realizada dia 03 de agosto de 2022



Fonte: O autor

A Figura 4 mostra o retorno da extração das notícias em destaque na página principal do site oficial do Grêmio, realizada no dia 03 agosto de 2022, cabe salientar que cada linha da Figura 4 corresponde a uma notícia em destaque.

3.1.2 Exemplo 2 de Algoritmo com *Scrapy*

Este segundo exemplo tem o objetivo de acessar o site Quotes to Scrape (mostrado na Figura 5), que está disponível pelo link https://quotes.toscrape.com/tag/humor/. Este site possui uma estrutura simples e visa servir como um site para realizar o processo de extração de diversas citações que estão listadas nele. Este exemplo pode ser encontrado na documentação do *Scrapy*.

Neste exemplo, para extrair os dados, cria-se inicialmente uma estrutura de repetição (linhas 10 até 14, na Figura 6) que busca e estrutura pares atributo-valor entre os elementos HTML que são divs e que tenham a classe *quote*, juntamente dos elementos HTML presentes dentro destas divs.

Figura 5: Imagem do site antes do processo de extração

Fonte: O autor

Figura 6: Código que extrai as principais citações

```python
import scrapy

class QuotesSpider(scrapy.Spider):
    name = 'quotes'
    start_urls = [
        'https://quotes.toscrape.com/tag/humor/',
    ]

    def parse(self, response):
        for quote in response.css('div.quote'):
            yield {
                'author': quote.xpath('span/small/text()').get(),
                'text': quote.css('span.text::text').get(),
            }

        next_page = response.css('li.next a::attr("href")').get()
        if next_page is not None:
            yield response.follow(next_page, self.parse)
```

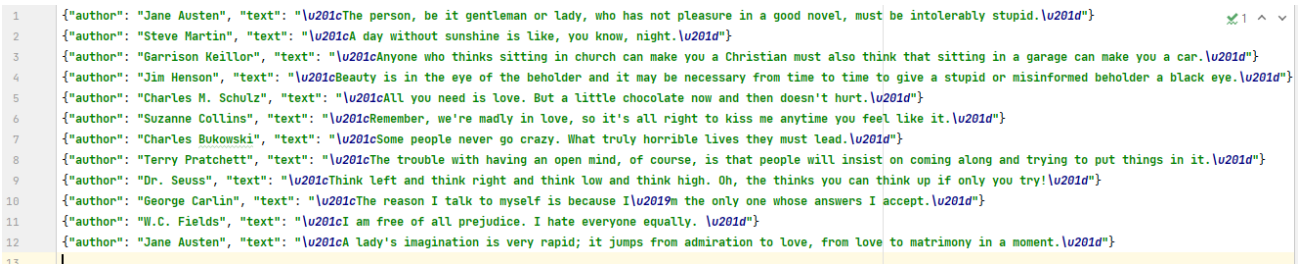
Fonte: O autor

Assim como no primeiro exemplo, este programa contém a estrutura básica de um *Spider*, que é a importação da biblioteca *Scrapy*, em seguida a criação da classe que recebe como parâmetro a sub biblioteca *Spider* e que possui os atributos name e *starts_urls*, além de possuir o método parse.

O diferencial deste exemplo é que ele busca todas as citações da página para em seguida identificar e realizar o *click* no botão *next* que redireciona para a próxima página que contém mais algumas citações. Nesta nova página é realizado a extração das citações e é verificado se existe outro botão *next*, caso não exista o fluxo do programa é encerrado.

O resultado da extração das citações é mostrado na Figura 6, onde cada linha corresponde a uma citação presente na página indicada no código da Figura 5 na linha 6.

Figura 7: Resultado da extração das citações



Fonte: O autor

3.2 *BEAUTIFUL SOUP*

A segunda ferramenta estudada no decorrer deste trabalho foi o *Beautiful Soup*[3], que é uma ferramenta utilizada para fazer *Web Scraping*, que auxilia na extração dos dados, pois possibilita uma aprendizagem rápida e fácil, uma vez que seus métodos são de fácil entendimento. Além disto, possui uma boa e abrangente documentação e uma comunidade ativa, o que auxilia no processo de aprendizagem e solução de dúvidas que venham a surgir.

Entretanto, o uso do *Beautiful Soup* possui uma desvantagem que é o fato dele possuir dependência de várias bibliotecas, uma vez que para se fazer uso de *Beautiful Soup*, são necessárias três bibliotecas no mínimo: a biblioteca *urllib*[4], a biblioteca do analisador externo e a própria biblioteca do *Beautiful Soup*.

3.2.1 Exemplo 1 de Algoritmo com *Beautiful Soup*

O primeiro exemplo tem por objetivo extrair os nomes dos estados do Brasil de uma página *Web* do Wikipédia (que pode ser acessada pelo link https://pt.wikipedia.org/wiki/Lista_de_capitais_do_Brasil e que pode ser visualizado na Figura 8), que possui uma estrutura simples, o que facilita o processo de extração das informações. Nesta página, as capitais estão listadas em um elemento “ul”, onde

[3] https://beautiful-soup-4.readthedocs.io/en/latest/
[4] https://docs.python.org/3/library/urllib.html

cada estado corresponde a um elemento “li” que possui um elemento “a” que referencia uma seção da página.

Neste exemplo, o código de extração de dados cria um par atributo-valor entre uma variável e uma lista que contém todos os “li” e que tenham a classe “tocleve-2”.

Figura 8: Imagem da página da Wikipédia

Fonte: O autor

Figura 9: Código que extrai os estados do Brasil

```
1  import urllib.request
2  from bs4 import BeautifulSoup
3  import html5lib
4
5  wiki = "https://pt.wikipedia.org/wiki/Lista_de_capitais_do_Brasil"
6  page = urllib.request.urlopen(wiki)
7  soup = BeautifulSoup(page, 'html5lib')
8  list = soup.find_all('li', attrs={'class': 'toclevel-2'})
9  for list_item in list[3:30]:
10     print(list_item.text.replace("2.",""))
```

Fonte: O autor

Neste código, primeiramente é realizado a importação das bibliotecas (linhas 1 até 3 da Figura 9) necessárias para a realização a extração das informações da página *Web*, sendo que dentre as bibliotecas importadas, a *urllib.request*[5] é responsável por possibilitar o tratamento das respostas da página *Web*; a biblioteca html5lib[6] contém um analisador externo necessário para analisar a resposta da página *Web*, enquanto que a biblioteca *Beautiful Soup* contém os métodos para procurar elementos na resposta da página já convertida.

[5] https://docs.python.org/3/library/urllib.html
[6] https://pypi.org/project/html5lib/

Após a realização das importações, é criada uma variável onde é armazenado o endereço da página que deseja-se acessar e extrair os dados (linha 5 da Figura 9), em seguida realiza-se a chamado do método *urlopen* e se envia por parâmetro a variável que contém a url da página (linha 6 da Figura 9).

Posteriormente, cria-se uma variável que receberá o retorno da análise da resposta da página *Web* feita pelo analisador externo, que neste exemplo foi o *html5* (linha 7 da Figura 9). É importante salientar que é a partir desta variável que se realiza as buscas de elementos da página e consecutivamente a extração das informações, neste exemplo esta variável é chamada de "*soup*".

Depois de analisada a resposta da página, é realizada a busca por todos os elementos "li" que possuam como atributo a classe "toclevel-2" (linha 8 na Figura 9) que é a classe presente no código da página HTML, em seguida é feito um laço de repetição para extrair o texto destes elementos e formatá-los retirando "2." do seu início (linhas 9 e 10 da Figura 9).

Figura 10: Resultado da extração dos estados brasileiros

```
1 Acre
2 Alagoas
3 Amapá
4 Amazonas
5 Bahia
6 Ceará
7 Distrito Federal
8 Espírito Santo
9 Goiás
10 Maranhão
11 Mato Grosso
12 Mato Grosso do Sul
13 Minas Gerais
14 Pará
15 Paraíba
16 Paraná
17 Pernambuco
18 Piauí
19 Rio Grande do Norte
20 Rio Grande do Sul
21 Rio de Janeiro
22 Rondônia
23 Roraima
24 Santa Catarina
25 São Paulo
26 Sergipe
27 Tocantins
```

Fonte: O autor

A Figura 10 mostra o resultado, apresentado no terminal do código de extração (apresentado na Figura 9). Neste retorno, são apresentados todos os estados brasileiros em ordem alfabética, com texto formatado para não apresentar o trecho "2." no início do nome do estado.

3.2.2 Exemplo 2 de Algoritmo com *Beautiful Soup*

Neste segundo exemplo, o objetivo do programa é extrair o nome dos artistas mais ouvidos no Spotify, segundo informações da matéria presente no G1 (que pode ser visualizado na Figura 11 e acessado através do link https://g1.globo.com/pop-arte/musica/noticia/2021/12/01/sertanejo-domina-lista-de-mais-ouvidos-do-spotify-em-2021.ghtml), e apresentá-los em formato diversificado, para tanto será usado neste exemplo outra biblioteca em complemento ao *Beautiful Soup*, que é o Pandas[7]. O Pandas é uma biblioteca que fornece métodos para análise e manipulação de dados de código aberto, de forma rápida, flexível e fácil de usar. Ainda é importante enfatizar que a biblioteca é toda construída sobre a linguagem de programação Python, além disso, cabe salientar que os nomes desses artistas estão apresentados em um formato de lista de elementos na página *Web*.

Figura 11: Site do G1

Fonte: O autor

Figura 12: Código que extrai o nome dos cantores

```
import pandas as pd
import urllib.request
from bs4 import BeautifulSoup
import html5lib

url_teste = "https://g1.globo.com/pop-arte/musica/noticia/2021/12/01/" \
            "sertanejo-domina-lista-de-mais-ouvidos-do-spotify-em-2021.ghtml"
page = urllib.request.urlopen(url_teste)
soup = BeautifulSoup(page, 'html5lib')
ul = soup.find('ul', attrs={'class': 'content-unordered-list'})
list = ul.findChildren('li')
A = []

for list_item in list:
    #print(list_item.text)
    A.append(list_item.text)

df = pd.DataFrame({'Artista': A})
print(df)
```

Fonte: O autor

[7] https://pandas.pydata.org/docs/index.html

Para extrair os nomes bastou buscar o elemento “ul” correspondente a lista e em seguida percorrer toda a lista e extrair os nomes dos artistas. Para tanto, foi utilizado o método “find” para buscar a lista (linha 10 na Figura 12), o qual criou um par atributo-valor entre variável “ul” e os elementos HTML que compõe a lista, em seguida, com o método “findChildren” montou-se uma lista contendo todos os elementos “li” daquela lista (linha 11 na Figura 12), que criou um par atributo-valor para cada elemento “li”.

Neste exemplo foi usado a biblioteca Pandas para exibir o resultado da extração em uma tabela com uma coluna somente. Para usar o Pandas, inicialmente foi realizado a importação da biblioteca pandas (linha 1 na Figura 12), posteriormente, foi criado um *Data Frame* (linha 18 na Figura 12) para exibir os nomes dos artistas e no final do programa foi realizada a exibição do *Data Frame* na saída padrão da IDE (linha 19 na Figura 12).

Na Figura 13 será apresentado o resultado da extração dos nomes dos artistas da página do G1 no formato de uma tabela com uma coluna somente, sendo que nesta coluna será colocado o nome dos artistas. Esta tabela, como já citado anteriormente, é construída a partir da utilização da biblioteca Pandas que possui diversas estruturas de dados, entre elas uma tabela.

Figura 13: Resultado da extração dos nomes dos artistas

	Artista
0	Os Barões Da Pisadinha
1	Gusttavo Lima
2	Marília Mendonça
3	Jorge & Mateus
4	Henrique & Juliano

Fonte: O autor

3.3 *SELENIUM*

Por fim, a terceira ferramenta estudada e que será utilizada para restante deste trabalho foi o *Selenium*[8][9]. O *Selenium* foi desenvolvido para construção de testes automatizados para aplicações *Web*, porém ele também é utilizado para o

[8] https://www.selenium.dev/
[9] https://selenium-python.readthedocs.io/

Web Scraping. Além disso, ele pode ser desenvolvido em várias linguagens de programação populares, como C#, Java, Python, Ruby, etc.

O *Selenium* possui um funcionamento amigável para quem está iniciando na automatização de testes e no *Web Scraping*, o que torna mais fácil a construção de códigos mais complexos. Outro ponto importante do *Selenium* é que ele pode trabalhar facilmente com conceitos básicos de Javascript (DOM) e pode lidar facilmente com solicitações AJAX e PJAX.

A principal vantagem do *Selenium* em relação a outras ferramentas estudadas é que ele possui mecanismos para construção das consultas nas interfaces Web que irão gerar os dados que serão armazenadas na base de dados.

3.3.1 Exemplo 1 de Algoritmo com *Selenium*

O primeiro exemplo tem como objetivo encontrar um elemento input em uma página e inserir um texto nele que corresponde a um argumento de uma consulta que irá gerar uma página HTML com os dados resultantes da consulta. Por meio desta ação é gerado uma atualização dos destaques na página principal. A página usada inicialmente nos testes foi https://www.walissonsilva.com/blog (mostrada na Figura 15), onde constam dados sobre cursos desenvolvidos pelo autor e ao realizar uma pesquisa com a palavra “data” são retornados os dados mostrados na Figura 16. Para efetuar esta ação o código de extração cria uma relação atributo-valor entre a variável que recebe o elemento *input* e as características do próprio elemento *input*.

Figura 14: Código que insere texto no elemento HTML *input*

```
1  from selenium import webdriver
2  from time import sleep
3
4  options = webdriver.ChromeOptions()
5  options.add_argument('--ignore-certificate-errors')
6  #options.add_argument('--incognito')
7  options.add_argument('--no-sandbox')
8  options.add_argument('--disable-dev-shm-usage')
9  navegador = webdriver.Chrome("chromedriver", options=options)
10 navegador.get("https://www.walissonsilva.com/blog")
11 sleep(3)
12 elemento = navegador.find_element_by_tag_name('input')
13 print(elemento)
14 elemento.send_keys('data')
```

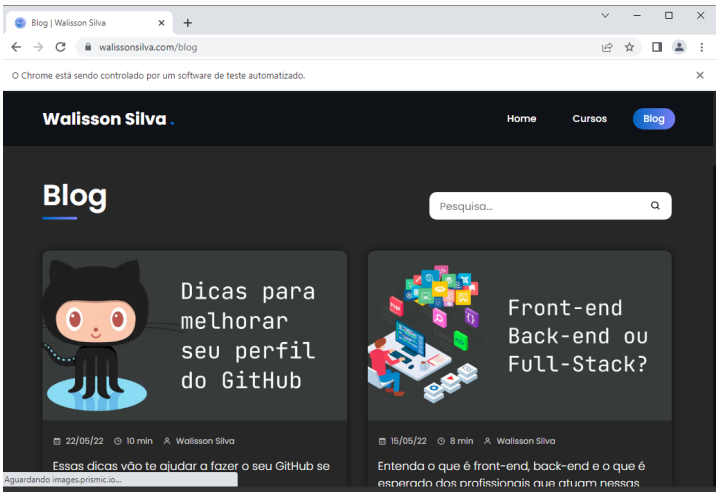
Fonte: O autor

Neste código, inicialmente é feita a importação das bibliotecas necessárias para a execução do programa, em seguida é iniciada uma janela do navegador escolhido para se trabalhar com o *Web Scraping*. Cabe salientar que deve-se colocar o driver correspondente ao navegador escolhido juntamente dos arquivos do programa para que o código possa ser executado.

Posteriormente é realizado o acesso a página escolhida e é chamado o método sleep que força uma espera para garantir que o navegador carregue corretamente a página desejada. Após o carregamento, é realizado a busca do elemento *input* através do método "*find_element_by_tag_name*" que retorna o elemento, em seguida é realizado a exibição do elemento *input* na saída padrão da *Integrated Development Environment* (IDE) ou Ambiente de Desenvolvimento Integrado e por fim é feito a inserção do texto no elemento *input*.

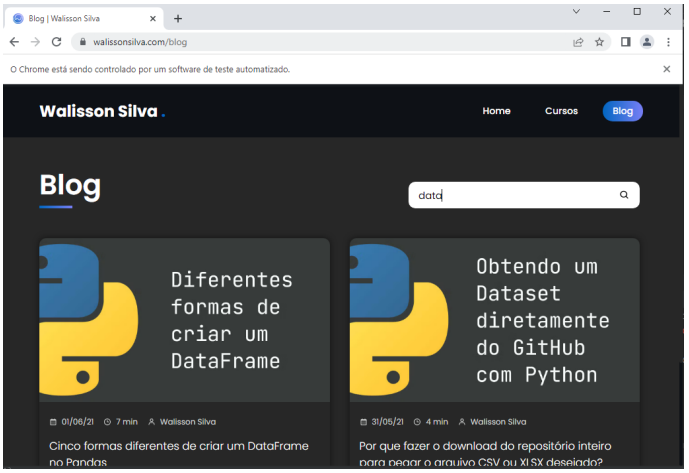
As Figuras 15 e 16, mostram um navegador que é aberto automaticamente ao se chamar a função *Chrome*() (linha 9 na Figura 14). Na Figura 14, o navegador mostra a página que se deseja abrir (linha 10 na Figura 14), em seguida na Figura 15 aparece a página acessada, sem nenhuma alteração feita pelo programa. Já a Figura 16, mostra o resultado da inserção da palavra "data" no campo *input* da página, pois a página realiza a busca de todas as matérias que possuem "data" no nome e o resultado da busca é retornado fazendo com que as matérias encontradas sejam colocadas somente elas exibidas na página. Esta ação de destacar as matérias encontradas ocorre sem a necessidade do sistema simular que o usuário houvesse pressionado o botão "*enter*" do teclado.

Figura 15: Navegador aberto com *Web Driver* na página indicada



Fonte: O autor

Figura 16: Resultado da inserção de valor no campo *input*

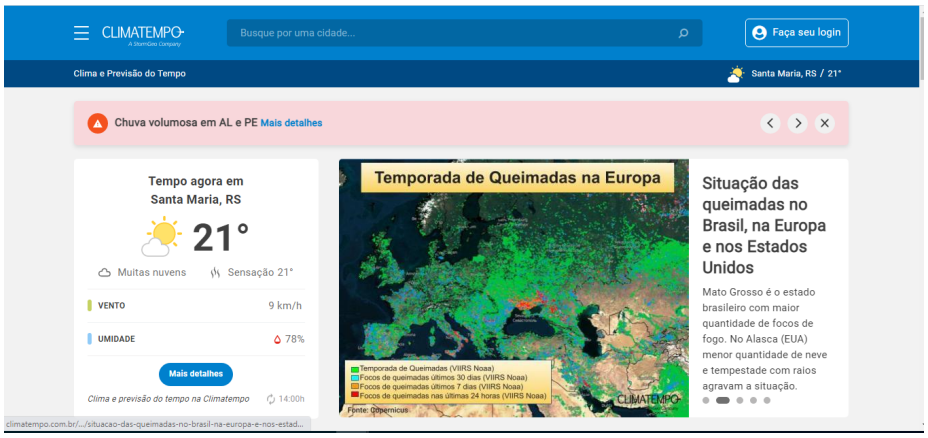


Fonte: O autor

3.3.2 Exemplo 2 de Algoritmo com *Selenium*

O objetivo deste segundo exemplo é consultar e extrair as condições climáticas de determinadas cidades no site da ClimaTempo (que pode ser acessado pelo link: https://www.climatempo.com.br/ e pode ser visualizado na Figura 17). Para tanto, foi criada uma classe separada, na qual foi colocado o método que realiza o acesso ao site do ClimaTempo e extrai os dados climáticos de cada cidade informada. Nesta classe são criadas relações de atributo-valor entre as variáveis que recebem os elementos HTML que exibem os dados climáticos e os valores dos próprios dados climáticos.

Figura 17: Site do Clima Tempo



Fonte: O autor

Figura 18: Código que extrai informações climáticas - Parte 1

```
1  from time import sleep
2  from selenium import webdriver
3  from selenium.webdriver.common.by import By
4
5  class newbot:
6
7      def __init__(self, nome_bot):
8          self.driver = webdriver.Chrome("chromedriver")
9
10     def climaTempo(self, cidades = []):
11
12         try:
13             for cidade in cidades:
14                 site = 'https://www.climatempo.com.br'
15                 self.driver.get(site)
16                 sleep(5)
17                 self.driver.find_element_by_xpath('//*[@id="bt_modalSearch_mobile"]/i').click()
18                 elemento = self.driver.find_element_by_id("searchGeneralMobile")
19                 print('****  ' + cidade + '  ****')
20
21                 for letra in cidade:
22                     elemento.send_keys(letra)
23                     sleep(0.5)
24
25                 self.driver.implicitly_wait(3)
```

Fonte: O autor

A Figura 18 mostra o início do código da classe *bott*. Na Figura 18 é mostrado a importação das bibliotecas (linhas 1 até 3 na Figura 18), a declaração da classe *newbott* (linha 5 na Figura 18) e declaração dos métodos __init__ e climaTempo. O método __init__ (linha 7 na Figura 18) é o método construtor da classe *newbott* que recebe por parâmetro o *self*, que aponta para a própria classe, e o nome do bott. Este método realiza a abertura do navegador escolhido. O método climaTempo recebe por parâmetro o *self*, que aponta para a própria classe, e um *array* de *Strings* com os nomes das cidades que deseja-se extrair as condições climáticas.

O método climaTempo possui um bloco *try…except* que permite verificar e tratar erros que ocorrem dentro do bloco *try*. Estes erros são tratados no bloco *except* (linhas 55 até 57 na Figura 20). Este bloco é utilizado neste exemplo, pois podem ocorrer erros em relação ao carregamento das páginas em virtude da conexão de *Internet*. Além disto, para tentar minimizar a ocorrência destes erros, é utilizado as funções *sleep* e *implicitly_wait* que promovem uma espera de alguns segundos para que o carregamento da página ocorra corretamente. Em virtude dos possíveis problemas de carregamento da página, também foi criado um bloco de repetição (linhas 21 até 23 na Figura 18) que realiza a simulação da digitação caracter por caracter dos nomes das cidades que se deseja extrair as informações e isto ocorre de maneira similar à que seria feita por um usuário normal, o que facilita o carregamento do *link* da página que contém as informações climáticas da cidade.

Figura 19: Código que extrai informações climáticas - Parte 2

```
26          li = self.driver.find_element_by_xpath('//*[@id="searchGeneralMobile_autocomplete"]/li[2]/a')
27          print(li.get_attribute('href'))
28          self.driver.get(li.get_attribute('href'))
29          sleep(6)
30          temperatura_minima = self.driver.find_element_by_xpath('//*[@id="min-temp-1"]').text
31          temperatura_maxima = self.driver.find_element_by_xpath('//*[@id="max-temp-1"]').text
32          chuva = self.driver.find_element_by_xpath(
33              '//*[@id="mainContent"]/div[4]/div[5]/div[1]/div[2]/div[2]/div[2]/div[1]/ul/li[2]/div/span').text\
34              .split('- ')
35          quantidade_chuva = chuva[0]
36          probabilidade_chuva = chuva[1]
37          vento = self.driver.find_element_by_xpath(
38              '//*[@id="mainContent"]/div[4]/div[5]/div[1]/div[2]/div[2]/div[2]/div[1]/ul/li[3]/div').text\
39              .split("-")
40          umidade_minima = self.driver.find_element_by_xpath(
41              '//*[@id="mainContent"]/div[4]/div[5]/div[1]/div[2]/div[2]/div[2]/div[1]/ul/li[4]/div/p/span[1]')\
42              .text
43          umidade_maxima = self.driver.find_element_by_xpath(
44              '//*[@id="mainContent"]/div[4]/div[5]/div[1]/div[2]/div[2]/div[2]/div[1]/ul/li[4]/div/p/span[2]')\
45              .text
46          dados = 'Temperatura Mínima: ' + temperatura_minima + '\n' \
47                  'Temperatura Máxima: ' + temperatura_maxima + '\n' \
48                  'Chuva (Probabilade): ' + quantidade_chuva + '(' + probabilidade_chuva + ')\n' \
49                  'Vento: ' + vento[1] + '\n' \
50                  'Umidade Mínima: ' + umidade_minima + '\n' \
51                  'Umidade Máxima: ' + umidade_maxima
```

Fonte: O autor

A página principal do site do ClimaTempo possui as informações climáticas da cidade na qual a localização do usuário está indicando, porém é possível acessar as informações do tempo de outras cidades por meio do campo de pesquisa. Quando o navegador não está maximizado, dependendo do tamanho ocupado por ele, este campo pode estar ocultado, ou seja, não estar visível normalmente. Para ele se tornar visível nestes casos, é necessário clicar no botão de pesquisa (botão com o ícone da lupa). A Figura 18 mostra o código que realiza a ação de busca e *click* no botão de pesquisa (linha 17 na Figura 18).

Após encontrar o campo, realiza-se a preenchimento, caracter por caracter (como mostrado na Figura 23), como já informado anteriormente (linhas 21 até 23 na Figura 18), em seguida é carregada uma lista com as cidades que possuem o nome digitado. Nesta lista, o primeiro elemento “li” é um título para lista, por isso que busca-se o segundo “li” (linha 26 na Figura 19) da lista para se obter o *link* da página com as informações climáticas da cidade desejada. Posteriormente é acessado a página do *link* obtido (linha 28 na Figura 19) e dessa página são extraídos os dados climáticos e é montada uma *String* com todos os dados da cidade para exibir estes dados na saída padrão da IDE (como mostrado na Figura 22).

Figura 20: Código que extrai informações climáticas - Parte 3

```
52                print(dados)
53                print('----------------------------------------------------------------------------------------------------')
54
55            except:
56                self.driver.close()
57                self.erro()
58
59
60        def erro(self):
61            self.climaTempo()
```

Fonte: O autor

A Figura 20 mostra o bloco *except* (linhas 55 até a 57 na Figura 20) que será executado em caso de erro no bloco *try*, além de mostrar o método erro (linhas 60 até a 61 na Figura 20) que é chamado dentro do bloco *except*.

Figura 21: Código que inicializa a classe que extrai os dados

```
1  import re
2  from bott import newbot
3
4  bott = newbot('roboClimaTempo')
5  bott.climaTempo(["Porto Alegre","Rio de Janeiro","Belo Horizonte"])
```

Fonte: O autor

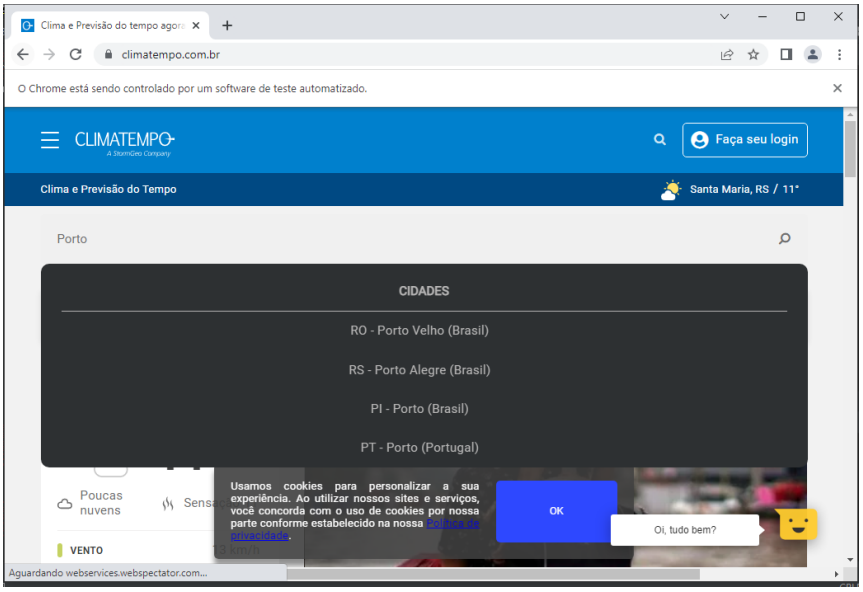
A Figura 21 exibe o código do arquivo main que cria um objeto da classe *newbott* e a inicializa (linha 4 na Figura 21), para assim poder acessar o método climaTempo que realiza o acesso ao site e a extração dos dados.

Figura 22: Resultado da extração dos dados climáticos

```
****  Porto Alegre  ****
https://www.climatempo.com.br/previsao-do-tempo/cidade/363/portoalegre-rs
Temperatura Mínima: 16°
Temperatura Máxima: 24°
Chuva (Probabilade): 0mm (0%)
Vento:  8km/h
Umidade Mínima: 45%
Umidade Máxima: 84%
----------------------------------------------------------------------------------------------------
****  Rio de Janeiro  ****
https://www.climatempo.com.br/previsao-do-tempo/cidade/321/riodejaneiro-rj
Temperatura Mínima: 15°
Temperatura Máxima: 29°
Chuva (Probabilade): 0mm (0%)
Vento:  15km/h
Umidade Mínima: 56%
Umidade Máxima: 70%
----------------------------------------------------------------------------------------------------
****  Belo Horizonte  ****
https://www.climatempo.com.br/previsao-do-tempo/cidade/107/belohorizonte-mg
Temperatura Mínima: 12°
Temperatura Máxima: 26°
Chuva (Probabilade): 0mm (0%)
Vento:  15km/h
Umidade Mínima: 37%
Umidade Máxima: 73%
----------------------------------------------------------------------------------------------------
```

Fonte: O autor

Figura 23: Execução da digitação caracter por caracter



Fonte: O autor

3.4 COMPARATIVO ENTRE AS FERRAMENTAS

Primeiramente, cabe salientar que todas as ferramentas mencionadas anteriormente são gratuitas e *open source*, o que significa que não é necessário pagar para usá-las e elas podem sempre serem incrementadas com novas funcionalidades criadas pelo próprio usuário.

Dentre as três ferramentas o *Scrapy* é o mais rápido, além de possuir um pacote completo com ferramentas necessárias para coletar os dados das páginas da *Web*, para fazer download de dados e analisá-los e para armazená-los no formato de sua escolha. Entretanto sua estrutura de arquivos possui uma maior complexidade, pelo fato de possuir uma estrutura de arquivos mais complexa, pois cada um tem uma função específica, requer um estudo mais aprofundado da ferramenta se comparada ao *Selenium* e ao *Beautiful Soup*, que com apenas um arquivo são de extrair os dados desejados, enquanto que *Scrapy* trabalha com uma divisão de tarefas entre os arquivos, o que requer maior estudo sobre o que cada arquivo dentro da estrutura do *Scrapy* é responsável, sendo assim não é o mais indicado para quem está iniciando no processo do *Web Scraping*.

O *Scrapy* possui suporte a *middleware* e extensões, pois foi feito para ser extensível. Além disso, o *Scrapy* permite adicionar seus proxies, manipular *cookies* e sessões e controlar a profundidade do rastreamento de uma maneira simples, o que

não é possível com *Selenium* e com *Beautiful Soup*, pois apesar de ambos permitirem a adição proxies, esta tarefa não é tão simples quanto no *Scrapy*. Outra vantagem do *Scrapy* é que ele funciona de forma assíncrona, além de usar menos memória e armazenamento da CPU.

O *Beautiful Soup*, como já citado anteriormente, não possui uma estrutura de arquivos complexa, como o *Scrapy*, o que o torna mais fácil de ser implementado, sendo assim é o mais indicado para quem está iniciando na extração de dados da *Web*. Possui uma comunidade online com várias soluções para diferentes problemas que pode enfrentar durante o uso desta biblioteca. Embora o *Beautiful Soup* permita solicitações paralelas, não é fácil de configurar e ainda assim perde quando comparado ao *Scrapy* em termos de velocidade. Se a tarefa de *Web Scraping* for pequena e não exija grande extração de dados, o *Beautiful Soup* pode ser a melhor escolha, pois não exige tanto tempo de estudo para entender seu funcionamento. Além disso, a documentação do *Beautiful Soup* é abrangente, o que permite um aprendizado mais rápido e fácil.

Por fim, o *Selenium* que dentre os três, não é nem o mais rápido, nem o mais lento, e assim como o *Beautiful Soup* não possui estrutura complexa para serem utilizados. Sua principal vantagem em relação aos demais é que ele possui mecanismos para executar *Web Scraping* mesmo se o conteúdo estiver em elementos JavaScript, além de lidar facilmente com solicitações AJAX e PJAX. O *Selenium* é bastante eficaz e pode lidar com tarefas em boa medida, além de poder executá-lo em diferentes linguagens de programação tais como Java, Python, node.js e Ruby. Outra vantagem é que pode-se usar o Selenium para controlar todos os principais navegadores da *Web*, como Chrome, Internet Explorer e Firefox. Isso é um acréscimo aos tópicos de longa data de problemas e soluções relacionados à biblioteca.

Entretanto, o Selenium por controlar navegadores headless, requer muitos recursos, o que reduz sua eficiência. Outra desvantagem do Selenium é que ele possui dificuldades em contornar 2FA, download de arquivos, teste de desempenho, link spidering, dependência de teste e códigos de resposta HTTP.

4 ESTUDO DOS PORTAIS

Visando a construção de uma base de dados foram elencados e estudados alguns sites que possuem dados importantes para o contexto agrícola, como por exemplo, dados relativos à produção em determinado local, a comercialização de determinado produto ou ainda informações relativas a população de um município qualquer.

Para tanto, foram realizadas algumas reuniões com professores do GIPAG para definir alguns sites através dos quais seria possível obter informações relevantes. A partir destas reuniões, definiu-se que a primeira versão da base de dados seria composta pelos dados presentes nos seguintes sites: Sistema IBGE de Recuperação Automática (SIDRA), Conab e IBGE Censo Demográfico de 2010.

No decorrer deste capítulo serão descritos detalhadamente os estudos de cada site citado anteriormente, com foco na apresentação da estrutura de consulta e de retorno dos dados.

4.1 SISTEMA IBGE DE RECUPERAÇÃO AUTOMÁTICA (SIDRA)

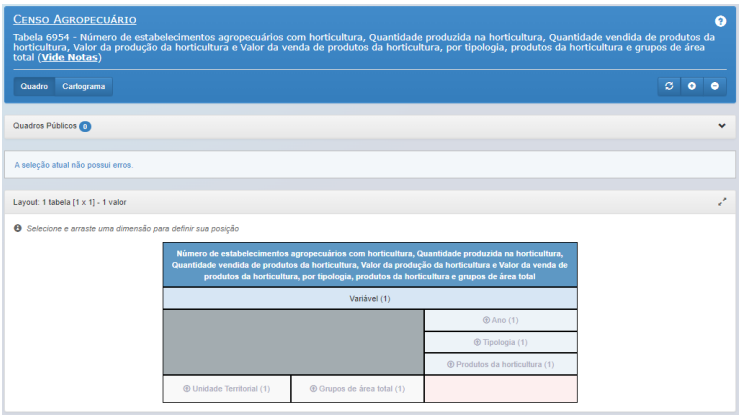
O primeiro site estudado foi o Sistema IBGE de Recuperação Automática (SIDRA), que contém dados da produção agrícola do Brasil. Este sistema possui dados de diversas pesquisas ligadas ao IBGE, como por exemplo, os dados do Censo Agropecuário de 2017, que estão divididos em 137 tabelas (visualizações) diferentes, onde cada tabela corresponde a uma página distinta, porém com uma interface de consulta similar para elas. Estas tabelas (visualizações) são identificadas por números inteiros, como mostrado na Figura 24. Cabe ressaltar que tabela aqui tem o sentido dado dentro do sistema, que também é usado em softwares de BI, onde tabela é um tipo de visualização (outros exemplos são gráficos de pizza, de barras, etc.) não devendo ser confundido com tabelas/relações dos Bancos de Dados Relacionais. Assim, as tabelas correspondem a visualizações distintas dos dados.

Para este trabalho buscou-se entender os dados do Censo Agropecuário 2017, para que fosse possível extrair-se, das diversas tabelas existentes no site, os dados relativos à produção agrícola. A partir de uma análise desta divisão dos

dados, foi possível constatar que o SIDRA, possui normalmente um par de tabelas, que estão numeradas em sequência e que possuem as mesmas informações, porém com um filtro de detalhamento diferente destas informações, por exemplo, tanto a tabela 6954 e 6953 possuem os mesmos dados, entretanto a tabela 6953 possui a possibilidade de filtrar esses dados pela condição do produtor em relação às terras e/ou grupo de atividade econômica, enquanto a tabela 6954 permite a filtragem dos dados pelo grupo de área total.

Nesta primeira versão da base de dados, buscou-se focar na extração dos produtos ligados a horticultura, sendo assim, foi decidido empenhar-se inicialmente na extração dados presentes na tabela 6954, que está disponível no link https://sidra.ibge.gov.br/tabela/6954 e está representada nas Figuras 24, 25, 26 e 27 (esta divisão em várias figuras foi feita pela impossibilidade de colocar em uma única figura todo conteúdo mostrado na página da tabela, a Figura 28 mostra toda página, mas a imagem não permite visualizar os dados). Esta decisão ocorreu, pois esta tabela possui dados importantes ligados ao número de estabelecimentos agropecuários com horticultura, a quantidade produzida na horticultura, a quantidade vendida de produtos da horticultura, o valor da produção da horticultura e o valor da venda de produtos da horticultura, além de permitir a filtragem desses dados pela tipologia (tipologia é uma variável que classifica a propriedade conforme a sua ligação com programas de incentivo e/ou conforme a agricultura a qual a propriedade é ligada), pelo produto, pelo grupo de área total e por localidade.

Figura 24: Tabela 6954 - Parte 1



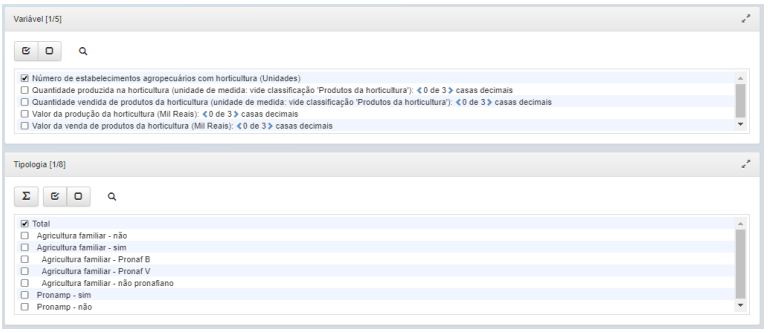
Fonte: O autor

A Figura 24 exibe uma parte crucial da estruturação das consultas no SIDRA, que é o quadro de layout, uma vez que este é responsável pela organização da(s) tabela(s) gerada(s) a partir da consulta montada.

Observando a Figura 24, a atual organização de layout geraria uma tabela para cada variável selecionada, sendo que a tabela possui as seguintes informações nas colunas: ano, tipologia e produtos de horticultura. Nas linhas estão as seguintes informações: unidade territorial e grupos de área total. Cabe salientar que caso seja selecionado mais tipologias, produtos, unidades territoriais e/ou grupos de área total, serão geradas mais linhas/colunas referentes aquela informação que teve mais opções selecionadas.

Portanto, é importante frisar que esta organização deve ser a mesma para todas as consultas, principalmente pelo fato de que ao se alterar a estrutura da tabela de retorno dos dados, consecutivamente será alterado o posicionamento dos dados.

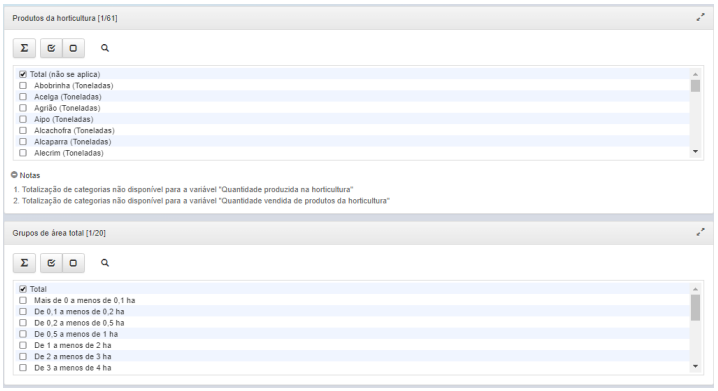
Figura 25: Tabela 6954 - Parte 2



Fonte: O autor

Na Figura 25 é mostrado as opções de variáveis, que são as informações possíveis de acessar-se nesta página, além de exibir as possibilidades de tipologia, que são: Total, Agricultura Familiar - sim, Agricultura Familiar - não, Agricultura Familiar - Pronaf B, Agricultura Familiar - Pronaf V, Agricultura Familiar - não pronafiano, Pronamp - sim e Pronamp - não.

Figura 26: Tabela 6954 - Parte 3



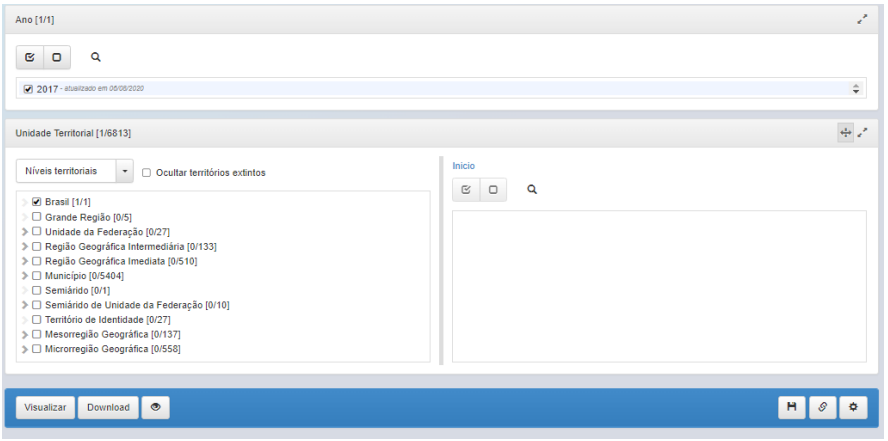
Fonte: O autor

Na Figura 26 é exposta a parte da página relacionada aos filtros de produto e grupo de área total, sendo que existem 61 opções para produto e 20 para grupo de área total. Os grupos de área total, permitem filtrar os dados por intervalos de tamanho de propriedade.

Os produtos aos quais se tem acesso nesta página são: total, abobrinha, acelga, agrião, aipo, alcachofra, alcaparra, alecrim, alface, alho-poró, almeirão, aspargo, batata-baroa (mandioquinha), batata-doce, berinjela, bertalha, beterraba, boldo, brócolis, bucha (esponja vegetal), camomila, cará, caruru, cebolinha, cenoura, chicória, chuchu, coentro, cogumelos, couve, couve-flor, erva-doce, ervilha (vagem),espinafre, gengibre, hortelã, inhame, jiló, lentilha, manjericão, maxixe, milho verde (espiga), morango, mostarda (semente), nabiça, nabo, orégano, pepino, pimenta, pimentão, quiabo, rabanete, repolho, rúcula, salsa, taioba, tomate (estaqueado), vagem (feijão vagem), outros produtos, sementes (produzidas para plantio) e mudas e outras formas de propagação (produzidas para plantio).

Os grupos de área total disponíveis são: total, mais de 0 a menos de 0,1 ha, de 0,1 a menos de 0,2 ha, de 0,2 a menos de 0,5 ha, de 0,5 a menos de 1 ha, de 1 a menos de 2 ha, de 2 a menos de 3 ha, de 3 a menos de 4 ha, de 4 a menos de 5 ha, de 5 a menos de 10 ha, de 10 a menos de 20 ha, de 20 a menos de 50 ha, de 50 a menos de 100 ha, de 100 a menos de 200 ha, de 200 a menos de 500 ha, de 500 a menos de 1.000 ha, de 1.000 a menos de 2.500 ha, de 2.500 a menos de 10.000 ha, de 10.000 ha e mais e produtor sem área.

Figura 27 - Tabela 6954 - Parte 4



Fonte: O autor

Figura 28 - Página Completa

Fonte: O autor

Na Figura 27 é mostrado a parte final da filtragem dos dados, nela estão os filtros do ano e da unidade territorial. No caso dos dados do Censo Agropecuário somente se tem acesso aos dados do ano de 2017 no SIDRA no momento do desenvolvimento deste trabalho. Com relação a unidade territorial, é importante salientar que ela permite filtrar os dados por país, estado, município e por regionalização geográfica.

Para demonstrar o retorno dos dados produzidos por uma consulta nesta página, foi realizado a produção de uma consulta manual na página, que terá a organização de layout das tabelas de retorno mostrada na Figura 29 e que possui todas as 61 opções de produtos selecionadas, todas as 5 variáveis selecionadas, as 3 primeiras tipologias marcadas, todos os grupos de área total selecionados e a unidade territorial solicitada foi o Brasil, como mostrado nas Figuras 30 e 31.

Figura 29 - Quadro de layout das tabelas de retorno

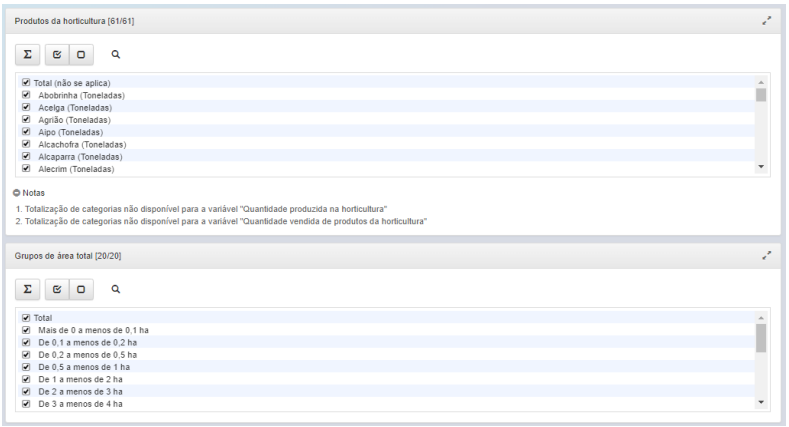
Número de estabelecimentos agropecuários com horticultura, Quantidade produzida na horticultura, Quantidade vendida de produtos da horticultura, Valor da produção da horticultura e Valor da venda de produtos da horticultura, por tipologia, produtos da horticultura e grupos de área total	
Unidade Territorial (1)	
Grupos de área total (20)	
Variável (5)	
	Ano (1)
	Tipologia (3)
Produtos da horticultura (61)	

Fonte: O autor

Figura 30 - Variáveis e Tipologias selecionadas

Fonte: O autor

Figura 31 - Produtos e Grupos de área total selecionados



Fonte: O autor

A consulta mostrada nas Figuras 29, 30 e 31, resulta na geração de 100 tabelas como a tabela mostrada em parte na Figura 32, sendo que destas 100 tabelas geradas, 20 tabelas são relacionadas a cada variável, pois é gerada uma tabela para cada possibilidade dentro da relação entre a variável e o grupos de área total. Cabe salientar que cada tabela possui 61 linhas, fora as linhas do cabeçalho,

onde cada linha representa um produto, além disso as tabelas possuem 3 colunas, que são as 3 tipologias selecionadas.

Figura 32 - Retorno da Consulta no SIDRA

Tabela 6954 - Número de estabelecimentos agropecuários com horticultura, Quantidade produzida na horticultura, Quantidade vendida de produtos da horticultura, Valor da produção da horticultura e Valor da venda de produtos da horticultura, por tipologia, produtos da horticultura e grupos de área total			
Brasil			
Grupos de área total - Total			
Variável - Número de estabelecimentos agropecuários com horticultura (Unidades)			
Ano - 2017			
Produtos da horticultura	Tipologia		
	Total	Agricultura familiar - não	Agricultura familiar - sim
Total	336.195	57.989	278.206
Abobrinha	34.855	5.924	28.931
Aceiga	5.184	1.065	4.119
Agrião	5.837	1.296	4.541
Aipo	1.188	250	938
Alcachofra	174	41	133

Fonte: O autor

4.2 CONAB

O segundo sistema estudado foi o sistema da Companhia Nacional de Abastecimento (Conab), que pode ser acessado pelo http://dw.ceasa.gov.br/ e que possui dados de comercialização realizada pelas unidades do Ceasa para cerca de 547 produtos. Alguns detalhes relevantes a respeito deste site é que ele possui uma interface de visualização de dados construída por um sistema de *Business Intelligence* ou Inteligência de Negócios, que apresenta os dados de comercialização com alto nível de detalhamento e, portanto, uma granularidade menor dos dados. Um exemplo deste alto nível de detalhamento pode ser visto no caso do produto alho, que poderia ser apresentado como um produto somente, porém o sistema tem o produto alho dividido em 3 produtos diferentes: alho, alho macho e alho poró.

Além disso, o sistema do Conab possui 2 divisões iniciais para a consulta, são elas: o PROHORT - Preço Diário e o PROHORT – SIMAB, sendo que o primeiro apresenta os dados referentes às médias simples das cotações diárias realizadas pelos mercados atacadistas, já a segunda divisão apresenta os dados relativos aos preços médios, as quantidades comercializadas e os valores totais da comercialização.

Neste trabalho, focou-se no estudo e extração dos dados presentes no sistema do PROHORT - SIMAB, que possui mais dados relevantes ao contexto da produção agrícola, uma vez que os dados de comercialização podem auxiliar a

análise do perfil dos consumidores em cada município e a relação do consumo no município com o que é mais produzido nele.

Outro ponto positivo desse sistemas é que ele possui cerca de 15 dimensões diferentes, sendo eles: Ceasa, Ceasa - UF, Ceasa - Município, Período de Comercialização - Ano, Período de Comercialização - Mês, Produto, Produto variedade, Produto setor, Produto grupo, Produto subgrupo, Origem Produto - País, Origem Produto - Região, Origem Produto - UF, Origem Produto - Meso-Microrregião e Origem Produto - Município.

Alguns exemplos de produtos que têm informações disponíveis neste sistema são: Abacate, abacaxi, abiu, abóbora, abobrinha, abrótea, açafrão, açaí, acelga, acerola, açúcar, adubo, agapanto, agrião, água de coco, água sanitária, aguardente, albacora, alc. p.conserva, alcachofra, alecrim, alface, alfafa, alfavaca, alfinete, algodão, alho, alho macho, alho poró, almeirão, alpiste, ameixa, ameixa importada, ameixa seca, amêndoa, amendoim, amido milho, amor perfeito, amora, anchova, angélica, anjo, antúrio, araticum, areca, ariacó, arraiá, arroz, arruda, artemizia, aspargo, asters, astromeria, atemoia, atum, avela, avenca, azaleia, azeite de oliva, azeitona, bacalhau, badejo e etc, que juntos formam, como já citado anteriormente cerca de 547 produtos.

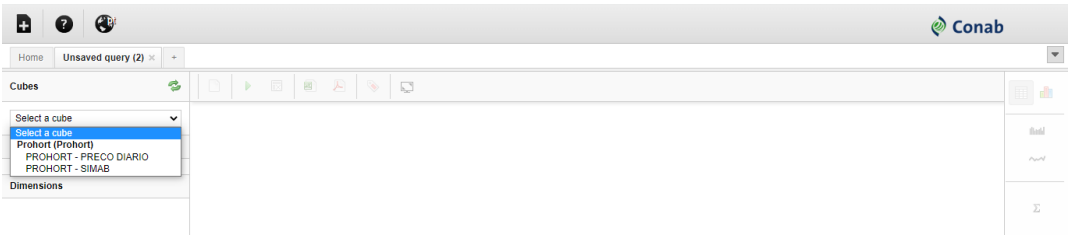
Para acessar estes dados, inicialmente, será carregada uma tela, como mostrado na Figura 33, na qual deve ser feito o *click* no botão realizar consulta e após isto, será exibido uma página, como mostrado na Figura 34.

Figura 33 - Página inicial do sistema da Conab

Fonte: O autor

Na Figura 34, é possível observar que existe um campo de seleção com o título de *Cubes*. Neste campo, é feita a seleção entre o PROHORT - Preço Diário e o PROHORT – SIMAB, para que assim possa-se carregar os filtros e as medidas, sendo que as medidas referem-se às informações que serão exibidas.

Figura 34 - Tela de acesso aos dados do Conab

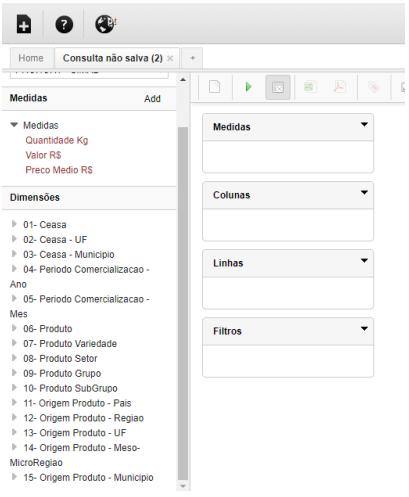


Fonte: O autor

Na Figura 35, são apresentadas as dimensões e as medidas, que são carregadas após o *click* na opção PROHORT – SIMAB no campo citado anteriormente. Agora inicia-se a construção da consulta, para tanto é necessário realizar o *click* no texto em vermelho da medida (Figura 35 – parte superior) que se deseja que apareça na visualização (equivalente a tabela no SIDRA) que será gerada para a consulta ou realizar o clicar e arrastar o texto até a divisão medidas. Isto fará com que a medida apareça na visualização resultante da consulta, cabe salientar que a ordem das medidas definirá a ordem em que elas irão aparecer como colunas na visualização resultante da consulta.

As demais dimensões e medidas, mostrados na Figura 35, permitem a mesma interação, ou seja, é possível tanto clicar diretamente neles, quanto clicar e arrastar.

Figura 35 - Organização da visualização de retorno

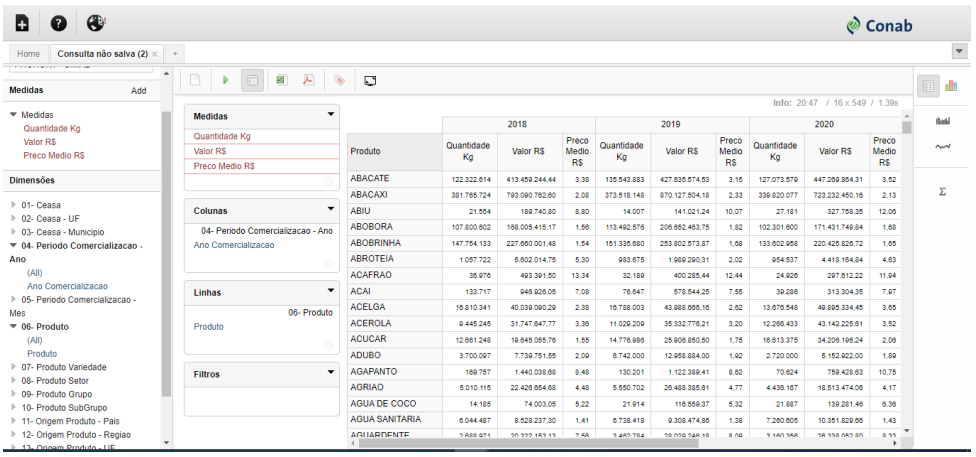


Fonte: O autor

Após montar a consulta, basta clicar no botão verde de *play* (Figura 36), que se encontra na barra acima do campo intitulado medidas, com isto, será gerado uma tabela com a organização e as informações solicitadas. Na Figura 36, é apresentada uma tabela gerada pela consulta das 3 medidas, com a informação do ano de

comercialização nas colunas e os produtos nas linhas. É importante frisar que como pode ser observado na Figura 36, os dados que estão colocados na divisão “Colunas”, irão aparecer acima das medidas.

Figura 36 - Exemplo de consulta e retorno no site da Conab

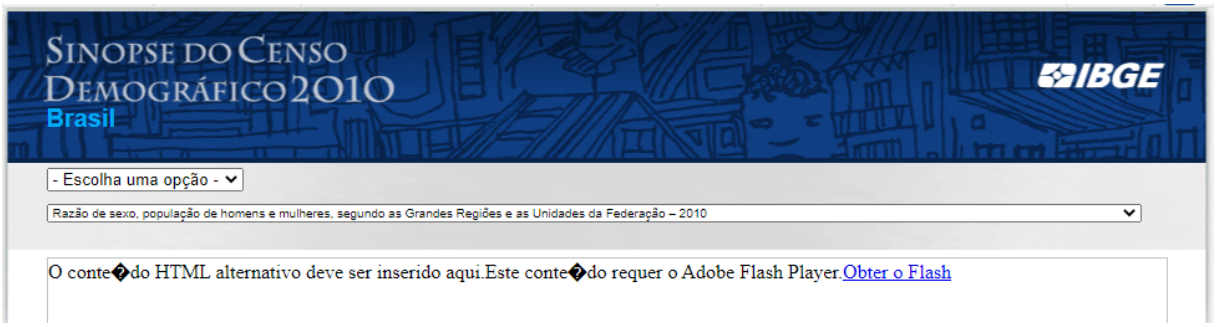


Fonte: O autor

4.3 IBGE CENSO DEMOGRÁFICO 2010

O terceiro site estudado foi o IBGE Censo Demográfico de 2010, que pode ser acessado pelo link https://www.ibge.gov.br/censo2010/apps/sinopse/index.php e que possui dados relativos à população de cada município conforme o Censo Demográfico de 2010, que até o momento do desenvolvimento deste trabalho, era o último Censo Demográfico com os dados definitivos tabulados e disponíveis. Na Figura 37, é apresentada a página inicial apresentada ao se acessar o link citado anteriormente.

Figura 37 - Censo Demográfico 2010 - Página Inicial



Fonte: O autor

Para se acessar as informações sobre os municípios, é necessário escolher a opção Brasil ou uma unidade federativa no primeiro campo, e a informação desejada

no segundo campo, porém após ser selecionado um dos campos, a página já será recarregada e irão aparecer informações. As opções possíveis para o segundo campo são as seguintes: Número de municípios, número de distritos, número de vilas, população, proporção da população do município da capital em relação à da Unidade da Federação, distribuição percentual, densidade demográfica, número de domicílios recenseadas, média de residentes por domicílios, dentre outras.

Na Figura 38, é mostrada o resultado exibido após selecionar o estado do Acre no primeiro campo e a informação 2.1 - População residente, total, urbana total e urbana na sede municipal, em números absolutos e relativos, com indicação da área total e densidade demográfica, segundo as Unidades da Federação e os municípios – 2010 no segundo campo. Com esta consulta é possível obter várias informações referentes a população para os todos os municípios do estado selecionado, além é possível acessar uma outra página com outros dados do município ao se clicar no nome do município na tabela.

Figura 38 - Dados exibidos após a seleção do Estado e da Informação

Município	Total	Urbana	Urbana na sede municipal	Total percentual	Urbana percentual	Urbana na sede municipal percentual	Área total Km2	Densidade demográfica da unidade territorial Hab/Km2
Acrelândia	12.538	5.916	5.916	100,00%	47,00%	47,00%	1807,9	6,94
Assis Brasil	6.072	3.700	3.700	100,00%	60,00%	60,00%	4974,2	1,22
Brasiléia	21.398	14.257	14.257	100,00%	66,00%	66,00%	3916,5	5,46
Bujari	8.471	3.693	3.693	100,00%	43,00%	43,00%	3034,8	2,79
Capixaba	8.798	3.929	3.929	100,00%	44,00%	44,00%	1702,6	5,17
Cruzeiro do Sul	78.507	55.326	55.326	100,00%	70,00%	70,00%	8779,2	8,94
Epitaciolândia	15.100	10.618	10.618	100,00%	70,00%	70,00%	1654,8	9,13
Feijó	32.412	16.636	16.636	100,00%	51,00%	51,00%	27974,6	1,16
Jordão	6.577	2.272	2.272	100,00%	34,00%	34,00%	5357,3	1,23
Mâncio Lima	15.206	8.750	8.750	100,00%	57,00%	57,00%	5453,0	2,79
Manoel Urbano	7.981	5.278	5.278	100,00%	66,00%	66,00%	10634,5	0,75
Marechal Thaumaturgo	14.227	3.969	3.969	100,00%	27,00%	27,00%	8191,7	1,74
Plácido de Castro	17.209	10.382	8.373	100,00%	60,00%	48,00%	1943,2	8,86
Porto Acre	14.880	1.982	1.982	100,00%	13,00%	13,00%	2604,7	5,71
Porto Walter	9.176	3.323	3.323	100,00%	36,00%	36,00%	6443,9	1,42
Rio Branco	336.038	308.545	308.545	100,00%	91,00%	91,00%	8835,7	38,03
Rodrigues Alves	14.389	4.315	4.315	100,00%	29,00%	29,00%	3077,0	4,68
Santa Rosa do Purus	4.691	1.892	1.892	100,00%	40,00%	40,00%	6145,6	0,76
Sena Madureira	38.029	25.112	25.112	100,00%	66,00%	66,00%	23751,3	1,60
Senador Guiomard	20.179	12.703	12.703	100,00%	62,00%	62,00%	2321,5	8,69
Tarauacá	35.590	19.351	19.351	100,00%	54,00%	54,00%	20171,0	1,76
Xapuri	16.091	10.330	10.330	100,00%	64,00%	64,00%	5347,3	3,01

Fonte: IBGE, Censo Demográfico 2010.
(1) Exclusive a população residente nas áreas urbanas isoladas. (2) Valores incluindo as águas interiores.

Fonte: O autor

5 SISTEMA DE EXTRAÇÃO DOS DADOS

Visando realizar a extração dos dados presentes nos sistemas citados no capítulo anterior, foram desenvolvidos inicialmente sistemas separados para tratar de maneira isolada inicialmente o desafio de extrair, formatar, agrupar ou qualquer outra operação necessária nos dados de cada um dos sistemas.

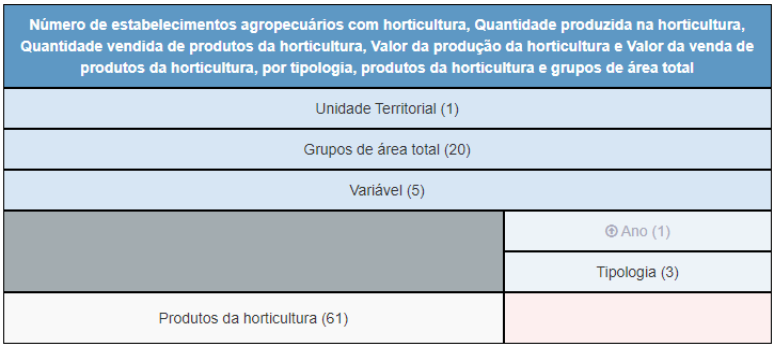
Neste capítulo, serão apresentadas funcionalidades de cada um dos sistemas de extração dos dados, além de detalhar-se as dificuldades enfrentadas e o desempenho de cada sistema de extração. Para tanto, o capítulo foi dividido em 3 subcapítulos, sendo cada um destes focado em um dos sites apresentados no capítulo 4.

Para o desenvolvimento dos sistemas de extração foi utilizado o Google Collaboratory[10], também conhecido como Colab, é um serviço de nuvem gratuito fornecido pelo próprio Google, que permite executar códigos Python e utilizar a mistura de texto com imagens e resultados do próprio código, o que facilita a produção de anotações relativas ao código. Além disso, por ser um serviço da Google ligado ao e-mail, facilita o compartilhamento com outros usuários.

5.1 SISTEMA IBGE DE RECUPERAÇÃO AUTOMÁTICA TABELA 6954

O sistema desenvolvido para extrair dados da Tabela 6954 do SIDRA (seção 4.1), visa construir a consulta utilizada como exemplo no capítulo anterior, para tanto, inicialmente, realizou-se as operações de arrastar os blocos para montar o layout da tabela de retorno dos dados.

Figura 29 - Quadro de layout das tabelas de retorno



Fonte: O autor

[10] https://colab.research.google.com/

Na Figura 39, é mostrado o código responsável por parte interações com a interface do SIDRA mostrada na Figura 29 para selecionar os dados que irão para a tabela/visualização criada, como por exemplo, na Figura 39, linha 48, que corresponde a ação de um usuário na interface do SIDRA que “arrasta” (*drag and drop*) um item para a consulta. Para realizar esta movimentação, foi utilizada a biblioteca *ActionChains* que possui entre outros métodos, o método *drag_and_drop* (Figura 39, linha 48), que com base em dois parâmetros, sendo o primeiro o elemento a ser movimentado e o segundo o destino da movimentação, movimenta o bloco apontado no primeiro parâmetro.

Cabe salientar que a busca pelos elementos da página, como mostrado na Figura 39, utiliza uma espera condicionada a fator específico, como por exemplo o elemento estar visível e ser selecionável (permitir “clicks” do mouse). Para utilizar esta função, é necessário realizar a importação da biblioteca *expected_conditions* e realizar a criação do *wait*, que define o tempo padrão de espera.

Figura 39 - Sistema de extração SIDRA Tabela 6954 - Parte 1

```
40  origem = wait.until(EC.visibility_of_element_located((By.ID, 'dim-C220')))
41
42  #Destino: //*[@id="dim-V"] bloco variável
43  destino = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="editor-layout"]/tbody/tr[1]/th')))
44
45  acao = ActionChains(driver)
46  time.sleep(2)
47  #Grupo de área total
48  acao.drag_and_drop(origem, destino).perform()
49  time.sleep(2)
50
51  #Origem: //*[@id="dim-C228"] Produtos
52  origem = wait.until(EC.visibility_of_element_located((By.ID, 'dim-C228')))
53
54  destino = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="editor-layout"]/tbody/tr[7]')))
55
56  acao = ActionChains(driver)
57  time.sleep(3)
58  #Produtos
59  acao.drag_and_drop(origem, destino).perform()
60  time.sleep(2)
```

Fonte: O autor

Na Figura 40, é mostrado parte do código que realiza a extração de dados do país referentes às variáveis para o grupo de área total e para tanto, são buscadas as colunas 1, 2 e 3 para cada linha em cada tabela. Cabe salientar que conforme mostrado anteriormente o layout das tabelas de retorno, apresenta os produtos nas linhas e as tipologias nas colunas, sendo que cada tabela se refere a uma relação entre a localidade, o grupo de área total e a variável.

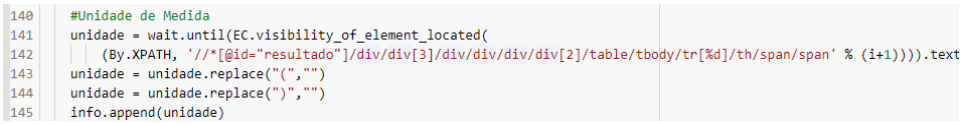
Figura 40 - Sistema de extração SIDRA Tabela 6954 - Parte 2

```
for i in range(1,(quantidade_produtos+1)):

    produto = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[1]/table/tbody/tr[%d]/th/span' % i))).text
    info.append(produto)

    info.append("Brasil")

    #Tabela 1 = Variável 1, Número de estabelecimentos
    numEstabelecimentoTotal = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[1]/table/tbody/tr[%d]/td[1]' % i))).text
    info.append(numEstabelecimentoTotal)
    numEstabelecimentoAGFNao = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[1]/table/tbody/tr[%d]/td[2]' % i))).text
    info.append(numEstabelecimentoAGFNao)
    numEstabelecimentoAGFSim = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[1]/table/tbody/tr[%d]/td[3]' % i))).text
    info.append(numEstabelecimentoAGFSim)

    #Tabela 2 = Variável 2, Quantidade Produzida
    qtdProduzidaTotal = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[2]/table/tbody/tr[%d]/td[1]' % i))).text
    info.append(qtdProduzidaTotal)
    qtdProduzidaAGFNao = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[2]/table/tbody/tr[%d]/td[2]' % i))).text
```

Fonte: O autor

Na Figura 41, é exibido o código responsável por capturar a unidade de medida a qual a quantidade produzida está vinculada, para tanto é realizada a busca do elemento (Figura 41, linhas 141 e 142) e é realizado um processamento para retirar os parênteses da string com o nome da unidade de medida.

Figura 41 - Sistema de extração SIDRA Tabela 6954 - Parte 3



```
    #Unidade de Medida
    unidade = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[2]/table/tbody/tr[%d]/th/span/span' % (i+1)))).text
    unidade = unidade.replace("(","")
    unidade = unidade.replace(")","")
    info.append(unidade)
```

Fonte: O autor

Na Figura 43, é exposto o código responsável por extrair dados dos grupos, sendo que dos dados desses grupos é extraído somente as informações da quantidade de estabelecimentos totais, quantidade de estabelecimentos ligados à Agricultura Familiar e não ligados à Agricultura Familiar. Para tanto, é utilizado a seguinte expressão “5*x+1”, onde x corresponde ao grupo de área total, que indica que deve ser ignorada a primeira tabela, pelo fato de que esta possui os valores referentes ao somatório de todos os grupos, e usadas as tabelas referentes à primeira variável, na Figura 42 é apresentado um pedaço de duas tabelas que relacionam um grupo de área total com a primeira variável, que é o número de estabelecimentos.

Figura 42 - Interface *Web* mostrando as tabelas

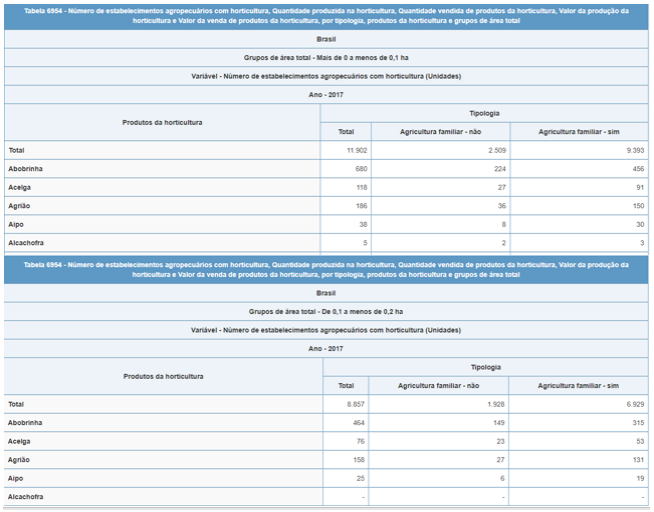


Tabela 6954 - Número de estabelecimentos agropecuários com horticultura, Quantidade produzida na horticultura, Quantidade vendida de produtos da horticultura, Valor da produção da horticultura e Valor da venda de produtos da horticultura, por tipologia, produtos da horticultura e grupos de área total

Brasil

Grupos de área total - Mais de 0 a menos de 0,1 ha

Variável - Número de estabelecimentos agropecuários com horticultura (Unidades)

Ano - 2017

Produtos da horticultura	Tipologia		
	Total	Agricultura familiar - não	Agricultura familiar - sim
Total	11.902	2.509	9.393
Abobrinha	680	224	456
Acelga	118	27	91
Agrião	186	36	150
Aipo	38	8	30
Alcachofra	5	2	3

Tabela 6954 - Número de estabelecimentos agropecuários com horticultura, Quantidade produzida na horticultura, Quantidade vendida de produtos da horticultura, Valor da produção da horticultura e Valor da venda de produtos da horticultura, por tipologia, produtos da horticultura e grupos de área total

Brasil

Grupos de área total - De 0,1 a menos de 0,2 ha

Variável - Número de estabelecimentos agropecuários com horticultura (Unidades)

Ano - 2017

Produtos da horticultura	Tipologia		
	Total	Agricultura familiar - não	Agricultura familiar - sim
Total	8.857	1.928	6.929
Abobrinha	464	149	315
Acelga	76	23	53
Agrião	158	27	131
Aipo	25	6	19
Alcachofra	-	-	-

Fonte: O autor

Figura 43 - Sistema de extração SIDRA Tabela 6954 - Parte 4

```
for x in range (1, 20):
  for i in range(1,(quantidade_produtos+1)):

    produto = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[1]/table/tbody/tr[%d]/th/span' % i))).text
    grupos.append(produto)

    grupos.append("Brasil")

    #Fórmula para pegar a tabela: 5x+y, onde o x vem do for e o y vem do número da variável
    #Tabela 1 = Variável 1, Número de estabelecimentos
    numEstabelecimentoTotal = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[%d]/table/tbody/tr[%d]/td[1]' % ((5*x+1),i)))).text
    grupos.append(numEstabelecimentoTotal)
    numEstabelecimentoAGFNao = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[%d]/table/tbody/tr[%d]/td[2]' % ((5*x+1),i)))).text
    grupos.append(numEstabelecimentoAGFNao)
    numEstabelecimentoAGFSim = wait.until(EC.visibility_of_element_located(
        (By.XPATH, '//*[@id="resultado"]/div/div[3]/div/div/div/div[%d]/table/tbody/tr[%d]/td[3]' % ((5*x+1),i)))).text
    grupos.append(numEstabelecimentoAGFSim)
```

Fonte: O autor

Na Figura 44, é mostrado uma parte do sistema que realiza o agrupamento dos dados obtidos com o código exibido na Figura 43, que são referentes aos grupos de área total. Estes agrupamentos visam diminuir os 19 grupos de área total para 5 grupos, que seguem a seguinte lógica: Grupo 1 - Número de Estabelecimentos com até 10 ha, Grupo 2 - Número de Estabelecimentos de 10 até 100 ha, Grupo 3 - Número de Estabelecimentos de 100 até 1000 ha, Grupo 4 - Número de Estabelecimentos com mais de 1000 ha e Grupo 5 - Número de Estabelecimentos com Produtor sem área. Este agrupamento foi realizado visando construir uma relação entre uma faixa de tamanhos de propriedade predeterminadas e a quantidade de estabelecimentos total, a quantidade de estabelecimentos ligados à Agricultura Familiar e a quantidade de estabelecimentos não ligados à Agricultura Familiar dentro desta faixa de tamanhos de propriedade em um município.

Figura 44 - Sistema de extração SIDRA Tabela 6954 - Parte 5

```
for k in range(0,quantidade_locais):
  for i in range(0,quantidade_produtos):
    print("------------------ GRUPO 1 ------------------")
    for j in range(0,9):
      aux = i+(j*quantidade_produtos)+(k*quantidade_total)
      print("Índice = %d" % aux)
      if gp_array[aux][2] != '..' and gp_array[aux][2] != 'X' and gp_array[aux][2] != '-':
        dado = int(str(gp_array[aux][2]).replace('.',''))
        auxGrupo += dado
      if gp_array[aux][3] != '..' and gp_array[aux][3] != 'X' and gp_array[aux][3] != '-':
        dadoAGFNao = int(str(gp_array[aux][3]).replace('.',''))
        auxGrupoAGFNao += dadoAGFNao
      if gp_array[aux][4] != '..' and gp_array[aux][4] != 'X' and gp_array[aux][4] != '-':
        dadoAGFSim = int(str(gp_array[aux][4]).replace('.',''))
        auxGrupoAGFSim += dadoAGFSim
    grupos.append(auxGrupo)
    gruposAGFSim.append(auxGrupoAGFSim)
    gruposAGFNao.append(auxGrupoAGFNao)
    auxGrupo = 0
    auxGrupoAGFSim = 0
    auxGrupoAGFNao = 0
```

Fonte: O autor

Na Figura 45, é mostrado o arquivo CSV gerado pelo sistema de extração de dados na Tabela 6954 do SIDRA, sendo importante salientar que para a construção deste arquivo foi realizada a execução do sistema descrito anteriormente, com a quantidade de produtos restringida a 15 produtos por localidade de forma a permitir que os dados fossem mostrados em uma figura (cabe ressaltar que são 61 produtos no total).

Figura 45 - Arquivo CSV SIDRA Tabela 6954

Produto	Local	qtd Estabelecimentos	qtd Estabelecimentos AGFNão	qtd Estabelecimentos AGFSim	qtd Produzida	qtd Produzida AGFNao	qtd Produzida AGFSim	Unidade de Medida
Total	Brasil	336195	57989	278206	..	..	..	Toneladas
Abobrinha	Brasil	34855	5924	28931	158518	35784	122733	Toneladas
Acelga	Brasil	5184	1065	4119	52239	18643	33595	Toneladas
Agrião	Brasil	5837	1296	4541	23222	8758	14464	Toneladas
Aipo	Brasil	1188	250	938	4711	1072	3639	Toneladas
Alcachofra	Brasil	174	41	133	826	364	462	Toneladas
Alcaparra	Brasil	13	5	8	2	1	1	Toneladas
Alecrim	Brasil	1867	461	1406	573	170	403	Toneladas
Alface	Brasil	108382	19344	89038	671509	239084	432425	Toneladas
Alho-porró	Brasil	4148	833	3315	12655	2946	9709	Toneladas
Almeirão	Brasil	15256	3025	12231	23485	7088	16398	Toneladas
Aspargo	Brasil	47	21	26	49	11	38	Toneladas
Batata-baroa (mandioquinha)	Brasil	2495	352	2143	51391	28913	22478	Toneladas
Batata-doce	Brasil	70860	11350	59510	350512	143456	207056	Toneladas
Berinjela	Brasil	11309	1979	9330	71914	18609	53305	Toneladas
Total	Alta Floresta	50	11	39	..	..	..	Toneladas
Abobrinha	Alta Floresta	28	5	23	31	1	30	Toneladas
Acelga	Alta Floresta	2	-	2	X	-	X	Toneladas
Agrião	Alta Floresta	-	-	-	-	-	-	Toneladas
Aipo	Alta Floresta	1	-	1	X	-	X	Toneladas
Alcachofra	Alta Floresta	-	-	-	-	-	-	Toneladas

Fonte: O autor

Por fim, é importante frisar que a execução deste sistema, que gerou a construção do arquivo CSV exibido na Figura 45, demorou cerca de 45 minutos, além de gerar uma requisição para cada localidade (Município, estado ou país). Por tal fator, este é o sistema que mais gerará requisições e processamento de dados, pois terá que ser feito um acesso ao site e a construção da consulta para cada município e para cada unidade federativa, além de realizar esta mesma busca para o país, ou seja, serão no mínimo 28 requisições feitas neste site, só para os dados dos estados e do país, mais as requisições dos municípios. Vale lembrar que para

cada localidade também é feito o processo de agrupamento das informações dos grupos de área total.

5.2 CONAB

O sistema de extração do Conab realiza o processo de automação da montagem de uma consulta para gerar uma tabela com todas as medidas (Figura 46, linhas 50 até 55), com os produtos nas linhas (Figura 46, linhas 59 e 60) e com o filtro Ceasa UF (Figura 46, linhas 64 e 65) e o ano de comercialização nas colunas.

Na Figura 46, é mostrado parte do código responsável pela formulação da consulta, sendo o código da linha 42, responsável pelo *click* no botão na página inicial, enquanto que nas linhas 45, 46 e 47 possuem o código que seleciona o PROHORT- SIMAB. Entre as linhas 50 e 65 são buscados os elementos da página e realizadas as ações necessárias para montar a consulta descrita anteriormente.

Figura 46 - Sistema de extração Conab - Parte 1

```
42  botao = wait.until(EC.element_to_be_clickable((By.CLASS_NAME,"run_query"))).click()
43
44  driver.implicitly_wait(15)
45  option = wait.until(EC.element_to_be_clickable(
46      (By.XPATH,"//*[@id='tab_panel']/div/div[2]/div[2]/select/optgroup/option[2]")))
47  option.click()
48
49  driver.implicitly_wait(15)
50  bQuantidade = wait.until(EC.element_to_be_clickable(
51      (By.XPATH,'//*[@id="tab_panel"]/div/div[2]/div[3]/div/div[1]/ul/li/ul/li[1]/a'))).click()
52  bValor = wait.until(EC.element_to_be_clickable(
53      (By.XPATH,'//*[@id="tab_panel"]/div/div[2]/div[3]/div/div[1]/ul/li/ul/li[2]/a'))).click()
54  bPreco = wait.until(EC.element_to_be_clickable(
55      (By.XPATH,'//*[@id="tab_panel"]/div/div[2]/div[3]/div/div[1]/ul/li/ul/li[3]/a'))).click()
56
57  bProdutoLista = wait.until(EC.element_to_be_clickable(
58      (By.XPATH,'//*[@id="tab_panel"]/div/div[2]/div[3]/div/div[2]/ul/li[6]/a'))).click()
59  bProduto = wait.until(EC.element_to_be_clickable(
60      (By.XPATH,'//*[@id="tab_panel"]/div/div[2]/div[3]/div/div[2]/ul/li[6]/ul/ul/li[2]/a'))).click()
61
62  bUFCeasa = wait.until(EC.element_to_be_clickable(
63      (By.XPATH,'//*[@id="tab_panel"]/div/div[2]/div[3]/div/div[2]/ul/li[2]/a'))).click()
64  bUF = wait.until(EC.element_to_be_clickable(
65      (By.XPATH,'//*[@id="tab_panel"]/div/div[2]/div[3]/div/div[2]/ul/li[2]/ul/ul/li[3]/a'))).click()
66
```

Fonte: O autor

Para a extração dos dados do sistema do Conab, optou-se pela estratégia de extrair os dados das unidades federativas, pelo fato de que a quantidade de unidades do Ceasa em relação a quantidade de municípios é baixa, ou seja, não seria interessante buscar a existência de uma unidade do Ceasa, numa busca município a município. Entretanto, mesmo buscando por unidade federativa, a maioria das unidades federativas não possuem informações no site do Conab, e por tal fator, foi necessário construir um código para preencher com “X” as colunas dos

estados que não possuíam valores (este trecho do código é apresentado na Figura 48).

Na Figura 47, é apresentada uma parte do código que é responsável por verificar se o estado do qual está se extraindo informações é o estado de Alagoas (Figura 47, linha 78). Esta verificação é necessária, pois o estado de Alagoas é o único que possui informações somente para os anos de 2019 e 2020, ou seja, não possui informações para os anos de 2018, 2021 e 2022, sendo assim é importante isolar um tratamento diferenciado para esta unidade federativa. Este tratamento consiste basicamente em preencher as colunas dos anos de 2018, 2021 e 2022 com “X”, é importante salientar que a parte do código responsável por preencher as colunas do ano de 2018, pode ser vista na Figura 47 nas linhas 98 e 99.

Figura 47 - Sistema de extração Conab - Parte 2

```
77  for numeroUF in range(2, 14):
78    if numeroUF != 4:
79      uf = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="table_27"]/table/thead/tr[1]/th[%d]/div' % numeroUF))).text
80      #for linha in range(1, 548):
81      for linha in range(1, linhas):
82        info.append(uf)
83        produto = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="table_27"]/table/tbody/tr[%d]/th/div' % linha))).text
84        info.append(produto)
85        for coluna in range(1, 16):
86          xpath = '//*[@id="table_27"]/table/tbody/tr[%d]/td[%d]' % (linha, (coluna+controle))
87          dado = wait.until(EC.visibility_of_element_located((By.XPATH, xpath)))
88          if dado.text != '':
89            info.append(dado.text)
90          else:
91            info.append('X')
92      controle += 15
93    else:
94      for linha in range(1, linhas):
95        info.append('AL')
96        produto = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="table_27"]/table/tbody/tr[%d]/th/div' % linha))).text
97        info.append(produto)
98        for coluna in range(1, 4):
99          info.append('X')
```

Fonte: O autor

Na Figura 47, é apresentado o código responsável por preencher com “X”, todas colunas dos estados que não possuem informações no site do Conab. É importante frisar que ao todo são 15 unidades federativas (Figura 48, linha 110), sem nenhuma informação presente no sistema do Conab. Para realizar este preenchimento é necessário realizar primeiramente o relacionamento entre cada unidade federativa com cada um dos produtos extraídos, e preencher as demais colunas com “X”.

Na Figura 49, é exibido parte do arquivo CSV gerado pelo sistema de extração de dados, nele é possível perceber que nem todos os produtos possuem todas as informações solicitadas na consulta. Para os casos em que a relação produto-informação estava vazia, o sistema foi projetado para preencher com “X”. Cabe salientar que para a realização desta tarefa, é executado um teste para

verificar se o texto do elemento *td* da tabela é vazio (Figura 47, linha 88), se ele for, é preenchido com “X”, senão é colocado o valor presente naquele *td* na tabela.

Figura 48 - Sistema de extração Conab - Parte 3

```
110  estados = ['AP','AM','BA','MA','MT','MS','PA','PB','PI','RN','RO','RR','SC','SE','TO']
111  lista_produtos = ['ABACATE','ABACAXI','ABIU','ABOBORA','ABOBRINHA','ABROTEIA','ACAFRAO','ACAI','ACELGA','
112  for estado in estados:
113    for produto in range(0, linhas):
114      info.append(estado)
115      info.append(lista_produtos[produto])
116      for i in range(1, 16):
117        info.append('X')
```

Fonte: O autor

Figura 49 - Arquivo CSV Conab

	UF	Produto	Quantidade(kg) em 2018	Valor (R$) em 2018	Preço Médio (R$) em 2018	Quantidade(kg) em 2019	Valor (R$) em 2019	Preço Médio (R$) em 2019	Quantidade(kg) em 2020
2	DF	ABACATE	1882534	8212645,92	4,36	2966337	12298987,11	4,15	2936433
3	DF	ABACAXI	12429201	43141355,28	3,47	14301523	34372194,32	2,4	11794449
4	DF	ABIU	10	240	24	X	X	X	X
5	DF	ABOBORA	1755883	3529135,07	2,01	577449	1488718,9	2,58	155514
6	DF	ABOBRINHA	2666731	4513809,24	1,69	3837755	5668691,64	1,48	3904646
7	DF	ABROTEIA	X	X	X	X	X	X	X
8	DF	ACAFRAO	X	X	X	X	X	X	X
9	DF	ACAI	X	X	X	X	X	X	X
10	DF	ACELGA	282741	345582,04	1,22	418798	544454,33	1,3	318653
11	DF	ACEROLA	2729	39631,87	14,52	2696	47527,92	17,63	1464
12	DF	ACUCAR	X	X	X	X	X	X	X
13	DF	ADUBO	59097	1657670,85	28,05	X	X	X	X
14	DF	AGAPANTO	X	X	X	X	X	X	X
15	DF	AGRIAO	196449	765733,67	3,9	248810	1081457,01	4,35	172285
16	DF	AGUA DE COCO	X	X	X	X	X	X	X
17	DF	AGUA SANITARIA	X	X	X	X	X	X	X
18	DF	AGUARDENTE	X	X	X	X	X	X	X
19	DF	AGULHAO	X	X	X	X	X	X	X
20	DF	ALBACORA	X	X	X	X	X	X	X
21	DF	ALC. P.CONSERVA	X	X	X	X	X	X	X
22	DF	ALCACHOFRA	5796	120443,66	20,78	1928	42123,8	21,85	1009
23	DF	ALECRIM	2412	77721,64	32,22	3528	155143,57	43,97	4803
24	DF	ALFACE	1045599	4853684,83	4,64	1536436	6408573,01	4,17	1431297
25	DF	ALFAFA	X	X	X	X	X	X	X
26	DF	ALFAVACA	X	X	X	X	X	X	X
27	DF	ALFINETE	X	X	X	X	X	X	X
28	DF	ALGODAO	X	X	X	X	X	X	X

Fonte: O autor

A execução deste sistema de extração foi realizada com um limitador de quantidade de linhas buscada e consecutivamente linhas extraídas, ou seja, com menos produtos. Para os testes individuais, o limitador recebeu o valor de 50, ou seja, para cada unidade federativa foram buscados e extraídos 50 produtos do total de 547 produtos, e estes testes levaram cerca de 30 minutos.

5.3 IBGE CENSO DEMOGRÁFICO 2010

O sistema de extração do IBGE Censo Demográfico de 2010 realiza o processo de automação da seleção das unidades federativas e a partir dessa seleção o sistema gera uma tabela com todos os municípios pertencentes ao estado selecionado, assim como descrito no capítulo 4.

Na Figura 50, é mostrado o trecho do código responsável por buscar e selecionar cada unidade federativa e ir mudando de unidade federativa conforme a

variável “numeroUF” (Figura 50, linha 46). Esta variável de controle do laço inicia com o valor 3, pois o primeiro estado é a terceira opção do campo de seleção, uma vez que a primeira opção é a opção padrão, que não possui valor (todas UFs são consideradas), e a segunda opção é o Brasil.

Figura 50 - Sistema de extração de dados da População

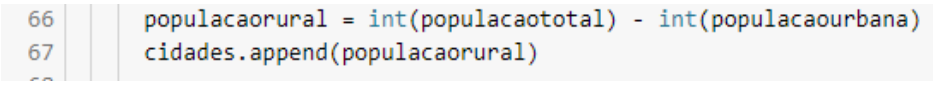
```
46  for numeroUF in range(3, 30):
47      opcaoUF = wait.until(EC.element_to_be_clickable((By.XPATH, '//*[@id="jumpMenu"]/option[%d]' % numeroUF))).click()
48      time.sleep(2)
49      uf = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="conteudo"]/h2'))).text
50      qtdMunicipios = driver.find_elements(By.XPATH, '//*[@id="div_tabela_dados"]/table/tbody/tr')
51      print("Quantidade de Municipios no %s = %d" % (uf, len(qtdMunicipios)))
52      for i in range(1, (len(qtdMunicipios)+1)):
53          nome = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="div_tabela_dados"]/table/tbody/tr[%d]/td[1]' % i))).text
54          cidades.append(nome)
55          cidades.append(uf)
56          cidades.append(siglas[numeroUF-3])
57
58          populacaototal = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="div_tabela_dados"]/table/tbody/tr[%d]/td[2]' % i))).text
59          populacaototal = populacaototal.replace('.','')
60          cidades.append(populacaototal)
61
62          populacaourbana = wait.until(EC.visibility_of_element_located((By.XPATH, '//*[@id="div_tabela_dados"]/table/tbody/tr[%d]/td[3]' % i))).text
63          populacaourbana = populacaourbana.replace('.','')
64          cidades.append(populacaourbana)
65
```

Fonte: O autor

Na Figura 50, também é mostrado outra parte do código, que é parte responsável por realizar a busca da quantidade de linhas da tabela (Figura 50, linha 50), para que o laço de repetição (Figura 50, linha 52), responsável por extrair os dados dos municípios, seja adaptado a quantidade de municípios que o estado possui.

Outro detalhe do sistema de extração de dados deste site, é que ele realiza o cálculo da população rural de cada município a partir da população total subtraída pela população urbana (Figura 51, linha 66), porém para isto, é necessário realizar a obtenção da população total (Figura 50, linha 58) e da população urbana (Figura 50, linha 58). Em seguida, foi necessário realizar uma pequena adaptação dos valores, para que o caractere ponto “.” fosse retirado (Figura 50, linha 59 e linha 63), antes de realizar as conversões que eram feitas para realizar o cálculo (Figura 51, linha 66).

Figura 51 - Cálculo da população rural



```
66      populacaorural = int(populacaototal) - int(populacaourbana)
67      cidades.append(populacaorural)
```

Fonte: O autor

A execução deste sistema de extração completo leva em torno de 55 minutos e gera um arquivo CSV com os dados extraídos, como pode ser visto na Figura 52. Cabe salientar que o tempo de execução é resultado principalmente da grande quantidade de municípios existente no Brasil, uma vez que, por exemplo, o estado

Minas Gerais possui 853 municípios, São Paulo possui 645 e o Rio Grande do Sul possui 496, cabe salientar que estes dados são referentes ao Censo Demográfico de 2010. Ao todo em 2010, o país possuía cerca de 5565 municípios.

Figura 52 - Arquivo CSV da população

	Nome do Munícipio	Nome UF	Sigla UF	População Total em 2010	População Urbana em 2010	População Rural em 2010	Área Territorial	Densidade Demográfica em 2010
2	Acrelândia	Acre	AC	12538	5916	6622	1807,9	6,94
3	Assis Brasil	Acre	AC	6072	3700	2372	4974,2	1,22
4	Brasiléia	Acre	AC	21398	14257	7141	3916,5	5,46
5	Bujari	Acre	AC	8471	3693	4778	3034,8	2,79
6	Capixaba	Acre	AC	8798	3929	4869	1702,6	5,17
7	Cruzeiro do Sul	Acre	AC	78507	55326	23181	8779,2	8,94
8	Epitaciolândia	Acre	AC	15100	10618	4482	1654,8	9,13
9	Feijó	Acre	AC	32412	16636	15776	27974,6	1,16
10	Jordão	Acre	AC	6577	2272	4305	5357,3	1,23
11	Mâncio Lima	Acre	AC	15206	8750	6456	5453	2,79
12	Manoel Urbano	Acre	AC	7981	5278	2703	10634,5	0,75
13	Marechal Thaumaturgo	Acre	AC	14227	3969	10258	8191,7	1,74
14	Plácido de Castro	Acre	AC	17209	10382	6827	1943,2	8,86
15	Porto Acre	Acre	AC	14880	1982	12898	2604,7	5,71
16	Porto Walter	Acre	AC	9176	3323	5853	6443,9	1,42
17	Rio Branco	Acre	AC	336038	308545	27493	8835,7	38,03
18	Rodrigues Alves	Acre	AC	14389	4315	10074	3077	4,68
19	Santa Rosa do Purus	Acre	AC	4691	1892	2799	6145,6	0,76
20	Sena Madureira	Acre	AC	38029	25112	12917	23751,3	1,6
21	Senador Guiomard	Acre	AC	20179	12703	7476	2321,5	8,69
22	Tarauacá	Acre	AC	35590	19351	16239	20171	1,76
23	Xapuri	Acre	AC	16091	10330	5761	5347,3	3,01
24	Água Branca	Alagoas	AL	19377	5101	14276	454,6	42,62
25	Anadia	Alagoas	AL	17424	8949	8475	189,5	91,96
26	Arapiraca	Alagoas	AL	214006	181481	32525	356,2	600,84
27	Atalaia	Alagoas	AL	44322	22457	21865	528,8	83,82
28	Barra de Santo Antônio	Alagoas	AL	14230	13242	988	138,4	102,79
29	Barra de São Miguel	Alagoas	AL	7574	6521	1053	76,6	98,86

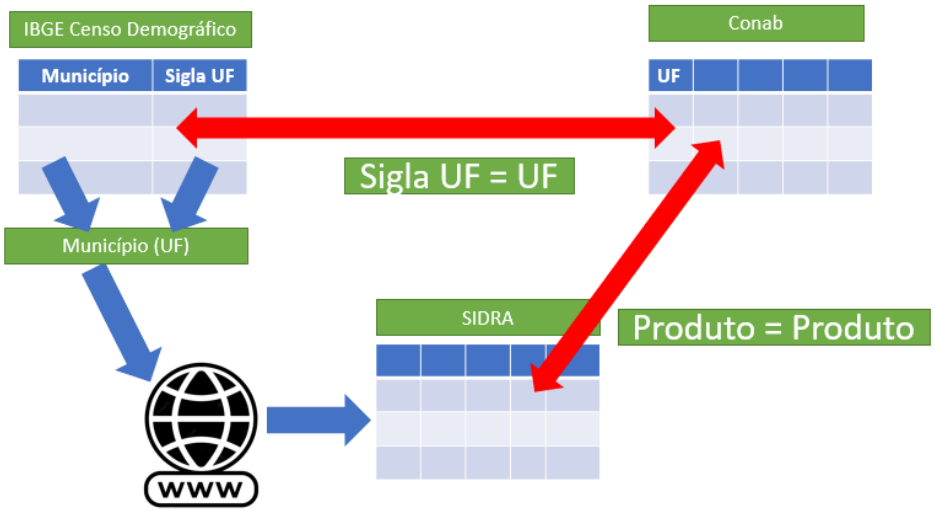
Fonte: O autor

6 BASE DE DADOS MONTADA

Neste capítulo, serão apresentados alguns dos principais dados da base de dados gerada e, os critérios de junção e de descarte dos dados obtidos pelos sistemas estudados no capítulo 5. Cabe enfatizar que a base de dados desenvolvida possui a junção dos dados extraídos por cada um dos 3 sistemas dos 3 portais.

Na Figura 53, é apresentada uma abstração do fluxo de junção dos dados entre as 3 bases para a geração de uma da base de dados única. É importante destacar que o fluxo se inicia a partir dos dados da população, presentes no site do IBGE Censo Demográfico de 2010. Após extrair os dados dos municípios e injetar as siglas dos estados entre os dados dos municípios (processo automatizado por código), inicia-se o processo de junção dos dados dos municípios com os dados do SIDRA. Para tanto, o sistema realiza a busca dos dados no SIDRA passando um parâmetro composto pelo nome do município e a sigla do estado, sendo assim serão retornados somente as informações daquele município. Partindo dessa premissa, o sistema incluirá os dados do IBGE Censo Demográfico de 2010 do município em cada linha gerada para as informações presentes no SIDRA relacionadas àquele município.

Figura 53 - Esquema de junção dos dados



Fonte: O autor

Posteriormente, inicia-se o processo de junção destes dados com os dados de comercialização presentes no Conab, para isto é feito duas comparações, sendo a primeira entre a sigla do estado presente nos dados do IBGE Censo Demográfico de 2010 e a unidade federativa presentes nos dados do Conab e segunda entre o

produto presente nos dados do SIDRA e o produto presente no Conab. Se ambas forem iguais, o sistema repete as informações de comercialização para cada linha da base única que está associada àquele produto/cultura e aquele estado.

Outro ponto a ser mencionado é que para a formação da base de dados unificada serão usados, inicialmente, somente os dados que permitam estabelecer uma junção entre as informações de produção e de comercialização. Portanto, nem todos os produtos presentes no sistema do Conab serão utilizados neste primeiro momento, principalmente pelo fato de que os dados do Conab mesclam produtos de diversas culturas como fruticultura, horticultura, pecuária, entre outras culturas. O SIDRA também possui algumas informações que não serão utilizadas, como por exemplo, os valores para os produtos: total, outros produtos, sementes (produzidas para plantio) e mudas e outras formas de propagação (produzidas para plantio).

Na Tabela 1, são apresentadas algumas das principais colunas presentes na base de dados, sendo que para cada coluna é apresentado o nome da coluna na base de dados, uma descrição da mesma e origem daquela informação.

Tabela 1 - Descrição das principais colunas

Código	Nome da Coluna	Descrição	Fonte
1	Nome do Município	Coluna que apresenta o nome do município, cabe frisar que esta informação será utilizada para construir o parâmetro de busca do município no SIDRA	Informação extraída do site do IBGE Censo Demográfico de 2010
2	Nome do Estado	Coluna que apresenta o nome do estado	Informação extraída do site do IBGE Censo Demográfico de 2010
3	Sigla do Estado	Coluna utilizada para apresentar a sigla do estado, cabe salientar que esta informação será utilizada para formular construir o parâmetro de busca do município no SIDRA e para estabelecer a relação com os dados do Conab	No momento da extração dos dados do IBGE Censo Demográfico é inserida esta informação via código através de uma lista com todas as siglas e uma relação entre elas e a numeração que define o estado dos municípios que estão sendo extraídos.

4	População Total do Município em 2010	Coluna que exibe a população total do município em 2010.	Informação extraída do site do IBGE Censo Demográfico de 2010
5	População Urbana do Município em 2010	Coluna que exibe a população urbana do município em 2010.	Informação extraída do site do IBGE Censo Demográfico de 2010
6	População Rural do Município em 2010	Coluna que exibe a população rural do município em 2010.	Informação obtida através do cálculo de subtração da população total pela população urbana.
7	Área Territorial do Município em 2010	Coluna que exibe a área territorial do município em 2010.	Informação extraída do site do IBGE Censo Demográfico de 2010
8	Densidade Demográfica do Município em 2010	Coluna que exibe a densidade territorial do município em 2010.	Informação extraída do site do IBGE Censo Demográfico de 2010
9	Nome do Produto	Coluna que mostra o nome do produto. Cabe salientar que tal informação será utilizada como base para estabelecer uma relação entre os dados do sistema do Conab e os dados do sistema do SIDRA.	Informação extraída do site do Conab e que também será extraída do site do SIDRA.
10	Quantidade comercializada do produto na unidade da Ceasa no estado 2018	Coluna que exibe a quantidade comercializada de um produto no estado em 2018. Cabe salientar que esta coluna é utilizada como referência para a comparação entre a quantidade vendida do produto no município e quantidade vendida do produto no estado.	Informação retirada do sistema do Conab.
11	Valor total da comercialização do produto pelas Ceasa no estado em 2018	Coluna que apresenta o valor total gerado pela comercialização do produto nas unidades do Ceasa no estado em 2018.	Informação retirada do sistema do Conab.

12	Preço médio do produto, em R$, praticado pelas Ceasa no estado em 2018	Coluna que exibe o preço médio do produto praticada pelas unidades do Ceasa no estado em 2018.	Informação retirada do sistema do Conab.
13	Quantidade comercializada do produto na unidade da Ceasa no estado 2019	Coluna que exibe a quantidade comercializada de um produto no estado em 2019. Cabe salientar que esta coluna é utilizada como referência para a comparação entre a quantidade vendida do produto no município e quantidade vendida do produto no estado.	Informação retirada do sistema do Conab.
14	Valor total da comercialização do produto pelas Ceasa no estado em 2019	Coluna que apresenta o valor total gerado pela comercialização do produto nas unidades do Ceasa no estado em 2019.	Informação retirada do sistema do Conab.
15	Preço médio do produto, em R$, praticado pelas Ceasa no estado em 2019	Coluna que exibe o preço médio do produto praticada pelas unidades do Ceasa no estado em 2019.	Informação retirada do sistema do Conab.
16	Quantidade comercializada do produto na unidade da Ceasa no estado 2020	Coluna que exibe a quantidade comercializada de um produto no estado em 2020. Cabe salientar que esta coluna é utilizada como referência para a comparação entre a quantidade vendida do produto no município e quantidade vendida do produto no estado.	Informação retirada do sistema do Conab.
17	Valor total da comercialização do produto pelas Ceasa no estado em 2020	Coluna que apresenta o valor total gerado pela comercialização do produto nas unidades do Ceasa no estado em 2020.	Informação retirada do sistema do Conab.
18	Preço médio do produto, em R$, praticado pelas Ceasa no estado em 2020	Coluna que exibe o preço médio do produto praticada pelas unidades do Ceasa no estado em 2020.	Informação retirada do sistema do Conab.
19	Quantidade comercializada do produto na unidade da	Coluna que exibe a quantidade comercializada de um produto no estado	Informação retirada do sistema do Conab.

	Ceasa no estado 2021	em 2021. Cabe salientar que esta coluna é utilizada como referência para a comparação entre a quantidade vendida do produto no município e quantidade vendida do produto no estado.	
20	Valor total da comercialização do produto pelas Ceasa no estado em 2021	Coluna que apresenta o valor total gerado pela comercialização do produto nas unidades do Ceasa no estado em 2021.	Informação retirada do sistema do Conab.
21	Preço médio do produto, em R$, praticado pelas Ceasa no estado em 2021	Coluna que exibe o preço médio do produto praticada pelas unidades do Ceasa no estado em 2021.	Informação retirada do sistema do Conab.
22	Quantidade comercializada do produto na unidade da Ceasa no estado 2022	Coluna que exibe a quantidade comercializada de um produto no estado em 2022. Cabe salientar que esta coluna é utilizada como referência para a comparação entre a quantidade vendida do produto no município e quantidade vendida do produto no estado.	Informação retirada do sistema do Conab.
23	Valor total da comercialização do produto pelas Ceasa no estado em 2022	Coluna que apresenta o valor total gerado pela comercialização do produto nas unidades do Ceasa no estado em 2022.	Informação retirada do sistema do Conab.
24	Preço médio do produto, em R$, praticado pelas Ceasa no estado em 2022	Coluna que exibe o preço médio do produto praticada pelas unidades do Ceasa no estado em 2022.	Informação retirada do sistema do Conab.
25	Quantidade de estabelecimentos	Coluna que mostra a quantidade de estabelecimentos que produzem determinado produto existente no município.	Informação retirada da visualização 6954 do SIDRA.
26	Quantidade de estabelecimentos ligados a Agricultura Familiar	Coluna que mostra a quantidade de estabelecimentos ligados à Agricultura Familiar e que produzem determinado	Informação retirada da visualização 6954 do SIDRA.

		produto existente no município.	
27	Quantidade de estabelecimentos não ligados a Agricultura Familiar	Coluna que mostra a quantidade de estabelecimentos não ligados à Agricultura Familiar e que produzem determinado produto existente no município.	Informação retirada da visualização 6954 do SIDRA.
28	Quantidade produzida por produto no município	Coluna que mostra a quantidade produzida de determinado produto no município.	Informação retirada da visualização 6954 do SIDRA.
29	Quantidade produzida por produto no município em estabelecimento ligados a Agricultura Familiar	Coluna que mostra a quantidade produzida de determinado produto no município em estabelecimentos ligados à Agricultura Familiar.	Informação retirada da visualização 6954 do SIDRA.
30	Quantidade produzida por produto no município em estabelecimento não ligados a Agricultura Familiar	Coluna que mostra a quantidade produzida de determinado produto no município em estabelecimentos não ligados à Agricultura Familiar.	Informação retirada da visualização 6954 do SIDRA.
31	Unidade de medida	Coluna que apresenta a unidade de medida utilizada para a informação da quantidade produzida.	Informação retirada da visualização 6954 do SIDRA.
32	Quantidade vendida do produto no município	Coluna que mostra a quantidade vendida de determinado produto em um município e que pode ser utilizada como referência para a comparação entre a quantidade comercializada do produto na unidade da Ceasa no estado e a quantidade vendida do produto no estado.	Informação retirada da visualização 6954 do SIDRA.
33	Quantidade vendida do produto vindo estabelecimento ligados a Agricultura Familiar no município.	Coluna que mostra a quantidade vendida de determinado produto, cuja origem vem de estabelecimentos ligados à Agricultura Familiar, em um município. Cabe salientar que esta coluna pode ser utilizada como referência para a comparação entre a quantidade	Informação retirada da visualização 6954 do SIDRA.

		comercializada do produto na unidade da Ceasa no estado e a quantidade vendida do produto no estado.	
34	Quantidade vendida do produto vindo estabelecimento não ligados a Agricultura Familiar no município.	Coluna que mostra a quantidade vendida de determinado produto, cuja origem vem de estabelecimentos não ligados à Agricultura Familiar, em um município. Cabe salientar que esta coluna pode ser utilizada como referência para a comparação entre a quantidade comercializada do produto na unidade da Ceasa no estado e a quantidade vendida do produto no estado.	Informação retirada da visualização 6954 do SIDRA.
35	Valor da produção total do produto no município	Coluna que mostra o valor da produção de determinado produto em um município.	Informação retirada da visualização 6954 do SIDRA.
36	Valor da produção dos estabelecimentos ligados a Agricultura Familiar do produto no município	Coluna que mostra o valor da produção de determinado produto, cuja origem vem de estabelecimentos ligados à Agricultura Familiar, em um município.	Informação retirada da visualização 6954 do SIDRA.
37	Valor da produção dos estabelecimentos não ligados a Agricultura Familiar do produto no município	Coluna que mostra o valor da produção de determinado produto, cuja origem vem de estabelecimentos não ligados à Agricultura Familiar, em um município.	Informação retirada da visualização 6954 do SIDRA.
38	Valor da venda do produto no município	Coluna que mostra o valor da venda de determinado produto em um município.	Informação retirada da visualização 6954 do SIDRA.
39	Valor da venda do produto, vindo de estabelecimentos ligados a Agricultura Familiar, no município	Coluna que mostra o valor da venda de determinado produto, cuja origem vem de estabelecimentos ligados à Agricultura Familiar, em um município.	Informação retirada da visualização 6954 do SIDRA.
40	Valor da venda do produto, vindo de estabelecimentos não ligados a Agricultura	Coluna que mostra o valor da venda de determinado produto, cuja origem vem de estabelecimentos não ligados à	Informação retirada da visualização 6954 do SIDRA.

	Familiar, no município	Agricultura Familiar, em um município.	
41	Grupo 1 - Número de Estabelecimentos com até 10 ha	Coluna que apresenta a quantidade de estabelecimentos com até 10 ha que existem em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
42	Grupo 2 - Número de Estabelecimentos de 10 até 100 ha	Coluna que apresenta a quantidade de estabelecimentos que tenham de 10 ha até 100 ha, existentes em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
43	Grupo 3 - Número de Estabelecimento de 100 até 1000 ha	Coluna que apresenta a quantidade de estabelecimentos que tenham de 100 ha até 1000 ha, existentes em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
44	Grupo 4 - Número de Estabelecimento com mais de 1000 ha	Coluna que apresenta a quantidade de estabelecimentos que tenham mais de 1000 ha, existentes em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
45	Grupo 5 - Número de Estabelecimentos com Produtor sem área	Coluna que apresenta a quantidade de estabelecimentos dos quais o produtor não é dono da terra por produto.	Informação proveniente do filtro por grupo de área total na tabela 6954 do SIDRA.
46	Grupo 1 - Número de Estabelecimentos ligados à Agricultura Familiar com até 10 ha	Coluna que apresenta a quantidade de estabelecimentos ligados à Agricultura Familiar com até 10 ha que existem em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
47	Grupo 2 - Número de Estabelecimentos ligados à Agricultura Familiar de 10 até 100 ha	Coluna que apresenta a quantidade de estabelecimentos ligados à Agricultura Familiar que tenham de 10 ha até 100 ha, existentes em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
48	Grupo 3 - Número de Estabelecimento ligados à Agricultura Familiar de 100 até 1000 ha	Coluna que apresenta a quantidade de estabelecimentos ligados à Agricultura Familiar que tenham de 100 ha até 1000 ha, existentes em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
49	Grupo 4 - Número de Estabelecimento ligados à	Coluna que apresenta a quantidade de estabelecimentos ligados à Agricultura	Informação obtida através do agrupamento de dados

	Agricultura Familiar com mais de 1000 ha	Familiar que tenham mais de 1000 ha, existentes em um município.	presentes na visualização 6954 no sistema SIDRA
50	Grupo 5 - Número de Estabelecimentos ligados à Agricultura Familiar com Produtor sem área	Coluna que apresenta a quantidade de estabelecimentos ligados à Agricultura Familiar dos quais o produtor não é dono da terra por produto.	Informação proveniente do filtro por grupo de área total na tabela 6954 do SIDRA.
51	Grupo 1 - Número de Estabelecimentos não ligados à Agricultura Familiar com até 10 ha	Coluna que apresenta a quantidade de estabelecimentos não ligados à Agricultura Familiar com até 10 ha que existem em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
52	Grupo 2 - Número de Estabelecimentos não ligados à Agricultura Familiar de 10 até 100 ha	Coluna que apresenta a quantidade de estabelecimentos não ligados à Agricultura Familiar que tenham de 10 ha até 100 ha, existentes em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
53	Grupo 3 - Número de Estabelecimento não ligados à Agricultura Familiar de 100 até 1000 ha	Coluna que apresenta a quantidade de estabelecimentos não ligados à Agricultura Familiar que tenham de 100 ha até 1000 ha, existentes em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
54	Grupo 4 - Número de Estabelecimento não ligados à Agricultura Familiar com mais de 1000 ha	Coluna que apresenta a quantidade de estabelecimentos não ligados à Agricultura Familiar que tenham mais de 1000 ha, existentes em um município.	Informação obtida através do agrupamento de dados presentes na visualização 6954 no sistema SIDRA
55	Grupo 5 - Número de Estabelecimentos não ligados à Agricultura Familiar com Produtor sem área	Coluna que apresenta a quantidade de estabelecimentos não ligados à Agricultura Familiar dos quais o produtor não é dono da terra por produto.	Informação proveniente do filtro por grupo de área total na tabela 6954 do SIDRA.

Fonte: O autor

7 CONSIDERAÇÕES FINAIS

No decorrer deste projeto, conseguiu-se aprender sobre o desenvolvimento de sistemas de extração de informação com ferramentas estudadas no capítulo 3, além de aprender as características específicas de cada ferramenta, o que facilita a tarefa de decidir a melhor ferramenta para cada situação específica. Neste trabalho utilizou-se o Selenium principalmente pelo fato de que o sistema de construção de consulta do SIDRA, requer o uso da função de *drag_and_drop*, que pode ser aplicada com o uso da biblioteca *ActionChains*.

Além disto, este trabalho conseguiu implementar o processo de *Web Scraping* nos sites inicialmente elencados, fazendo com que fosse possível acessar vários dados relativos ao contexto da produção e comercialização dos produtos agrícolas do país.

Cabe salientar que a base de dados apresentada no capítulo representa parte do que será a base de dados que pretende-se produzir, e isto decorre de fato de que para a construção da base completa é necessário realizar uma busca completa nos sites elencados, o que geraria uma carga de requisições grande, principalmente no SIDRA. Esta carga de requisições deve ser tratada de modo que o sistema de extração de dados não seja considerado com um ataque de negação de serviço, uma vez que realiza muitas requisições, pois isso pode gerar um bloqueio do IP da máquina de origem das requisições.

Pretende-se continuar o desenvolvimento desta base de dados, com a inclusão de novas informações provenientes de outros sites, por isso no subcapítulo a seguir, será apresentado as ideias para continuidade do desenvolvimento da base de dados.

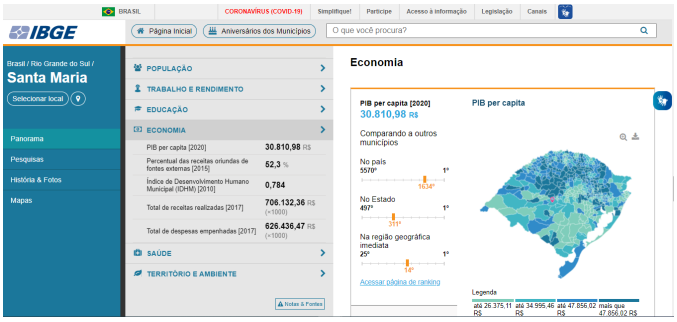
Por fim, é importante frisar o fato de que os dados atualmente presentes na base de dados construída já permitem ter um olhar integrado de contexto agrícola, permitindo uma análise mais profunda a respeito da produção e comercialização dos produtos agrícolas, uma vez que apresenta aos pesquisadores esses dados já unidos. Tal ideia vale principalmente para os pesquisadores focados na região central do estado, uma vez que atualmente os dados presentes na base de dados estão focados nos produtos de Horticultura.

7.1 CONTINUIDADE DO TRABALHO

Como citado anteriormente, pretende-se dar prosseguimento no desenvolvimento da base de dados, para tanto elencou-se alguns sistemas para os quais é necessário analisar se os dados presentes neles são realmente relevantes e caso sejam, é preciso examinar a estrutura de montagem de consulta, a estrutura de retorno dos dados e elaborar uma estratégia para a extração destes dados e a posterior junção deles na base de dados atual.

As Figuras 54 e 55 apresentam dois sites que estão elencados para a continuidade deste trabalho, são eles o site do Cidades@ IBGE e a tabela 6956 do SIDRA.

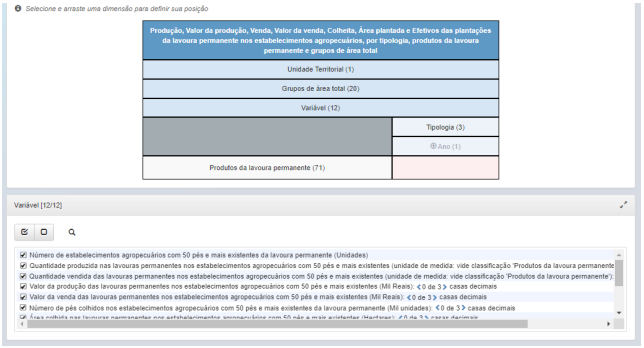
Figura 54 - Cidades@ IBGE e outros dados dos municípios



Fonte: O autor

O site Cidades@ IBGE possui dados estimativos da população de 2021, além de possuir dados relativos à Economia e à Educação do município. Alguns exemplos destes dados são: PIB per capita, Índice de Desenvolvimento Humano Municipal (IDHM), salário médio mensal dos trabalhadores formais, taxa de escolarização de 6 a 14 anos de idade, IDEB – anos iniciais e dos anos finais do ensino fundamental (Rede pública) e etc.

Figura 55 - SIDRA Tabela 6956



Fonte: O autor

A tabela 6956 do SIDRA possui uma estrutura, no geral, similar a tabela 6954, porém possui um foco nos produtos da Fruticultura, além de possuir outras variáveis, sendo elas: o número de estabelecimentos agropecuários com 50 pés e mais existentes da lavoura permanente (Unidades), a quantidade produzida nas lavouras permanentes nos estabelecimentos agropecuários com 50 pés e mais existentes, a quantidade vendida das lavouras permanentes nos estabelecimentos agropecuários com 50 pés e mais existentes, o valor da produção das lavouras permanentes nos estabelecimentos agropecuários com 50 pés e mais existentes (Mil Reais), o valor da venda das lavouras permanentes nos estabelecimentos agropecuários com 50 pés e mais existentes (Mil Reais), número de pés colhidos nos estabelecimentos agropecuários com 50 pés e mais existentes da lavoura permanente (Mil unidades), a área colhida das lavouras permanentes nos estabelecimentos agropecuários com 50 pés e mais existentes (Hectares), área total existente na data de referência nas lavouras permanentes nos estabelecimentos agropecuários com 50 pés e mais existentes (Hectares), o número de pés existentes nos estabelecimentos agropecuários com 50 pés e mais existentes da lavoura permanente (Mil unidades), o número de estabelecimentos agropecuários com menos de 50 pés existentes da lavoura permanente (Unidades), o número de pés existentes nos estabelecimentos agropecuários com menos de 50 pés existentes da lavoura permanente (Mil unidades) e o valor da venda das lavouras permanentes nos estabelecimentos agropecuários com menos de 50 pés existentes (Mil Reais). É importante destacar que os filtros presentes nesta tabela são iguais aos filtros presentes na tabela 6954.

Além destes sites citados anteriormente, pretende-se adicionar novas colunas a partir de um processamento de algumas informações presentes na atual versão da base de dados, por exemplo, adicionar uma coluna que apresenta a porcentagem que o número de estabelecimentos ligados a Agricultura Familiar representa no total de número de estabelecimentos em um determinado município.

Por fim, cabe reforçar que os próximos passos do desenvolvimento deste trabalho serão decididos em conjunto com os membros GIPAG, ou seja, as definições de novas colunas e de novos sites para se extrair dados, pois o objetivo é melhorar cada vez mais a base de dados desenvolvida até o momento neste trabalho, para que ela gradualmente torne-se uma base com um volume de dados significativos e realmente relevantes para que assim, esses dados possam ser tornar

o pilar para o desenvolvimento de pesquisas e consecutivamente influenciar na tomada de decisões relativas ao contexto agrícola.

REFERÊNCIAS

BITTENCOURT, Amanda Suarez *et al*. Sistema de Informações Georreferenciado para o Gerenciamento da Fruticultura e da Olericultura. *In*: SEMANA GEOMÁTICA, 6., 2016, Santa Maria. **Anais eletrônicos** [...]. Santa Maria: UFSM, 2016. ISSN: 2179-4243. Disponível em: https://www.politecnico.ufsm.br/vigeomatica/anais_vi_geomatica.pdf. Acesso em: 01 ago. 2022

FERRARA, Emilio *et al*. Web data extraction, applications and techniques: A survey. **ACM Transactions on Intelligent Systems and Technology**, *[S.l.]*, v. 11, n. 2, 35 p., jan. 2020.

LAENDER, Alberto H. F. *et al*. A Brief Survey of Web Data Extraction Toolst. **Special Interest Group on Management of Data (SIGMOD)**, Nova York, v. 31, n. 2, p. 84-93, jun. 2002. ISSN:0163-5808.

LAGE, Juliano Palmieri *et al*. Automatic generation of agents for collecting hidden Web pages for data extraction. **Data & Knowledge Engineering**, *[S. l.]*, v. 49, n. 2, p. 177-196, mai. 2004. ISSN: 0169-023X. DOI: 10.1016/j.datak.2003.10.003.

LUIZ, Alfredo José Barreto; MAIA, Aline de Holanda Nunes. Análise gráfica dos dados do Censo Agropecuário: subsídios para determinação de calendários agrícolas. **Revista da Estatística Universidade Federal de Ouro Preto (UFOP)**, Ouro Preto, v. 3, n. 3, p. 174-177, mai. 2014. ISSN: 2237-8111. Disponível em: https://periodicos.ufop.br/rest/article/view/3395/2658. Acesso em: 01 ago. 2022

MITCHELL, Ryan. **Web Scraping com Python**. 2. ed. Tradução: Lúcia A. Kinoshita. São Paulo: Novatec, 2018. Título original: Web Scraping with Python. ISBN 978-1-49198-557-1

MOENS, Marie-Francine. **Information Extraction**: Algorithms and Prospects in a Retrieval Context. Bélgica: Springer: 2006.

SWEIGART, Al. **Automatize tarefas maçantes com Python**. Tradução: Lúcia A. Kinoshita. São Paulo: Novatec, 2015. Título original: Automate the Boring Stuff with Python. ISBN: 978-1-59327-599-0

ZHANG, Shou; BALOG, Krisztian. Web Table Extraction, Retrieval, and Augmentation: A Survey. **Knowledge-Based Systems**, *[S.l.]*, vol. 70, p. 301-323, jan. 2014.