UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Eduardo Andrello Yasuda

Desenvolvimento de ferramenta computacional para levantamento de dados paramétricos para auxílio em projeto conceitual

São Carlos

2022

Eduardo Andrello Yasuda

Desenvolvimento de ferramenta computacional para levantamento de dados paramétricos para auxílio em projeto conceitual

Monografia apresentada ao Curso de Engenharia Aeronáutica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Aeronáutico.

Orientador: Prof. Dr. Álvaro Martins Abdalla

São Carlos

2022

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Y11d

Yasuda, Eduardo Andrello
Desenvolvimento de ferramenta computacional para levantamento de dados paramétricos para auxílio em projeto conceitual / Eduardo Andrello Yasuda; orientador Álvaro Martins Abdalla. São Carlos, 2022.

Monografia (Graduação em Engenharia Aeronáutica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2022.

1. Python. 2. Projeto Conceitual. 3. Web Scraper. 4. Características de aeronaves. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO
Approval sheet

Candidato / *Student*: Eduardo Andrello Yasuda
Título do TCC / *Title* : Desenvolvimento de Ferramenta Computacional para levantamento de dados paramétricos para auxílio em projeto conceitual
Data de defesa / *Date*: 20/07/2022

Comissão Julgadora / *Examining committee*	**Resultado** / *result*
Professor Doutor Fernando Martini Catalano	APROVADO
Instituição / Affiliation: EESC - SAA	
Professor Doutor Manoel Rodrigues Alves	Aprovado
Instituição / Affiliation: USP - IAU	

Presidente da Banca / Chair of the Examining Committee:

Professor Doutor Fernando Martini Catalano
(assinatura / *signature*)

Aos meus amigos e familiares que me acompanharam e apoiaram ao longo dos últimos anos

AGRADECIMENTOS

Aos meus pais, Eduardo e Fátima, por toda ajuda e suporte incondicional dados durante todos os anos e que foram essenciais para a realização deste trabalho.

Ao meu tio Felício, que sempre me lembrou das minhas prioridades em todos os nossos encontros.

A todos os meus amigos, a quem tive o prazer de conhecer e compartilhar esses últimos 6 anos.

"Everything can be taken from a man but one thing:
the last of the human freedoms—to choose one's attitude
in any given set of circumstances, to choose one's own way."
Viktor E. Frankl

RESUMO

YASUDA, E. A. **Notas sobre o desenvolvimento de base de dados para aeronaves de combate**. 2022. 68p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2022.

Este trabalho tem como objetivo auxiliar os futuros alunos da matéria de Projetos de Aeronaves I a obter dados do maior número de aeronaves possível. Para tal, foi necessária a criação de um programa cuja função é conhecida como "web scraper", utilizando a linguagem de programação Python e o site de pesquisa Wikipédia. O escopo da primeira versão do programa é a criação de uma base de dados com informações sobre aeronaves de combate e suas principais características. Caso sejam necessárias informações sobre outros tipos de aeronaves, tal programa pode ser modificado livremente para atender as necessidades do projeto.O resultado do trabalho foi a criação de uma tabela em Excel contendo dados como nome, tripulação, comprimento, envergadura, peso, potência, entre outros dados de 1.307 aeronaves (incluindo aeronaves fora de operação). A partir desta tabela é possível filtrar as características desejáveis e aplicáveis para a disciplina de Projetos de Aeronaves I.

Palavras-chave: python; web scraper; aeronaves de combate; características de aeronaves; projeto conceitual.

ABSTRACT

YASUDA, E. A. **Development of computational tool to collect parametric data in aid to conceptual projects**. 2022. 68p. Monograph (Conclusion Course Paper) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2022.

This work aims to help future students of the Aircraft Projects I subject to obtain data from as many aircraft as possible. For this, it was necessary to create a program whose function is known as "web scraper", using the Python programming language and the Wikipedia research site. The scope of the first version of the program is the creation of a database with information about combat aircraft and their main characteristics. If information on other types of aircraft is required, this program can be freely modified to meet the project's needs. The result of the work was the creation of an Excel table containing data such as name, crew, length, wingspan, weight, power, among other data for 1,307 aircraft (including out-of-operation aircraft). From this table, it is possible to filter the desirable and applicable characteristics for the Aircraft Projects I discipline.

Keywords: python; web scraper; fighter aircraft; aircraft characteristics; conceptual project.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

HTML	HyperText Markup Language
API	Application Programming Interface
bs4	Beautiful Soup
na	Not available
Regex	Regular Expressions
USP	Universidade de São Paulo
USPSC	Campus USP de São Carlos

SUMÁRIO

1 INTRODUÇÃO

1.1 Escopo do trabalho

Este trabalho teve como objetivo a criação de uma ferramenta para o levantamento automatizado de dados sobre aeronaves de combate, de forma a auxiliar a extração de informações para a disciplina de Projetos de Aeronaves I. Para isso, foi desenvolvido um programa na linguagem de programação Python 3.10 capaz de acessar e extrair dados de páginas da web. A fonte de informações utilizadas para o projeto foi o site de pesquisas Wikipédia, utilizado devido à sua ampla base de aeronaves e melhor disposição de informações.

A criação de uma base de dados também poderia ser feita de forma manual. No entanto, o trabalho seria exaustivo e a base teria que ser periodicamente manualmente revisada. Tarefa que, apesar de possível, não geraria valor. O tempo gasto por um membro do grupo para atualizar uma base seria melhor utilizado na estruturação do projeto ou auxiliando outras frentes em análises ou em sessões de problem solving.

1.2 Revisão Bibliográfica

Python é uma linguagem de programação high-level desenhada para ser entendida e escrita de forma simples por humanos e de fácil leitura e processamento para computadores, amplamente usada por desenvolvedores e pesquisadores. Atualmente existem inúmeras aplicações da linguagem, que, se descritas nesta revisão bibliográfica, fugiriam do escopo do trabalho. A revisão bibliográfica conterá uma breve história de sua criação, sua estrutura básica e principais aplicações no desenvolvimento do trabalho.

Desenvolvido em 1991 por Guido van Russum, programador e matemático holandês, o Python é uma linguagem de programação de alto-nível e de uso geral, amplamente utilizada por desenvolvedores e pesquisadores de software. É uma linguagem fácil de ser entendida e aprendida, também é uma linguagem open-source, significando que todos da comunidade podem contribuir para seu desenvolvimento.

No prefácio do livro "Programming Python"Lutz (1996, p.6, tradução nossa), Guido diz:

> "Há mais de seis anos, em dezembro de 1989, eu estava procurando por um projeto de programação como "hobby"que me mantivesse ocupado durante a semana próxima ao Natal. Meu escritório estaria fechado, mas eu tinha um computador em casa e não muito mais do que isso em mãos. Eu decidi escrever um interpretador para a nova linguagem de scripting sobre a qual eu vinha pensando

> ultimamente: uma descendente da ABC que agradaria a hackers de Unix/C. Escolhi Python como um título provisório para o projeto, sendo que eu estava num humor um pouco irreverente (e sendo também um grande fã do Monty Python's Flying Circus)"

1.2.1 Python Reserved Words

Como toda forma de comunicação, o Python também tem seu vocabulário próprio, apesar de se limitar apenas a algumas palavras quando comparado ao vocabulário humano, essas palavras são chamadas *Reserved Words*. Elas tem um único significado para o programa e não podem ser utilizadas para nomear variáveis.

Reserved Words (vocabulário reservado) é são principal forma de comunicação entre o programador e o programa, toda vez que uma *reserved word* é identificada, o programa a executará. No Quadro 1, a lista das palavras que compõem esse vocabulário.

Quadro 1 – Python Reserved words

and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	
class	finally	is	return	
continue	for	lambda	try	
def	from	nonlocal	while	

Fonte: Severance (2009, p. 5)

1.2.2 Principais funções usadas neste trabalho

1.2.2.1 *Import*

Import é uma das *reserved words* cuja função é importar um módulo para o programa atual. Um módulo é um arquivo/programa pode conter funções, classes, variáveis e até uma rotina pré-determinada para ser executada. Alguns desses módulos precisam ser instalados para poderem ser utilizados no programa, outros já são automaticamente instalados junto ao Python (*Python Standard Libary*).

Para o programa ser mais eficiente e evitar o uso constante da função *import*, cada módulo é importado somente uma vez por sessão e suas funções e rotinas podem ser usadas livremente. Também é possível escrever um módulo personalizado para determinada tarefa e importá-lo para uso em outro programa.

Os módulos utilizados no programa estão listados a seguir:

- Selenium

- Selenium webdriver
- Beautiful Soup
- Requests
- Pandas
- re

As funções de cada módulo serão detalhadas ao longo do trabalho. A sintaxe utilizada para importar cada módulo está listada no Quadro 2.

Quadro 2 – Módulos utilizados

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import requests
import pandas as pd
import re
```

Fonte: Autor

1.2.2.2 *For* loops

Um *loop* contendo a função *for* é utilizado para realizar iterações sobre uma sequência de dados (por exemplo uma lista de números ou nomes).

> *"Rather than always iterating over an arithmetic progression of numbers (like in Pascal), or giving the user the ability to define both the iteration step and halting condition (as C), Python's for statement iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence." Silva, J. R. (2020).*

Um exemplo de como um loop iterativo funciona pode ser observado na figura 1.

Figura 1 – Exemplo de um loop

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Fonte: w3 Schools (2022)

Neste exemplo também são apresentadas duas novas funções: *if* e *continue*. Essas funções são utilizadas com frequência durante o desenvolvimento do programa. No exemplo, a ideia principal do programa é 'imprimir' todos os itens da lista 'frutas' exceto os que são "banana". Isso é feito da seguinte forma:

- Primeiro define-se os itens que entrarão na variável 'fruits', que por definição é uma lista;
- Com a lista pronta, a função; *for* começará a iteração com cada item da lista individualmente e de forma sequencial, armazenando cada item sob a variável 'x';
- Em seguida, o item armazenado sob a variável 'x' é comparado com a palavra 'banana';
- *Se* a variável for igual a 'banana', a função *continue* é acionada, o que fará com que o loop passe para a próxima palavra, sem "imprimir"a palavra 'banana';
- Caso a palavra seja diferente de 'banana', ela sera 'impressa'.

Um exemplo prático deste *loop*, aplicado no desenvolvimento do programa, é exibido na figura 2.

Figura 2 – *For loop* com função *if* e *continue*

```
# Identifica todos os links que direcionan para páginas de aeronaves da tabela
for link in table.select("td:nth-of-type(1) a "):
    # Cria e adiciona todos os links de aeronaves na lista 'Links_list'
    url = BASE_URL + link['href']
    if link['href'] == "/wiki/HAL_FGFA": continue
    Links_list.append(url)
```

Fonte: Autor

Neste trecho, o objetivo do código é montar uma lista com todos os links presentes em uma tabela da web, excluindo o link cuja url termina em /wiki/HAL_FGFA, pois se trata de uma aeronave repetida.

1.2.2.3 *Try* e *Except*

Um programa em Python é executado linha por linha pelo interpretador do programa. Caso exista algum erro, o programa é completamente interrompido e uma mensagem de erro ('*Traceback*') é exibida, de forma que a próxima linha de comando não é executada. Um exemplo de situação em que esse tipo de problema pode ocorrer é mostrado na figura 3.

Figura 3 – Exemplo de código que pode gerar uma interrupção

```
inp = input('Enter Fahrenheit Temperature: ')
fahr = float(inp)
cel = (fahr - 32.0) * 5.0 / 9.0
print(cel)

# Code: http://www.py4e.com/code3/fahren.py
```

Fonte: Severance (2009)

O código da figura 3 exige que uma entrada seja dada para a conversão da temperatura de Fahrenheit para Celsius. O esperado é que um número seja dado como entrada, no entanto se a entrada for uma palavra, o código exibirá uma mensagem de erro, conforme a figura 4.

Figura 4 – Exemplo de entrada que gera a interrupção do programa

```
python fahren.py
Enter Fahrenheit Temperature:fred
Traceback (most recent call last):
  File "fahren.py", line 2, in <module>
    fahr = float(inp)
ValueError: could not convert string to float: 'fred'
```

Fonte: Severance (2009)

A estrutura condicional *Try* e *Except* existe para tratar destes erros, que podem ser esperados ou inesperados, de forma que a execução do programa não seja interrompida. Portanto, essa estrutura é utilizada em casos onde a probabilidade de um erro ocorrer é conhecida e uma outra sequência de códigos seja executada caso o erro aconteça. Essa sequência extra de códigos é ignorada caso não exista erro na primeira.

O programa começa tentando executar a primeira função (*Try*), caso não ocorra nenhum erro, o programa ignora a segunda função (*Except*). Caso ocorra um erro na primeira tentativa, o programa executará a segunda parte.

No exemplo dado, essa estrutura ficaria conforme a figura 5.

Figura 5 – Exemplo de estrutura *Try/Except*

```
inp = input('Enter Fahrenheit Temperature:')
try:
    fahr = float(inp)
    cel = (fahr - 32.0) * 5.0 / 9.0
    print(cel)
except:
    print('Please enter a number')

# Code: http://www.py4e.com/code3/fahren2.py
```

Fonte: Severance (2009)

Com isso, existem 2 possibilidades de comportamento do código. A primeira, caso um número seja dado como entrada, seria executá-lo normalmente e a segunda, caso haja um erro, é exibir uma mensagem pedindo para que um número seja inputado. A figura 6 mostra os 2 possíveis cenários.

Figura 6 – Exemplo de respostas utilizando *Try/Except*

```
python fahren2.py
Enter Fahrenheit Temperature:72
22.22222222222222

python fahren2.py
Enter Fahrenheit Temperature:fred
Please enter a number
```

Fonte: (SEVERANCE, 2009)

No programa, a aplicação do *Try/Except*, pode ser vista na figura 8.

Figura 7 – Exemplo de estrutura *Try/Except* aplicada no programa

```
# ----------------------------------------
if re.match('Wingspan:', text):
    if wingspan_count == 1:
        pass
    else:
        wingspan_count = 1
        try:
            wingspan1 = re.findall('([0-9]+.?[0-9]+) [m]', text)
            wingspan = float(wingspan1[0])
            Wingspan_list.append(wingspan)
            print(wingspan)
        except:
            Wingspan_list.append("na")
```

Fonte: Autor

O objetivo deste trecho do programa é criar uma lista contendo as envergaduras das aeronaves. No entanto, nem todas as aeronaves possuem essa informação disponível. Então, caso não seja possível extrair essa informação, o programa executará a função *Except*, onde o valor 'na' será adicionado na lista de envergaduras.

1.2.2.4 *Regular Expressions*

Regular expressions, ou Regex, foram amplamente utilizado neste trabalho. Elas possibilitam a identificação dos dados que são necessários, a partir da descrição dos caracteres que compõem a informação buscada. *"Regular Expressions are a notation for describing sets of character strings. When a particular string is in the set described by a regular expression, we often say that the regular expression matches the string."* Cox, R (2007)

Um exemplo do uso de Regex pode ser observado na Figura 8.

Figura 8 – Exemplo de uso das *Regular Expressions*

```
177             try:
178                 weight1 = re.findall('([0-9]+,?[0-9]+) [kg]', text)
179                 weight = weight1[0]
180                 Empty_weight_list.append(weight)
181                 print(weight)
182             except:
183                 Empty_weight_list.append("na")
```

Fonte: Autor

Neste exemplo, a Regex foi empregada na linha 178 para identificar o valor do peso vazio da aeronave. Lê-se: Extrair (todos os dados caracteres que estão dentro dos parênteses) os caracteres que contém 1 ou mais dígitos de 0 a 9, seguidos ou não de ",", seguidos de 1 ou mais dígitos de 0 a 9.

A tabela 1 ilustra os símbolos utilizados pelas *Regular Expressions*.

Tabela 1 – Símbolos utilizados pelas Regex

Symbol(s)	**Meaning**	**Example**	**Example Matches**
*	Matches the preceding character, sub expression, or bracketed character, 0 or more times	a*b*	aaaaaaaa, aaabbbbb, bbbbbb
+	Matches the preceding character, subexpression, or bracketed character, 1 or more times	a+b+	aaaaaaaab, aaabbbbb, abbbbbb
[]	Matches any character within the brackets (i.e., "Pick any one of these things"	[A-Z]*	APPLE, CAPITALS, QWERTY
()	A grouped subexpression (these are evaluated first, in the "order of operations" of regular expressions)	(a*b)*	aaabaab, abaaab, ababaaaaab
{m,n}	Matches the preceding character, subexpression, or bracketed character between m and n times (inclusive)	a{2,3}b{2,3}	aabbb, aaabbb, aabb
[^]	Matches any single character that is not in the brackets	[Â-Z]*	apple, lowercase, qwerty
|	Matches any character, string of characters, or subexpression, separated by the "I" (note that this is a vertical bar, or "pipe," not a capital "i")	b(a|i|e)d	bad, bid, bed
.	Matches any single character (including symbols, numbers, a space, etc.	b.d	bad, bzd, b$d, b d
^	Indicates that a character or subexpression occurs at the beginning of a string	^a	apple, asdf, a
	An escape character (this allows you to use "special" characters as their literal meaning)	.	.
$	Often used at the end of a regular expression, it means "match this up to the end of the string." Without it, every regular expression has a defacto ".*" at the end of it, accepting strings where only the rst part of the string matches. This can be thougt of as analogous to the $^{s}ymbol$.	[A-Z]*[a-z]*$	ABCabc, zzzyx, Bob
?!	"Does not contain." This odd pairing of symbols, immediately preceding a character (or regular expression), indicates that that character should not be found in that specic place in the larger string. This can be tricky to use; after all, the character might be found in a different part of the string. If trying to eliminate a character entirely, use in conjunction with a ^ and $ at either end.	^((?![A-Z]).)*$	no-caps-here, $ymb0ls a4e f!ne

Fonte: Mitchell, R (2015, p.27)

1.2.3 Web Scraping

1.2.3.1 O que é?

Web Scraping é um método automático para obter grandes quantidades de informações de *websites*. A maioria desses dados são desestruturados e em formato HTML, que após serem separados são convertidos e estruturados em uma planilha ou base de dados para serem usados em outras aplicações. Existem várias maneiras de performar um *web scraping* para obter dados de sites da internet. Elas incluem serviços online, API's ou criando um código manualmente do zero. Muitos sites famosos, como Google, Twitter, Facebook, StackOverflow, etc, possuem API's que permitem ao usuário acessar dados de maneira estruturada. Essa seria a melhor alternativa, no entanto existem sites que não permitem que usuários acessem grandes quantidades de dados de maneira estruturada ou simplesmente não possuem a opção de mostrar os dados de forma estruturada. Nessas situações, a melhor alternativa é usar um *Web Scraper* para extrair os dados desses sites.

1.2.3.2 Usos do Web Scraping

Algumas aplicações do uso de *web scrapers* incluem, mas não se limitam a Geeksforgeeks (2021):

- **Monitoramento de Preços**: Empresas podem utilizar a ferramenta para coletar dados de seus produtos e de competidores, assim como observar como sua estratégia de preços está sendo impactada. Podendo usar os dados coletados para ajustar o *pricing* de seus produtos para o ponto ótimo, obtendo maiores receitas.

- **Pesquisas de Mercado**: *Web scraping* pode ser utilizado por empresas de pesquisa de mercado. Dados de alta qualidade obtidos em grandes volumes podem ser muito úteis para empresas analisarem tendências de consumo e melhor compreender qual direção deve ser tomada no futuro.

- **Monitoramento de notícias**: Extrair dados de sites de notícias facilitam a criação de relatórios detalhados sobre determinada notícia para uma empresa. Isso é ainda mais relevante para empresas que frequentemente estão nos noticiários ou que dependem de notícias diárias para seu funcionamento.

- **Análise de Sentimento**: Para entender o sentimento geral dos consumidores em relação aos seus produtos, empresas podem utilizar *Web Scraping*. Isso ajuda no desenvolvimento de produtos que os usuários anseiam, gerando uma vantagem competitiva.

- **Marketing por E-mail**: O uso de *Web Scraping* para propagandas por email é um tanto controverso. Empresas podem pegar listas de emails de diversos websites para enviar emails promocionais e de marketing para todas as pessoas listadas.

1.2.4 Python para *Web Scraping*

Atualmente, Python é uma linguagem popular para se desenvolver um *web scraper*, pois consegue lidar facilmente com os processos envolvidos. Seja por meio de *libaries* como a *Beautiful Soup* ou *Selenium*, seja por meio da ampla comunidade que compartilha dicas, experiências e também tira dúvidas de novos usuários.

A *Beautiful Soup* (bs4) é uma poderosa ferramenta para *web scraping*. Com ela é possível criar uma *parse tree* que pode ser usada para extrair dados em HTML de um *web site*. Como dito anteriormente, a maioria dos dados dos sites não é estruturada, o que dificulta a leitura e identificação dos dados que estamos procurando. Esta *libary* estrutura todo o código HTML de um site e permite a localização de itens específicos de maneira mais simples. Geeksforgeeks (2021)

2 DESENVOLVIMENTO

2.1 Metodologia

Para desenvolver o programa de forma estruturada e simplificada, duas frentes de programação foram criadas: a primeira analisa a página da web que e cria uma lista com todas as aeronaves de combate e a segunda, responsável por acessar cada link dessa lista, que são as páginas das aeronaves, e extrair os dados necessários. Ambas as partes serão detalhadas a seguir, juntamente com os pacotes necessários para a sua execução.

Todos os dados coletados de cada acesso aos links das páginas obtidas foram salvos em listas geradas pelo programa, em seguida transformados em um *dataframe*. Os dados do *dataframe* foram transformados em uma planilha de Excel para futura análise. Por se tratar de um grande volume de informação extraída, alguns *outputs* apresentaram divergências com os valores reais, principalmente devido à forma como algumas informações são codificadas nas páginas, o que dificulta o programa a identificar a informação correta.

Para contornar esse problema, que se repete algumas vezes ao longo das análises, foi implementado o método *'Try/Except'*. Esse método consiste em simplesmente tentar extrair o dado uma vez (*'Try'*) e, caso a extração apresente algum erro, uma subrotina com 100% de certeza de funcionamento é acionada (*'Except'*). Neste caso a rotina *'Except'* retornará o valor 'na' e o adicionará à base de dados. O programa foca em extrair os seguintes dados sobre as aeronaves:

- Tripulação (Crew) [Pilotos]
- Comprimento (Length) [m]
- Envergadura (Wingspan) [m]
- Altura (Height) [m]
- Área da Asa (Wing Aerea) [m^2]
- Peso Vazio (Empty Weight) [kg]
- MTOW [Kg]
- Velocidade Máxima (Maximum Speed) [km/h]
- Velocidade de Cruzeiro (Cruise Speed)[km/h]
- Teto de Serviço (Service Ceiling)[m]
- Tração(Thrust) [kN] ou Potência (Power) [hp]

Uma tabela para avaliação, como a Tabela 4, foi gerada para avaliar o desempenho geral do programa.

Tabela 2 – Avaliação da performance do programa

Dado	**Crew**	**Length**	**Wingspan**	**Height**	**Wing Aerea**	**Empty Weight**
Quantidade	1307	1307	1307	1307	1307	1307
Dados encontrados	xx	xx	xx	xx	xx	xx
Dados não encontrados	xx	xx	xx	xx	xx	xx
Máximo	xx	xx	xx	xx	xx	xx
Mínimo	xx	xx	xx	xx	xx	xx

Fonte: Análise própria

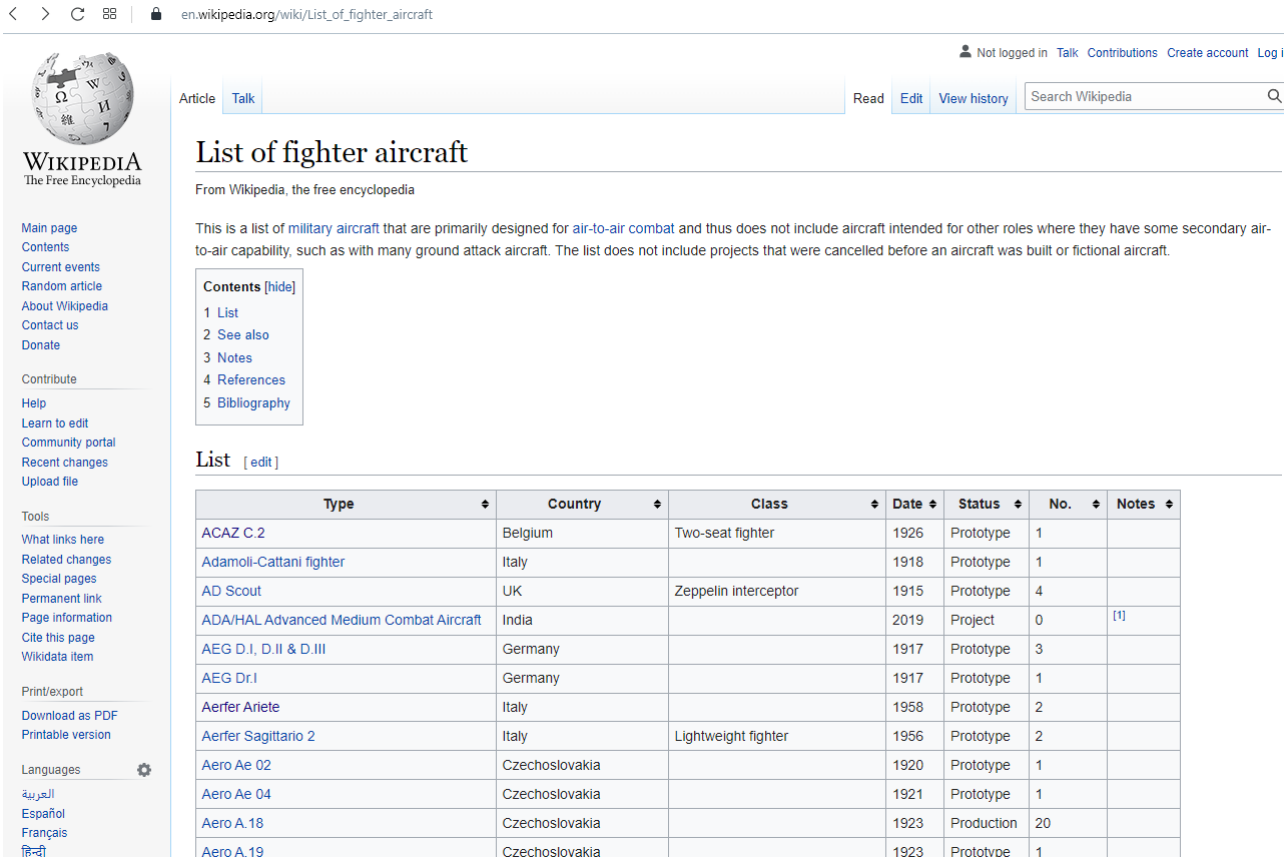
Como uma grande quantidade de aeronaves foi levantada e avaliada pelo programa, a verificação de erros uma a uma seria contra-produtiva. Como alternativa, foi escolhido avaliar os resultados máximos e mínimos de cada item e verificar se os valores divergem da faixa esperada.

2.2 Identificação e criação da lista de aeronaves

2.2.1 Acesso à pagina da web

O funcionamento da ferramenta depende do acesso à pagina que contém a tabela com os nomes e links das aeronaves que desejamos extrair as informações. A página que contém essas informações pode ser encontrada em "https://en.wikipedia.org/wiki/List_-of_fighter_aircraft", que é exibida na Figura 9.

Figura 9 – Página de referência para extração dos links



Fonte: (WIKIPEDIA, 2022b)

O programa acessa a página utilizando os pacotes *Selenium* e *Beautiful Soup*. A função desses pacotes é garantir o acesso e sua leitura da página desejada, respectivamente. As linhas que representam essa seção do programa estão detalhadas na Figura 10, com comentários acima de cada linha de comando, indicados pelo símbolo #.

Figura 10 – Código para acesso à pagina da web

```
#Setup selenium
chrome_options = Options()

chrome_options.add_argument("--headless") # faz com que o browser não abra durante o processo

# instala automaticamente o webdriver para o Google Chrome, necessário para acessar a web
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
#-------------------------------------------------------------------------------

BASE_URL = "https://en.wikipedia.org"

#Acessa a página principal, que contém a lista de aeronaves
page = requests.get("https://en.wikipedia.org/wiki/List_of_fighter_aircraft")


# Leitura da página que contém a tabela com a lista de aeronaves

# A estrutura da página é em html, esse comando faz a quebra de todas as linhas de código da página
soup = BeautifulSoup(page.content, 'html.parser')
```

Fonte: Autor

Ao acessar a página, que está em HTML, o comando ”html.parser” lê e estrutura a página em linhas de HTML para fatilitar a interpretação do código da página, pois em seguida será necessário identificar a tabela que contém os links de cada aeronave.

2.2.1.1 Pacotes utilizados

Para essa seção os pacotes da Figura 11 foram utilizados:

Figura 11 – Pacotes utilizados para o acesso à pagina da web

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import requests
```

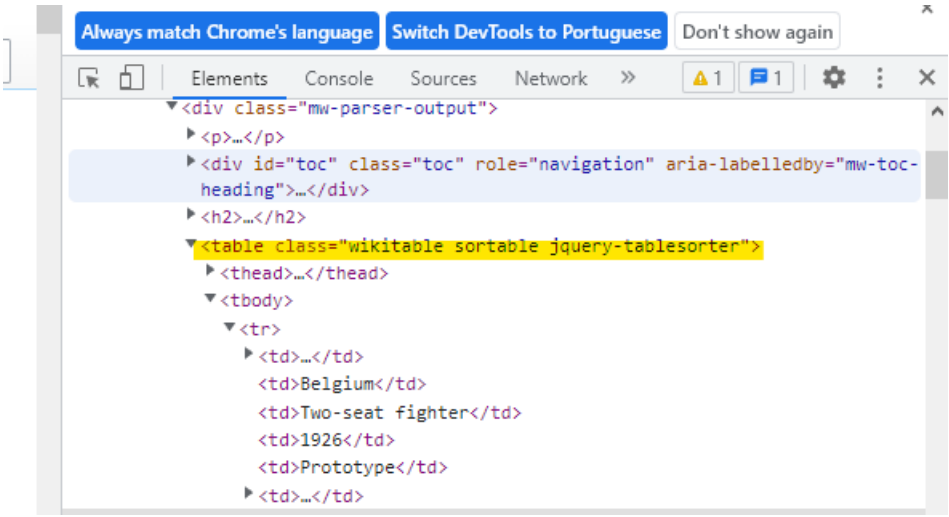
Fonte: Autor

É importante novamente ressaltar que antes do uso do programa é necessária a instalação destes e dos demais pacotes, eles podem ser instalados diretamente pela janela de comandos, utilizando o comando -pip install nome_do_pacote.

2.2.2 Identificação e levantamento da lista de aeronaves

Com o código da página em HTML preparado para ser interpretado, é necessário identificar o tipo de tabela que contém os dados necessários. Para isso, ao clicar em cima da tabela com o comando 'Inspecionar elemento', obtemos que a tabela é da classe *'wikitable sortable'*, como mostra a Figura 12.

Figura 12 – Classe da tabela



Fonte: (WIKIPEDIA, 2022b)

A partir desta informação, as linhas de código da Figura 13 geram o comando para extrair e criar três listas (Figura 14), contendo os links para cada aeronave, a data de registro do primeiro voo e o status atual, esses três dados são adicionados nas listas denominadas 'Links_list', 'date_list' e 'status_list', respectivamente.O objetivo da criação da lista de links é evitar que o programa tenha que ler a tabela da página inicial toda vez que for executar um loop para acessar os links da tabela. As outras duas listas serão utilizadas como filtros para futuras análises.

Figura 13 – Código para a leitura da tabela e criação da lista de links

```
# Identifica a tabela que contém a lista de aeronaves, o formato da tabela é 'wikitable sortable'
table = soup.find('table', {"class": 'wikitable sortable'})

# Identifica todos os links que direcionan para páginas de aeronaves da tabela
for link in table.select("td:nth-of-type(1) a "):
    # Cria e adiciona todos os links de aeronaves na lista 'Links_list'
    url = BASE_URL + link['href']
    if link['href'] == "/wiki/HAL_FGFA": continue
    Links_list.append(url)

for date in table.select("td:nth-of-type(4)"):
    date_list.append(date.text)

for status in table.select("td:nth-of-type(5)"):
    status_list.append(status.text)
```

Fonte: Autor

Observações: Para o levantamento da lista de links foi necessário a criação de um *conditional statement* utilizando a função *continue*. O motivo dessa alteração se deve ao fato da aeronave HAL FGFA possuir 2 links na mesma coluna que levam à mesma aeronave, o Sukhoi 57 (PAK FA), gerando uma linha repetida na base de dados. Então, essa linha extra deve ser ignorada pelo programa para evitar erros na estrutura do *dataframe* (estrutura que consolida todas as listas em uma tabela).

Figura 14 – Listas de links, datas e status

```
21  Links_list = []  # Lista que será usada de referência para acesso às paginas das aeronaves
22  date_list = []  # Lista que contém o ano de inicio da operação da aeronave
23  status_list = []  # Lista que mostra o status da aeronave
```

Fonte: Autor

2.3 Acesso aos links da lista e extração de dados (Parte 1)

A partir deste ponto, se inicia o *loop* iterativo entre a lista de links originada na seção anterior e o comando para extração de dados. Antes do início da extração, é necessário a criação das listas onde cada tipo de informação será armazenada (figura 15). A posição das informações que cada lista contém corresponde à posição da aeronave na lista de links.

Figura 15 – Listas de informações a serem extraídas

```
52  # Cada tipo de dado terá uma lista própia, eles são adicionados de acordo com a respectiva posição da aeronave na lista
53
54  # Listas
55  Title_list = []
56  Crew_list = []  # em pessoas
57  Length_list = []  # em metros
58  Wingspan_list = []  # em metros
59  Height_list = []
60  Wing_area_list = []
61  Empty_weight_list = []
62  MTOW_list = []
63  Maximum_speed_list = []
64  Cruise_speed_list = []
65  Service_ceiling_list = []
66  Powerplant_list = []
```

Fonte: Autor

O *loop* se inicia com o programa acessando o primeiro link da 'Links_list', utilizando o comando *get* do pacote *Selenium*. Os dados que serão extraídos estão em HTML e estão sob a categoria 'li'.

Para facilitar a identificação dessa categoria, o primeiro passo que o código executa é separar todas as linhas de HTML que estão identificadas com o marcador 'li' (linha 75). Em seguida, o primeiro dado a ser extraído da página é o nome da aeronave (que em seguida é adicionado à sua respectiva lista). O nome da aeronave vem acompanhado da palavra "Wikipedia", sendo que é necessário uma linha de código para remover a palavra 'Wikipedia' (linha 78), restando somente o nome da aeronave. Para isso, são utilizadas as *Regular Expressions* importadas no modulo "re". O trecho do código correspondente à essa seção é exibido na figura 16.

Figura 16 – Início do *loop* para extração das informações

```
67  # ----------------------------------------------------------------------
68
69  counter = 0
70
71  for link in Links_list:
72      driver.get(link)
73      # driver.get("https://en.wikipedia.org/wiki/Bloch_MB.150")
74      #   items = driver.find_elements(By.TAG_NAME, "li")
75      items = driver.find_elements_by_tag_name("li")
76      #    items = driver.find_elements_by_css_selector('li')
77      title = driver.title
78      title1 = re.findall('(.+)- [ Wikipedia]', title)
79      title = title1[0]
80      Title_list.append(title)
81      print(title)
```

Fonte: Autor

Observação: O contador, presente na linha 69, tem a função de garantir que caso o código falhe em adicionar algum valor em uma lista, o valor "na" seja adicionado na referida lista. Esse contador está relacionado com os *conditional statements* ao final do código (figura 17), evitando o erro de consolidação das listas no *dataframe* ao final do programa.

Figura 17 – Mecanismo para preencher listas caso ocorra algum erro (lista não exaustiva)

```
261      counter = counter + 1
262
263      # ITEMS SEM INFORMAÇOES
264      if len(Crew_list) != counter:
265          Crew_list.append("na")
266
267      if len(Length_list) != counter:
268          Length_list.append("na")
269
270      if len(Wingspan_list) != counter:
271          Wingspan_list.append("na")
272
273      if len(Height_list) != counter:
274          Height_list.append("na")
```

Fonte: Autor

2.4 Extração de dados - Parte 2

Como dito anteriormente, a primeira parte da extração de dados identifica todos os marcadores ”li” da página em HTML. Esses marcadores são linhas de texto que contém os dados que precisamos. Essas linhas são armazenadas na variável ”items”, que por definição será uma lista com as linhas selecionadas.

A segunda parte da extração de dados consiste efetivamente em extrair os dados das aeronaves e se baseia no uso de *for loops* e *Regular Expressions* ao longo de cada linha da variável ”items”.

Antes do início do *loop*, é necessário a criação de outro mecanismo para evitar que dupla contagens aconteçam, pois em algumas páginas da web podem existir mais de uma informação sobre o mesmo item. O programa considera somente a primeira informação que aparecer. A figura 18 ilustra o mecanismo de contadores utilizados para evitar que esse problema ocorra e a figura 19, uma aplicação desse mecanismo.

Figura 18 – Mecanismo para evitar dupla contagem

```
83      crew_count = 0
84      length_count = 0
85      wingspan_count = 0
86      height_count = 0
87      wing_area_count = 0
88      empty_weight_count = 0
89      MTOW_count = 0
90      maximum_speed_count = 0
91      cruise_speed_count = 0
92      service_ceiling_count = 0
93      powerplant_count = 0
```

Fonte: Autor

Figura 19 – Exemplo de aplicação do mecanismo

```
129     if re.match('Wingspan:', text):
130         if wingspan_count == 1:
131             pass
132         else:
133             wingspan_count = 1
134             try:
135                 wingspan1 = re.findall('([0-9]+.?[0-9]+) [m]', text)
136                 wingspan = float(wingspan1[0])
137                 Wingspan_list.append(wingspan)
138                 print(wingspan)
139             except:
140                 Wingspan_list.append("na")
```

Fonte: Autor

Neste exemplo, caso uma informação sobre a envergadura da aeronave já tenha sido adicionada à lista de envergaduras, o contador passará a valer 1. Caso exista uma segunda linha contendo a mesma informação, o programa entenderá que o contador já está com valor 1 e ignorará a segunda informação através do comando *pass* (linhas 130 e 131).

Seguindo com o mesmo exemplo, cuja estrutura de extração de dados é similar às demais dessas seção,a estrutura condicional *Try/Except*, jutamente com o uso das *Regular Expressions* são utilizadas para identificar, extrair e adicionar os dados nas respectivas listas.

Neste caso, o programa identificar a linha que contem a os caracteres ”Wingspan:” (linha 129). Caso a linha ”Wingspan:” não exista, o contador inicial do programa adicionará o valor ”na” na respectiva posição da lista de envergaduras.

Caso essa linha exista, o programa tentará (*Try*) extrair o valor da envergadura em metros (m) da aeronave. Para identificar o valor da envergadura, faz-se uso das *ReGex* (linha 135). Nesta linha, entende-se: encontrar todos os elementos que contém um ou mais dígitos de 0 a 9, seguidos, ou não, de um ponto, seguidos de um ou mais dígitos de 0 a 9, seguidos de ”m”.

Caso a linha ’Wingspan:’ exista, mas a informação não esteja no formato adequado para extração, uma exceção (*Except*) é acionada, adicionando o valor ”na” na respectiva posição da lista de envergaduras.

A mesma lógica é aplicada para os demais itens que compõem a base de dados, com leves modificações em relação à linha da *regular expression* a ser identificada. O exemplo da Figura 20, ilustra a situação onde uma leve modificação na *regular expression* foi feita.

Figura 20 – Exemplo de extração da potência da aeronave

```
247        if re.match('Powerplant:', text):
248            if powerplant_count == 1:
249                pass
250            else:
251                powerplant_count = 1
252                try:
253                    powerplant1 = re.findall('.([0-9]+.?[0-9]+)\s[kN]', text)
254                    powerplant = float(powerplant1[0])
255                    Powerplant_list.append(powerplant)
256                    print(powerplant)
257
258                except:
259                    Powerplant_list.append("na")
```

2.5 Consolidando as informações

Após a execução do *loop* de extração, todas as listas estão prontas para serem consolidadas em uma única tabela. O módulo Pandas (pd) é o responsável por realizar esta tarefa. A Figura 21 ilustra esta parte. Primeiramente, é necessário nomear cada lista (linha 308) e em seguida utilizar a função ”pd.DataFrame” (linha 315) para converter o conjunto de listas em um dataframe, que é uma tabela com todas as listas consolidadas.

Para facilitar a visualização e análise dos dados coletados pelo programa, também é possível converter o *dataframe* gerado para um arquivo .csv (linha 317).

Figura 21 – Consolidação das informações

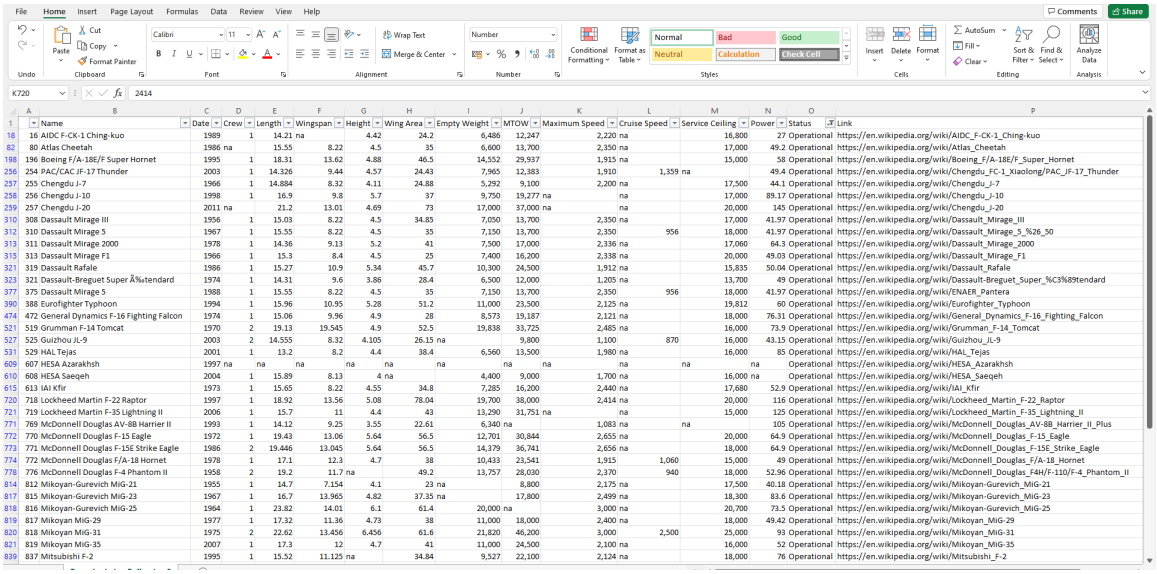
```
308 database = {"Name": Title_list, "Date": date_list, "Crew": Crew_list, "Length": Length_list,
309             "Wingspan": Wingspan_list,
310             "Height": Height_list,
311             "Wing Area": Wing_area_list, "Empty Weight": Empty_weight_list, "MTOW": MTOW_list,
312             "Maximum Speed": Maximum_speed_list,
313             "Cruise Speed": Cruise_speed_list, "Service Ceiling": Service_ceiling_list, "Power": Powerplant_list,
314             "Status": status_list, "Link": Links_list}
315 database = pd.DataFrame(database)
316
317 database.to_csv("Base_de_dados_Refinada.csv")
318
319 driver.close()
```

Fonte: Autor

2.6 Resultados

O programa demora em média 3 horas e meia para finalizar todas as rotinas e, ao final, gera uma planilha em .csv com todos os resultados obtidos pelo *scraper*. Ao todo foram analisadas 1307 aeronaves entre 742 protótipos, 7 projetos, 507 produzidas e 51 em operação. A figura 22 mostra um trecho do resultado do programa, exibindo somente as aeronaves de combate atualmente em operação, seguida do quadro de análise de performance do programa.

Figura 22 – Consolidação das informações



Name	Date	Crew	Length	Wingspan	Height	Wing Area	Empty Weight	MTOW	Maximum Speed	Cruise Speed	Service Ceiling	Power	Status	Link
16 AIDC F-CK-1 Ching-kuo	1989	1	14.21	na	4.42	24.2	6,486	12,247	2,220	na	16,800	27	Operational	https://en.wikipedia.org/wiki/AIDC_F-CK-1_Ching-kuo
80 Atlas Cheetah	1986	na	15.55	8.22	4.5	35	6,600	13,700	2,350	na	17,000	49.2	Operational	https://en.wikipedia.org/wiki/Atlas_Cheetah
196 Boeing F/A-18E/F Super Hornet	1995	1	18.31	13.62	4.88	46.5	14,552	29,937	1,915	na	15,000	58	Operational	https://en.wikipedia.org/wiki/Boeing_F/A-18E/F_Super_Hornet
254 PAC/CAC JF-17 Thunder	2003	1	14.326	9.44	4.57	24.43	7,965	12,383	1,910	1,359	na	49.4	Operational	https://en.wikipedia.org/wiki/Chengdu_FC-1_Xiaolong/PAC_JF-17_Thunder
255 Chengdu J-7	1966	1	14.884	8.32	4.11	24.88	5,292	9,100	2,200	na	17,500	44.1	Operational	https://en.wikipedia.org/wiki/Chengdu_J-7
256 Chengdu J-10	1998	1	16.9	9.8	5.7	37	9,750	19,277	na	na	17,000	89.17	Operational	https://en.wikipedia.org/wiki/Chengdu_J-10
257 Chengdu J-20	2011	na	21.2	13.01	4.69	73	17,000	37,000	na	na	20,000	145	Operational	https://en.wikipedia.org/wiki/Chengdu_J-20
308 Dassault Mirage III	1956	1	15.03	8.22	4.5	34.85	7,050	13,700	2,350	na	17,000	41.97	Operational	https://en.wikipedia.org/wiki/Dassault_Mirage_III
310 Dassault Mirage 5	1967	1	15.55	8.22	4.5	35	7,150	13,700	2,350	956	18,000	41.97	Operational	https://en.wikipedia.org/wiki/Dassault_Mirage_5_%26_50
311 Dassault Mirage 2000	1978	1	14.36	9.13	5.2	41	7,500	17,000	2,336	na	17,060	64.3	Operational	https://en.wikipedia.org/wiki/Dassault_Mirage_2000
313 Dassault Mirage F1	1966	1	15.3	8.4	4.5	25	7,400	16,200	2,338	na	20,000	49.03	Operational	https://en.wikipedia.org/wiki/Dassault_Mirage_F1
319 Dassault Rafale	1986	1	15.27	10.9	5.34	45.7	10,300	24,500	1,912	na	15,835	50.04	Operational	https://en.wikipedia.org/wiki/Dassault_Rafale
321 Dassault-Breguet Super Ã‰tendard	1974	1	14.31	9.6	3.86	28.4	6,500	12,000	1,205	na	13,700	49	Operational	https://en.wikipedia.org/wiki/Dassault-Breguet_Super_%C3%89tendard
375 Dassault Mirage 5	1988	1	15.55	8.22	4.5	35	7,150	13,700	2,350	956	18,000	41.97	Operational	https://en.wikipedia.org/wiki/ENAER_Pantera
388 Eurofighter Typhoon	1994	1	15.96	10.95	5.28	51.2	11,000	23,500	2,125	na	19,812	60	Operational	https://en.wikipedia.org/wiki/Eurofighter_Typhoon
472 General Dynamics F-16 Fighting Falcon	1974	1	15.06	9.96	4.9	28	8,573	19,187	2,121	na	18,000	76.31	Operational	https://en.wikipedia.org/wiki/General_Dynamics_F-16_Fighting_Falcon
519 Grumman F-14 Tomcat	1970	2	19.13	19.545	4.9	52.5	19,838	33,725	2,485	na	16,000	73.9	Operational	https://en.wikipedia.org/wiki/Grumman_F-14_Tomcat
525 Guizhou JL-9	2003	2	14.555	8.32	4.105	26.15	na	9,800	1,100	870	16,000	43.15	Operational	https://en.wikipedia.org/wiki/Guizhou_JL-9
529 HAL Tejas	2001	1	13.2	8.2	4.4	38.4	6,560	13,500	1,980	na	16,000	85	Operational	https://en.wikipedia.org/wiki/HAL_Tejas
607 HESA Azarakhsh	1997	na	na	na	na	na	na	na	na	na	na	na	Operational	https://en.wikipedia.org/wiki/HESA_Azarakhsh
608 HESA Saeqeh	2004	1	15.89	8.13	4	na	4,400	9,000	1,700	na	16,000	na	Operational	https://en.wikipedia.org/wiki/HESA_Saeqeh
613 IAI Kfir	1973	1	15.65	8.22	4.55	34.8	7,285	16,200	2,440	na	17,680	52.9	Operational	https://en.wikipedia.org/wiki/IAI_Kfir
718 Lockheed Martin F-22 Raptor	1997	1	18.92	13.56	5.08	78.04	19,700	38,000	2,414	na	20,000	116	Operational	https://en.wikipedia.org/wiki/Lockheed_Martin_F-22_Raptor
719 Lockheed Martin F-35 Lightning II	2006	1	15.7	11	4.4	43	13,290	31,751	na	na	15,000	125	Operational	https://en.wikipedia.org/wiki/Lockheed_Martin_F-35_Lightning_II
769 McDonnell Douglas AV-8B Harrier II	1993	1	14.12	9.25	3.55	22.61	6,340	na	1,083	na	na	105	Operational	https://en.wikipedia.org/wiki/McDonnell_Douglas_AV-8B_Harrier_II_Plus
770 McDonnell Douglas F-15 Eagle	1972	1	19.43	13.06	5.64	56.5	12,701	30,844	2,655	na	20,000	64.9	Operational	https://en.wikipedia.org/wiki/McDonnell_Douglas_F-15_Eagle
771 McDonnell Douglas F-15E Strike Eagle	1986	2	19.446	13.045	5.64	56.5	14,379	36,741	2,656	na	18,000	64.9	Operational	https://en.wikipedia.org/wiki/McDonnell_Douglas_F-15E_Strike_Eagle
772 McDonnell Douglas F/A-18 Hornet	1978	1	17.1	12.3	4.7	38	10,433	23,541	1,915	1,060	15,000	49	Operational	https://en.wikipedia.org/wiki/McDonnell_Douglas_F/A-18_Hornet
776 McDonnell Douglas F-4 Phantom II	1958	2	19.2	11.7	na	49.2	13,757	28,030	2,370	940	18,000	52.96	Operational	https://en.wikipedia.org/wiki/McDonnell_Douglas_F4H/F-110/F-4_Phantom_II
812 Mikoyan-Gurevich MiG-21	1955	1	14.7	7.154	4.1	23	na	8,800	2,175	na	17,500	40.18	Operational	https://en.wikipedia.org/wiki/Mikoyan-Gurevich_MiG-21
815 Mikoyan-Gurevich MiG-23	1967	1	16.7	13.965	4.82	37.35	na	17,800	2,499	na	18,300	83.6	Operational	https://en.wikipedia.org/wiki/Mikoyan-Gurevich_MiG-23
816 Mikoyan-Gurevich MiG-25	1964	1	23.82	14.01	6.1	61.4	20,000	na	3,000	na	20,700	73.5	Operational	https://en.wikipedia.org/wiki/Mikoyan-Gurevich_MiG-25
817 Mikoyan MiG-29	1977	1	17.32	11.36	4.73	38	11,000	18,000	2,400	na	18,000	49.42	Operational	https://en.wikipedia.org/wiki/Mikoyan_MiG-29
818 Mikoyan MiG-31	1975	2	22.62	13.456	6.456	61.6	21,820	46,200	3,000	2,500	25,000	93	Operational	https://en.wikipedia.org/wiki/Mikoyan_MiG-31
819 Mikoyan MiG-35	2007	1	17.3	12	4.7	41	11,000	24,500	2,100	na	16,000	52	Operational	https://en.wikipedia.org/wiki/Mikoyan_MiG-35
837 Mitsubishi F-2	1995	1	15.52	11.125	na	34.84	9,527	22,100	2,124	na	18,000	76	Operational	https://en.wikipedia.org/wiki/Mitsubishi_F-2

Tabela 3 – Avaliação da performance do programa (parte 1)

Dado	**Crew**	**Length**	**Wingspan**	**Height**	**Wing Aerea**	**Empty Weight**
Quantidade	1307	1307	1307	1307	1307	1307
Dados encontrados	882	1200	1141	1061	1127	1097
Dados não encontrados	425	107	166	246	180	210
Máximo	8	30.97	25	16.08	167	27604
Mínimo	1	4.26	5.78	1.1	4.7	129

Fonte: Análise própria

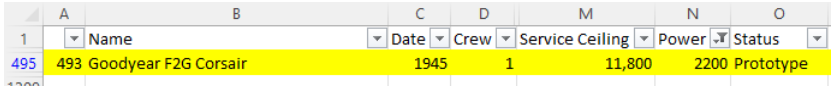
Tabela 4 – Avaliação da performance do programa (parte 2)

Dado	**MTOW**	**Max Speed**	**Cruise Speed**	**Service Ceiling**	**Power**
Quantidade	1307	1307	1307	1307	1307
Dados encontrados	380	1152	137	957	1097
Dados não encontrados	927	155	1170	350	210
Máximo	63504	3661	2500	27400	2200
Mínimo	652	106	110	1000	2.9

Fonte: Análise própria

A partir da avaliação dos resultados de máximo e mínimo, verificou-se que a coluna de Potência (Power) possui um valor máximo acima do esperado, que em média seria abaixo de 100 kN para cada motor. Portanto a aeronave, identificada como ”Goodyear F2G Corsair”, cujo valor corresponde aos 2200 kN indicados foi verificada e, como resultado, o valor extraído pelo programa se refere à potência do motor da aeronave na unidade ”kW” e não em ”kN”.

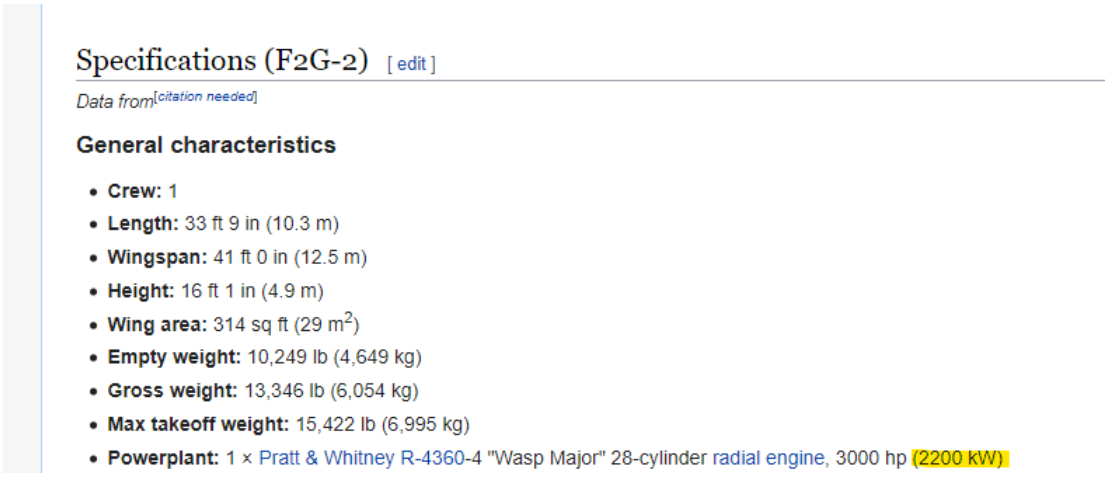
Figura 23 – Identificação da aeronave



	A	B	C	D	M	N	O
1		Name	Date	Crew	Service Ceiling	Power	Status
495	493	Goodyear F2G Corsair	1945	1	11,800	2200	Prototype

Fonte: Análise própria

Figura 24 – Verificação da potência



Specifications (F2G-2) [edit]

Data from[citation needed]

General characteristics

- **Crew:** 1
- **Length:** 33 ft 9 in (10.3 m)
- **Wingspan:** 41 ft 0 in (12.5 m)
- **Height:** 16 ft 1 in (4.9 m)
- **Wing area:** 314 sq ft (29 m^2)
- **Empty weight:** 10,249 lb (4,649 kg)
- **Gross weight:** 13,346 lb (6,054 kg)
- **Max takeoff weight:** 15,422 lb (6,995 kg)
- **Powerplant:** 1 × Pratt & Whitney R-4360-4 "Wasp Major" 28-cylinder radial engine, 3000 hp (2200 kW)

Fonte: (WIKIPEDIA, 2022a)

Este problema foi contornado utilizando a estratégia ’*Easier to Ask for Forgiveness Than Permission*’, que consiste em utilizar múltiplos ’*Try/Except*’ em uma mesma secção do código para identificar se os dados relacionados ao motor da aeronave estão em hp ou kN. Com isso uma coluna adicional foi criada para separar os dados em kN e hp. Observação: caso o dado esteja em kW, o programa automaticamente converterá a unidade para hp.

3 CONCLUSÃO

Sendo a primeira versão do programa, ele ainda passará por futuras revisões e refinamentos de lógica. No entanto, avaliando seu funcionamento e a quantidade e qualidade dos dados extraídos, podemos considerar a primeira versão do programa satisfatória.

Das 11 características avaliadas pelo programa, 10 delas apresentaram resultados satisfatórios, sendo que a maioria dos dados não encontrados estão relacionados ao Peso Máximo de Decolagem (MTOW) e à Velocidade de Cruzeiro (Cruize Speed). A ausência desses dados pode ser explicada por 2 fatores:

- O primeiro fator está relacionado à forma em que esses dados são disponibilizados nas páginas examinadas, se eles divergirem do formato esperado, o programa não fará a adição dos valores na base de dados. Este fator poderá ser corrigido em futuras revisões do programa;
- O segundo fator, ocorre pela pura ausência dos dados nas páginas analisadas e não tem como ser evitado.

Futuras revisões do programa podem aumentar a eficiência de extração de dados de páginas da web que apresentam o primeiro tipo de erro. Também é necessário aprimorar a seção de extração da potência do motor, pois existem aeronaves cuja unidade de potência utilizada não está em ”kN”, mas sim em ”kW”, por exemplo. Esta revisão seria útil, por exemplo, para entender a evolução da relação Peso x Potência das aeronaves ao longo do tempo.

Para o objetivo da disciplina de Projetos I, a base de dados gerada pelo programa pode ser utilizada para a análise de dados das aeronaves atualmente em operação. Caso seja necessário incluir aeronaves das demais categorias, deve-se atentar para valores que se destoam da faixa esperada e, caso sim, sempre verificar se há um possível erro na informação coletada.

4 PRÓXIMOS PASSOS

Para a melhoria contínua do programa, foram pensados alguns próximos passos (não exaustivos) que podem ser implementados nas próximas versões do programa:

- Implementar interface (em python), com filtros que possibilitem selecionar o tipo de informação a ser exibida. Por exemplo: selecionar somente aeronaves que foram feitas a partir de uma certa data ou aeronaves com uma potência específica
- Incluir nova função para extrair o tipo de aeronave de combate que foi extraída. Por exemplo: Fighter, bomber-fighter, training-fighter, stealth-fighter, etc.
- Incluir função para plotar gráficos comparativos. Por exemplo: dispersão entre potência das aeronaves e ano de fabricação, relação peso x potência, área da asa x peso, etc.

REFERÊNCIAS

COX, R. **Regular Expression Matching Can Be Simple And Fast (but is slow in Java, Perl, PHP, Python, Ruby, ...)**. [*S.l.: s.n.*], 2007.

GEEKSFORGEEKS. **History of Web Scraping**. 2021. Disponível em: https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/.

LUTZ, M. **Programming Python, 1st edition**. [*S.l.: s.n.*]: O'Reilly Associates, 1996.

MITCHELL, R. **Web Scraping with Python**. [*S.l.: s.n.*]: O'Reilly, 2015.

PYTHON For Loops. 2022. Disponível em: https://www.w3schools.com/python/python_for_loops.asp. Acesso em: 15 mar. 2022.

SEVERANCE, C. R. **Exploring Data Using Python 3**. [*S.l.: s.n.*], 2009.

SILVA, J. R. **Python Tutorial Release 3.8.1 Guido van Rossum and the Python development team**. [*S.l.: s.n.*]: Python Software Foundation, 2020.

WIKIPEDIA. **Goodyear F2G Corsair**. 2022. Disponível em: https://en.wikipedia.org/wiki/Goodyear_F2G_Corsair.

WIKIPEDIA. **List of Fighter Aircraft**. 2022. Disponível em: https://en.wikipedia.org/wiki/List_of_fighter_aircraft.

APÊNDICES

APÊNDICE A – CÓDIGO DESENVOLVIDO

```
# Web scraper usando Selenium + bs4
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import requests

import pandas as pd
from datetime import datetime
import re

# Setup selenium
chrome_options = Options()

chrome_options.add_argument("--headless")  # faz com que o browser não
abra durante o processo

# instala automaticamente o webdriver para o Google Chrome, necessário
para acessar a web
driver = webdriver.Chrome(service=Service(ChromeDriverManager()
.install()))
# ---------------------------------------------------------------------
Links_list = []  # Lista que será usada de referência para
acesso às paginas das aeronaves
date_list = []  # Lista que contém o ano de início da
operação da aeronave
status_list = []  # Lista que mostra o status da aeronave

BASE_URL = "https://en.wikipedia.org"

# Acessa a página principal, que contém a lista de aeronaves
page = requests.get("https://en.wikipedia.
org/wiki/List_of_fighter_aircraft")

# Leitura da página que contém a tabela com a lista de aeronaves
```

```
# A estrutura da página é em html, esse comando faz a quebra
de todas as linhas de código da página
soup = BeautifulSoup(page.content, 'html.parser')

# Identifica a tabela que contém a lista de aeronaves, o formato
da tabela é 'wikitable sortable'
table = soup.find('table', {"class": 'wikitable sortable'})

# Identifica todos os links que direcionan para páginas de
aeronaves da tabela
for link in table.select("td:nth-of-type(1) a "):
    # Cria e adiciona todos os links de aeronaves na lista 'Links_list'
    url = BASE_URL + link['href']
    if link['href'] == "/wiki/HAL_FGFA": continue
    Links_list.append(url)

for date in table.select("td:nth-of-type(4)"):
    date_list.append(date.text)

for status in table.select("td:nth-of-type(5)"):
    status_list.append(status.text)
# ----------------------------------------------------------------------

# Cada tipo de dado terá uma lista própria, eles são adicionados
de acordo com a respectiva posição da aeronave na lista

# Listas
Title_list = []
Crew_list = []  # em pessoas
Length_list = []  # em metros
Wingspan_list = []  # em metros
Height_list = []
Wing_area_list = []
Empty_weight_list = []
MTOW_list = []
Maximum_speed_list = []
Cruise_speed_list = []
Service_ceiling_list = []
```

```
Thrust_list = []
Power_list = []
# ----------------------------------------------------------------------

counter = 0

for link in Links_list:
    driver.get(link)
    # driver.get("https://en.wikipedia.org/wiki/Bloch_MB.150")
    #   items = driver.find_elements(By.TAG_NAME, "li")
    items = driver.find_elements_by_tag_name("li")
    #    items = driver.find_elements_by_css_selector('li')
    title = driver.title
    title1 = re.findall('(.+)- [ Wikipedia]', title)
    title = title1[0]
    Title_list.append(title)
    print(title)

    crew_count = 0
    length_count = 0
    wingspan_count = 0
    height_count = 0
    wing_area_count = 0
    empty_weight_count = 0
    MTOW_count = 0
    maximum_speed_count = 0
    cruise_speed_count = 0
    service_ceiling_count = 0
    powerplant_count = 0

    for item in items:

        text = item.text

        if re.match('Crew:', text):
            if crew_count == 1:
                pass
            else:
                crew = re.findall('([0-9])', text)
```

```
            if len(crew) == 0:
                crew = "na"

            else:
                crew = float(crew[0])
            Crew_list.append(crew)
            print(crew)
            crew_count = 1

    # ----------------------------------------
    if re.match('Length:', text):
        if length_count == 1:
            pass
        else:
            length_count = 1
            try:
                length1 = re.findall('([0-9]+.?[0-9]+) [m]', text)
                length = float(length1[0])
                Length_list.append(length)
                print(length)
            except:
                Length_list.append("na")
                print(length)

    # ----------------------------------------
    if re.match('Wingspan:', text):
        if wingspan_count == 1:
            pass
        else:
            wingspan_count = 1
            try:
                wingspan1 = re.findall('([0-9]+.?[0-9]+) [m]', text)
                wingspan = float(wingspan1[0])
                Wingspan_list.append(wingspan)
                print(wingspan)
            except:
                Wingspan_list.append("na")

    # ----------------------------------------
```

```
if re.match('Height:', text):
    if height_count == 1:
        pass
    else:
        height_count = 1
        try:
            height1 = re.findall('([0-9]+.?[0-9]+) [m]', text)
            height = float(height1[0])
            Height_list.append(height)
            print(height)
        except:
            Height_list.append("na")

# ----------------------------------------
if re.match('Wing area:', text):
    if wing_area_count == 1:
        pass
    else:
        wing_area_count = 1

        try:
            Wing_area1 = re.findall('([0-9]+.?[0-9]+) [m^2]', text)
            Wing_area = float(Wing_area1[0])
            Wing_area_list.append(Wing_area)
            print(Wing_area)
        except:
            Wing_area_list.append("na")

# ----------------------------------------
if re.match('Empty weight:', text):
    if empty_weight_count == 1:
        pass
    else:
        empty_weight_count = 1
        try:
            weight1 = re.findall('([0-9]+,?[0-9]+) [kg]', text)
            weight = weight1[0]
            Empty_weight_list.append(weight)
            print(weight)
```

```
            except:
                Empty_weight_list.append("na")

    # ----------------------------------------
    if re.match('Max takeoff weight:', text):
        if MTOW_count == 1:
            pass
        else:
            MTOW_count = 1
            try:
                MTOW1 = re.findall('.([0-9]+,?[0-9]+)\s[kg]', text)
                MTOW = MTOW1[0]
                MTOW_list.append(MTOW)
                print(MTOW)
            except:
                MTOW = (text.split("[")[0]).split(" ")[-1]
                MTOW_list.append(MTOW)
                print(MTOW)

    # ----------------------------------------
    if re.match('Maximum speed:', text):
        if maximum_speed_count == 1:
            pass
        else:
            maximum_speed_count = 1
            try:
                Max_speed1
                =re.findall('.([0-9],?[0-9]+)\s[k][m][/][h]', text)
                Max_speed = Max_speed1[0]
                Maximum_speed_list.append(Max_speed)
                print(Max_speed)

            except:
                Maximum_speed_list.append("na")
    # else:Maximum_speed_list.append("na")
    # ----------------------------------------
    if re.match('Cruise speed:', text):
        if cruise_speed_count == 1:
            pass
```

```
    else:
        cruise_speed_count = 1

        try:
            Cruise_speed1 =
            re.findall('.\s([0-9],?[0-9]+)\s[k][m][/][h]', text)
            Cruise_speed = Cruise_speed1[0]
            Cruise_speed_list.append(Cruise_speed)
            print(Cruise_speed)
        except:
            Cruise_speed_list.append("na")

# ----------------------------------------
if re.match('Service ceiling:', text):
    if service_ceiling_count == 1:
        pass
    else:
        service_ceiling_count = 1
        try:
            service_ceiling1 =
            re.findall('([0-9]+,?[0-9]+) [m]', text)
            service_ceiling = service_ceiling1[0]
            Service_ceiling_list.append(service_ceiling)
            print(service_ceiling)
        except:
            Service_ceiling_list.append("na")

# ----------------------------------------

if re.match('Powerplant:', text):
    if powerplant_count == 1:
        pass
    else:
        powerplant_count = 1
        try:
            thrust1 =
            re.findall('.([0-9]+.?[0-9]+)\s[k][N]', text)
            thrust = float(thrust1[0])
            Thrust_list.append(thrust)
```

```
                print(thrust)

            except:
                Thrust_list.append("na")
                try:
                    power1 = re.findall
                    ('.([0-9]+.?[0-9]+)\s[h][p]', text)
                    power = float(power1[0])
                    Power_list.append(power)
                    print(power)

                except:
                    try:
                        power1 =
                        re.findall('.([0-9]+.?[0-9]+)\s[k][W]', text)
                        power = float(power1[0]) * 1.34102
                        Power_list.append(power)
                        print(power)
                    except:
                        Power_list.append("na")

counter = counter + 1

# ITEMS SEM INFORMAÇOES
if len(Crew_list) != counter:
    Crew_list.append("na")

if len(Length_list) != counter:
    Length_list.append("na")

if len(Wingspan_list) != counter:
    Wingspan_list.append("na")

if len(Height_list) != counter:
    Height_list.append("na")

if len(Wing_area_list) != counter:
    Wing_area_list.append("na")
```

```
    if len(Empty_weight_list) != counter:
        Empty_weight_list.append("na")

    if len(MTOW_list) != counter:
        MTOW_list.append("na")

    if len(Maximum_speed_list) != counter:
        Maximum_speed_list.append("na")

    if len(Cruise_speed_list) != counter:
        Cruise_speed_list.append("na")

    if len(Service_ceiling_list) != counter:
        Service_ceiling_list.append("na")

    if len(Thrust_list) != counter:
        Thrust_list.append("na")

    if len(Power_list) != counter:
        Power_list.append("na")


    painel = {"Name": Title_list, "Crew": Crew_list,
    "Length [m]": Length_list,
              "Wingspan [m]": Wingspan_list,
              "Height [m]": Height_list,
              "Wing Area [m²]": Wing_area_list, "Empty Weight [kg]":
              Empty_weight_list, "MTOW [kg]": MTOW_list,
              "Maximum Speed [km/h]": Maximum_speed_list,
              "Cruise Speed [km/h]": Cruise_speed_list, "Service
              Ceiling [m]":Service_ceiling_list, "Thrust [kN]":
              Thrust_list, "Power [hp]": Power_list}

    painel_df = pd.DataFrame(painel)

    painel_df.to_csv("Painel_Geral_refinado_v2.csv")

database = {"Name": Title_list, "Date": date_list, "Crew": Crew_list,
"Length [m]": Length_list,
```

```
                "Wingspan [m]": Wingspan_list,
                "Height [m]": Height_list,
                "Wing Area [m²]": Wing_area_list, "Empty Weight [kg]":
                Empty_weight_list, "MTOW [kg]": MTOW_list,
                "Maximum Speed [km/h]": Maximum_speed_list,
                "Cruise Speed [km/h]": Cruise_speed_list, "Service Ceiling
                [m]": Service_ceiling_list, "Thrust [kN]": Thrust_list,
                "Power [hp]": Power_list,
                "Status": status_list, "Link": Links_list}
database = pd.DataFrame(database)

database.to_csv("Base_de_dados_Refinada_v2.csv")

driver.close()

# ----------------------------------------

# ------------------------------------------------------------------------------------------

# ------------------------------------------------------------------
```

APÊNDICE B – AERONAVES OPERACIONAIS

Tabela 5 – Tabela de aeronaves operacionais

Position	Name	Date	Crew	Length	Wingspan	Height	Wing Area
16	**AIDC F-CK-1 Ching-kuo**	1989	1	14.21	na	4.42	24.2
80	**Atlas Cheetah**	1986	na	15.55	8.22	4.5	35
196	**Boeing F/A-18E/F Super Hornet**	1995	1	18.31	13.62	4.88	46.5
254	**PAC/CAC JF-17 Thunder**	2003	1	14.326	9.44	4.57	24.43
255	**Chengdu J-7**	1966	1	14.884	8.32	4.11	24.88
256	**Chengdu J-10**	1998	1	16.9	9.8	5.7	37
257	**Chengdu J-20**	2011	na	21.2	13.01	4.69	73
308	**Dassault Mirage III**	1956	1	15.03	8.22	4.5	34.85
310	**Dassault Mirage 5**	1967	1	15.55	8.22	4.5	35
311	**Dassault Mirage 2000**	1978	1	14.36	9.13	5.2	41
313	**Dassault Mirage F1**	1966	1	15.3	8.4	4.5	25
319	**Dassault Rafale**	1986	1	15.27	10.9	5.34	45.7
321	**Dassault-Breguet Super Ã‰tendard**	1974	1	14.31	9.6	3.86	28.4
375	**Dassault Mirage 5**	1988	1	15.55	8.22	4.5	35
388	**Eurofighter Typhoon**	1994	1	15.96	10.95	5.28	51.2
472	**General Dynamics F-16 Fighting Falcon**	1974	1	15.06	9.96	4.9	28
519	**Grumman F-14 Tomcat**	1970	2	19.13	19.545	4.9	52.5
525	**Guizhou JL-9**	2003	2	14.555	8.32	4.105	26.15
529	**HAL Tejas**	2001	1	13.2	8.2	4.4	38.4
607	**HESA Azarakhsh**	1997	na	na	na	na	na
608	**HESA Saeqeh**	2004	1	15.89	8.13	4	na
613	**IAI Kfir**	1973	1	15.65	8.22	4.55	34.8
718	**Lockheed Martin F-22 Raptor**	1997	1	18.92	13.56	5.08	78.04
719	**Lockheed Martin F-35 Lightning II**	2006	1	15.7	11	4.4	43
769	**McDonnell Douglas AV-8B Harrier II**	1993	1	14.12	9.25	3.55	22.61
770	**McDonnell Douglas F-15 Eagle**	1972	1	19.43	13.06	5.64	56.5
771	**McDonnell Douglas F-15E Strike Eagle**	1986	2	19.446	13.045	5.64	56.5
772	**McDonnell Douglas F/A-18 Hornet**	1978	1	17.1	12.3	4.7	38
776	**McDonnell Douglas F-4 Phantom II**	1958	2	19.2	11.7	na	49.2
812	**Mikoyan-Gurevich MiG-21**	1955	1	14.7	7.154	4.1	23
815	**Mikoyan-Gurevich MiG-23**	1967	1	16.7	13.965	4.82	37.35
816	**Mikoyan-Gurevich MiG-25**	1964	1	23.82	14.01	6.1	61.4
817	**Mikoyan MiG-29**	1977	1	17.32	11.36	4.73	38
818	**Mikoyan MiG-31**	1975	2	22.62	13.456	6.456	61.6
819	**Mikoyan MiG-35**	2007	1	17.3	12	4.7	41
837	**Mitsubishi F-2**	1995	1	15.52	11.125	na	34.84
928	**Northrop F-5**	1959	1	14.6876	8.13	4.077	17.3
1059	**Saab JAS 39 Gripen**	1988	1	14.1	8.4	4.5	30
1079	**Shenyang J-8**	1969	1	21.39	9.344	5.41	42.2
1080	**Shenyang J-11**	1998	1	21.9	14.7	5.92	52.84
1081	**Shenyang J-15**	2009	1	22.28	15	5.92	67.84
1082	**Shenyang J-16**	2011	2	na	na	na	na
1153	**Sukhoi Su-17**	1966	1	19.02	13.68	5.12	38.5
1154	**Sukhoi Su-27**	1977	1	21.9	14.7	5.92	62
1155	**Sukhoi Su-30**	1989	2	21.935	14.7	6.36	62
1156	**Sukhoi Su-34**	1990	2	23.34	14.7	6.09	62.04
1157	**Sukhoi Su-33**	1987	1	22	14.7	5.93	67.84
1158	**Sukhoi Su-35**	1988	1	21.9	15.3	5.9	62
1161	**Sukhoi Su-57**	2010	1	20.1	14.1	4.6	78.8
1162	**Sukhoi Su-30MKI**	2000	2	21.935	14.7	6.36	62
1284	**Xi'an JH-7**	1988	2	22.32	12.8	6.22	42.2

Fonte: Elaborada pelo autor.

Tabela 6 – Tabela de aeronaves operacionais (cont.)

Position	Name	Empty Weight	MTOW	Maximum Speed	Cruise Speed	Service Ceiling	Power
16	**AIDC F-CK-1 Ching-kuo**	6486	12247	2220	na	16800	27
80	**Atlas Cheetah**	6600	13700	2350	na	17000	49.2
196	**Boeing F/A-18E/F Super Hornet**	14552	29937	1915	na	15000	58
254	**PAC/CAC JF-17 Thunder**	7965	12383	1910	1359	na	49.4
255	**Chengdu J-7**	5292	9100	2200	na	17500	44.1
256	**Chengdu J-10**	9750	19277	na	na	17000	89.17
257	**Chengdu J-20**	17000	37000	na	na	20000	145
308	**Dassault Mirage III**	7050	13700	2350	na	17000	41.97
310	**Dassault Mirage 5**	7150	13700	2350	956	18000	41.97
311	**Dassault Mirage 2000**	7500	17000	2336	na	17060	64.3
313	**Dassault Mirage F1**	7400	16200	2338	na	20000	49.03
319	**Dassault Rafale**	10300	24500	1912	na	15835	50.04
321	**Dassault-Breguet Super Â‰tendard**	6500	12000	1205	na	13700	49
375	**Dassault Mirage 5**	7150	13700	2350	956	18000	41.97
388	**Eurofighter Typhoon**	11000	23500	2125	na	19812	60
472	**General Dynamics F-16 Fighting Falcon**	8573	19187	2121	na	18000	76.31
519	**Grumman F-14 Tomcat**	19838	33725	2485	na	16000	73.9
525	**Guizhou JL-9**	na	9800	1100	870	16000	43.15
529	**HAL Tejas**	6560	13500	1980	na	16000	85
607	**HESA Azarakhsh**	na	na	na	na	na	na
608	**HESA Saeqeh**	4400	9000	1700	na	16000	na
613	**IAI Kfir**	7285	16200	2440	na	17680	52.9
718	**Lockheed Martin F-22 Raptor**	19700	38000	2414	na	20000	116
719	**Lockheed Martin F-35 Lightning II**	13290	31751	na	na	15000	125
769	**McDonnell Douglas AV-8B Harrier II**	6340	na	1083	na	na	105
770	**McDonnell Douglas F-15 Eagle**	12701	30844	2655	na	20000	64.9
771	**McDonnell Douglas F-15E Strike Eagle**	14379	36741	2656	na	18000	64.9
772	**McDonnell Douglas F/A-18 Hornet**	10433	23541	1915	1060	15000	49
776	**McDonnell Douglas F-4 Phantom II**	13757	28030	2370	940	18000	52.96
812	**Mikoyan-Gurevich MiG-21**	na	8800	2175	na	17500	40.18
815	**Mikoyan-Gurevich MiG-23**	na	17800	2499	na	18300	83.6
816	**Mikoyan-Gurevich MiG-25**	20000	na	3000	na	20700	73.5
817	**Mikoyan MiG-29**	11000	18000	2400	na	18000	49.42
818	**Mikoyan MiG-31**	21820	46200	3000	2500	25000	93
819	**Mikoyan MiG-35**	11000	24500	2100	na	16000	52
837	**Mitsubishi F-2**	9527	22100	2124	na	18000	76
928	**Northrop F-5**	4347	11192	1741	na	15800	16
1059	**Saab JAS 39 Gripen**	6800	14000	2460	na	15240	54
1079	**Shenyang J-8**	10371	18879	2300	na	18000	47.1
1080	**Shenyang J-11**	16380	33000	2500	na	19000	132
1081	**Shenyang J-15**	17500	32500	na	na	20000	122.6
1082	**Shenyang J-16**	17700	35000	na	na	na	na
1153	**Sukhoi Su-17**	12160	19430	1400	na	14200	76.4
1154	**Sukhoi Su-27**	16380	30450	2500	na	19000	75.22
1155	**Sukhoi Su-30**	17700	34500	2120	na	17300	74.5
1156	**Sukhoi Su-34**	22500	45100	1900	1300	17000	132
1157	**Sukhoi Su-33**	18400	33000	2300	na	17000	74.5
1158	**Sukhoi Su-35**	19000	34500	2400	1170	18000	86.3
1161	**Sukhoi Su-57**	18000	35000	2135	na	20000	88.3
1162	**Sukhoi Su-30MKI**	18400	38800	2120	na	17300	123
1284	**Xi'an JH-7**	14500	28475	1808	na	16000	54.29

Fonte: Elaborada pelo autor.