UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE LORENA

ALEXANDRE HENRIQUE CONSTANTINO LEAL

Desenvolvimento de programas em Python para auxiliar a extração e visualização dos dados de compostos binários metálicos

Lorena

2021

ALEXANDRE HENRIQUE CONSTANTINO LEAL

Desenvolvimento de programas em Python para auxiliar a extração e visualização dos dados de compostos binários metálicos

Trabalho de Graduação apresentado à Escola de Engenharia de Lorena da Universidade de São Paulo como requisito parcial para a conclusão do curso de Engenharia Física.

Orientador: Prof. Dr. Luiz Tadeu Fernandes Eleno

Lorena

2021

AGRADECIMENTOS

Aos meus avós que tornaram este momento possível, sempre me incentivando e ajudando. Especialmente ao meu avô que sempre me ajudou ao longo de toda a minha jornada.

Ao Prof. Dr. Luiz Tadeu Fernandes Eleno pela constante orientação e ajuda ao longo do desenvolvimento do trabalho.

A todos os professores da EEL, que com muita paciência e dedicação participaram ativamente da minha formação

A todos os amigos que de alguma forma fizeram parte dessa jornada eu agradeço com um forte abraço.

Aos meus amigos de faculdade por todo o apoio e ajuda durante todo o período acadêmico.

À minha namorada por sua compreensão, presença e incansável apoio ao longo da elaboração deste trabalho.

“Você não pode ensinar nada a ninguém, mas pode ajudar a pessoas a descobrirem por si mesmas.”

Galileu Galilei

RESUMO

LEAL, AHC. **Desenvolvimento de programa em Python para auxiliar a extração e visualização dos dados de compostos binários metálicos.** 2021. Número de folhas 88f. Monografia – Escola de Engenharia de Lorena, Universidade de São Paulo, Lorena, 2021.

A quantidade de dados presente no mundo vem aumentando exponencialmente nos últimos anos e, cada vez mais informações estão presentes nos servidores da internet, incluindo informações e dados dos materiais e suas características intrínsecas. Utilizando a plataforma *Web* da *Materials Project* e sua API, a monografia teve por objetivo criar um programa de extração das informações dos compostos binários metálicos de alumínio, boro e silício com os demais trinta metais de transição e, em seguida, gerar as visualizações dos parâmetros físicos desejados. Para isso, dois programas principais foram feitos em Python, desenvolvidos para a extração, geração da base de dados extraída dos compostos metálicos, acesso a base de dados criada e geração das visualizações das propriedades desejadas em gráficos. Os programas produzidos permitem uma boa extração dos dados e visualização de suas propriedades elásticas. Um dos resultados obtidos foi a e relação de eficiência entre a extração via API e a extração via Web Scraping, utilizando a API foi possível extrair os dados em média de forma 441,6 vezes mais rápido via API. Utilizando os programas desta monografia como modelos é possível, com alguns ajustes leves, extrair as informações de quaisquer compostos binários que for desejável e, em seguida, gerar visualizações de suas novas bases de dados, sendo um ótimo começo para outros projetos com análise dos dados dos compostos binários da plataforma.

Palavras-chave: Python. Programa. API. Compostos Binários. *Materials Project*.

ABSTRACT

LEAL, AHC. **Software development in Python for extract and visualize data of metallic binary compounds.**2021. Number of sheets 88. Monograph – Escola de Engenharia de Lorena, Universidade de São Paulo, Lorena, 2021.

The amount of data present in the world has been increasing exponentially in recent years, and more and more information is present on internet servers, including information and data on materials and their intrinsic characteristics. Using the Materials Project web platform and its API, the monograph aimed to create a program to extract information from the metallic binary compounds of aluminum, boron and silicon with the other thirty transition metals and then generate the parameter views desired physical properties, belonging to the database generated from the data extracted from the platform. For this, two main programs were made using Python, developed for the extraction, generation of the metallic compounds database, access of the created database and generation of the visualizations of the desired properties. The produced programs allow a good data extraction and visualization of its elastic properties. One of the results obtained was the efficiency ratio between extraction via API and extraction via Web Scraping, using the API it was possible to extract data on average 441.6 times faster via API. Using the programs in this monograph as models it is possible, with some light adjustments, to extract information from any binary compounds you desire and then generate its visualizations from its new databases, making it a great start for other projects with data analysis platform binary compounds.

Keywords: Python. Software. API. Binary Compounds. *Materials Project.*

LISTA DE FIGURAS

Figura 1 – Interaface inicial referente as informações iniciais do programa.....39
Figura 2 – Selecionando o elemento e iniciando a extração da API 39
Figura 3 – Geração do *Dataframe* com os dados da *MAPI* utilizando a *Pymatgen e a Pandas* 40
Figura 4 - *DataFrame* com as novas colunas de *Method, elasticity, spacegroup* e seus respectivos valores.....41
Figura 5 - *DataFrame* com as novas colunas de *Method, elasticity, spacegroup* e seus respectivos valores.....41
Figura 6 - *DataFrame* com as novas colunas de *Method, elasticity, spacegroup* e seus respectivos valores.....41
Figura 7 – Interface para rodar o *Web Scraping*.....42
Figura 8 – Inputs do *Web Scraping* 42
Figura 9 – Login com Google utilizado pelo *Web Scraping* 43
Figura 10 – Programa na página inicial, aguardando confirmação para prosseguir.....43
Figura 11 – Programa aguardando a validação de que está na página principal do *Materials Project* e pode prosseguir com o Web Scraping.....44
Figura 12 – Extração dos dados via *Web Scraping* 44
Figura 13 – Finalização da execução do *Web Scraping* e geração da base de dados local 45
Figura 14 – DataFrame final com os valores de K_{VRH} e G_{VRH} obtidos por ML via *Web Scraping*.....45
Figura 15 – Coluna Decomposes_To obtida via *Web Scraping* e colunas E_Young e Coef_Poisson calculadas com os valores extraídos de K_{VRH} e G_{VRH} obtidos por DFT ou ML. 46
Figura 16 – Aba gerada a partir do primeiro *DataFrame* extraído da API 46
Figura 17 – Aba referente ao *DataFrame* transformado, com os valores das colunas *spacegroup* e *elasticity* explodidos em mais colunas 47
Figura 18 – Aba referente ao *DataFrame* completo com a decomposição do composto e seus valores previstos de K_{VRH} e G_{VRH} extraídos via *Web Scraping*.....47
Figura 19 – Aba referente ao *DataFrame* final com as colunas de Módulo de Young e Coeficiente de Poisson também calculados.....48
Figura 20 – Interface inicial do programa gerador de visualizações aguardando o elemento químico desejado para análise 49

Figura 21 – Fim da execução do programa gerado por listas de tamanho diferentes 49
Figura 22 – Primeiro *DataFrame* “impresso” na tela com as informações de energia de formação para os compostos binários de 25% de alumínio. 50
Figura 23 – Gráficos das propriedades dos compostos binários com 75% de alumínio..... 51
Figura 24 – Gráficos das propriedades dos compostos binários com 66% de boro..... 52
Figura 25 – Gráficos das propriedades dos compostos binários com 25% de silício 53

Figura A-1 Criando o ambiente virtual do programa.....58
Figura A-2 Instalando o pacote conda, com o gerenciador de pacotes pip o Python e alguns extras 58
Figura A-3 Ativando o ambiente de trabalho ..... 58Figura A-4 Instalando o Jupyter Notebook 59
Figura H-1 *DataFrames* de energia de formação para os compostos binários de Al com os metais de transição nas proporções de 25%, 33%, 50%, 66% e 75% da esquerda para a direita. 79
Figura H-2 *DataFrames* de Módulo de Young para os compostos binários de Al com os metais de transição nas proporções de 25%, 33%, 50%, 66% e 75% da esquerda para a direita. 79
Figura H-3 – *DataFrames* do Coeficiente de Poisson para os compostos binários de Al com os metais de transição nas proporções de 25%, 33%, 50%, 66% e 75% da esquerda para a direita. 80
Figura H-4 – *DataFrames* do Módulo de Volume (K_{VRH}) para os compostos binários de Al com os metais de transição nas proporções de 25%, 33%, 50%, 66% e 75% da esquerda para a direita. 80
Figura H-5 – *DataFrames* do Módulo de Cisalhamento (G_{VRH}) para os compostos binários de Al com os metais de transição nas proporções de 25%, 33%, 50%, 66% e 75% da esquerda para a direita. 81
Figura I-1 - Gráficos das propriedades dos compostos binários com 33% de silício 82
Figura I-2 - Gráficos das propriedades dos compostos binários com 50% de silício 82
Figura I-3 - Gráficos das propriedades dos compostos binários com 66% de silício 83
Figura I-4 - Gráficos das propriedades dos compostos binários com 75% de silício 83
Figura I-5 - Gráficos das propriedades dos compostos binários com 25% de alumínio 84
Figura I-6 - Gráficos das propriedades dos compostos binários com 33% de alumínio 84

Figura I-7 - Gráficos das propriedades dos compostos binários com 50% de alumínio 85
Figura I-8 - Gráficos das propriedades dos compostos binários com 66% de alumínio 85
Figura I-9 - Gráficos das propriedades dos compostos binários com 25% de boro 86
Figura I-10 - Gráficos das propriedades dos compostos binários com 33% de boro 86
Figura I-11 - Gráficos das propriedades dos compostos binários com 50% de boro 87
Figura I-12 - Gráficos das propriedades dos compostos binários com 75% de boro 87

LISTA DE TABELAS

Tabela 1 - Relação da simetria do sistema cristalino com o número de constantes elásticas independentes 21

LISTA DE SIGLAS

AMT	*Alumineto de Metal de Transição*
API	*Application Programing Interface*
ETL	*Extract Transform Load*
CSV	*Comma-Separated Values*
IDE	*Integrated Development Environment*
DFT	*Density Functional Theory*
GGA	*Generalized Gradiente Approximation*
JSON	*JavaScript Object Notation*
SQL	*Structured Query Language*
HTML	*HyperText Markup Language*
URL	*Uniform Resource Locator*
MAPI	*Materials Application Programing Interface*
ML	*Machine Learning*
PDF	*Portable Document Format*
REST	*REpresentational State Transfer*
VASP	*Vienna Ab Initio Simulation Package*

LISTA DE SÍMBOLOS

σ	Tensão
ε	Deformação
E	Módulo de elasticidade/Young
C	Constante de rigidez elástica
s	Constante de complacência
E_f	Energia de formação
K	Módulo volumétrico
K_V	Módulo volumétrico de Voigt
K_R	Módulo volumétrico de Reuss
K_{VRH}	Módulo volumétrico de Voigt-Reuss-Hill
G	Módulo de cisalhamento
G_V	Módulo de cisalhamento de Voigt
G_R	Módulo de cisalhamento de Reuss
G_{VRH}	Módulo de cisalhamento de Voigt-Reuss-Hill
v	Coeficiente de Poisson
M	Módulo longitudinal
$\boldsymbol{\lambda}$	Primeiro parâmetro de Lamé
$\boldsymbol{\mu}$	Segundo parâmetro de Lamé
R_G	Razão de Pugh
P_C	Pressão de Cauchy
ζ	Parâmetro de deslocamento interno de Kleinman
A_L	Anisotropia elástica

SUMÁRIO

1 INTRODUÇÃO 17

1.1 Motivação 17

1.2 Justificativa 18

1.3 Organização do trabalho 18

2 FUNDAMENTAÇÃO TEÓRICA 19

2.1 Propriedades elásticas do material 19

2.1.1 Tensor elástico 19

2.1.2 Influência da simetria cristalina sobre o tensor elástico 21

2.1.3 Módulos elásticos de Voigt, Reuss e Hill 25

2.1.4 Módulo de Young (E) 27

2.1.5 Coeficiente de Poisson isotrópico ($\boldsymbol{\nu}$) 27

2.2 Density Functional Theory (DFT) 27

2.2.1 *Materials Project* 28

2.3 Energia de formação 28

2.4 Python 29

2.4.1 Bilbiotecas, módulos e *Frameworks* 29

2.5 Application Programming Interface *(API)* 31

2.6 *Web Scraping* 31

3 METODOLOGIA 33

3.1 Materiais e Tecnologias empregadas 33

3.1.1 Pacote Miniconda 3 33

3.1.2 IDE Jupyter Notebook 33

3.1.3 Chromedriver 33

3.1.4 Google Chrome 34

3.1.5 Materials Application Programming Interface (MAPI) 34

3.1.6 Python e suas bibliotecas 34

3.2 Métodos 35

3.2.1 Instalação do pacote Miniconda 3 35

3.2.2 Ambiente Virtual 35

3.2.3 Instalação do Jupyter Notebook e das bibliotecas 36

3.2.4 Importação das bibliotecas e definição de funções e variáveis 36

3.2.5 Extrator dos dados via API 36

3.2.6 Transformação dos dados 36

3.2.7 *Web Scraper* dos dados faltantes 37
3.2.8 Cálculo dos parâmetros desejados e da geração da base de dados local..... 37
3.2.9 Geração das visualizações..... 37
4 RESULTADOS E DISCUSSÃO..... 39
4.1 Extração dos dados via MAPI..... 39
4.2 Formatação do *DataFrame*..... 40
4.3 *Web Scraping* e planilha final..... 42
4.3.1 Geração da base de dados local..... 46
4.3.2 Geração de gráficos..... 48
5 CONCLUSÃO..... 54
REFERÊNCIAS..... 56
APÊNDICE A - Código de instalação do ambiente virtual e Jupyter Notebook..... 59
APÊNDICE B - Código ETL dos dados..... 61
APÊNDICE C - Código extrator dos dados da MAPI..... 64
APÊNDICE D - Código formatação do *DataFrame* gerado na extração da MAPI..... 65
APÊNDICE E - *Web Scraping* da dos valores de estabilidade..... 67
APÊNDICE F - Código do cálculo dos parâmetros e geração do arquivo como base de dados local..... 73
APÊNDICE G – Código de geração dos gráficos..... 74
APÊNDICE H – *DataFrames* das propriedades por composição dos compostos metálicos para o alumínio..... 80
APÊNDICE I –Gráficos das propriedades por composição dos compostos binários de alumínio, boro e silício com metais de transição..... 83

1 INTRODUÇÃO

Vivemos a era do Big Data! São gerados e coletados mais dados do que nós conseguimos processar. Esses dados as vezes são úteis e são armazenados em bases de dados que permitem acessar tais informações de uma maneira simples posteriormente (FIA, 2020). As APIs e os *Web Scrapings* são maneiras de extrair estes dados quando são advindos da internet. A evolução das APIs permitiu o compartilhamento dos dados de forma mais ágil e simples, possibilitando a extração de conjuntos de informações desejados via código.

A plataforma *Materials Project* possui diversos dados sobre as propriedades dos seus materiais possuindo uma API para fazer a extração (JAIN *et al.*, 2013). A extração, filtragem, visualização e armazenamento destes dados são funcionalidades muito desejáveis para uma análise das propriedades do material e, o trabalho teve enfoque em montar um programa de extração dos dados referentes a estabilidade elástica dos compostos binários metálicos de alumínio, boro e silício da plataforma.

A linguagem Python permite automatizar tarefas chatas e repetitivas que poderiam ser feitas manualmente, como uma consulta em uma API ou o *download* de uma informação disposta na página da *Web* (SWEIGART, 2015). Por isso o trabalho se propôs a utilizar essa nova ferramenta muito útil na esfera dos dados para montar dois programas, um voltado para a extração e armazenamento das informações, e o outro para acesso e geração das visualizações a partir da base de dados.

Com isso o desejo do trabalho é de extrair os dados dos materiais desejados de forma automatizada e mais rápida por meio da plataforma e, em seguida visualizar tais dados. Criando um modelo que com alguns ajustes possa ser útil para a extração e visualização de outras informações da plataforma de forma mais fácil.

1.1 Motivação

A motivação para a realização deste projeto estava em produzir uma ferramenta capaz de auxiliar na obtenção dos dados das propriedades físicas dos compostos binários metálicos de alumínio, boro e silício. Assim, estavam dentre os objetivos:

1. Desenvolver um *software* que possibilitasse a geração de bases de dados locais dos compostos binários de alumínio, boro e silício. Que futuramente também servisse como um modelo que possa ser adaptado para extrair quaisquer compostos binários.

2. Desenvolver um *software* de visualização dos dados extraídos e armazenados nas bases de dados locai que possibilitasse a geração de gráficos das propriedades físicas dos compostos binários metálicos.

1.2 Justificativa

O desejo de utilizar essas ferramentas e montar esse projeto foi algo mais pessoal, tinha como objetivo me aprofundar no tema e desenvolver um pouco mais minhas habilidades na programação e montar uma automação com ETL de dados e visualização. Além de permitir que outros estudantes tenham meios de extrair vários dados da plataforma de forma mais rápida.

1.3 Organização do trabalho

No Capítulo 2 são apresentados os conceitos físicos e computacionais teóricos para a compreensão da metodologia, dos resultados e das conclusões do trabalho. No Capítulo 3 está explicada a metodologia empregada no desenvolvimento da monografa, com os *softwares* empregados e as linguagens utilizadas. O Capítulo 4 expõe os resultados obtidos no desenvolvimento do programa para a extração das informações e geração da base de dados, e do programa de acesso e visualização das informações. O Capítulo 5 é responsável por concluir o trabalho com algumas conclusões feitas sobre os programas, baseada nas informações obtidas no trabalho e sobre algumas possíveis melhorias que podem ser feitas para permitir o programa extrair os dados de outra maneira. Por fim seguem os Apêndices com todos os códigos abordados nos resultados e que compõe o programa e, as imagens de interface, tabelas e gráficos gerados pelos programas. Os códigos finais podem ser acessados no repositório do *Github*:https://github.com/Alexandre-Leal/Binary-Compound-Data-Collector-RPA

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Propriedades elásticas do material

2.1.1 Tensor elástico

Um corpo sólido qualquer deformar-se-á ao ser aplicada uma força de tração ou compressão uniaxial sobre duas de suas superfícies opostas de seu corpo. A força aplicada gerará uma tensão mecânica interna, de compressão ou de tração respectivamente, que é responsável pela deformação física do material. Se a tensão gerada pela força aplicada estiver abaixo de um determinado limite, chamada de tensão limite de escoamento, o corpo não apresentará nenhuma deformação plástica e permanecerá no regime elástico, ou seja, ao ser retirada a tensão o corpo retornará à sua forma inicial (CALLISTER e RETHWISCH, 2016). Essa relação de tensão-deformação no regime elástico é conhecida como Lei de Hooke, descrita na equação (1), a constante de proporcionalidade *E* é chamada de módulo de Young ou módulo de elasticidade (CALLISTER e RETHWISCH, 2016). A Lei de Hooke pode ser escrita de forma generalizada como na equação (2), descrevendo para uma determinada tensão homogênea σ_{ij} o valor da deformação resultante ε_{kl} ocasionada no cristal (NYE, 1985).

$$\sigma = E \,.\, \varepsilon \tag{1}$$

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{2}$$

Sendo C_{ijkl} as constantes de rigidez elástica do cristal em questão. Dessa forma, a equação (2) passa a representar um conjunto de 9 equações que ao todo possuem 81 constantes elásticas C_{ijkl}. Mas, como σ_{ij} é a parte simétrica do tensor de tensão, a consideração abaixo pode ser feita abaixo (NYE, 1985).

$$\sigma_{ij} = \sigma_{ji} \;\Rightarrow\; C_{ijkl} = C_{jikl} \tag{3}$$

Além disso, a simetria do tensor de deformação implica na igualdade da equação (4) (NYE, 1985).

$$\varepsilon_{kl} = \varepsilon_{lk} \;\Rightarrow\; C_{ijkl} = C_{jikl} \tag{4}$$

A partir das considerações feitas nas equações (3) e (4) por conta da simetria dos tensores de tensão-deformação, o número de constantes elásticas independentes C_{ijkl} é reduzido de 81 para 36 (NYE, 1985).

A tensão pode ser obtida por meio da derivada parcial da energia potencial elástica *U* pela deformação ε_{ij} do material através da equação (5) (WANG, LEE e KAN, 2016 *apud* SINGH *et al.*, 2021).

$$\sigma_{ij} = \frac{\partial U}{\partial \varepsilon_{ij}} \tag{5}$$

A partir das equações (2) e (5) pode-se montar a relação presente na equação (6) (SINGH *et al.*, 2021).

$$C_{ijkl} = \frac{\partial \sigma_{ij}}{\partial \varepsilon_{kl}} \Rightarrow C_{ijkl} = \frac{\partial^2 U}{\partial \varepsilon_{kl} \partial \varepsilon_{ij}} \tag{6}$$

Por conta da arbitrariedade na ordem da diferenciação presente na equação (6), $C_{ijkl} = C_{jikl}$. Essa relação também é chamada de simetria maior do tensor de rigidez e reduz o número de constantes elásticas independentes C_{ijkl} de 36 para 21 (SINGH *et al.*, 2021).

Utilizando a notação matricial e abreviando para as constantes elásticas C_{ijkl} seus sufixos "*ij*" para um valor de 1 a 6 e "*kl*" para um valor de 1 a 6, por conta da simetria do segundo ranque dos tensores de tensão-deformação, é possível escrever as equações para as 36 constantes elásticas C_{nm}, das quais 21 são independentes (NYE, 1985). Dessa maneira, a notação de Voigt para a matriz dos tensores de tensão e deformação pode ser expressa no formato da equação (7) (SINGH *et al.*, 2021).

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{bmatrix} \tag{7}$$

De maneira similar, o tensor de complacência elástica, com $s_{ij} = C_{ij}^{-1}$, pode ser escrito na forma matricial como na equação (8) (SINGH *et al.*, 2021).

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} & s_{15} & s_{16} \\ s_{21} & s_{22} & s_{23} & s_{24} & s_{25} & s_{26} \\ s_{31} & s_{32} & s_{33} & s_{34} & s_{35} & s_{36} \\ s_{41} & s_{42} & s_{43} & s_{44} & s_{45} & s_{46} \\ s_{51} & s_{52} & s_{53} & s_{54} & s_{55} & s_{56} \\ s_{61} & s_{62} & s_{63} & s_{64} & s_{65} & s_{66} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} \tag{8}$$

2.1.2 Influência da simetria cristalina sobre o tensor elástico

Como visto acima, qualquer sistema cristalino possuirá no máximo 36 constantes de rigidez elástica C_{ijkl} das quais, dependendo da simetria do sistema cristalino, até 21 podem ser independentes, como acontece para o sistema Monoclínico. Basicamente quanto mais simétrico for o sistema cristalino menos constantes elásticas independentes ele possuirá, um exemplo disso é o sistema cúbico que possui apenas 3 constantes independentes (C_{11}, C_{12}, C_{44}).

A tabela 1 mostra a quantidade de constantes de rigidez elásticas independentes para cada sistema cristalino. Os sistemas cristalinos Romboédrico e Tetragonal apresentam quantidades de constantes elásticas independente dependendo da classe de Laue, por conta disto ela foi deixada entre parênteses nesses casos. A partir da tabela 1 fica mais evidente que o aumento da assimetria do sistema cristalino gera o aumento do seu número de constantes elásticas independentes presentes na sua matriz de rigidez elástica (MOUHAT e COUDERT, 2014), basta olhar o crescimento da quantidade de constantes independentes com o aumento das diferenciações dos parâmetros da rede.

Tabela 1 - Relação da simetria do sistema cristalino com o número de constantes elásticas independentes

Sistema cristalino	**Quantidade de C_{ij} independentes**	**Ângulos entre parâmetros**	**Relação dos parâmetros de rede**
Cúbico	3	$\alpha = \beta = \gamma = 90°$	$a = b = c$
Hexagonal	5	$\alpha = \beta = 90°, \gamma = 120°$	$a_1 = a_2 = a_3 \neq c$
Romboédrico (I) - ($\bar{3}$m)	6	$\alpha = \beta = \gamma \neq 90°$	$a = b = c$
Romboédrico (II) - ($\bar{3}$)	7	$\alpha = \beta = \gamma \neq 90°$	$a = b = c$
Tetragonal (I) - (4/mmm)	6	$\alpha = \beta = \gamma = 90°$	$a = b \neq c$
Tetragonal (II) - (4/m)	7	$\alpha = \beta = \gamma = 90°$	$a = b \neq c$
Ortorrômbico	9	$\alpha = \beta = \gamma = 90°$	$a \neq b \neq c$
Monoclínico	13	$\alpha = \gamma = 90° \neq \beta$	$a \neq b \neq c$
Triclínico	21	$\alpha \neq \beta \neq \gamma \neq 90°$	$a \neq b \neq c$

Fonte: Adaptada de MOUHAT e COUDERT (2014) e CALLISTER (2016)

Cada sistema cristalino deve obedecer determinadas condições, necessárias e suficientes, para que o sistema seja elasticamente estável. Tais condições podem ser descritas matematicamente na forma de inequações que relacionam as constantes de rigidez elástica da matriz de rigidez (MOUHAT e COUDERT, 2014). A seguir estão as matrizes de constantes elásticas e as condições matemáticas de estabilidade para cada sistema cristalino.

Para a matriz, os símbolos • foram utilizados para representar as constantes elásticas diferentes de zero da matriz simétrica e as constates elásticas iguais a zero foram deixadas em branco.

1.1.1.1. Cúbico

O sistema cúbico apresenta 3 constantes elásticas independentes: C_{11}, C_{12}, C_{44} e sua estabilidade pode ser descrita matematicamente pelas 3 inequações (10) que são originadas da sua matriz de constantes elásticas (9) (MOUHAT e COUDERT, 2014).

$$\begin{bmatrix} C_{11} & C_{12} & C_{12} & & & \\ \bullet & C_{11} & C_{12} & & & \\ \bullet & \bullet & C_{11} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ & & & & & C_{44} \end{bmatrix} \tag{9}$$

$$C_{11} - C_{12} > 0, \qquad C_{11} + 2C_{12} > 0, \qquad C_{44} > 0 \tag{10}$$

1.1.1.2. Hexagonal

O sistema Hexagonal apresenta 5 constantes elásticas independentes: C_{11}, C_{12}, C_{13}, C_{33}, C_{44} e sua estabilidade pode ser descrita matematicamente pelas 3 inequações (12) que são originadas da sua matriz de constantes elásticas (11) (MOUHAT e COUDERT, 2014).

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & & & \\ \bullet & C_{11} & C_{13} & & & \\ \bullet & \bullet & C_{33} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ & & & & & (C_{11} - C_{12})/2 \end{bmatrix} \tag{11}$$

$$C_{11} > |C_{12}|, \qquad 2C_{13}^2 < C_{33}(C_{11} + C_{12}), \qquad C_{44} > 0 \tag{12}$$

1.1.1.3. Tetragonal (I) - 4/mmm

O sistema Tetragonal (I) apresenta 6 constantes elásticas independentes: C_{11}, C_{12}, C_{13}, C_{33}, C_{44}, C_{66} e sua estabilidade pode ser descrita matematicamente pelas 4 inequações (13) que são originadas da sua matriz de constantes elásticas (14) (MOUHAT e COUDERT, 2014).

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & & & \\ \bullet & C_{11} & C_{13} & & & \\ \bullet & \bullet & C_{33} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ & & & & & C_{66} \end{bmatrix} \tag{13}$$

$$C_{11} > |C_{12}|, \qquad 2C_{13}^2 < C_{33}(C_{11} + C_{12}), \qquad C_{44} > 0, \qquad C_{66} > 0 \tag{14}$$

1.1.1.4. Tetragonal (II) - 4/m

O sistema Tetragonal (II) apresenta 7 constantes elásticas independentes: $C_{11}, C_{12}, C_{13}, C_{16}, C_{33}, C_{44}, C_{66}$ e sua estabilidade pode ser descrita matematicamente pelas 4 inequações (16) que são originadas da sua matriz de constantes elásticas (15) (MOUHAT e COUDERT, 2014).

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & & & C_{16} \\ \bullet & C_{11} & C_{13} & & & -C_{16} \\ \bullet & \bullet & C_{33} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ \bullet & \bullet & & & & C_{66} \end{bmatrix} \tag{15}$$

$$\begin{gathered} C_{11} > |C_{12}|, \qquad 2C_{13}^2 < C_{33}(C_{11} + C_{12}), \qquad C_{44} > 0, \\ 2C_{16}^2 > C_{66}(C_{11} - C_{12}) \end{gathered} \tag{16}$$

1.1.1.5. Romboédrico (I) - $\bar{3}$m

O sistema Romboédrico (I) apresenta 6 constantes elásticas independentes: $C_{11}, C_{12}, C_{13}, C_{14}, C_{33}, C_{44}$ e sua estabilidade pode ser descrita matematicamente pelas 4 inequações (18) que são originadas da sua matriz de constantes elásticas (17) (MOUHAT e COUDERT, 2014).

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & & \\ \bullet & C_{11} & C_{13} & -C_{14} & & \\ \bullet & \bullet & C_{33} & & & \\ \bullet & \bullet & & C_{44} & & \\ & & & & C_{44} & C_{14} \\ & & & & \bullet & C_{66} \end{bmatrix} \tag{17}$$

$$C_{11} > |C_{12}|, \qquad C_{44} > 0, \tag{18}$$

$$C_{13}^2 < \frac{1}{2}C_{33}(C_{11} + C_{12}),$$

$$C_{14}^2 < \frac{1}{2}C_{44}(C_{11} - C_{12}) \equiv C_{44}C_{66}$$

Sendo $C_{66} = \frac{1}{2}(C_{11} - C_{12})$. Assim como no caso do sistema cristalino Hexagonal.

1.1.1.6. Romboédrico (II) - $\bar{3}$

O sistema Romboédrico (II) apresenta 7 constantes elásticas independentes: C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{33}, C_{44} e sua estabilidade pode ser descrita matematicamente pelas 4 inequações (26) que são originadas da sua matriz de constantes elásticas (25) (MOUHAT e COUDERT, 2014).

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & \\ \bullet & C_{11} & C_{13} & -C_{14} & -C_{15} & \\ \bullet & \bullet & C_{33} & & & \\ \bullet & \bullet & & C_{44} & & -C_{15} \\ \bullet & \bullet & & & C_{44} & C_{14} \\ & & & \bullet & \bullet & C_{66} \end{bmatrix} \tag{19}$$

$$\begin{gathered} C_{11} > |C_{12}|, \qquad C_{44} > 0, \\ C_{13}^2 < \frac{1}{2}C_{33}(C_{11} + C_{12}), \\ C_{14}^2 + C_{15}^2 < \frac{1}{2}C_{44}(C_{11} - C_{12}) \equiv C_{44}C_{66} \end{gathered} \tag{20}$$

Sendo $C_{66} = \frac{1}{2}(C_{11} - C_{12})$. Assim como no caso do sistema cristalino Romboédrico (I).

1.1.1.7. Ortorrômbico

O sistema Ortorrômbico apresenta 9 constantes elásticas independentes: C_{11}, C_{12}, C_{13}, C_{22}, C_{23}, C_{33}, C_{44}, C_{55}, C_{66} e sua estabilidade pode ser descrita matematicamente pelas 6 inequações (22) que são originadas da sua matriz de constantes elásticas (27) (MOUHAT e COUDERT, 2014).

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & & & \\ \bullet & C_{22} & C_{23} & & & \\ \bullet & \bullet & C_{33} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ & & & & & C_{66} \end{bmatrix} \tag{21}$$

$$C_{11} > 0, \quad C_{44} > 0, \quad C_{55} > 0, \quad C_{66} > 0, \quad C_{11}C_{22} > C_{12}^2, \\ C_{11}C_{22}C_{33} + 2C_{12}C_{13}C_{23} - C_{11}C_{23}^2 - C_{22}C_{13}^2 - C_{33}C_{12}^2 > 0 \tag{22}$$

1.1.1.8. Monoclínico e Triclínico

Os sistemas cristalinos monoclínico e triclínico apresentam 13 e 21 constantes de rigidez elásticas independentes respectivamente e suas condições de estabilidade apresentam sistemas de inequações maiores e mais complexos e que não serão abordados no desenvolvimento deste trabalho.

2.1.3 Módulos elásticos de Voigt, Reuss e Hill

Para um material policristalino, os tensores de tensão e deformação variam na posição **r**. Dessa maneira a expressão da Lei de Hooke generalizada para um sistema policristalino linear e homogêneo, utilizando a notação de Voigt simplificada, pode ser dada pela equação (23) (SINGH *et al.*, 2021).

$$\sigma_i(\boldsymbol{r}) = C_{ij}(\boldsymbol{r})\varepsilon_j(\boldsymbol{r}) \tag{23}$$

Para calcular as constantes elásticas e propriedades mecânicas efetivas num material policristalino é preciso considerar os valores médios no sistema de coordenadas **r**. As equações (24) e (25)permitem determinar o valor médio da tensão $\langle\sigma\rangle$ e o valor médio da deformação $\langle\varepsilon\rangle$ para um sistema estatisticamente uniforme (SINGH *et al.*, 2021).

$$\langle\sigma\rangle = \frac{1}{\mathrm{V}}\int \sigma(\boldsymbol{r})\, d\boldsymbol{r} \tag{24}$$

$$\langle\varepsilon\rangle = \frac{1}{\mathrm{V}}\int \varepsilon(\boldsymbol{r})\, d\boldsymbol{r} \tag{25}$$

Sendo V o volume e assumindo que $\sigma(\boldsymbol{r})$e $\varepsilon(\boldsymbol{r})$ variam pouco e continuamente na posição do espaço **r.** Existem várias maneiras de calcular as constantes elásticas efetivas, sendo algumas das mais conhecidas as de Voigt, Reuss e Voigt-Reuss-Hill (SINGH *et al.*, 2021).

No esquema de Voigt é considerado que o campo da deformação é constante ao longo do material e, portanto, $\varepsilon(\boldsymbol{r}) = \varepsilon$, sendo ε independente de $\boldsymbol{r}$. Em contrapartida, o esquema de Reuss se baseia na premissa de que o campo da tensão é uniforme ao longo do material e, portanto, $\sigma(\boldsymbol{r}) = \sigma$ e é independente de **r**. No modelo de Voigt-Reuss-Hill, é feita uma média aritmética entre os valores dos modelos de Voigt e Reuss, o que acabou gerando uma boa estimativa quando comparada com valores medidos de forma experimental. Assim, o modelo Voigt-Reuss-Hill mostrou-se ser o mais verossímil dentre os três modelos (SINGH *et al.*, 2021).

❖ *Bulk Modulus* ou Módulo volumétrico (K)

O módulo volumétrico K é a razão entre a tensão aplicada e a deformação volumétrica de um determinado material, dessa maneira, ele é um parâmetro que descreve a capacidade do material a resistir às mudanças de volume quando aplicada uma tensão sobre este (POPLAVKO, 2019). As equações (26), (27) e (28) representam matematicamente o cálculo do módulo volumétrico K para os modelos de Voigt, Reuss e Voigt-Reuss-Hill respectivamente (SINGH *et al.*, 2021).

$$K_V = \frac{1}{9}\left[(C_{11} + C_{22} + C_{33}) + 2(C_{12} + C_{23} + C_{31})\right] \tag{26}$$

$$\frac{1}{K_R} = (s_{11} + s_{22} + s_{33}) + 2(s_{12} + s_{23} + s_{31}) \tag{27}$$

$$K_{VRH} = \frac{K_V + K_R}{2} \tag{28}$$

❖ Módulo de cisalhamento (G)

O módulo de cisalhamento, também chamado de módulo de rigidez, é a razão entre a tensão de cisalhamento e a deformação de cisalhamento (POPLAVKO, 2019). As equações (29), (30) e (31) são utilizadas para calcular os módulos de cisalhamento nos modelos de Voigt, Reuss e Voigt-Reuss-Hill respectivamente (SINGH *et al.*, 2021).

$$G_V = \frac{1}{15}\left[(C_{11} + C_{22} + C_{33}) - (C_{12} + C_{23} + C_{31}) + 3(C_{44} + C_{55} + C_{66})\right] \tag{29}$$

$$\frac{1}{G_R} = 4(s_{11} + s_{22} + s_{33}) - 4(s_{12} + s_{23} + s_{31}) + 3(s_{44} + s_{55} + s_{66}) \tag{30}$$

$$G_{VRH} = \frac{G_V + G_R}{2} \tag{31}$$

Para o cálculo das propriedades abaixo, nas equações onde constam os módulos elástico *K* e *G*, foram utilizados os módulos elásticos K_{VRH} e G_{VRH} respectivamente, obtidos por meio do modelo de Voigt-Reuss-Hill.

2.1.4 Módulo de Young (E)

O módulo de Young, ou também chamado de módulo elástico, representa a capacidade de um determinado material se alongar ou se comprimir ainda no regime elástico quando aplicada sobre ele algum tipo de tensão (POPLAVKO, 2019). Sendo ele a constante de proporcionalidade na relação tensão-deformação (NYE, 1985). O módulo de Young pode ser calculado a partir dos módulos volumétrico K e de cisalhamento G utilizando a equação (32) (SINGH *et al.*, 2021).

$$E = \frac{9KG}{3K + G} \tag{32}$$

2.1.5 Coeficiente de Poisson isotrópico ($\boldsymbol{\nu}$)

O coeficiente de Poisson indica o quanto a seção transversal de uma amostra deforma quando o material é submetido à tensão longitudinal, aplicada ao longo de seu comprimento, e que, portanto, deforma a amostra longitudinalmente. Esse coeficiente permite acompanhar o comportamento da deformação da seção transversal do material conforme o corpo se deforma longitudinalmente (POPLAVKO, 2019). O coeficiente de Poisson para um determinado material isotrópico pode ser calculado utilizando a equação (33) (SINGH *et al.*, 2021).

$$\nu = \frac{3K - E}{6K} = \frac{3K - 2G}{2(3K + G)} \tag{33}$$

2.2 Density Functional Theory (DFT)

A DFT, também chamada em português de Teoria do Funcional da Densidade, possui dois teoremas principais que são utilizados para o método de cálculos de primeiros princípios quando se busca prever os valores das propriedades do material. Estes dois principais teoremas são o de Hohenberg, Kohn e Sham (HOHENBERG, KOHN e SHAM, 1990).

O primeiro teorema afirma que a energia no estado fundamental é um funcional da densidade eletrônica. Isso implica que a densidade eletrônica é a única variável que influencia a energia do estado fundamental e, portanto, é a única que precisa ser calculada para se

determinar a energia e outras propriedades. O segundo teorema afirma que a densidade eletrônica do estado fundamental é a densidade que minimiza a energia do sistema.

As contas da DFT poderiam chegar em resultados exatos, no entanto existe a presença do potencial de correlação e troca cujo valor não é conhecido. Para resolver este problema, existem diversas considerações quanto ao valor deste potencial. Para os valores presentes neste trabalho, que foram extraídos da plataforma do *Materials Project*, a consideração feita pelo *Materials Project* para o potencial de correlação e troca foi a aproximação *Generalized Gradient Approximation* (GGA).

2.2.1 *Materials Project*

Os valores presentes no *Materials Project* são calculados utilizando a DFT a p *software* VASP (JAIN *et al.*, 2011), para calcular o valor da energia total dos compostos a conta com a DFT é realizada e um ajuste é feito utilizando a aproximação GGA (JAIN *et al.*, 2011).

Para obter as constantes elásticas o *Materials Project* utiliza o método dos primeiros princípios da DFT, utilizando simulações computacionais para tentar prever os resultados das propriedades elásticas dos materiais. Inicialmente é realizado um relaxamento de toda a tensão no cristal, até que ela se torne zero. Em seguida perturbações são aplicadas nos vetores da rede e o tensor de tensão resultante é calculado. Por fim o ajuste do tensor é realizado utilizando as relações constitutivas linear de tensão-deformação de elasticidade linear e as propriedades derivadas do tensor são calculadas (JONG *et al.*, 2015).

2.3 Energia de formação

A energia de formação de um composto ($H_f^{A_nB_m}$) pode ser obtida na DFT através da diferença entre a energia total do composto com as energias potenciais químicas dos elementos separados. Na DFT a energia de formação de um composto pode ser calculada segundo a equação (34).

$$H_f^{A_nB_m} = E(A_nB_m) - n\mu_A - m\mu_B \quad (34)$$

Onde μ_A e μ_B são as energias potenciais químicas dos elementos químicos A e B e E(A_nB_m) é a energia total do composto A_nB_m.

2.4 Python

2.4.1 Bilbiotecas, módulos e *Frameworks*

De forma simples, podemos dizer que uma biblioteca possui um conjunto de módulos que agregam funcionalidades diferentes à ela e a compõem (LUNDH, 2001).

Os *Frameworks* são conjuntos de módulos que também ajudam a desenvolver aplicações *Web*. Podem se parecer muito com as bibliotecas em alguns casos, mas em geral possuem duas grandes diferenças em comparação a elas. A primeira é que ao utilizar uma biblioteca o programador chama seus métodos e controla a ação, já em um *Framework* esse controle é invertido, sendo definido pelo *Framework* as ações que ser utilizadas. A segunda é que uma biblioteca realiza operações específicas enquanto os *Frameworks* possuem um fluxo básico e o restante deve ser construído pelo programador (WASEEM, 2021).

- NumPy

A NumPy é uma biblioteca *open source* para a computação cientifica. Ela contém diversas funções matemáticas que são muito úteis. A NumPy substitui algumas das funcionalidades do Matlab e Mathematica, possibilitando uma rápida prototipação no próprio código. A NumPy permite deixar o código mais enxuto do que se utilizase apenas Python para obter os mesmos resultados, pois suas funções matemáticas permitem simplificar muito o código resultando em um código mais eficiente e enxuto (IDRIS, 2013).

- Pandas

A Pandas é uma biblioteca ideal para trabalhar com dados arquivados na forma de planilhas ou bancos de dados relacionais, com dados armazenados de forma tabular. Ela é a ferramenta ideal no Python para explorar, limpar, e processar os dados. A biblioteca Pandas suporta integração com diversos tipos de formatos como: CSV, Excel, SQL, JSON, entre outros. A biblioteca possui maneiras de filtrar as informações muito eficientemente, podendo utilizar diversas condições diferentes para determinar a informação a ser filtrada (MCKINNEY *et al.*, 2021).

- Pymatgen

A Pymatgen é uma biblioteca que permite a extração dos dados do sistema do *Materials Project* via API, denominada MAPI. Ela possui um empacotador conveniente para a MAPI para possibilitar a extração de múltiplos dados do sistema em uma só consulta utilizando REST (ONG, et al., 2013).

- Selenium

O Selenium é um *Framework* que possui uma série de ferramentas úteis para simular um usuário e testar o funcionamento das aplicações, no entanto, ele também permite a extração de dados via *browser* o que também é conhecido como *Web Scraping* (CHACON, 2021). Essa extração ocorre por meio das informações dispostas na tela do seu navegador automático (*WebDriver*) e por conta disto, essa extração também foi chamada de *Screen Scraping* (GUNDECHA, 2014).

- Getpass

O módulo Getpass faz parte da biblioteca padrão do Python e permite omitir informações no código, ele é bastante útil quando se deseja deixar a informação digitada pelo usuário oculta, como no caso de senhas (LUNDH, 2001).

- Sys

O módulo Sys faz parte da biblioteca padrão do Python e permite controlar a rotina do programa com algumas funções, podendo por exemplo forçar o término da execução do programa no momento desejado (LUNDH, 2001).

- Matplotlib

A Matplotlib é uma biblioteca para plotagens 2D que permite a geração de pontos, histogramas, gráficos de barras, gráficos de linhas com marcadores ou apenas de linhas e muitos outros utilizando Python (MORGAN, 2018).

2.5 Application Programming Interface *(API)*

O termo API é um acrônimo de *Application Programming Interface* e, como o nome sugere, as APIs são as interfaces entre dois *softwares*, que permitem com que esses programas "conversem" entre si, compartilhando as informações entre eles. A API possibilitou a obtenção de dados de um programa para outro de maneira mais ágil e simples. Além disso, elas também podem ser utilizadas como via para armazenar informações no sistema, permitindo que o programa compartilhe os dados com o programa na outra ponta da API. As APIs possuem algumas arquiteturas diferentes entre si e, essas variações, implicam no seu tipo, que podem ser classificadas como: REST, SOAP, Webhooks, GraphQL, WebSocket, gRPC, EDI, MQTT, AMQP ou *Server-sent event* (LANE, 2020).

A maioria das APIs pedem identificação do usuário para que o programa possa estar acessando e extraindo os dados do outro programa. Essa identificação é feita por meio de credenciais que servem como identificadores únicos para aquele usuário. Dentre as opções de credenciais para uma API estão nome, senha, token, chaves etc. (RAPIDAPI, 2021).

No caso da API do *Materials Project* (MAPI) a credencial necessária é a chave da API e para obter a chave da API basta criar uma conta no sistema e acessar a página API, nela a chave está disposta (ONG *et al.*, 2014).

Após fazer a autenticação com a sua credencial, o usuário pode obter os dados por meio de bibliotecas de requests, que no Python seria a *requests*. No entanto, no caso da MAPI existe uma biblioteca ainda melhor desenvolvida especificamente para a extração de grandes quantidades de dados dos compostos presentes nela, a biblioteca *Pymatgen*. Com ela é possível fazer as consultas das propriedades para os compostos indicados bastando seguir o modelo do código descrito na sua documentação (ONG *et al.*, 2013).

2.6 *Web Scraping*

Web Scraping é a prática de extrair os dados da internet de forma automatizada por meio de programas, sendo uma prática tão antiga quanto a própria internet. No entanto, no passado essa prática foi conhecida com outros nomes, como: *screen scraping*, *data mining* ou *web harvesting* (MITCHELL, 2018).

3 METODOLOGIA

Esta seção apresenta as tecnologias empregadas para a elaboração dos *softwares* de extração, formatação e visualização de dados e as metodologias empregadas para atingir o resultado.

3.1 Materiais e Tecnologias empregadas

Para a elaboração do projeto foram utilizados um notebook com 8Gb de RAM e um processador Intel (R) Core (TM) i5-5200U CPU @ 2.20GHz 2.20 GHz e uma conexão de internet de 100 Mb além das seguintes tecnologias:

3.1.1 Pacote Miniconda 3

O pacote Miniconda 3é um instalador mínimo para o Conda e que já possui com ele algumas bibliotecas úteis no desenvolvimento do *software*, como a biblioteca *pip* (ANACONDA, 2017). Para o instalador Miniconda3 foi utilizado o Windows 64-bit pois era o compatível com o sistema operacional do computador, que era um Windows 10.

3.1.2 IDE Jupyter Notebook

A IDE (*Integrated Development Environment*) ou também conhecido como Ambiente de Desenvolvimento Integrado é uma aplicação *web open-source* que permite a criação de códigos com visualizações e tratamentos de dados. Além disso ele permite o uso de blocos de código que são muito úteis para desenvolver o código em partes, gerando ótimas visualizações durante a execução (PROJECT JUPYTER, 2021).

A IDE Jupyter Notebook foi utilizada para desenvolver os códigos em Python e serviu como interpretador para o *software* desenvolvido em Python.

3.1.3 Chromedriver

O Chromedriver é um servidor que implementa o WebDriver, uma ferramenta *open-source* para automações na *Web* utilizando o navegador Google Chrome. Além disso, a versão utilizada para o ChromeDriver foi a 96.0.4664.45, a que é compatível com a versão do navegador que estava instalado no computador.

Ele foi o programa utilizado para rodar o Google Chrome automático que foi desenvolvido utilizando o *Framework* Selenium em Python.

3.1.4 Google Chrome

O Google Chrome é um navegador *Web* amplamente utilizado no mundo, o qual, assim como supracitado, o próprio driver (Chromedriver) também utiliza como meio para rodar os códigos de *Web Scraping* no navegador. O Google Chrome possui a função de Inspecinar elemento que permite acessar o código *HyperText Markup Language* (HTML) do programa.

3.1.5 Materials Application Programming Interface (MAPI)

A *Materials Application Programming Interface* (MAPI) é uma API aberta que permite acessar os dados do *Materials Poject* utilizando os princípios de *REpresentational State Transfer* (REST), assim um *Uniform Resource Identifier* (URI), identificador que designa determinados dados do composto procurado (ONG *et al. 2014*).

3.1.6 Python e suas bibliotecas

A linguagem empregada no desenvolvimento do *software* foi o Python, ela foi escolhida por se tratar de uma linguagem com vastas bibliotecas com usos pertinentes para a elaboração do *software*, como*:* Pip, Pymatgen, Pandas, Matplotlib, NumPy.

- ❖ Pip

A Pip é uma biblioteca com um pacote de instalação para o Python, e foi utilizada para instalar as demais bibliotecas, como a Pandas,NumPy e Pymatgen.

- ❖ Pymatgen

A Pymatgen foi utilizada para extrair os dados da MAPI, e graças a ela foi possível consultar e extrair os dados das propriedades físicas do material de forma mais rápida e eficiente.

- ❖ NumPy

A biblioteca NumPy foi utilizada para realizar operações matemáticas e alguns recursos especiais como valores não encontrados. Em conjunto com a Pandas ela foi utilizada para: o tratamento dos dados extraídos e a geração da base de dados local.

- Pandas

A biblioteca Pandas foi utilizada para tratar os dados e gerar o arquivo final com as informações obtidas e tratadas. Dessa maneira a Pandas foi empregada para formatar as informações em *DataFrames* e, em seguida, para gerar a base de dados local em um arquivo Excel.

- Selenium

O Selenium foi usado para o desenvolvimento do *Web Scraping* das informações que não estavam disponíveis na API, mas apenas na interface *Web* do Site. Graças a ela foi possível montar a parte do código referente ao *Web Scraping* das informações via interface *Web*

- Matplotlib

A Matploblib foi utilizada para a geração dos gráficos personalizado das propriedades dos materiais para cada composição dos compostos binários com metais de transição.

3.2 Métodos

O desenvolvimento do projeto está separado em 3 grandes etapas: a instalação e preparo do ambiente virtual para o desenvolvimento do código (3.2.1 a 3.2.3), o ETL das informações gerando a base de dados locais (3.2.4 a 3.2.8) e a visualização dos dados extraídos (3.2.9).

3.2.1 Instalação do pacote Miniconda 3

O primeiro passo para o desenvolvimento do projeto foi a instalação do pacote Miniconda 3 que já instala o Python e o instalador de pacotes *Pip* e Conda que permitiram instalar as demais bibliotecas.

3.2.2 Ambiente Virtual

Em seguida, com o instalador de pacotes conda, uma partição da máquina responsável por conter apenas informações desse projeto foi criada, passo muito útil para separar um desenvolvimento dos demais. Portanto, usando o pacote conda, foi criado o ambiente virtual do projeto para separá-lo dos demais desenvolvimentos presentes na máquina e retirar possíveis

interferências que as bibliotecas de outras versões de outros desenvolvimentos pudessem ocasionar (APÊNDICE A).

3.2.3 Instalação do Jupyter Notebook e das bibliotecas

Logo em seguida, utilizando o instalador conda, a IDE Jupyter Notebook foi instalada (APÊNDICE A). Após isso, todas as bibliotecas necessárias para o desenvolvimento do projeto também foram instaladas utilizando os pacotes conda ou Pip.

3.2.4 Importação das bibliotecas e definição de funções e variáveis

As bibliotecas instaladas foram ativadas dentro do código para realizar a extração e transformação dos dados. Além disso, algumas definições foram feitas, como a chave da API, a lista com os elementos de transição a serem combinados, a lista com os elementos que o usuário pode combinar com os de transição e o elemento que o usuário deseja combinar, com essas informações foi possível dar continuidade na extração (APÊNDICE B).

3.2.5 Extrator dos dados via API

A primeira parte do desenvolvimento foi a confecção do programa extrator de dados via API, realizando consultas na MAPI e coletando valores de determinadas propriedades de um composto indicado. Para isso foi utilizada a biblioteca Pymatgen que permitiu a consulta e extração das propriedades de diversos compostos através do MPRester, utilizou-se o token para realizar a autenticação na API (APÊNDICE C).

3.2.6 Transformação dos dados

Após a extração dos dados via Pymatgen foi necessário armazená-los em forma de tabelas e para isso as bibliotecas Pandas e NumPy foram utilizadas, gerando novas colunas, deletando colunas desnecessárias, renomeando colunas e reordenando as informações de acordo com os objetivos. Essas bibliotecas permitiram a transformação de dicionários para novas colunas, tornando possível a organização adequada dos dados dentro dos *DataFrames* (APÊNDICE D).

3.2.7 *Web Scraper* dos dados faltantes

Para conseguir acessar os dados faltantes na API, predição de $K_{VRH,}$ G_{VRH} e *Decomposes To*, foi necessário elaborar em uma parte do código um *Web Scraping* destas três informações faltantes via interface *Web*.

Para isso, o *Framework* Selenium do Python foi utilizado no desenvolvimento desta etapa, permitindo a confecção de um *script* que navegou entre as páginas dos compostos e extraiu informações desejadas (APÊNDICE E).

Ao utilizar o código do (APÊNDICE E) você poderá realizar o login com uma conta de email USP, ou logar manualmente antes de dar o ok no input para prosseguir.

Assim o *script* irá logar na interface Web do site e em seguida irá acessar individualmente cada uma das páginas dos compunds-ids extraídos via API de uma base de dados igual a cópia do df extraído via API (APÊNDICE E).

3.2.8 Cálculo dos parâmetros desejados e da geração da base de dados local

Finalmente, com os valores de K_{VRH} e G_{VRH} de todos os compostos binários metálicos, as relações de tensão-deformação como o Módulo de Young e o Coeficiente de Poisson, podem ser calculadas, gerando a base de dados local com todos os dados extraídos, transformados e calculados (APÊNDICE F).

3.2.9 Geração das visualizações

A partir deste ponto um segundo programa foi confeccionado, separadamente, mas no mesmo ambiente virtual criado na seção 3.2.2. A separação foi feita para permitir que as visualizações pudessem ser geradas por um segundo programa independente utilizando as bases de dados gerada pelo *software* de extração. Assim, as visualizações puderam ser geradas a partir das bases de dados locais criadas para o alumínio, boro e silício.

Como o código foi escrito separado, as bibliotecas necessárias precisaram ser importadas, como no caso da Matplotlib que permitiu a geração dos gráficos. Além disso, os parâmetros desejados foram inseridos na forma de listas para indicar ao código qual arquivo Excel e quais parâmetros filtrar para posteriormente gerar a visualização.

Dessa forma os dados da base de dados local foram separados em diversos *DataFrames* diferentes a partir de sua composição referente ao elemento químico escolhido pelo usuário e,

após isso, todos essas tabelas filtradas foram utilizadas para gerar os gráficos personalizados de todos os parâmetros para cada composição de cada um dos compostos binário metálicos (APÊNDICE G).

4 RESULTADOS E DISCUSSÃO

4.1 Extração dos dados via MAPI

O MPRester da Pymatgen possibilitou extrair os dados por consultas na MAPI. Com o computador e rede de conexão informados foi possível extrair mais de 9200 dados em até 40 segundos, sendo que o código também empilhou os dados coletados de cada composto em um único *DataFrame* neste mesmo tempo.

Para a extração de dados ocorrer o usuário deve informar o elemento químico desejado e que será combinado com os 30 metais de transição expostos. Em seguida, todos os compostos metálicos binários entre o elemento selecionado e os metais de transição serão extraídos da MAPI. Mas para isso o usuário deverá em seguida permitir o início da extração API, via comandos na interface do programa, a Figura 1 apresenta essa interface inicial onde o elemento desejado para a combinação deverá ser escolhido.

Figura 1 – Interaface inicial referente as informações iniciais do programa

```
================================================================================================================================
                                        METAIS DE TRANSIÇÃO UTILIZADOS PARA A COMBINAÇÃO
================================================================================================================================

Lista dos metais de transição: [Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn, Y, Zr, Nb, Mo, Tc, Ru, Rh, Pd, Ag, Cd, La, Hf, Ta, W, R
e, Os, Ir, Pt, Au, Hg]

Número total de elementos: 30

================================================================================================================================
                                        DEFININDO O ELEMENTO A SER COMBINADO
================================================================================================================================


Olá, seja bem-vindo ao extrator de dados!

Para prosseguir selecione um dos símbolos químicos abaixo para extrair suas combinações com os 30 metais de transição listados
acima.

Os símbolos químicos disponíveis são:
[Al, B, Si]

Insira o símbolo químico desejado:
```

Fonte: Autoria própria

Figura 2 – Selecionando o elemento e iniciando a extração da API

```
================================================================================================================================
                                        DEFININDO O ELEMENTO A SER COMBINADO
================================================================================================================================

Olá, seja bem-vindo ao extrator de dados!

Para prosseguir selecione um dos símbolos químicos abaixo para extrair suas combinações com os 30 metais de transição listados
acima.

Os símbolos químicos disponíveis são:
[Al, B, Si]

Insira o símbolo químico desejado: Si

================================================================================================================================
                                        EXTRAÇÃO DOS DADOS VIA API
================================================================================================================================

Observação: A partir de agora algumas perguntas para prosseguir serão realizadas, as opções estarão entre colchetes -> [y/n] qu
e siginificam "yes or no" para prosseguir ou não prosseguir.


Podemos iniciar a extração dos dados dos compostos SiM? [y/n]
```

Fonte: Autoria própria

A *Pymatgen* possibilitou a extração dos dados de fórmula química, magnetização total, energia de formação, grupo espacial (conjunto de dados), elasticidade (conjunto de dados), densidade, sistema cristalino e volume da célula unitária para cada um dos compostos binários metálicos extraídos. No entanto o formato das informações vindas da consulta da API era de dicionário e, por conta disto, a Pandas foi utilizada para transformar essa estrutura de dicionários no *DataFrame* da Figura 3.

Assim, em menos de 1 minuto, os dados das propriedades acima foram extraídos no formato de dicionário e em seguida armazenados no *DataFrame* da Figura 3.

Figura 3 – Geração do *Dataframe* com os dados da *MAPI* utilizando a *Pymatgen e a Pandas*

```
================================================================================================================================
                                                EXTRAÇÃO DOS DADOS VIA API
================================================================================================================================

Observação: A partir de agora algumas perguntas para prosseguir serão realizadas, as opções estarão entre colchetes -> [y/n] qu
e siginificam "yes or no" para prosseguir ou não prosseguir.

Podemos iniciar a extração dos dados dos compostos SiM? [y/n]y

Extraindo os dados e gerando o DataFrame...

DataFrame gerado com sucesso!

Tempo de execução --- 28.54 seconds ---
```

	material_id	pretty_formula	total_magnetization	formation_energy_per_atom	spacegroup	elasticity	density	crystal_system	volume
0	mp-7822	Sc5Si3	0.001283	-0.768762	{'symprec': 0.1, 'source': 'spglib', 'symbol':...	{'G_Reuss': 72.0, 'G_VRH': 72.0, 'G_Voigt': 72...	3.252083	hexagonal	315.592481
1	mp-2841	ScSi2	0.004605	-0.371364	{'symprec': 0.1, 'source': 'spglib', 'symbol':...	{'G_Reuss': -184.0, 'G_VRH': -83.0, 'G_Voigt':...	3.305653	hexagonal	50.799400
2	mp-9969	ScSi	0.003176	-0.837596	{'symprec': 0.1, 'source': 'spglib', 'symbol':...	{'G_Reuss': 66.0, 'G_VRH': 70.0, 'G_Voigt': 74...	3.336116	orthorhombic	72.712176
3	mp-1190215	Sc5Si3	1.106745	-0.290534	{'symprec': 0.1, 'source': 'spglib', 'symbol':...	None	3.038162	hexagonal	337.813713
4	mp-1209086	Sc5Si4	0.017014	-0.765365	{'symprec': 0.1, 'source': 'spglib', 'symbol':...	None	3.269256	orthorhombic	684.930707

Fonte: Autoria própria

4.2 Formatação do *DataFrame*

As informações extraídas da API e inicialmente armazenadas no *DataFrame* ficaram dispostas exatamente como na Figura 3. No entanto, as colunas *spacegroup* e *elasticity* apresentaram dicionários como sendo seus valores. Isso ocorreu pois, nesses dois casos, o valor do dicionário extraído pela *Pymatgen* relacionado às chaves *spacegroup* e *elasticity* também eram dicionários, ou seja, para as colunas *spacegroup* e *elasticity* os valores armazenados eram dicionários e, portanto, foi preciso realizar um novo processo de transformação das chaves destes dois dicionários em novas colunas do *DataFrame* adicionando também seus respectivos valores. Para isso, foram adicionadas ao *DataFrame* novas colunas: *spacegroup_number*, *point_group*, *G_Reuss*, *G_VRH*, *G_Voigt*, *K_Reuss*, *K_VRH*, *K_Voigt*, *C11*, *C12*, *C13*, *C14*, *C15*, *C16*, *C21*, *C22*, *C23*, *C24*, *C25*, *C26*, *C31*, *C32*, *C33*, *C34*, *C35*, *C36*, *C41*, *C42*, *C43*, *C44*, *C45*, *C46*, *C51*, *C52*, *C53*, *C54*, *C55*, *C56*, *C61*, *C62*, *C63*, *C64*, *C65*, *C66*; com os valores

das respectivas chaves dos dicionários de *spacegroup* e *elasticity*. Após isso, as colunas *elasticity* e *spacegroup* foram deletadas e a coluna *Method* contendo a informação do método utilizado para a obtenção dos valores de K_{VRH} e G_{VRH} foi gerada indicando o método "DFT" para os valores extraídos via API. Dessa maneira, o *DataFrame* resultante desse processo ficou como nas Figuras 4, 5 e 6.

Figura 4 - *DataFrame* com as novas colunas de *Method*, *elasticity*, *spacegroup* e seus respectivos valores

	material_id	pretty_formula	total_magnetization	formation_energy_per_atom	density	crystal_system	cell_volume	spacegroup_number	point_group	G_Reuss	G_VRH	G_Voigt	K_Reuss	K_VRH	K_Voigt
0	mp-1209129	Sc4Al	0.073436	0.521431	2.503547	cubic	274.337263	227.0	m-3m	NaN	NaN	NaN	NaN	NaN	NaN
1	mp-999204	ScAl3	0.000114	-0.361746	2.916630	tetragonal	71.679549	139.0	4/mmm	63.0	64.0	65.0	94.0	94.0	94.0
2	mp-813	ScAl2	0.001419	-0.486889	3.009750	cubic	109.151154	227.0	m-3m	69.0	69.0	69.0	88.0	88.0	88.0
3	mp-11220	Sc2Al	0.175192	-0.350948	3.024175	hexagonal	128.369562	194.0	6/mmm	44.0	44.0	45.0	70.0	70.0	71.0
4	mp-978498	Sc3Al	1.255009	-0.247560	3.091582	cubic	86.931890	221.0	m-3m	34.0	36.0	37.0	52.0	52.0	52.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
246	mp-1183189	Al3Au	0.000144	-0.113982	6.933301	tetragonal	66.560267	139.0	4/mmm	NaN	NaN	NaN	NaN	NaN	NaN
247	mp-1228956	AlAu4	0.020434	-0.136008	14.968187	monoclinic	90.397495	12.0	2/m	NaN	NaN	NaN	NaN	NaN	NaN
248	mp-569558	AlAu2	0.000005	-0.316636	13.667781	orthorhombic	204.552664	62.0	mmm	31.0	32.0	34.0	117.0	118.0	119.0
249	mp-10871	AlAu	0.000229	-0.243324	11.097747	cubic	33.509015	221.0	m-3m	-19.0	-21.0	-23.0	147.0	147.0	147.0
250	mp-30550	AlAu2	0.000099	-0.313114	13.709394	orthorhombic	509.829422	58.0	mmm	NaN	NaN	NaN	NaN	NaN	NaN

Fonte: Autoria própria

Figura 5 - *DataFrame* com as novas colunas de *Method*, *elasticity*, *spacegroup* e seus respectivos valores

elastic_anisotropy	universal_anisotropy	poisson_ratio	Method	warnings	C11	C12	C13	C14	C15	C16	C21	C22	C23	C24	C25	C26
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.22	0.22	0.22	DFT	NaN	167.804222	70.389204	49.418412	0.000006	0.0	0.0	70.389204	167.804222	49.418412	0.0	-0.000005	0.0
0.01	0.01	0.19	DFT	NaN	186.213145	38.590199	38.590199	0.000000	0.0	0.0	38.601216	186.220569	38.601216	0.0	0.000000	0.0
0.20	0.20	0.24	DFT	NaN	132.370197	60.584140	27.470937	0.000000	0.0	0.0	59.502663	133.880264	27.905192	0.0	0.000000	0.0
0.38	0.38	0.22	DFT	NaN	86.080589	34.968520	34.968567	0.000000	0.0	0.0	34.960180	86.053022	34.960149	0.0	0.000000	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.41	0.41	0.37	DFT	NaN	140.329484	94.081719	103.654049	0.000000	0.0	0.0	92.963969	169.610283	87.755593	0.0	0.000000	0.0
1.11	1.11	0.57	DFT	NaN	98.541017	170.864924	170.864924	0.000000	0.0	0.0	170.864924	98.541017	170.864924	0.0	0.000000	0.0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Fonte: Autoria própria

Figura 6 - *DataFrame* com as novas colunas de *Method*, *elasticity*, *spacegroup* e seus respectivos valores

C31	C32	C33	C34	C35	C36	C41	C42	C43	C44	C45	C46	C51	C52	C53	C54	C55	C56	C61	C62	C63	C64	C65	C66
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
51.912518	51.912518	167.624716	0.000019	0.000019	0.0	0.000000	1.413079e-07	0.000000	62.359826	-3.964194e-07	7.065399e-08	1.413079e-07	0.000000	0.000000	3.964194e-07	62.359826	7.065398e-08	0.920994	-0.920994	0.000000	0.0	-6.142580e-09	92.094019
38.593454	38.593454	186.213449	0.000000	0.000000	0.0	0.010863	4.175399e-03	0.000328	66.327780	0.000000e+00	0.000000e+00	4.610909e-03	0.011125	0.000777	0.000000e+00	66.327454	0.000000e+00	0.004883	0.001448	0.011805	0.0	0.000000e+00	66.325966
27.566648	27.566648	137.841489	0.000000	0.000000	0.0	-0.000077	1.800964e-04	0.000020	47.328072	0.000000e+00	0.000000e+00	3.160338e-05	0.000012	0.000069	0.000000e+00	47.279779	0.000000e+00	-0.045384	0.003946	-0.015270	0.0	0.000000e+00	35.600452
34.964214	34.964131	86.066357	0.000000	0.000000	0.0	-0.096626	-8.267627e-02	-0.082669	44.594887	0.000000e+00	0.000000e+00	8.267729e-02	0.096626	0.082670	0.000000e+00	44.595711	0.000000e+00	0.000000	0.000000	0.000000	0.0	0.000000e+00	44.595288
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
103.264366	87.645093	195.655017	0.000000	0.000000	0.0	0.000018	-2.051332e-05	-0.000014	37.884676	0.000000e+00	0.000000e+00	-1.455003e-05	-0.000028	-0.000007	0.000000e+00	34.227856	0.000000e+00	-0.000268	-0.000142	-0.001838	0.0	0.000000e+00	22.145306
70.864924	170.864924	98.541017	0.000000	0.000000	0.0	0.000000	0.000000e+00	0.000000	-14.292538	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000e+00	-14.292538	0.000000e+00	0.000000	0.000000	0.000000	0.0	0.000000e+00	-14.292538
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

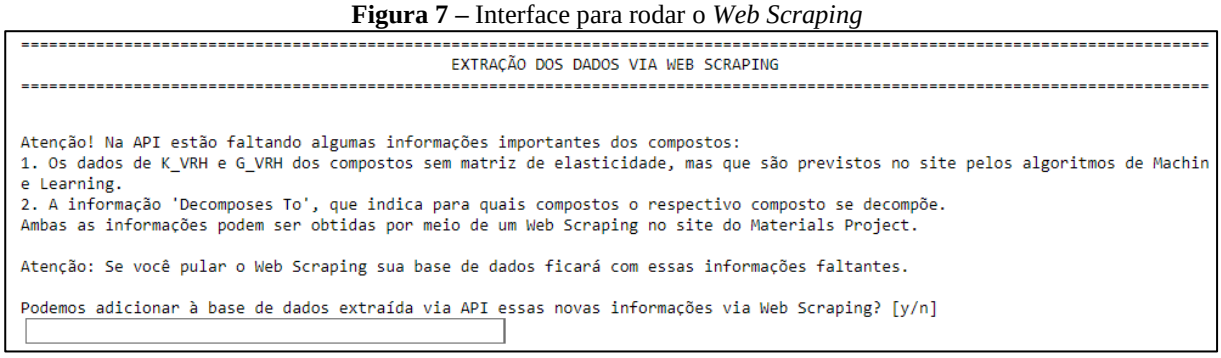
Fonte: Autoria própria

A extração via API permitiu a geração de uma base de dados local bem ampla e de forma eficiente e rápida, ficando faltantes apenas os valores de K_{VRH} e G_{VRH} não calculados no sistema do *Materials Project*, além da informação de se tal composto é estável ou se decompõe para outro(s) composto(s) mais estáveis.

4.3 *Web Scraping* e planilha final

Em seguida o programa pergunta ao usuário se tem interesse em fazer o *Web Scraping* para coletar essas informações faltantes, ou se ele deseja pular tal etapa. Isso pois o *Web Scraping* dessas informações está na ordem de 600 dados e levou cerca de 15 minutos para coletar todos os dados e preencher os *DataFrames* com a informação faltante. Isso mostrou o quão mais efetivo é extrair dados diretamente de uma API do que coletando-os via *Web Scraping*. No entanto, como os dados previstos de K_{VRH} e G_{VRH} por *Machine Learning* não estavam disponíveis na API e são pertinentes para a análise do conjunto mais completo de informações, este foi o caminho encontrado para extrair tais informações da plataforma e adicioná-las nos itens faltantes do *DataFrame*. Na Figura 7 podemos ver a interface para a extração via Web Scraping. Espera-se que o programa leve 12 minutos em média para realizar a extração destas informações complementares, com condições de computador e internet semelhantes.

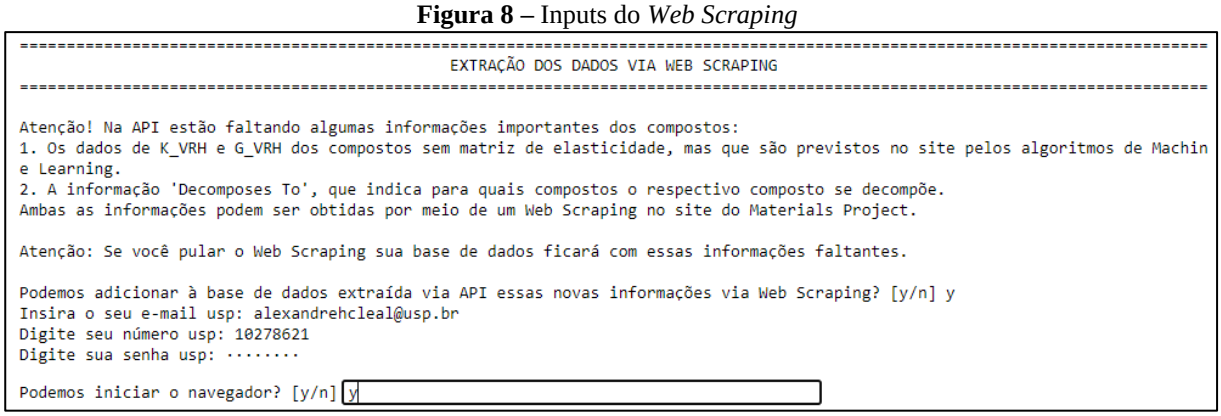
Figura 7 – Interface para rodar o *Web Scraping*



Fonte: Autoria própria

Após decidir fazer o *Web Scraping*, o usuário terá de colocar suas credenciais USP como na Figura 8.

Figura 8 – Inputs do *Web Scraping*

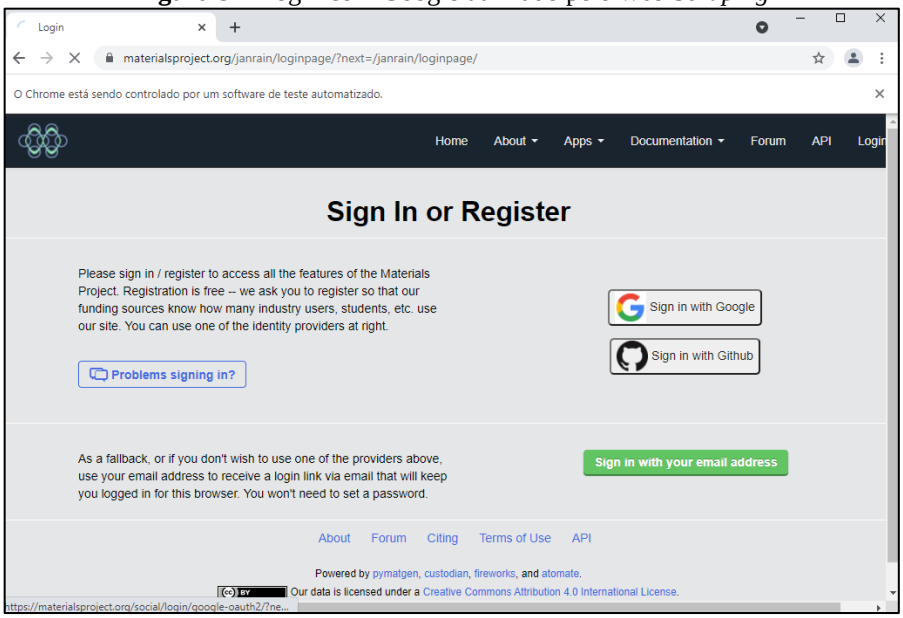


Fonte: Autoria própria

Esses inputs são necessários pois para conseguir obter as informações previstas por ML dos parâmetros K_{VRH} e G_{VRH}, isso pois, para conseguir visualizar e em seguida extrair tais dados

é necessário que o navegador esteja logado no site. Caso o navegador não esteja logado tais informações não irão aparecer! Buscando contornar o problema o *software* é capaz de, com esses dados, realizar o login utilizando o login com Google disposto na Figura 9. Após inserir as credenciais e permitir que o navegador inicie, o programa irá abrir uma aba anônima no Google Chrome controlada automaticamente na URL do login com Google, igual a que está exposta na Figura 9.

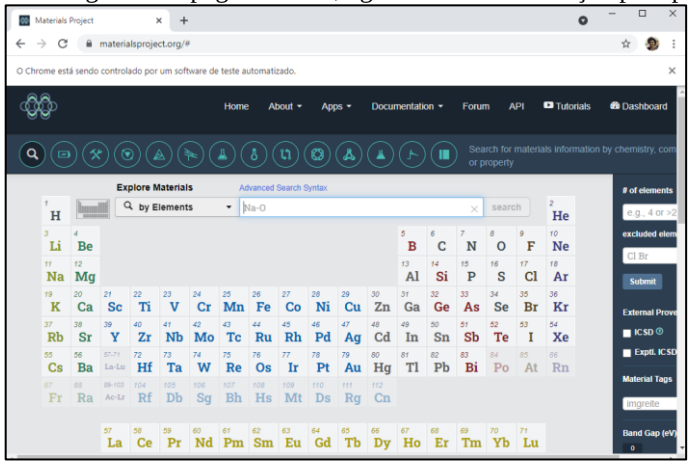
Figura 9 – Login com Google utilizado pelo *Web Scraping*



Fonte: Autoria própria

Em seguida irá navegar até a URL da Figura 9 e iniciar o processo automático de login, inserindo e-mail, clicando no botão de entrar, inserindo as credenciais USP e em seguida confirmando no botão. Após essas etapas o programa estará finalmente na página principal do *Materials Project* só que aguardando uma confirmação para prosseguir, como está disposta na Figura 10.

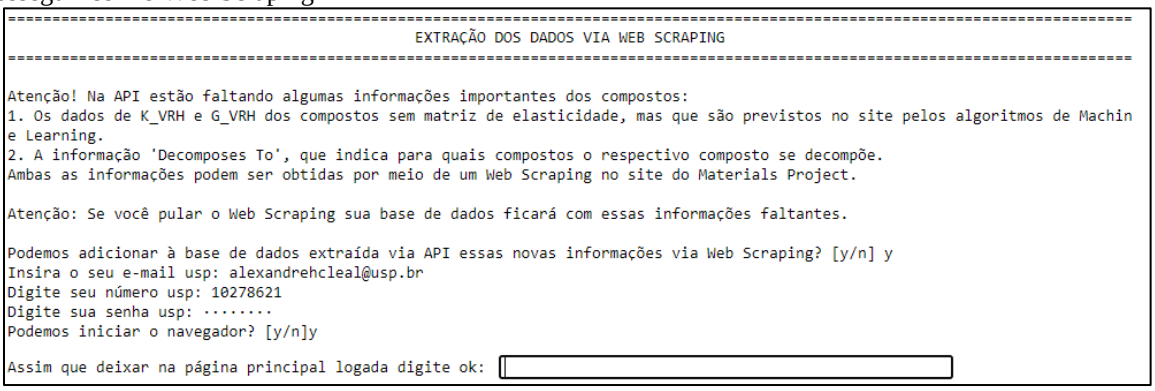
Figura 10 – Programa na página inicial, aguardando confirmação para prosseguir



Fonte: Autoria própria

Note que na Figura 10 uma imagem da conta apareceu no lugar do ícone cinza, isso indica o login que foi realizado pelo programa. No entanto, durante 2 semanas o botão de login do site não estava funcionando e levava para uma URL legado que não estava funcionando. Para contornar este problema, é pedido para o usuário que se certifique de que está na página principal e logado, como no exemplo da Figura 10, para então inserir ok e prosseguir, como pode ser visualizado na Figura 11.

Figura 11 – Programa aguardando a validação de que está na página principal do *Materials Project* e pode prosseguir com o Web Scraping

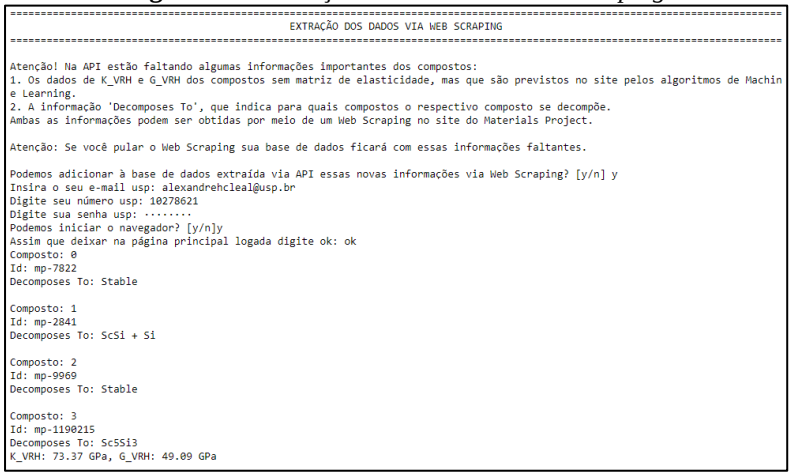


Fonte: Autoria própria

O “ok” pedido pelo programa na Figura 11 permite que o usuário utilize outros meios para realizar o Login além das credenciais USP, podendo realizar o login com um e-mail qualquer, copiando o link enviado no e-mail para o navegador automático, tornando o programa mais flexível para outras pessoas realizarem logins de forma diferentes.

Após a permissão do usuário ser concedida, o programa continuará a comandar o navegador de forma automática extraindo os dados de decomposição e previsões por *Machine Learning* dos parâmetros K_{VRH} e G_{VRH} para todos os compostos binários extraídos via API presentes no *DataFrame*. O começo dessa extração pode ser visto na Figura 12.

Figura 12 – Extração dos dados via *Web Scraping*



Fonte: Autoria própria

O final da extração está disposto na Figura 13, com 633,71 segundos de execução. Sendo o maior tempo registrado pela máquina para executar *Web Scraping* foi 1411,92 tendo uma média considerável de 12 minutos. Os dados faltantes foram adicionados ao DataFrame, resultando nas colunas exibidas na Figura 14, assim como a indicação do método ML para os casos em que o K_{VRH} e G_{VRH} foram previstos por *Machine Learning*.

Figura 13 – Finalização da execução do *Web Scraping* e geração da base de dados local

```
Composto: 249
Id: mp-10871
Decomposes To: AlAu

Composto: 250
Id: mp-30550
Decomposes To: AlAu2
K_VRH: 123.97 GPa, G_VRH: 31.26 GPa

Podemos fechar o navegador? [y/n]y

DataFrame editado com sucesso!

Tempo de execução --- 633.71 seconds ---
DataFame após a inclusão dos dados coletados pelo Web Scraping:
```

Fonte: Autoria própria

A coluna *Method* indica se o K_{VRH} e G_{VRH} foram previstos por ML ou calculados por DFT, note que para as linhas que possuem apenas K_{VRH} e G_{VRH} o método foi ML e as informações foram coletadas via *Web Scraping*.

Figura 14 – DataFrame final com os valores de K_{VRH} e G_{VRH} obtidos por ML via *Web Scraping*

G_VRH	G_Voigt	K_Reuss	K_VRH	K_Voigt	elastic_anisotropy	universal_anisotropy	poisson_ratio	Method
72.00	72.0	88.0	89.00	90.0	0.08	0.08	0.18	DFT
-83.00	18.0	100.0	103.00	107.0	-5.41	-5.41	1.05	DFT
70.00	74.0	101.0	101.00	101.0	0.64	0.64	0.22	DFT
49.09	NaN	NaN	73.37	NaN	NaN	NaN	NaN	ML
59.20	NaN	NaN	89.73	NaN	NaN	NaN	NaN	ML
...	...	...	...	...	...	...	...	...
27.24	NaN	NaN	118.00	NaN	NaN	NaN	NaN	ML
26.84	NaN	NaN	114.50	NaN	NaN	NaN	NaN	ML
26.78	NaN	NaN	114.40	NaN	NaN	NaN	NaN	ML
-3.00	2.0	38.0	39.00	40.0	-6.31	-6.31	0.53	DFT
12.00	15.0	13.0	15.00	17.0	3.46	3.46	0.18	DFT

Fonte: Autoria própria

Após isso, as colunas com o Módulo de Young e Coeficiente de Poisson são calculadas e adicionadas ao *DataFrame*, a Figura 15 mostra um recorte do *DataFrame* com essas novas colunas e seus respectivos valores. Por fim o *DataFrame* final é transformado em uma aba da

planilha. A planilha com todos os *DataFrames* relevantes é gerada ao final, formando a base de dados local.

Figura 15 – Coluna Decomposes_To obtida via *Web Scraping* e colunas E_Young e Coef_Poisson calculadas com os valores extraídos de K_{VRH} e G_{VRH} obtidos por DFT ou ML.

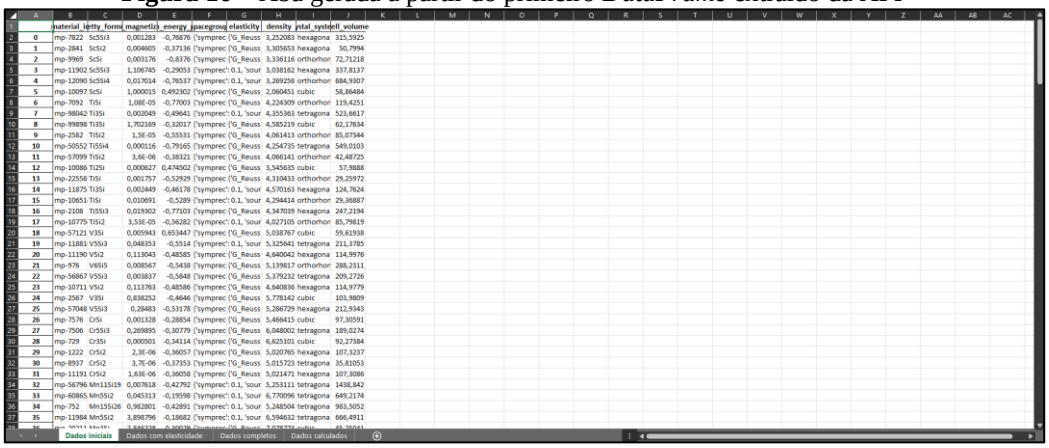
C65	C66	Decomposes_To	E_Young	Coef_Poisson
0.0	77.346971	Stable	170.123894	0.181416
0.0	-18.798929	ScSi + Si	-340.446903	1.050885
0.0	59.169670	Stable	170.589812	0.218499
NaN	NaN	Sc5Si3	120.414561	0.226467
NaN	NaN	Sc5Si3 + ScSi	145.583434	0.229590
...	...	...	...	...
NaN	NaN	Au + Si	75.881020	0.392823
NaN	NaN	Au + Si	74.684398	0.391289
NaN	NaN	Au + Si	74.524807	0.391427
0.0	10.268203	Hg + Si	-9.236842	0.539474
0.0	7.401528	Hg + Si	28.421053	0.184211

Fonte: Autoria própria

4.3.1 Geração da base de dados local

A base de dados local foi gerada ao longo de toda a execução do programa, nela as versões relevantes dos *DataFrames*, os das Figuras Figura 3, Figura 4, Figura 14 e Figura 15, foram salvos no formato de abas ao longo de um único arquivo Excel. Por fim, um único arquivo Excel com essas abas foi gerado. As Figuras 16, 17, 18 e 19 mostram as abas do arquivo gerado como base de dados.

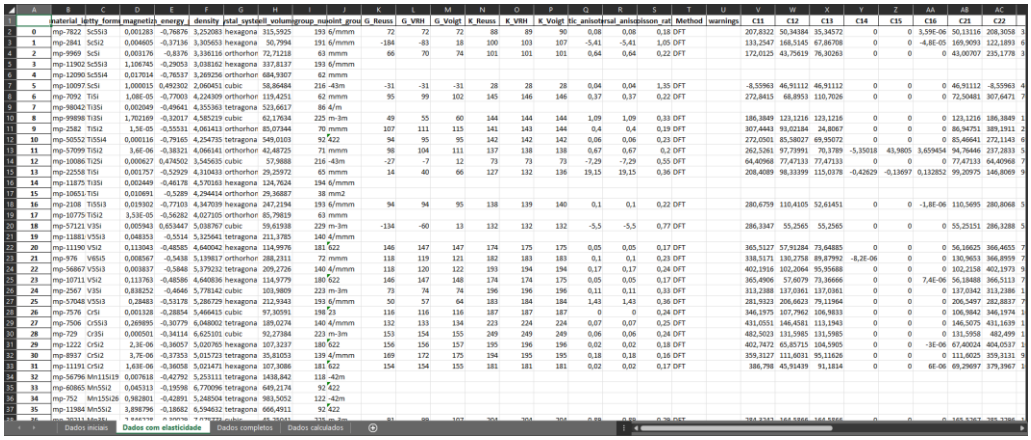
Figura 16 – Aba gerada a partir do primeiro *DataFrame* extraído da API



Fonte: Autoria própria

Ao longo do processo algumas transformações e novas extrações foram necessárias. A primeira extração via API permitiu obter muitos dados condensados na forma de dicionários denrto das colunas *spacefroup* e *elasticity* que precisaram ser alocados em novas colunas para facilitar a visualização originando a tabela da Figura 17.

Figura 17 – Aba referente ao *DataFrame* transformado, com os valores das colunas *spacegroup* e *elasticity* explodidos em mais colunas

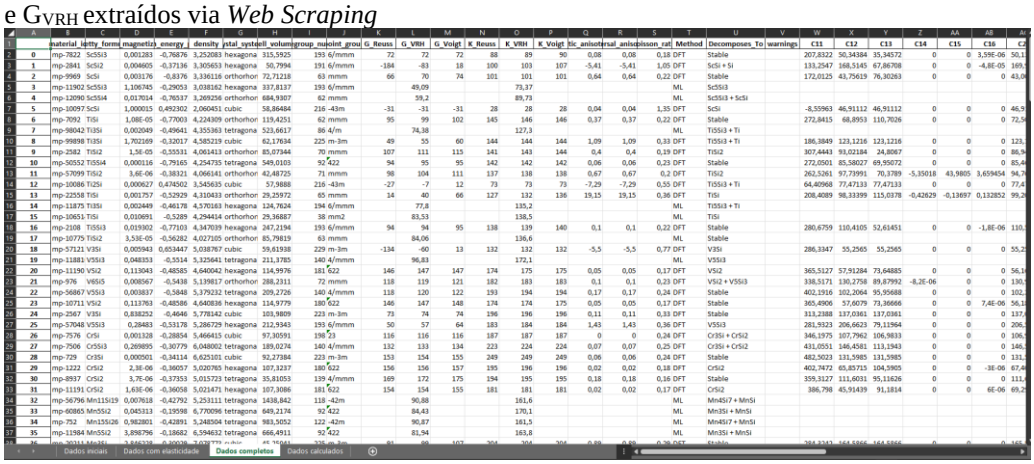


	material_id	ty_formula	magnetic	energy_pa	density_pa	stal_system	ell_volume	group_num	oint_group	G_Reuss	G_VRH	G_Voigt	K_Reuss	K_VRH	K_Voigt	tic_anisotr	ersal_anis	sson_ratio	Method	warnings	C11	C12	C13	C14	C15	C16	C25	C22
0	mp-7822	ScSiI3	0,001283	-0,76876	3,252083	hexagona	315,5925	193	6/mmm	72	72	72	88	89	90	0,08	0,08	0,18	DFT		207,8322	50,34384	35,34572	0	0	3,59E-06	50,13116	208,3058
1	mp-2841	ScSi2	0,004605	-0,37136	3,305653	hexagona	50,7994	191	6/mmm	-184	-83	18	100	103	107	-5,41	-5,41	1,05	DFT		133,2547	168,5145	67,86708	0	0	-4,8E-05	169,9093	122,1893
2	mp-9969	ScSi	0,003176	-0,8376	3,336116	orthorhor	72,71218	63	mmm	66	70	74	101	101	101	0,64	0,64	0,22	DFT		172,0125	43,75619	76,30263	0	0	0	43,00707	235,1778
3	mp-11902	Sc5Si3	1,106745	-0,29053	3,038162	hexagona	337,8137	193	6/mmm																			
4	mp-12090	Sc5Si4	0,017014	-0,76537	3,269256	orthorhor	684,9307	62	mmm																			
5	mp-10097	ScSi	1,000015	0,492302	2,060451	cubic	58,86484	216	-43m	-31	-31	-31	28	28	28	0,04	0,04	1,35	DFT		-8,55963	46,91112	46,91112	0	0	0	46,91112	-8,55963
6	mp-7092	TiSi	1,08E-05	-0,77003	4,224309	orthorhor	119,4251	62	mmm	95	99	102	145	146	146	0,37	0,37	0,22	DFT		272,8415	68,8953	110,7026	0	0	0	72,50481	307,6471
7	mp-98042	Ti3Si	0,002049	-0,49641	4,355363	tetragona	523,6617	86	4/m																			
8	mp-99898	Ti3Si	1,702169	-0,32017	4,585219	cubic	62,17634	225	m-3m	49	55	60	144	144	144	1,09	1,09	0,33	DFT		186,3849	123,1216	123,1216	0	0	0	123,1216	186,3849
9	mp-2582	TiSi2	1,5E-05	-0,55531	4,061413	orthorhor	85,07344	70	mmm	107	111	115	141	143	144	0,4	0,4	0,19	DFT		307,4443	93,02184	24,8067	0	0	0	86,94751	389,1911
10	mp-50552	TiSSi4	0,000116	-0,79165	4,254735	tetragona	549,0103	92	422	94	95	95	142	142	142	0,06	0,06	0,23	DFT		272,0501	85,58027	69,95072	0	0	0	85,46641	272,1143
11	mp-57099	TiSi2	3,6E-06	-0,38321	4,066141	orthorhor	42,48725	71	mmm	98	104	111	137	138	138	0,67	0,67	0,2	DFT		262,5261	97,73991	70,3789	-5,35018	43,9805	3,659454	94,76446	237,2833
12	mp-10086	Ti2Si	0,000627	0,474502	3,545635	cubic	57,9888	216	-43m	-27	-7	12	73	73	73	-7,29	-7,29	0,55	DFT		64,40968	77,47133	77,47133	0	0	0	77,47133	64,40968
13	mp-22558	TiSi	0,001757	-0,52929	4,310433	orthorhor	29,25972	65	mmm	14	40	66	127	132	136	19,15	19,15	0,36	DFT		208,4089	98,33399	115,0378	-0,42629	-0,13697	0,132852	99,20975	146,8069
14	mp-11875	Ti3Si	0,002449	-0,46178	4,570163	hexagona	124,7624	194	6/mmm																			
15	mp-10651	TiSi	0,010691	-0,5289	4,294414	orthorhor	29,36887	38	mm2																			
16	mp-2108	Ti5Si3	0,019302	-0,77103	4,347039	hexagona	247,2194	193	6/mmm	94	94	95	138	139	140	0,1	0,1	0,22	DFT		280,6759	110,4105	52,61451	0	0	-1,8E-06	110,5695	280,8068
17	mp-10775	Ti5Si2	3,53E-05	-0,56282	4,027105	orthorhor	85,79819	63	mmm																			
18	mp-57121	V3Si	0,005943	0,653447	5,038767	cubic	59,61938	229	m-3m	-134	-60	13	132	132	132	-5,5	-5,5	0,77	DFT		286,3347	55,2565	55,2565	0	0	0	55,25151	286,3288
19	mp-11881	V5Si3	0,048353	-0,5514	5,325641	tetragona	211,3785	140	4/mmm																			
20	mp-11190	VSi2	0,113043	-0,48585	4,640042	hexagona	114,9976	181	622	146	147	147	174	175	175	0,05	0,05	0,17	DFT		365,5127	57,91284	73,64885	0	0	0	56,16625	366,4655
21	mp-976	V6Si5	0,008567	-0,5438	5,139817	orthorhor	288,2311	72	mmm	118	119	121	182	183	183	0,1	0,1	0,23	DFT		338,5171	130,2758	89,87992	-8,2E-06	0	0	130,9653	366,8959
22	mp-56867	V5Si3	0,003837	-0,5848	5,379232	tetragona	209,2726	140	4/mmm	118	120	122	193	194	194	0,17	0,17	0,24	DFT		402,1916	102,2064	95,95688	0	0	0	102,2158	402,1973
23	mp-10711	VSi2	0,113763	-0,48588	4,640836	hexagona	114,9779	180	622	146	147	148	174	174	175	0,05	0,05	0,17	DFT		365,4906	57,6079	73,36666	0	0	7,4E-06	56,18488	366,5113
24	mp-2567	V3Si	0,838252	-0,4646	5,778142	cubic	103,9809	223	m-3m	73	74	74	196	196	196	0,11	0,11	0,33	DFT		313,2388	137,0361	137,0361	0	0	0	137,0342	313,2386
25	mp-57048	V5Si3	0,28483	-0,53178	5,286729	hexagona	212,9343	193	6/mmm	50	57	64	183	184	184	1,43	1,43	0,36	DFT		281,9323	206,6623	79,11964	0	0	0	206,5497	282,8837
26	mp-7576	CrSi	0,001328	-0,28854	5,466415	cubic	97,30591	198	23	116	116	116	187	187	187	0	0	0,24	DFT		346,1975	107,7962	106,9833	0	0	0	106,9842	346,1974
27	mp-7506	CrSSi3	0,269895	-0,30779	6,048002	tetragona	189,0274	140	4/mmm	132	133	134	223	224	224	0,07	0,07	0,25	DFT		431,0551	146,4581	113,1943	0	0	0	146,5075	431,1639
28	mp-729	CrSi	0,000501	-0,34114	6,625101	cubic	92,27384	223	m-3m	153	154	155	249	249	249	0,06	0,06	0,24	DFT		482,5023	131,5985	131,5985	0	0	0	131,5958	482,499
29	mp-1222	CrSi2	2,3E-06	-0,36057	5,020765	hexagona	107,3237	180	622	156	156	157	195	196	196	0,02	0,02	0,18	DFT		402,7472	65,85715	104,5905	0	0	-3E-06	67,40024	404,0537
30	mp-8937	CrSi2	3,7E-06	-0,37353	5,015723	tetragona	35,81053	139	4/mmm	169	172	175	194	195	195	0,18	0,18	0,16	DFT		359,3127	111,6031	95,11626	0	0	0	111,6025	359,3131
31	mp-11191	CrSi2	1,63E-06	-0,36058	5,021471	hexagona	107,3086	181	622	154	154	155	181	181	181	0,02	0,02	0,17	DFT		386,798	45,91439	91,1814	0	0	6E-06	69,29697	379,3967
32	mp-56796	Mn11Si19	0,007618	-0,42792	5,253111	tetragona	1438,842	118	-42m																			
33	mp-60865	Mn5Si2	0,045313	-0,13598	6,770096	tetragona	649,2174	92	422																			
34	mp-752	Mn15Si26	0,982801	-0,42891	5,248504	tetragona	983,5052	122	-42m																			
35	mp-11984	Mn5Si2	3,898796	-0,18682	6,594632	tetragona	666,4911	92	422																			

Fonte: Autoria própria

No entanto, mesmo após a extração via API os dados de K_{VRH} e G_{VRH} estavam faltantes para alguns compostos, além dos dados de decomposição. Para isso o *Web Scraping* dessas informações foi realizado originando a aba da Figura 18.

Figura 18 – Aba referente ao *DataFrame* completo com a decomposição do composto e seus valores previstos de K_{VRH} e G_{VRH} extraídos via *Web Scraping*

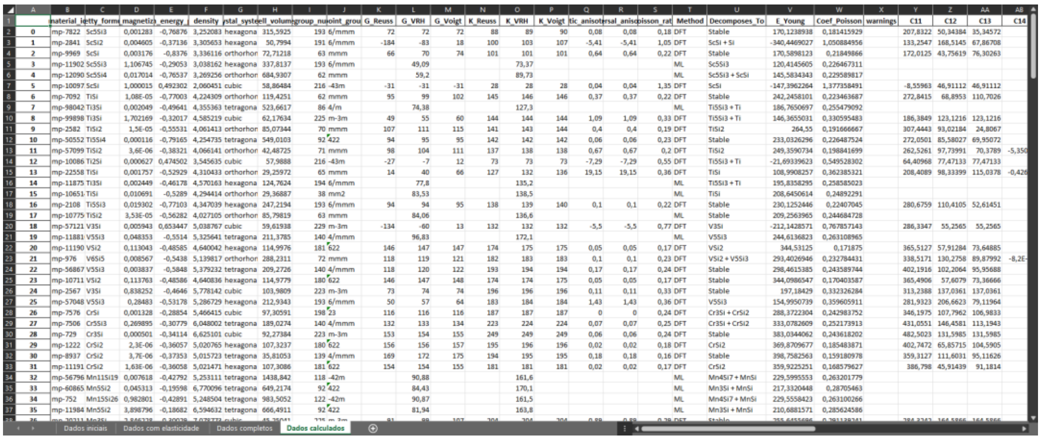


	material_id	ty_formula	magnetic	energy_pa	density_pa	stal_system	ell_volume	group_num	oint_group	G_Reuss	G_VRH	G_Voigt	K_Reuss	K_VRH	K_Voigt	tic_anisotr	ersal_anis	sson_ratio	Method	Decomposes_To	warnings	C11	C12	C13	C14	C15	C16	C2
0	mp-7822	ScSiI3	0,001283	-0,76876	3,252083	hexagona	315,5925	193	6/mmm	72	72	72	88	89	90	0,08	0,08	0,18	DFT	Stable		207,8322	50,34384	35,34572	0	0	3,59E-06	50,1
1	mp-2841	ScSi2	0,004605	-0,37136	3,305653	hexagona	50,7994	191	6/mmm	-184	-83	18	100	103	107	-5,41	-5,41	1,05	DFT	ScSi + Si		133,2547	168,5145	67,86708	0	0	-4,8E-05	169,1
2	mp-9969	ScSi	0,003176	-0,8376	3,336116	orthorhor	72,71218	63	mmm	66	70	74	101	101	101	0,64	0,64	0,22	DFT	Stable		172,0125	43,75619	76,30263	0	0	0	43,0
3	mp-11902	Sc5Si3	1,106745	-0,29053	3,038162	hexagona	337,8137	193	6/mmm				49,09		73,37				ML	ScSiI3								
4	mp-12090	Sc5Si4	0,017014	-0,76537	3,269256	orthorhor	684,9307	62	mmm				59,2		89,73				ML	Sc5Si3 + ScSi								
5	mp-10097	ScSi	1,000015	0,492302	2,060451	cubic	58,86484	216	-43m	-31	-31	-31	28	28	28	0,04	0,04	1,35	DFT	ScSi		-8,55963	46,91112	46,91112	0	0	0	46,9
6	mp-7092	TiSi	1,08E-05	-0,77003	4,224309	orthorhor	119,4251	62	mmm	95	99	102	145	146	146	0,37	0,37	0,22	DFT	Stable		272,8415	68,8953	110,7026	0	0	0	72,5
7	mp-98042	Ti3Si	0,002049	-0,49641	4,355363	tetragona	523,6617	86	4/m				74,38		127,3				ML	Ti5Si3 + Ti								
8	mp-99898	Ti3Si	1,702169	-0,32017	4,585219	cubic	62,17634	225	m-3m	49	55	60	144	144	144	1,09	1,09	0,33	DFT	Ti5Si3 + Ti		186,3849	123,1216	123,1216	0	0	0	123,
9	mp-2582	TiSi2	1,5E-05	-0,55531	4,061413	orthorhor	85,07344	70	mmm	107	111	115	141	143	144	0,4	0,4	0,19	DFT	TiSi2		307,4443	93,02184	24,8067	0	0	0	86,9
10	mp-50552	Ti5Si4	0,000116	-0,79165	4,254735	tetragona	549,0103	92	422	94	95	95	142	142	142	0,06	0,06	0,23	DFT	Stable		272,0501	85,58027	69,95072	0	0	0	85,4
11	mp-57099	TiSi2	3,6E-06	-0,38321	4,066141	orthorhor	42,48725	71	mmm	98	104	111	137	138	138	0,67	0,67	0,2	DFT	TiSi2		262,5261	97,73991	70,3789	-5,35018	43,9805	3,659454	94,7
12	mp-10086	Ti2Si	0,000627	0,474502	3,545635	cubic	57,9888	216	-43m	-27	-7	12	73	73	73	-7,29	-7,29	0,55	DFT	Ti5Si3 + Ti		64,40968	77,47133	77,47133	0	0	0	77,4
13	mp-22558	TiSi	0,001757	-0,52929	4,310433	orthorhor	29,25972	65	mmm	14	40	66	127	132	136	19,15	19,15	0,36	DFT	TiSi		208,4089	98,33399	115,0378	-0,42629	-0,13697	0,132852	99,2
14	mp-11875	Ti3Si	0,002449	-0,46178	4,570163	hexagona	124,7624	194	6/mmm		77,8				135,2				ML	Ti3Si + Ti								
15	mp-10651	TiSi	0,010691	-0,5289	4,294414	orthorhor	29,36887	38	mm2		83,53				138,5				ML	TiSi								
16	mp-2108	Ti5Si3	0,019302	-0,77103	4,347039	hexagona	247,2194	193	6/mmm	94	94	95	138	139	140	0,1	0,1	0,22	DFT	Stable		280,6759	110,4105	52,61451	0	0	-1,8E-06	110,
17	mp-10775	Ti5Si2	3,53E-05	-0,56282	4,027105	orthorhor	85,79819	63	mmm		84,06				136,6				ML	Stable								
18	mp-57121	V3Si	0,005943	0,653447	5,038767	cubic	59,61938	229	m-3m	-134	-60	13	132	132	132	-5,5	-5,5	0,77	DFT	V3Si		286,3347	55,2565	55,2565	0	0	0	55,2
19	mp-11881	V5Si3	0,048353	-0,5514	5,325641	tetragona	211,3785	140	4/mmm		96,83				172,1				ML	V5Si3								
20	mp-11190	VSi2	0,113043	-0,48585	4,640042	hexagona	114,9976	181	622	146	147	147	174	175	175	0,05	0,05	0,17	DFT	VSi2		365,5127	57,91284	73,64885	0	0	0	56,1
21	mp-976	V6Si5	0,008567	-0,5438	5,139817	orthorhor	288,2311	72	mmm	118	119	121	182	183	183	0,1	0,1	0,23	DFT	V5Si3 + VSSi3		338,5171	130,2758	89,87992	-8,2E-06	0	0	130,
22	mp-56867	V5Si3	0,003837	-0,5848	5,379232	tetragona	209,2726	140	4/mmm	118	120	122	193	194	194	0,17	0,17	0,24	DFT	Stable		402,1916	102,2064	95,95688	0	0	0	102,
23	mp-10711	VSi2	0,113763	-0,48586	4,640836	hexagona	114,9779	180	622	146	147	148	174	174	175	0,05	0,05	0,17	DFT	Stable		365,4906	57,6079	73,36666	0	0	7,4E-06	56,1
24	mp-2567	V3Si	0,838252	-0,4646	5,778142	cubic	103,9809	223	m-3m	73	74	74	196	196	196	0,11	0,11	0,33	DFT	Stable		313,2388	137,0361	137,0361	0	0	0	137,
25	mp-57048	V5Si3	0,28483	-0,53178	5,286729	hexagona	212,9343	193	6/mmm	50	57	64	183	184	184	1,43	1,43	0,36	DFT	V5Si3		281,9323	206,6623	79,11964	0	0	0	206,
26	mp-7576	CrSi	0,001328	-0,28854	5,466415	cubic	97,30591	198	23	116	116	116	187	187	187	0	0	0,24	DFT	Cr3Si + CrSi2		346,1975	107,7962	106,9833	0	0	0	106,
27	mp-7506	CrSSi3	0,269895	-0,30779	6,048002	tetragona	189,0274	140	4/mmm	132	133	134	223	224	224	0,07	0,07	0,25	DFT	Cr3Si + CrSi2		431,0551	146,4581	113,1943	0	0	0	146,
28	mp-729	CrSi	0,000501	-0,34114	6,625101	cubic	92,27384	223	m-3m	153	154	155	249	249	249	0,06	0,06	0,24	DFT	Stable		482,5023	131,5985	131,5985	0	0	0	131,
29	mp-1222	CrSi2	2,3E-06	-0,36057	5,020765	hexagona	107,3237	180	622	156	156	157	195	196	196	0,02	0,02	0,18	DFT	CrSi2		402,7472	65,85715	104,5905	0	0	-3E-06	67,4
30	mp-8937	CrSi2	3,7E-06	-0,37353	5,015723	tetragona	35,81053	139	4/mmm	169	172	175	194	195	195	0,18	0,18	0,16	DFT	Stable		359,3127	111,6031	95,11626	0	0	0	111,
31	mp-11191	CrSi2	1,63E-06	-0,36058	5,021471	hexagona	107,3086	181	622	154	154	155	181	181	181	0,02	0,02	0,17	DFT	CrSi2		386,798	45,91439	91,1814	0	0	6E-06	69,2
32	mp-56796	Mn11Si19	0,007618	-0,42792	5,253111	tetragona	1438,842	118	-42m		90,88				161,6				ML	Mn4Si7 + MnSi								
33	mp-60865	Mn5Si2	0,045313	-0,13598	6,770096	tetragona	649,2174	92	422		84,43				170,1				ML	Mn3Si + MnSi								
34	mp-752	Mn15Si26	0,982801	-0,42891	5,248504	tetragona	983,5052	122	-42m		90,87				161,5				ML	Mn4Si7 + MnSi								
35	mp-11984	Mn5Si2	3,898796	-0,18682	6,594632	tetragona	666,4911	92	422		81,94				163,8				ML	Mn3Si + MnSi								

Fonte: Autoria própria

Por fim, alguns parâmetros como o Módulo de Young e Coeficiente de Poisson precisavam ser calculados para todos os compostos, utilizando seus módulos de cisalhamento e volume, o que resultou na última aba representada pela Figura 19.

Figura 19 – Aba referente ao *DataFrame* final com as colunas de Módulo de Young e Coeficiente de Poisson também calculados



Fonte: Autoria própria

Finalmente após essa última aba ser gerada o arquivo Excel com todas as abas foi salvo finalizando a fase de extração, tratamento e armazenamento dos dados em uma base de dados local. Dessa maneira ao fim da execução do programa de extração de dados, um Excel com todos os dados coletados e tratados foi gerado, permitindo guardar essas informações para futuras análises ou modelagens feitas a partir do próprio computador, sem ter que gastar tempo requisitando e coletando tais informações novamente.

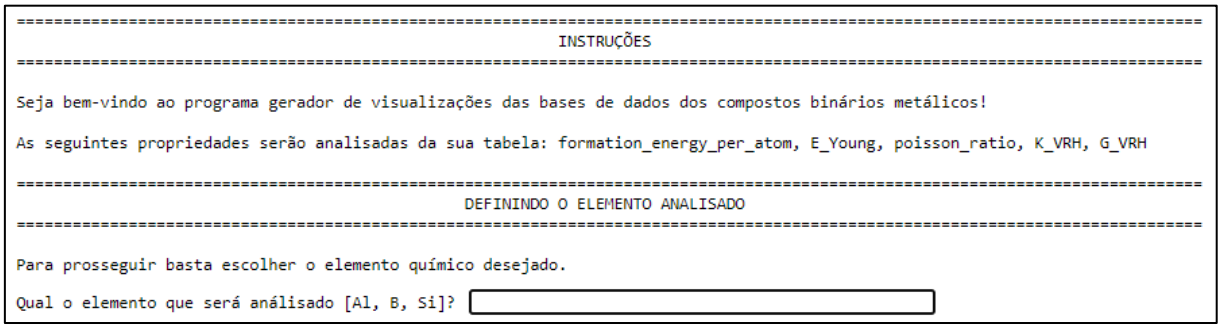
A geração das bases de dados poderia ser expandida para mais compostos binários, o trabalho focou em combinar alumínio, boro e silício com os trinta metais de transição listados, no entanto, dado um elemento químico qualquer pelo usuário e uma lista de combinação qualquer, toda combinação desejada de composto binário poderia ser extraída. Portanto, com leves ajustes no código o usuário pode alterar a lista de elementos químicos permitidos e inserir outros elementos além do Al, B e Si para combinar com a lista de metais de transição. Além disso, a lista de combinações também pode ser alterada para conter quaisquer materiais que se desejar combinar. Assim, o código com pequenos ajustes permite obter a base de dados local das propriedades elásticas retiradas do sistema de qualquer combinação binária desejada.

4.3.2 Geração de gráficos

A geração dos gráficos se deu por meio do segundo programa, feito exclusivamente para analisar as bases de dados geradas. Como dito anteriormente, essa decisão permitiu que o programa pudesse gerar visualizações de múltiplas bases de dados locais sem precisar extraí-las no mesmo instante. Permitindo que as visualizações fossem feitas após a extração de vários compostos.

Para iniciar a geração das visualizações o usuário deve inserir na interface inicial do programa o elemento que ele irá analisar, escolhendo um dentre os elementos químicos disponíveis, como disposto na Figura 20.

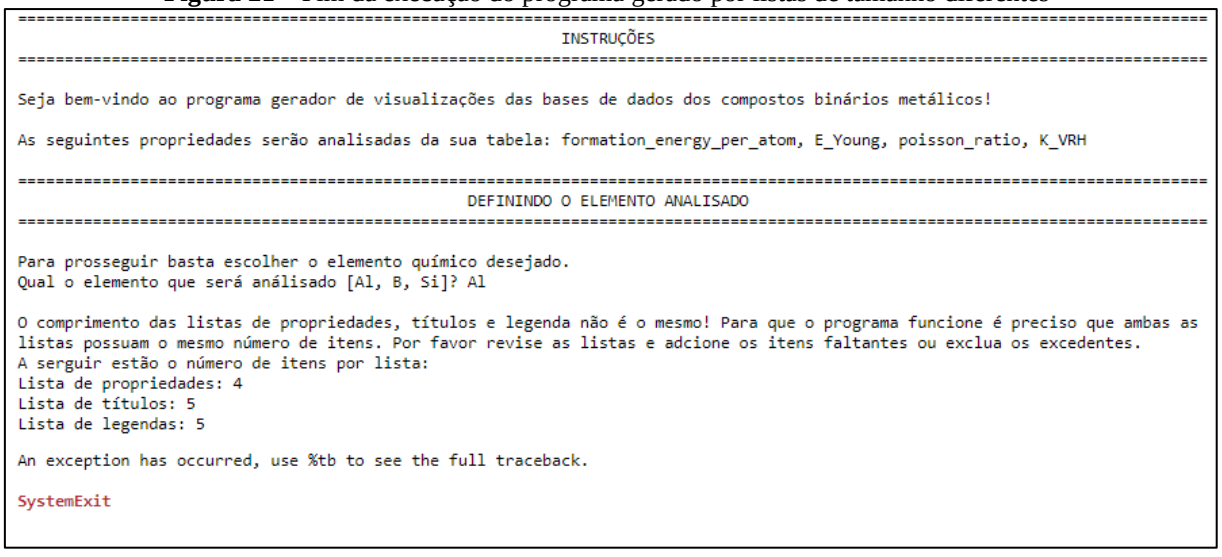
Figura 20 – Interface inicial do programa gerador de visualizações aguardando o elemento químico desejado para análise



Fonte: Autoria própria

No entanto, a geração dos gráficos se baseia em três listas: propriedades, títulos e legendas, para criar os gráficos personalizados para cada propriedade. Se por algum motivo, uma dessas três listas possuir um comprimento diferente dos demais uma mensagem de erro aparece informando que existe algum dado excedente ou faltante como na Figura 21.

Figura 21 – Fim da execução do programa gerado por listas de tamanho diferentes



Fonte: Autoria própria

Após inserir o elemento químico desejado o programa filtra a base de dados e gera 5 novos *DataFrames* para cada propriedade da lista, um para cada uma das proporções definidas 25%, 33%, 50%, 66% e 75%. No exemplo da Figura 20, 25 *DataFrames* foram gerados durante a execução. Cada *DataFrame* possui 2 colunas, uma para a fórmula química do composto analisado e a outra para o valor da propriedade analisada. A

Figura 22 mostra o primeiro *DataFrame* "impresso" na tela, para as energias de formação dos compostos binários metálicos com 25% de Al. Assim como este os demais 24

DataFrames também foram gerados nesta etapa (APÊNDICE H) e serviram como base para a geração de seus respectivos gráficos.

Figura 22 – Primeiro *DataFrame* "impresso" na tela com as informações de energia de formação para os compostos binários de 25% de alumínio.

```
================================================================================
                        DEFININDO O ELEMENTO ANALISADO
================================================================================

Qual o elemento que será análisado [Al, B, Si]? Al

Foram encontrados 21 compostos.

Elementos faltantes no gráfico:25%Al
4º elemento da 1ª linha
10º elemento da 1ª linha
6º elemento da 2ª linha
3º elemento da 3ª linha
4º elemento da 3ª linha
5º elemento da 3ª linha
6º elemento da 3ª linha
7º elemento da 3ª linha
10º elemento da 3ª linha
Total: 9
```

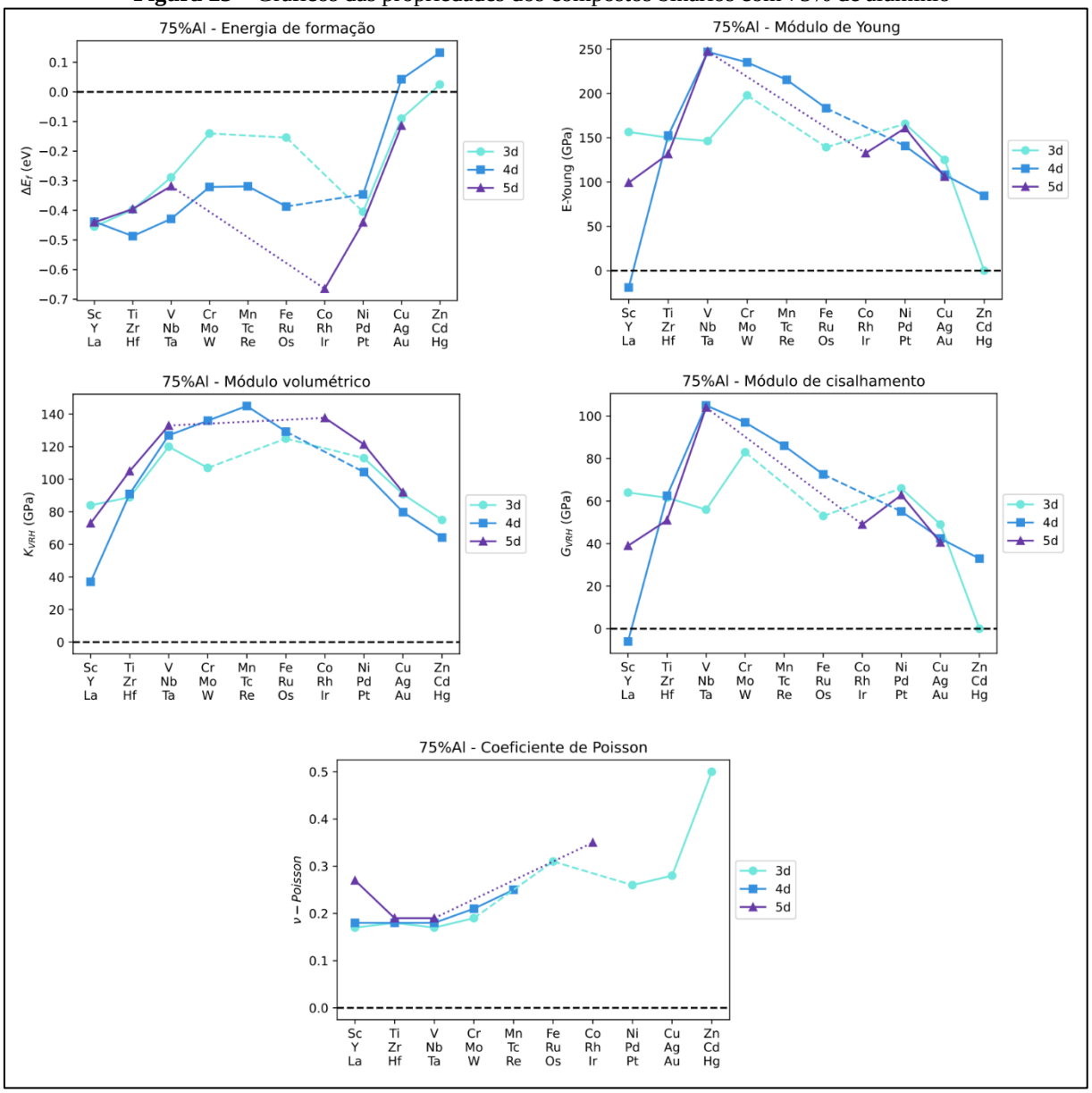
	pretty_formula	formation_energy_per_atom
235	AlPt3	-0.692685
169	AlPd3	-0.612381
155	AlRh3	-0.491680
76	AlNi3	-0.433437
115	Zr3Al	-0.296003
22	Ti3Al	-0.279486
8	Sc3Al	-0.268877
129	AlMo3	-0.241469
197	Hf3Al	-0.221002
183	La3Al	-0.201390
52	AlFe3	-0.199436
244	AlAu3	-0.197910
84	AlCu3	-0.189439
101	Y3Al	-0.185329
127	Nb3Al	-0.173775
60	AlCo3	-0.148929
25	AlV3	-0.135406
144	AlTc3	-0.112514
170	AlAg3	-0.073814
44	Mn3Al	-0.037628
176	AlCd3	0.102806

Fonte: Autoria própria

O mesmo processo pode ser feito para o Boro (B) e para o Silício (Si) já que suas bases também podem ser geradas com o programa de extração via API e *Web Scraping*. Assim todos os *DataFrames* e 75 gráficos puderam ser gerados, o (APÊNDICE H) consta todos os DataFrames pertencentes ao alumínio, mas para o boro e silício o formato da informação seria o mesmo.

Com os 75 *DataFrames* gerados, seus respectivos 75 gráficos podem ser traçados. As Figuras 24, 25 e 26 contém os gráficos das propriedades das proporções com o maior número de compostos para o alumínio, boro e silício respectivamente. Por fim, o programa salva cada um dos gráficos individualmente como um arquivo PDF, permitindo o armazenamento de tais visualizações no computador. Os gráficos das demais composições dos compostos binários com alumínio, boro e silício podem ser encontrados no (APÊNDICE I).

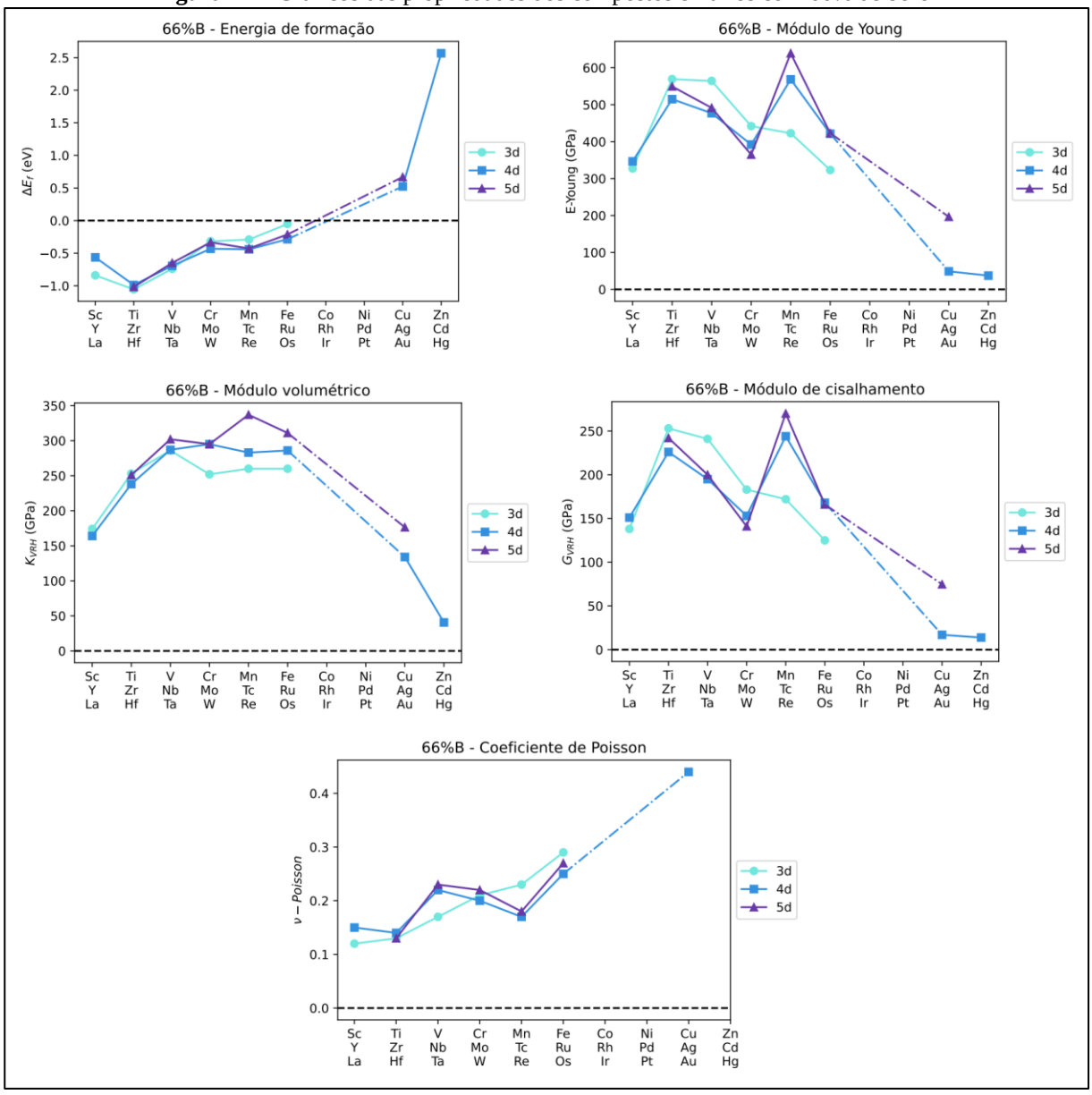
Figura 23 – Gráficos das propriedades dos compostos binários com 75% de alumínio



Fonte: Autoria própria

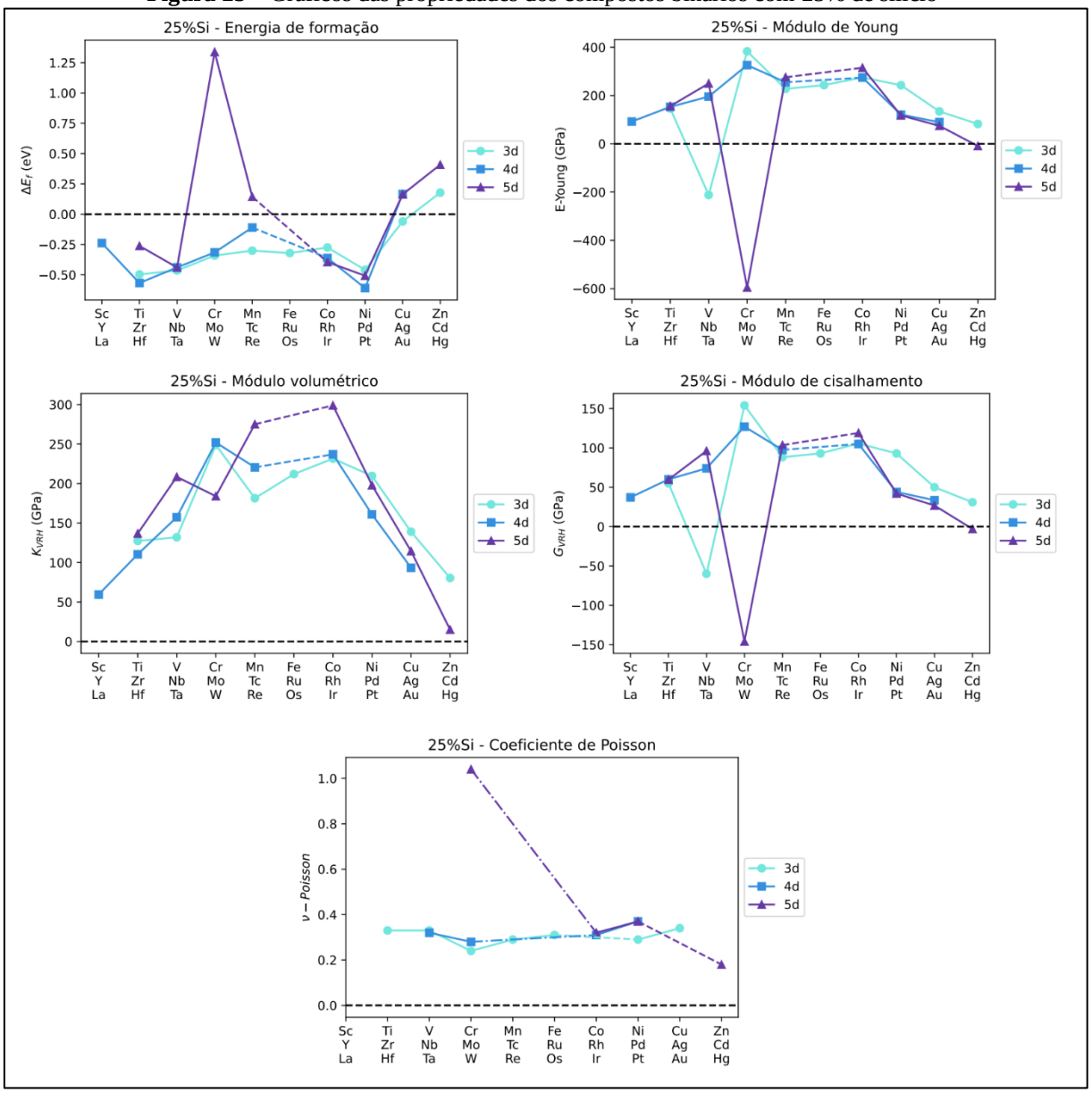
A partir do conjunto de gráficos é possível notar o comportamento igual entre as curvas do Módulo de Cisalhamento e Módulo de Young, o que faz todo sentido já que E é um múltiplo de G, sendo, portanto, diretamente proporcional àquele.

Figura 24 – Gráficos das propriedades dos compostos binários com 66% de boro



Fonte: Autoria própria

Figura 25 – Gráficos das propriedades dos compostos binários com 25% de silício



Fonte: Autoria própria

Os gráficos para as demais composições dos compostos binários com silício podem ser encontrados no (APÊNDICE I).

Da mesma maneira, a visualização para os demais compostos binários também pode ser obtida, para qualquer elemento desejado, bastando apenas leves ajustes no código e possuir a base de dados local destes compostos no com suas composições químicas e valores a serem analisados.

5 CONCLUSÃO

A partir do programa montado foi possível extrair os dados de todos os compostos binários metálicos com alumínio, boro ou silício presentes na plataforma *Materials Project*. Além disso, com esses dados extraídos o programa permite gerar arquivos Excel com todas as informações, permitindo salvar cada etapa de transformação ou adição de informações novas como uma nova aba da planilha, que por sua vez pode ser utilizada como uma base de dados local. Em seguida, um segundo programa possibilita acessar estas bases locais e gerar a visualização dos dados de suas colunas.

Com algumas pequenas adaptações na parte inicial do código do programa de extração, seria possível expandir a extração de dados para outros compostos binários. Dessa forma, indicando os elementos permitidos e os metais, ou outros elementos químicos, na lista de combinação, seria possível extrair e gerar uma base de dados para quaisquer compostos binários desejados. Dessa maneira, o programa serve como base para a extração de dados de outros compostos binários via API dos compostos binários, além de permitir o *Scraping* dos dados previstos por ML de K_{VRH} e G_{VRH} dos compostos, podendo ser ainda mais aprimorado para se tornar um programa de extração ainda mais completo ou que atenda outras necessidades desejadas, como extrair outras características via API ou *Web Scraping*. A implementação de novas extrações no *Web Scraping* é um pouco mais complexa, mas um programador que possua um Python intermediário já dará conta de inserir as demais informações que deseja extrair utilizando como modelo o código molde já programado.

O segundo programa permite a visualização das informações das colunas da base de dados local para as bases dos compostos binários extraídos de alumínio, boro e silício. Ajustando as listas iniciais no código é possível selecionar o parâmetro desejado que será plotado, além de definir seu título e legenda do eixo Y. Ajustando a função para ser igual à lista de combinação do programa de extração é possível gerar visualizações para as bases de dados locais de quaisquer compostos binários. Podendo, portanto, servir como um código base para novas adaptações que permitirão gerar, com leves ajustes, novas visualizações para outros compostos binários.

Quando comparada velocidade da extração via API com relação à via *Web Scraping*, ficou evidente o quão mais prudente e otimizado é extrair dados diretamente de uma API. Enquanto aproximadamente 9200 dados foram extraídos em média em 30 segundos por uma API, apenas, mas apenas 500 dados foram extraídos em uma média de 12 minutos. Ou seja, a API teve uma

velocidade de extração de dados 441,6 vezes mais veloz para o mesmo computador e internet especificados nos materiais do trabalho. Um motivo para essa diferença está no tempo que o navegador automatizado leva para extrair cada informação, precisando extrair uma a uma, carregando página por página. Além disso o login também precisa ser feito no site o que pode tomais mais alguns segundos da execução total. Com relação a isso é possível concluir que a extração das informações deve ser feita sempre via API e apenas quando necessário a extração via *Web Scraping* deve ser realizada.

Por fim, a linguagem Python se mostrou muito versátil e útil para trabalhar com dados, tanto para a parte de extração, transformação quanto visualização. Provando-se bastante poderosa ao sempre ter uma biblioteca que suprisse a necessidade durante o desenvolvimento. Com a biblioteca Pymatgen foi extremamente fácil extrair os dados de forma rápida e prática; graças a Pandas e NumPy a transformação dos dicionários em *DataFrames* bem definidos e organizados foi possível; o Selenium possibilitou o *Web Scraping* de forma completa de todos os dados desejados da maneira mais ágil possível; a Matplotlib permitiu a criação de gráficos totalmente estilizados que suprissem exatamente as necessidades de visualização do trabalho e, não menos importante, a flexibilidade da sintaxe e facilidade para o encadeamento de componentes lógicos possibilitou a programação de dois códigos relativamente pequenos capazes de gerar bases de dados e visualizações personalizadas, mostrando mais uma vez o quão bom o Python pode ser para trabalhar com dados.

O link do código ainda está sendo melhorado e pode ser acompanhado e obtido no repositório do *Github* disponível na introdução.

REFERÊNCIAS

ANACONDA. Miniconda, 2017. Disponivel em: <https://docs.conda.io/en/latest/miniconda.html>. Acesso em: 20 nov. 2021.

BEWLAY, B. P. et al. TiAl alloys in commercial aircraft engines. **Materials at High Temperatures**, 33, 30 jun. 2016. 549-559. Disponivel em: <https://doi.org/10.1080/09603409.2016.1183068>.

CALLISTER, W. D.; RETHWISCH, D. G. **Ciência e engenharia de materiais**. 9ª. ed. Rio de Janeiro: LTC, 2016.

CHACON, V. Aprenda A Utilizar O Selenium Para Web Scraping. **Lets Code**, 2021. Disponivel em: <https://www.letscode.com.br/blog/aprenda-a-utilizar-o-selenium-para-web-scraping>. Acesso em: 15 nov. 2021.

CHROMEDRIVER. ChromeDriver. **ChromeDriver**, 2021. Disponivel em: <https://chromedriver.chromium.org/home>. Acesso em: 20 nov. 2021.

DEY, G. K. Physical metallurgy of nickel aluminides. In: SADHANA **Sādhanā**. [S.l.]: Indian Academy of Sciences, v. 46, 2021. Cap. 28, p. 247–262.

FIA. Data Mining: O que é, Para que serve e Tipos de técnicas. **Fundação Instituição de Administração**, 2020. Disponivel em: <https://fia.com.br/blog/data-mining/>. Acesso em: 25 nov. 2021.

GUNDECHA, U. **Learning Selenium Testing Tools with Python**. Birmingham: Packt Publishing, 2014.

HARRISON, W. A. **Electronic structure and the properties of solids**. Dover. ed. Nova Iorque: Dover Publications, 1989.

HOHENBERG, P. C.; KOHN, W.; SHAM, L. J. The beginnings and some thoughts. **Advances in Quantum Chemistry**, 1990. 7–26.

IDRIS, I. **NumPy Beginner's Guide**. 2ª. ed. Birmingham: Packt Publishing, 2013.

JAIN, A. et al. A high-throughput infrastructure for density functional theory calculations. **Computational Materials Science**, jun. 2011. 2295-2534.

JAIN, A. et al. Formation enthalpies by mixing GGA and GGA+U calculations. **Phys. Rev. B**, jul. 2011. 045115.

JAIN, A. et al. The Materials Project: A materials genome approach to accelerating materials innovation. **APL Materials**, 2013. 011002.

JONG, M. et al. Charting the complete elastic properties of inorganic crystalline compounds. **Scientific Data**, 17 mar. 2015.

KUBE, C. M. Elastic anisotropy of crystals. **AIP Advances**, 13 set. 2016. Disponivel em: <https://doi.org/10.1063/1.4962996>.

LANDIS, G. A. **Refino de Materiais para Produção de Matriz Solar na Lua**. NASA. Cleveland, p. 25. 2006.

LANE, K. Intro to APIs: What Is an API? **Postman**, 2020. Disponivel em: <https://blog.postman.com/intro-to-apis-what-is-an-api/>. Acesso em: 11 17 2021.

LUNDH, F. **Python Standard Library**. [S.l.]: O'Reilly, 2001.

MAVKO, G.; MUKERJI, T.; DVORKIN, J. **The Rock Physics Handbook:** Tools for Seismic Analysis of Porous Media. 2. ed. Nova Iorque: Cambridge University Press, 2009.

MCKINNEY, W. **Pandas:** powerful Python data analysis. [S.l.]: [s.n.], v. Release 1.3.4, 2021.

MITCHELL, R. **Web Scraping with Python**. 2ª. ed. Sebastopol: O'Reilly Media, 2018.

MORGAN, P. **Data Analysis from Scratch with Python**. [S.l.]: AI Sciences LLC, 2018.

MOUHAT, F.; COUDERT, F. X. Necessary and Sufficient Elastic Stability Conditions in Various Crystal Systems. **Physical Review B**, Paris, 05 dez. 2014.

NYE, J. F. **Physical properties of crystals:** their representation by tensors and matrices. [S.l.]: Oxford university press, 1985.

ONG, S. P. et al. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. **Computational Materials Science**, 2013. 314-319. Disponivel em: <https://pymatgen.org>. Acesso em: 02 out. 2021.

ONG, S. P. et al. The Materials Application Programming Interface (API): A. **Computational Materials Science**, 18 out. 2014. 2009-125.

PAVLIC, O. Design of Mg alloys: The effects of Li concentration on the structure. **Journal of Alloys and Compounds**, 2017. 15-25.

POPLAVKO, Y. M. Mechanical properties of solids. In: POPLAVKO, Y. M. **Electronic Materials**. [S.l.]: Elsevier, 2019. Cap. 2, p. 71-93.

PROJECT JUPYTER. Jupyter. **Jupyter**, 2021. Disponivel em: <https://jupyter.org>. Acesso em: 25 out. 2021.

RANGANATHAN, S. I.; OSTOJA-STARZEWSKI, M. Universal Elastic Anisotropy Index. **Physical Review Letters**, Urbana, 1 ago. 2008. 055504-1 a 4. Disponivel em: <https://link.aps.org/doi/10.1103/PhysRevLett.101.055504>.

RAPIDAPI. API Credentials – What are API Credentials? **The RapidAPI Blog**, 2021. Acesso em: 16 nov. 2021.

RAYNER-CANHAM, G. **Descriptive inorganic chemistry**. 6. ed. New York: Freeman and Company, 2014.

RELAN, K. **Building REST APIs with Flask**. [S.l.]: Apress, 2019.

SINGH, S. MechElastic: A Python Library for Analysis of mechanical and elastic properties of bulk and 2D materials. **Computer Physics Communications**, 2021. 108068.

SWEIGART, A. **Automate the Boring Stuff with Python**. 1. ed. São Francisco: No Sartch Press, v. 1, 2015.

WANG, L.; LEE, D.; KAN, C.-D. Work conjugate pair of stress and strain in molecular dynamics. **International Journal of Smart and Nano Materials**, 10 out. 2016.

WASEEM, M. Python Frameworks: What Are The Top 5 Frameworks In Python? **edureka**, 2021. Disponivel em: <https://www.edureka.co/blog/python-frameworks/>. Acesso em: 11 19 2021.

APÊNDICE A - Código de instalação do ambiente virtual e Jupyter Notebook

Figura A-1 Criando o ambiente virtual do programa

```
(base) C:\Users\alexa\Desktop\Faculdade\TCCv2>conda create -n TCC python=3.8
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Fonte: Autoria própria

Figura A-2 Instalando o pacote conda, com o gerenciador de pacotes *pip* o Python e alguns extras

```
  added / updated specs:
    - python=3.8


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2021.10.26 |       haa95532_2         115 KB
    pip-21.2.2                 |   py38haa95532_0         1.9 MB
    ------------------------------------------------------------
                                           Total:         2.0 MB

The following NEW packages will be INSTALLED:

  ca-certificates    pkgs/main/win-64::ca-certificates-2021.10.26-haa95532_2
  certifi            pkgs/main/win-64::certifi-2021.10.8-py38haa95532_0
  openssl            pkgs/main/win-64::openssl-1.1.1l-h2bbff1b_0
  pip                pkgs/main/win-64::pip-21.2.2-py38haa95532_0
  python             pkgs/main/win-64::python-3.8.12-h6244533_0
  setuptools         pkgs/main/win-64::setuptools-58.0.4-py38haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.36.0-h2bbff1b_0
  vc                 pkgs/main/win-64::vc-14.2-h21ff451_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
  wheel              pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_1
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py38haa95532_2


Proceed ([y]/n)? y
```

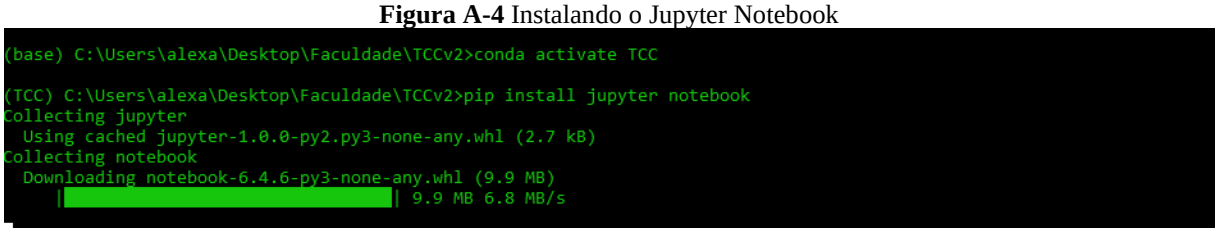
Fonte: Autoria própria

Figura A-3 Ativando o ambiente de trabalho

```
Proceed ([y]/n)? y


Downloading and Extracting Packages
pip-21.2.2          | 1.9 MB    | ############################################################################ | 100%
ca-certificates-2021 | 115 KB    | ############################################################################ | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate TCC
#
# To deactivate an active environment, use
#
#     $ conda deactivate


(base) C:\Users\alexa\Desktop\Faculdade\TCCv2>conda activate TCC

(TCC) C:\Users\alexa\Desktop\Faculdade\TCCv2>
```

Fonte: Autoria própria

Figura A-4 Instalando o Jupyter Notebook



Fonte: Autoria própria

APÊNDICE B - Código ETL dos dados

```
from pymatgen.ext.matproj import MPRester
import pandas as pd
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver import ActionChains
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.chrome.service import Service
from getpass import getpass
from datetime import time
import math
import NumPy as np
import time
import sys

## Definições principais

# Para alterar os elementos à serem combinados com o elemento_analisado, basta alterar esta lista
# Metais que serão combinados com o elemento em questão
metais_de_transicao = ['Sc','Ti','V','Cr','Mn','Fe','Co','Ni','Cu','Zn','Y','Zr','Nb','Mo','Tc','Ru', 'Rh','Pd','Ag','Cd', 'La','Hf','Ta','W','Re','Os', 'Ir', 'Pt','Au', 'Hg']

# Definindo as combinações permitidas
elementos_permitidos = ['Al', 'B', 'Si']

# Chave da API - Para acessar o Pymatgen
API_KEY ='6AsndhXp3MvcQu1J'
```

```
## Definindo funções
# Função de mensagem
def mensagem(mensagem):
    '''
    Envia uma mensagem e averigua a confirmação do usuário para continuar a execução ou parar o programa.
    '''
    msg = input(mensagem)
    while msg.lower() != 'y':
        msg = input(mensagem)

# Função cabeçalho
def cabecalho(titulo):
    '''
    Printa o cabeçalho indicado.
    '''
    print()
    print('=' * 127)
    print('{:^127}'.format(titulo))
    print('=' * 127)
    print()

# Cabeçalho dos metais de transição
cabecalho('METAIS DE TRANSIÇÃO UTILIZADOS PARA A COMBINAÇÃO')
print(f'''Lista dos metais de transição: [{', '.join(metais_de_transicao)}]

Número total de elementos: {len(metais_de_transicao)}''')

# Cabeçalho do elemento analisado
cabecalho('DEFININDO O ELEMENTO A SER COMBINADO')
```

```
# Definindo o elemento químico a ser combinado com os metais de transição
elemento_analisado = input(f'''Olá, seja bem-vindo ao extrator de dados!

Para prosseguir selecione um dos símbolos químicos abaixo para extrair suas combinações com os {len(metais_de_transicao)} metais de transição listados acima.

Os símbolos químicos disponíveis são:
[{', '.join(elementos_permitidos)}]

Insira o símbolo químico desejado: ''')
while elemento_analisado not in elementos_permitidos:
    elemento_analisado = input(f'''Símbolo químico inválido! Os símbolos químicos disponíveis são: [{', '.join(elementos_permitidos)}]
''')

# Definindo o nome do arquivo
if elemento_analisado == 'Al':
    nome_arquivo_bd = 'Aluminetos.xlsx'
elif elemento_analisado == 'B':
    nome_arquivo_bd = 'Boretos.xlsx'
else:
    nome_arquivo_bd = 'Silicetos.xlsx'
```

APÊNDICE C - Código extrator dos dados da MAPI

```
# Cabeçalho da extração via API
cabecalho('EXTRAÇÃO DOS DADOS VIA API')

print('Observação: A partir de agora algumas perguntas para prosseguir serão realizadas, as opções estarão entre colchetes -> [y/n] que siginificam "yes or no" para prosseguir ou não prosseguir.\n')

mensagem(f'Podemos iniciar a extração dos dados dos compostos {elemento_analisado}M? [y/n]')

print('\nExtraindo os dados e gerando o DataFrame...')

start_time = time.time()

df_dados_api = pd.DataFrame() # Creating the Final Database

with MPRester(API_KEY) as mpr:
    for metal in metais_de_transicao:
        formula = f'{elemento_analisado}-{metal}'
        query = mpr.query(formula,['material_id', 'pretty_formula','total_magnetization', 'formation_energy_per_atom' , 'spacegroup','elasticity','density',  'crystal_system', 'volume'])
        df_query = pd.DataFrame(query)
        df_dados_api = df_dados_api.append(df_query)

print(f"'\nDataFrame gerado com sucesso!

Tempo de execução --- {(time.time() - start_time):.2f} seconds ---'")
```

APÊNDICE D - Código formatação do *DataFrame* gerado na extração da MAPI

```
# Renomenado a coluna do volume da célula unitária
df_dados_api.rename(columns = {'volume':'cell_volume'}, inplace = True)

# Resetando os índices do Dataframe
df_dados_api.reset_index(drop=True, inplace=True)

# Criando a Workbook no Excel
writer = pd.ExcelWriter(nome_arquivo_bd, engine='xlsxwriter')

# Criando a Worksheet dentro da Workbook
df_dados_api.to_excel(writer, sheet_name = r'Dados iniciais')

# Gerando as novas colunas
df_dados_api[['spacegroup_number', 'point_group','G_Reuss', 'G_VRH', 'G_Voigt', 'K_Reuss',
'K_VRH', 'K_Voigt', 'elastic_anisotropy', 'universal_anisotropy', 'poisson_ratio', 'Method']] =
np.nan


## Formatando os dicionários de Spacegroup e Elasticity em colunas
for n in range(len(df_dados_api)):
    # Separando o dicionário do Spacegroup
    df_dados_api.loc[df_dados_api.index[n], 'spacegroup_number'] =
df_dados_api['spacegroup'][n]['number']
    df_dados_api.loc[df_dados_api.index[n], 'point_group'] =
df_dados_api['spacegroup'][n]['point_group']

    # Itera para as linhas que possuirem elasticidade
    if df_dados_api.loc[n, 'elasticity'] != None:
        for nova_coluna in ['G_Reuss', 'G_VRH', 'G_Voigt', 'K_Reuss', 'K_VRH', 'K_Voigt',
'elastic_anisotropy', 'universal_anisotropy', 'poisson_ratio']:
```

```
            df_dados_api.loc[df_dados_api.index[n], nova_coluna] =
df_dados_api['elasticity'][n][nova_coluna]

        # Analisa se o id possui avisos e os adiciona em uma nova coluna
        try:
            df_dados_api.loc[df_dados_api.index[n], 'warnings'] =
df_dados_api['elasticity'][n]['warnings']
        except ValueError:
            pass

        # Extraindo Cij do tensor de elasticidade
        for i in range(6):
            for j in range(6):
                df_dados_api.loc[df_dados_api.index[n], f'C{i+1}{j+1}'] =
df_dados_api['elasticity'][n]['elastic_tensor_original'][i][j]

# Retirando as colunas Spacegroup e Elasticity
df_dados_api.drop(columns = ['spacegroup', 'elasticity'], inplace=True)

# Indicando o método
df_dados_api.loc[df_dados_api['K_VRH'].notnull(), 'Method'] = 'DFT'

# Exportando a planilha com as novas colunas
df_dados_api.to_excel(writer, sheet_name='Dados com elasticidade ')

# Fazendo uma cópia
df_id = df_dados_api.copy()
```

APÊNDICE E - *Web Scraping* da dos valores de estabilidade

```
# Cabeçalho da extração via Web Scraping
cabecalho('EXTRAÇÃO DOS DADOS VIA WEB SCRAPING')

# Permite o usuário realizar o Web Scraping ou pular, como preferir
webscraping_pergunta = input('''Atenção! Na API estão faltando algumas informações
importantes dos compostos:
1. Os dados de K_VRH e G_VRH dos compostos sem matriz de elasticidade, mas que são
previstos no site pelos algoritmos de Machine Learning.
2. A informação 'Decomposes To', que indica para quais compostos o respectivo composto se
decompõe.
Ambas as informações podem ser obtidas por meio de um Web Scraping no site do Materials
Project.

Atenção: Se você pular o Web Scraping sua base de dados ficará com essas informações
faltantes.

Podemos adicionar à base de dados extraída via API essas novas informações via Web
Scraping? [y/n] ''')

while webscraping_pergunta.lower() not in ['y', 'n']:
    webscraping_pergunta = input('''Insira um valor válido! "y" para rodar o Web Scraping ou
"n" para pulá-lo.
Atenção: Se você pular o Web Scraping sua base de dados ficará com essas informações
faltantes.''')

## Coletando os dados de estabilidade, K_VRH e G_VRH previstos pelo programa de
Machine Learning
if webscraping_pergunta == 'y':

    # Coletando dados com o usuário
    email = input('Insira o seu e-mail usp: ')
```

```
    numero_usp = input('Digite seu número usp: ')
    senha_usp = getpass('Digite sua senha usp: ')

    # Outras constantes
    stability_XPATH =
'/html/body/div/div/div[3]/div[1]/div[2]/div[2]/table[1]/tbody/tr[6]/td/span'

    google_email_xpath = r'//*[@id="identifierId"]'
    google_continuar_xpath = r'//*[@id="identifierId"]'
    google_confirmar_xpath =
r'//*[@id="view_container"]/div/div/div[2]/div/div[2]/div/div[1]/div/div/button/span'

    num_usp_xpath = r'//*[@id="username"]'
    usp_senha_xpath = r'//*[@id="password"]'
    usp_login_btn_xpath = r'//*[@id="login-main-content"]/div[1]/form/button'

    url = f'https://materialsproject.org/janrain/loginpage/?next=/janrain/loginpage/'

    ## Iniciando o navegador
    mensagem('Podemos iniciar o navegador? [y/n]')

    start_time = time.time()

    # Inicializando o navegador
    s = Service('chromedriver.exe')
    browser = webdriver.Chrome(service = s)

    browser.get(url) # Navegando para a url indicada
    wait = WebDriverWait(browser, 10)
    action = ActionChains(browser)

    # Fazendo login
    try:
```

```
        btn_login_google = wait.until(EC.presence_of_element_located((By.XPATH,
r'/html/body/div/div[2]/div[2]/div[1]/a/button')))
        action.move_to_element(btn_login_google).click().perform()

    except TimeoutException:
        print(f'Erro browser demorou muito para carregar ou o elemento não foi encontrado.')
        browser.close()

    wait.until(EC.presence_of_element_located((By.XPATH,
google_email_xpath))).send_keys(email)
    wait.until(EC.presence_of_element_located((By.XPATH,
google_continuar_xpath))).send_keys(Keys.ENTER)

    wait.until(EC.presence_of_element_located((By.XPATH,
num_usp_xpath))).send_keys(numero_usp)
    wait.until(EC.presence_of_element_located((By.XPATH,
usp_senha_xpath))).send_keys(senha_usp)
    wait.until(EC.presence_of_element_located((By.XPATH, usp_login_btn_xpath))).click()

    wait.until(EC.presence_of_element_located((By.XPATH,
google_confirmar_xpath))).click()

    home_page_ok = input('Assim que deixar na página principal logada digite ok: ')
    while home_page_ok != 'ok':
        home_page_ok = input('Assim que deixar na página principal logada digite ok: ')

    wait.until(EC.element_to_be_clickable((By.XPATH,
r'//*[@id="searchfield"]/form/div[2]/div/button')))

    ## Extraindo estabilidade e predições de K_VRH e G_VRH
    # Extraindo estabilidade
    for n in range(len(df_id)):
        erro_estabilidade = 0
```

```
        material_id = df_id['material_id'][n]

        # Definindo a url
        url = f'https://materialsproject.org/materials/{material_id}/'
        browser.get(url) # Navegando para a url indicada

        try:
            WebDriverWait(browser, 1).until(EC.presence_of_element_located((By.XPATH,
stability_XPATH)))
            stability = browser.find_element(By.XPATH, stability_XPATH).text # Valor do
elemento web
            df_id.loc[df_id.index[n], 'Decomposes_To'] = stability
            print(f"'Composto: {n}
Id: {material_id}
Decomposes To: {stability}'")

        except TimeoutException:
            print(f"'Composto: {n}
Id: {material_id}
Decomposes To: Não encontrado'")
            erro_estabilidade = 1
            pass

    # Extraindo K_VRH e G_VRH
        K_VRH_material = df_id['K_VRH'][n]

        if erro_estabilidade == 1:
            print(f'K_VRH: Não encontrado, G_VRH: Não encontrado \n')

        elif math.isnan(K_VRH_material):
            try:
```

```
            wait.until(EC.element_to_be_clickable((By.XPATH, r'//*[@id="kg-
prediction"]/button'))).click()

            K_VRH_predicao = wait.until(EC.presence_of_element_located((By.XPATH,
r'//*[@id="kg-prediction"]/div/table/tbody/tr/td[1]/span'))).text
            G_VRH_predicao = wait.until(EC.presence_of_element_located((By.XPATH,
r'//*[@id="kg-prediction"]/div/table/tbody/tr/td[2]/span'))).text

            print(f'K_VRH: {K_VRH_predicao}, G_VRH: {G_VRH_predicao} \n')

            K_VRH_predicao = float(K_VRH_predicao[:5])
            G_VRH_predicao = float(G_VRH_predicao[:5])

            df_id.loc[df_id.index[n], 'K_VRH'] = K_VRH_predicao
            df_id.loc[df_id.index[n], 'G_VRH'] = G_VRH_predicao
            df_id.loc[df_id.index[n], 'Method'] = 'ML' # Previstos segundo o algoritmo de
Machine Learning

        except TimeoutException:
            print(f'K_VRH: Não encontrado, G_VRH: Não encontrado (TimeoutException)\n')
            pass
    else: # ver de tirar
        df_id.loc[df_id.index[n], 'Method'] = 'DFT' # Previstos segundo o algoritmo de
Machine Learning
        print()

end_time = time.time()

mensagem('Podemos fechar o navegador? [y/n]')

# Fechando o navegador
browser.close()
```

```
    print(f"'\nDataFrame editado com sucesso!

Tempo de execução --- {(end_time - start_time):.2f} seconds ---'")
# Gravando a planilha após o Web Scraping
df_id.to_excel(writer, sheet_name='Dados completos')
```

APÊNDICE F - Código do cálculo dos parâmetros e geração do arquivo como base de dados local

```
# Cálculando o módulo de Young
df_id['E_Young'] = (9 * df_id['K_VRH'] * df_id['G_VRH'])/(3 * df_id['K_VRH'] +
df_id['G_VRH']) # Capacidade de resistir a alongamento ou compressão longitudinal

# Cálculando da razão ou coeficiente de Poisson
df_id['Coef_Poisson'] = (3 * df_id['K_VRH'] - df_id['E_Young']) / (6 * df_id['K_VRH']) #
Compressão ou alongamento longitudinal

# Criando Worksheet com a coluna de E_Young
df_id.to_excel(writer, sheet_name='Dados calculados')
writer.save()

print()
print(f'Arquivo {nome_arquivo_bd} salvo com sucesso. Fim da execução!')
```

APÊNDICE G – Código de geração dos gráficos

```
import pandas as pd
import NumPy as np
import math
import matplotlib.pyplot as plt

## Definindo as propriedades a serem extraidas
propriedades_lista = ['formation_energy_per_atom', 'E_Young', 'poisson_ratio', 'K_VRH',
'G_VRH']
propriedades_lista_titulos = ['Energia de formação', 'Módulo de Young', 'Coeficiente de
Poisson', 'Módulo volumétrico', 'Módulo de cisalhamento']
grafico_legenda_lista = [r'$\Delta E_f$ (eV)', r'E-Young (GPa)', r'$\nu - Poisson$',
'$K_{VRH}$ (GPa)', '$G_{VRH}$ (GPa)']



## Definindo os elementos permitidos e os diretórios dos arquivos
elementos_permitidos = ['Al', 'B', 'Si']

elemento_analisado = input(f"Qual o elemento que será análisado [{',
'.join(elementos_permitidos)}]? ")

while elemento_analisado not in elementos_permitidos:
    elemento_analisado = input(f"'Símbolo químico inválido! Os símbolos químicos
disponíveis são: [{', '.join(elementos_permitidos)}] '")

# Definindo o nome do arquivo
if elemento_analisado == 'Al':
    nome_arquivo_bd = 'Bases de dados\Aluminetos.xlsx'
elif elemento_analisado == 'B':
    nome_arquivo_bd = 'Bases de dados\Boretos.xlsx'
else:
    nome_arquivo_bd = 'Bases de dados\Silicetos.xlsx'
```

```
dados = pd.read_excel(nome_arquivo_bd, sheet_name = 'Dados calculados')

# Definindo funções
def resetando_df_modelo():
    '''
    Função para deixar as tabelas/dataframes no formato adequado para o script do gráfico
    '''
    data = {'Metal':  ['Sc','Ti','V','Cr','Mn','Fe','Co','Ni','Cu','Zn','Y','Zr','Nb','Mo','Tc','Ru',
'Rh','Pd','Ag','Cd', 'La','Hf','Ta','W','Re','Os', 'Ir', 'Pt','Au', 'Hg'],
            'Propriedade': [np.nan]*30,
            }
    # Create the pandas DataFrame
    df_modelo = pd.DataFrame(data)
    return df_modelo

## Transformando dataframes e gerando os gráficos
for idxx, prop in enumerate(propriedades_lista):
    df_prop = dados[['pretty_formula', prop]]

    # Determinando os Dataframes
    search_for =
['Sc3','Ti3','V3','Cr3','Mn3','Fe3','Co3','Ni3','Cu3','Zn3','Y3','Zr3','Nb3','Mo3','Tc3','Ru3',
'Rh3','Pd3','Ag3','Cd3','Hf3','Ta3','W3','Re3','Os3', 'Ir3', 'Pt3','Au3', 'Hg3', 'La3']
    # Filtrando compostos com 3 metais de transição
    df_prop_25por = df_prop[df_prop['pretty_formula'].str.contains('|'.join(search_for))]

    search_for =
['Sc2','Ti2','V2','Cr2','Mn2','Fe2','Co2','Ni2','Cu2','Zn2','Y2','Zr2','Nb2','Mo2','Tc2','Ru2',
'Rh2','Pd2','Ag2','Cd2','Hf2','Ta2','W2','Re2','Os2', 'Ir2', 'Pt2','Au2', 'Hg2', 'La2']
    # Filtrando compostos com 3 metais de transição
    df_prop_33por = df_prop[df_prop['pretty_formula'].str.contains('|'.join(search_for))]
```

```
    df_prop_50por = df_prop

    # Filtrando compostos com 3 metais de transição
    df_prop_66por = df_prop[df_prop['pretty_formula'].str.contains(f'{elemento_analisado}2')]

    df_prop_75por = df_prop[df_prop['pretty_formula'].str.contains(f'{elemento_analisado}3')]

    filtro_25_75 =
['2','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21','22','23','24','25','26','27','28','29','30']
    filtro_50 =
['2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21','22','23','24','25','26','27','28','29','30']
    filtro_33_66 =
['3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20','21','22','23','24','25','26','27','28','29','30']

    porcent_lista = ['25%', '33%', '50%', '66%', '75%']
    filtros_lista = [filtro_25_75, filtro_50, filtro_33_66]
    df_lista = [df_prop_25por, df_prop_33por, df_prop_50por, df_prop_66por, df_prop_75por]


    prop_titulo = propriedades_lista_titulos[idxx]
    legenda_do_grafico = grafico_legenda_lista[idxx]

    for idx, df in enumerate(df_lista):
        porcentagem = porcent_lista[idx]
        nome_do_arquivo = f'{prop_titulo}-{porcentagem}{elemento_analisado}.pdf'
        titulo_do_grafico = f'{porcentagem}{elemento_analisado} - {prop_titulo}'

        teste1 = idx % 2
        teste2 = idx // 2
```

```
if teste1 == 1:
    for i in filtro_33_66:
        df = df[~df.pretty_formula.str.contains(i)]
elif teste2 == 1:
    for i in filtro_50:
        df = df[~df.pretty_formula.str.contains(i)]
else:
    for i in filtro_25_75:
        df = df[~df.pretty_formula.str.contains(i)]

# Remove duplicatas e fica com a de menor energia de formação
df = df.sort_values(by = [prop]).drop_duplicates(subset='pretty_formula', keep='first')
print(f'\nForam encontrados {len(df)} compostos.')

df2 = df.copy()

for ireplace in [f'{elemento_analisado}', '2', '3', '4']:
    df2['pretty_formula'] = df2['pretty_formula'].str.replace(ireplace,'')

# Zerando a tabela do df_modelo
df_modelo = resetando_df_modelo()

# Adiciona os valores no modelo desejado
for element in df2['pretty_formula']:
    valor = df.loc[df2['pretty_formula'] == element, prop].iloc[0]
    df_modelo.loc[df_modelo.Metal == element, 'Propriedade'] = valor

# extraindo nomes dos elementos e criando legendas
d3 = df_modelo['Metal'][:10]
d4 = df_modelo['Metal'][10:20]
d5 = df_modelo['Metal'][20:]
L = [f'{x}\n{y}\n{z}' for x, y, z in zip(d3, d4, d5)]
```

```
        # Gerando o gráfico
        fig = plt.figure()

        l10 = range(10)

        plt.plot(l10, df_modelo['Propriedade'][:10], marker = 'o', color = '#6FE7DD', linestyle =
'-',  label='3d')
        plt.plot(l10, df_modelo['Propriedade'][10:20], marker = 's', color = '#3490DE', linestyle =
'-', label='4d')
        plt.plot(l10, df_modelo['Propriedade'][20:], marker = '^', color = '#6639A6', linestyle = '-
', label='5d')

        # Titulo do gráfico
        plt.title(titulo_do_grafico)

        # adiciona linhas tracejadas caso haja dados faltantes
        is_NaN = df_modelo.isnull()
        row_has_NaN = is_NaN.any(axis=1)
        rows_with_NaN = df_modelo[row_has_NaN]
        index_of_rows_with_NaN = list(rows_with_NaN.index)

        print(f'\nElementos faltantes no gráfico:{porcentagem}{elemento_analisado}')
        for i in index_of_rows_with_NaN:
            j = i % 10
            grupo = i // 10
            print(f'{j + 1}º elemento da {grupo + 1}ª linha')
            if j > 0 and j < 9:
                c = [j-1, j+1]
                pi = df_modelo['Propriedade'][i-1]
                pf = df_modelo['Propriedade'][i+1]
                color = ['#6FE7DD', '#3490DE', '#6639A6']
                plt.plot(c, [pi, pf], color[i // 10], linestyle = '--')
```

```
            # Repetição dos ifs até encontrar o ponto subsequente, dependendo da distância traceja de uma maneira diferente
            controlador = 8
            while math.isnan(pf) and j < controlador and controlador > 1:
                c = [j-1, j+(10-controlador)]
                pf = df_modelo['Propriedade'][i+(10-controlador)]
                color = ['#6FE7DD', '#3490DE', '#6639A6']
                if controlador == 8:
                    linha = '-.'
                else:
                    linha = ':'
                plt.plot(c, [pi, pf], color[i // 10], linestyle = linha)
                controlador -= 1

        print(f'Total: {len(index_of_rows_with_NaN)}')
        display(df)

        # formatação
        plt.axhline(0, linestyle='--', color='k')
        plt.ylabel(legenda_do_grafico)
        plt.legend(loc = 'center left', bbox_to_anchor=(1, 0.5))
        plt.xticks(l10, L)
        plt.tight_layout()
        plt.savefig(f'Graficos/{nome_do_arquivo}.pdf')
```

APÊNDICE H – *DataFrames* das propriedades por composição dos compostos metálicos para o alumínio

Figura H-2 *DataFrames* de energia de formação para os compostos binários de Al com os metais de transição nas proporões de 25%, 33%, 50%, 66% e 75% da esquerda para a direita.

	pretty_formula	formation_energy_per_atom
235	AlPt3	-0.692685
169	AlPd3	-0.612381
155	AlRh3	-0.491680
76	AlNi3	-0.433437
115	Zr3Al	-0.296003
22	Ti3Al	-0.279486
8	Sc3Al	-0.268877
129	AlMo3	-0.241469
197	Hf3Al	-0.221002
183	La3Al	-0.201390
52	AlFe3	-0.199436
244	AlAu3	-0.197910
84	AlCu3	-0.189439
101	Y3Al	-0.185329
127	Nb3Al	-0.173775
60	AlCo3	-0.148929
25	AlV3	-0.135406
144	AlTc3	-0.112514
170	AlAg3	-0.073814
44	Mn3Al	-0.037628
176	AlCd3	0.102806

	pretty_formula	formation_energy_per_atom
238	AlPt2	-0.853658
162	AlPd2	-0.833424
80	AlNi2	-0.501586
143	AlTc2	-0.412661
123	Zr2Al	-0.358074
3	Sc2Al	-0.350948
248	AlAu2	-0.316636
108	Y2Al	-0.314966
126	Nb2Al	-0.293135
211	AlRe2	-0.262899
194	Hf2Al	-0.262791
203	Ta2Al	-0.247720
26	AlV2	-0.145029
56	AlFe2	-0.141820
38	AlCr2	-0.121507
172	AlAg2	-0.075495
12	Ti2Al	0.572197

	pretty_formula	formation_energy_per_atom
158	AlRh	-1.101652
233	AlPt	-1.041411
222	AlIr	-0.958980
159	AlPd	-0.915719
147	AlRu	-0.672735
69	AlNi	-0.657566
65	AlCo	-0.610262
122	ZrAl	-0.462376
7	ScAl	-0.446304
102	YAl	-0.418543
179	LaAl	-0.411507
11	TiAl	-0.407164
217	AlOs	-0.400855
193	HfAl	-0.394234
245	AlAu	-0.394012
59	AlFe	-0.327249
212	AlRe	-0.315780
42	MnAl	-0.272440
94	AlCu	-0.213428
174	AlAg	-0.039721

	pretty_formula	formation_energy_per_atom
228	Al2Pt	-0.894034
146	Al2Ru	-0.718840
161	Al2Pd	-0.618174
219	Al2Os	-0.562104
119	ZrAl2	-0.537820
100	YAl2	-0.531464
180	LaAl2	-0.501039
2	ScAl2	-0.486889
195	HfAl2	-0.441321
13	TiAl2	-0.429396
242	Al2Au	-0.423042
88	Al2Cu	-0.178066
208	Al2W	-0.035495
97	Al2Zn	0.007792

	pretty_formula	formation_energy_per_atom
223	Al3Ir	-0.664651
110	ZrAl3	-0.486934
6	ScAl3	-0.454931
237	Al3Pt	-0.440558
182	LaAl3	-0.440391
103	YAl3	-0.437962
124	NbAl3	-0.428580
79	Al3Ni	-0.404993
10	TiAl3	-0.397650
187	HfAl3	-0.395293
153	Al3Ru	-0.386943
167	Al3Pd	-0.346222
130	Al3Mo	-0.321089
141	Al3Tc	-0.318806
202	TaAl3	-0.318202
27	Al3V	-0.288522
55	Al3Fe	-0.154243
37	Al3Cr	-0.140379
246	Al3Au	-0.113982
90	Al3Cu	-0.089984
96	Al3Zn	0.024896
175	Al3Ag	0.042948
177	Al3Cd	0.132653

Fonte: Autoria própria

Figura H-3 *DataFrames* de Módulo de Young para os compostos binários de Al com os metais de transição nas proporões de 25%, 33%, 50%, 66% e 75% da esquerda para a direita.

	pretty_formula	E_Young
82	AlCu3	0.000000
25	AlV3	29.383984
74	AlNi3	46.618705
185	La3Al	46.956522
176	AlCd3	49.705100
127	Nb3Al	52.071429
101	Y3Al	78.837989
244	AlAu3	79.362397
20	Ti3Al	87.513812
4	Sc3Al	87.750000
171	AlAg3	88.691479
61	AlCo3	127.542010
121	Zr3Al	131.964996
197	Hf3Al	132.002397
169	AlPd3	142.448540
226	AlPt3	193.853078
58	AlFe3	194.436242
44	Mn3Al	205.436620
155	AlRh3	209.897811
144	AlTc3	230.367774
129	AlMo3	307.820352

	pretty_formula	E_Young
12	Ti2Al	-15.360577
243	AlAu2	75.717391
108	Y2Al	76.153846
172	AlAg2	81.346154
109	Zr2Al	108.211144
3	Sc2Al	109.133858
80	AlNi2	111.932773
56	AlFe2	120.024742
162	AlPd2	138.137143
194	Hf2Al	144.182278
201	Ta2Al	145.800000
238	AlPt2	159.004458
26	AlV2	201.238652
126	Nb2Al	210.297345
38	AlCr2	260.850227
143	AlTc2	320.440551
211	AlRe2	352.881637

	pretty_formula	E_Young
43	MnAl	-103.194379
249	AlAu	-66.150000
179	LaAl	84.031088
102	YAl	87.906977
231	AlPt	90.793220
174	AlAg	103.399873
7	ScAl	104.661818
168	AlPd	140.400000
19	TiAl	162.957975
94	AlCu	167.830049
122	ZrAl	168.612245
50	AlFe	173.863636
214	AlRe	175.027624
193	HfAl	183.345882
69	AlNi	188.129032
158	AlRh	260.765217
147	AlRu	264.110014
65	AlCo	290.225954
222	AlIr	297.728518
217	AlOs	303.428571

	pretty_formula	E_Young
208	Al2W	-9.056962
161	Al2Pd	82.622951
242	Al2Au	89.612069
97	Al2Zn	100.242632
180	LaAl2	100.434783
83	Al2Cu	105.756677
228	Al2Pt	106.288732
100	YAl2	144.000000
2	ScAl2	164.108108
9	TiAl2	194.400000
119	ZrAl2	207.697183
195	HfAl2	212.142857
146	Al2Ru	256.306291
219	Al2Os	341.395804

	pretty_formula	E_Young
98	YAl3	-19.028571
96	Al3Zn	0.000000
177	Al3Cd	84.454935
182	LaAl3	99.313953
246	Al3Au	106.112266
175	Al3Ag	108.179761
90	Al3Cu	125.039634
191	HfAl3	131.680328
223	Al3Ir	132.651394
55	Al3Fe	139.310748
167	Al3Pd	140.678539
30	Al3V	146.322581
17	TiAl3	150.167342
117	ZrAl3	152.566531
1	ScAl3	156.485549
237	Al3Pt	160.887902
79	Al3Ni	165.733333
153	Al3Ru	183.356358
37	Al3Cr	197.844059
141	Al3Tc	215.412668
130	Al3Mo	235.104950
124	NbAl3	246.944444
202	TaAl3	247.491054

Fonte: Autoria própria

Figura H-3 – *DataFrames* do Coeficiente de Poisson para os compostos binários de Al com os metais de transição nas proporões de 25%, 33%, 50%, 66% e 75% da esquerda para a direita.

	pretty_formula	poisson_ratio
4	Sc3Al	0.22
101	Y3Al	0.23
129	AlMo3	0.27
22	Ti3Al	0.27
115	Zr3Al	0.27
76	AlNi3	0.29
185	La3Al	0.30
58	AlFe3	0.31
234	AlPt3	0.32
44	Mn3Al	0.32
93	AlCu3	0.34
61	AlCo3	0.39
127	Nb3Al	0.45
25	AlV3	0.47
144	AlTc3	NaN
155	AlRh3	NaN
169	AlPd3	NaN
170	AlAg3	NaN
176	AlCd3	NaN
197	Hf3Al	NaN
244	AlAu3	NaN

	pretty_formula	poisson_ratio
3	Sc2Al	0.24
143	AlTc2	0.26
211	AlRe2	0.27
38	AlCr2	0.27
108	Y2Al	0.27
123	Zr2Al	0.28
126	Nb2Al	0.28
51	AlFe2	0.28
194	Hf2Al	0.29
232	AlPt2	0.32
162	AlPd2	0.35
201	Ta2Al	0.35
172	AlAg2	0.36
248	AlAu2	0.37
80	AlNi2	0.40
12	Ti2Al	0.54
26	AlV2	NaN

	pretty_formula	poisson_ratio
95	AlCu	-2.58
50	AlFe	0.16
65	AlCo	0.23
193	HfAl	0.24
122	ZrAl	0.24
179	LaAl	0.24
11	TiAl	0.25
42	MnAl	0.26
102	YAl	0.26
7	ScAl	0.28
222	AlIr	0.28
147	AlRu	0.28
158	AlRh	0.28
217	AlOs	0.29
159	AlPd	0.31
69	AlNi	0.31
233	AlPt	0.32
245	AlAu	0.36
214	AlRe	0.37
174	AlAg	NaN

	pretty_formula	poisson_ratio
2	ScAl2	0.19
180	LaAl2	0.19
119	ZrAl2	0.19
149	Al2Ru	0.19
9	TiAl2	0.20
219	Al2Os	0.20
100	YAl2	0.20
195	HfAl2	0.21
89	Al2Cu	0.27
228	Al2Pt	0.36
242	Al2Au	0.36
161	Al2Pd	0.38
208	Al2W	0.51
97	Al2Zn	NaN

	pretty_formula	poisson_ratio
6	ScAl3	0.17
27	Al3V	0.17
10	TiAl3	0.18
124	NbAl3	0.18
110	ZrAl3	0.18
105	YAl3	0.18
188	HfAl3	0.19
37	Al3Cr	0.19
202	TaAl3	0.19
130	Al3Mo	0.21
141	Al3Tc	0.25
79	Al3Ni	0.26
182	LaAl3	0.27
90	Al3Cu	0.28
55	Al3Fe	0.31
223	Al3Ir	0.35
96	Al3Zn	0.50
153	Al3Ru	NaN
167	Al3Pd	NaN
175	Al3Ag	NaN
177	Al3Cd	NaN
237	Al3Pt	NaN
246	Al3Au	NaN

Fonte: Autoria própria

Figura H-4 – *DataFrames* do Módulo de Volume (K_{VRH}) para os compostos binários de Al com os metais de transição nas proporões de 25%, 33%, 50%, 66% e 75% da esquerda para a direita.

	pretty_formula	K_VRH
185	La3Al	40.00
101	Y3Al	49.00
176	AlCd3	51.01
4	Sc3Al	52.00
171	AlAg3	91.55
121	Zr3Al	100.30
16	Ti3Al	108.40
197	Hf3Al	121.00
244	AlAu3	123.40
82	AlCu3	129.00
169	AlPd3	151.80
25	AlV3	159.00
127	Nb3Al	162.00
58	AlFe3	174.00
76	AlNi3	177.00
44	Mn3Al	187.00
61	AlCo3	187.00
155	AlRh3	200.80
144	AlTc3	202.50
226	AlPt3	212.40
129	AlMo3	225.00

	pretty_formula	K_VRH
108	Y2Al	55.0
3	Sc2Al	70.0
12	Ti2Al	71.0
172	AlAg2	94.0
109	Zr2Al	100.0
194	Hf2Al	113.0
51	AlFe2	117.0
248	AlAu2	118.0
26	AlV2	141.3
162	AlPd2	158.0
126	Nb2Al	161.0
201	Ta2Al	162.0
80	AlNi2	185.0
38	AlCr2	186.0
238	AlPt2	205.0
143	AlTc2	224.0
211	AlRe2	255.0

	pretty_formula	K_VRH
179	LaAl	53.00
102	YAl	60.00
5	ScAl	63.92
50	AlFe	85.00
174	AlAg	87.59
19	TiAl	101.10
245	AlAu	104.00
122	ZrAl	108.00
94	AlCu	113.00
193	HfAl	117.00
42	MnAl	141.00
159	AlPd	148.00
78	AlNi	158.00
65	AlCo	179.00
233	AlPt	184.00
158	AlRh	196.00
212	AlRe	199.40
147	AlRu	202.00
222	AlIr	229.00
217	AlOs	236.00

	pretty_formula	K_VRH
180	LaAl2	55.00
97	Al2Zn	74.37
100	YAl2	80.00
2	ScAl2	88.00
89	Al2Cu	90.00
242	Al2Au	105.00
9	TiAl2	108.00
161	Al2Pd	112.00
119	ZrAl2	113.00
195	HfAl2	120.00
228	Al2Pt	129.00
149	Al2Ru	155.00
208	Al2W	159.00
219	Al2Os	191.00

	pretty_formula	K_VRH
98	YAl3	37.00
177	Al3Cd	64.21
182	LaAl3	73.00
96	Al3Zn	75.00
175	Al3Ag	79.85
6	ScAl3	84.00
17	TiAl3	88.97
91	Al3Cu	90.81
117	ZrAl3	91.05
246	Al3Au	92.00
167	Al3Pd	104.40
191	HfAl3	105.00
37	Al3Cr	107.00
79	Al3Ni	113.00
27	Al3V	120.00
237	Al3Pt	121.40
55	Al3Fe	125.00
124	NbAl3	127.00
153	Al3Ru	129.30
202	TaAl3	133.00
130	Al3Mo	136.00
221	Al3Ir	137.70
141	Al3Tc	145.00

Fonte: Autoria própria

Figura H-5 – *DataFrames* do Módulo de Cisalhamento (G_{VRH}) para os compostos binários de Al com os metais de transição nas proporões de 25%, 33%, 50%, 66% e 75% da esquerda para a direita.

	pretty_formula	G_VRH
82	AlCu3	0.00
25	AlV3	10.00
74	AlNi3	16.00
185	La3Al	18.00
127	Nb3Al	18.00
176	AlCd3	18.58
244	AlAu3	28.49
101	Y3Al	32.00
20	Ti3Al	32.00
171	AlAg3	33.13
4	Sc3Al	36.00
61	AlCo3	46.00
197	Hf3Al	50.07
121	Zr3Al	51.52
169	AlPd3	53.01
226	AlPt3	71.91
58	AlFe3	74.00
44	Mn3Al	78.00
155	AlRh3	79.16
144	AlTc3	87.90
129	AlMo3	121.00

	pretty_formula	G_VRH
12	Ti2Al	-5.00
243	AlAu2	27.00
172	AlAg2	30.00
108	Y2Al	30.00
80	AlNi2	40.00
109	Zr2Al	41.00
3	Sc2Al	44.00
56	AlFe2	44.00
162	AlPd2	51.00
201	Ta2Al	54.00
194	Hf2Al	56.00
238	AlPt2	58.00
26	AlV2	79.69
126	Nb2Al	82.00
38	AlCr2	103.00
143	AlTc2	127.00
211	AlRe2	139.00

	pretty_formula	G_VRH
95	AlCu	-666.00
43	MnAl	-32.00
249	AlAu	-21.00
231	AlPt	32.00
179	LaAl	34.00
104	YAl	35.00
174	AlAg	39.67
7	ScAl	41.00
168	AlPd	52.00
214	AlRe	64.00
19	TiAl	66.17
122	ZrAl	68.00
69	AlNi	72.00
193	HfAl	74.00
50	AlFe	75.00
158	AlRh	102.00
147	AlRu	103.00
222	AlIr	116.00
65	AlCo	118.00
217	AlOs	118.00

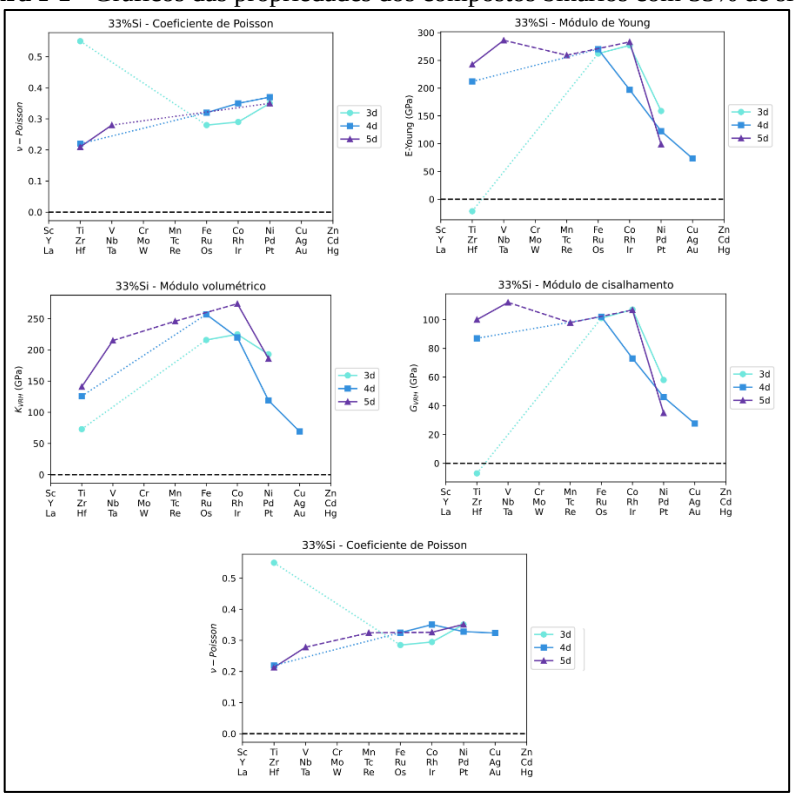
	pretty_formula	G_VRH
208	Al2W	-3.0
161	Al2Pd	30.0
242	Al2Au	33.0
228	Al2Pt	39.0
97	Al2Zn	39.3
83	Al2Cu	40.0
180	LaAl2	42.0
100	YAl2	60.0
2	ScAl2	69.0
9	TiAl2	81.0
119	ZrAl2	87.0
195	HfAl2	88.0
146	Al2Ru	103.0
219	Al2Os	142.0

	pretty_formula	G_VRH
98	YAl3	-6.00
96	Al3Zn	0.00
177	Al3Cd	32.97
182	LaAl3	39.00
246	Al3Au	40.57
175	Al3Ag	42.45
223	Al3Ir	49.00
90	Al3Cu	49.00
191	HfAl3	51.00
55	Al3Fe	53.00
167	Al3Pd	55.15
30	Al3V	56.00
17	TiAl3	61.61
117	ZrAl3	62.49
237	Al3Pt	62.89
1	ScAl3	64.00
79	Al3Ni	66.00
153	Al3Ru	72.55
37	Al3Cr	83.00
141	Al3Tc	86.00
130	Al3Mo	97.00
202	TaAl3	104.00
124	NbAl3	105.00

Fonte: Autoria própria

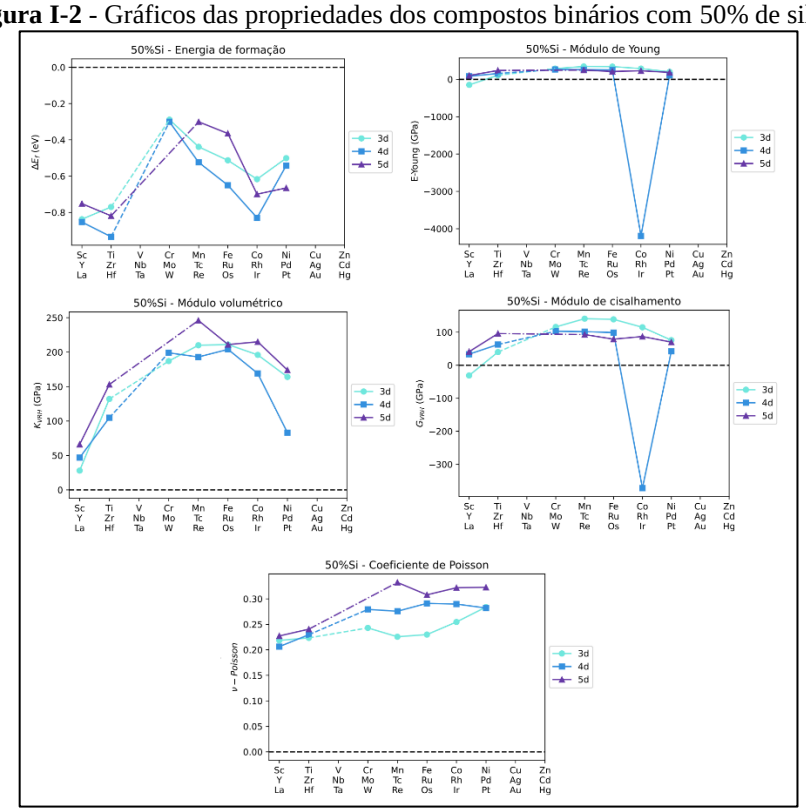
APÊNDICE I –Gráficos das propriedades por composição dos compostos binários de alumínio, boro e silício com metais de transição

Figura I-1 - Gráficos das propriedades dos compostos binários com 33% de silício



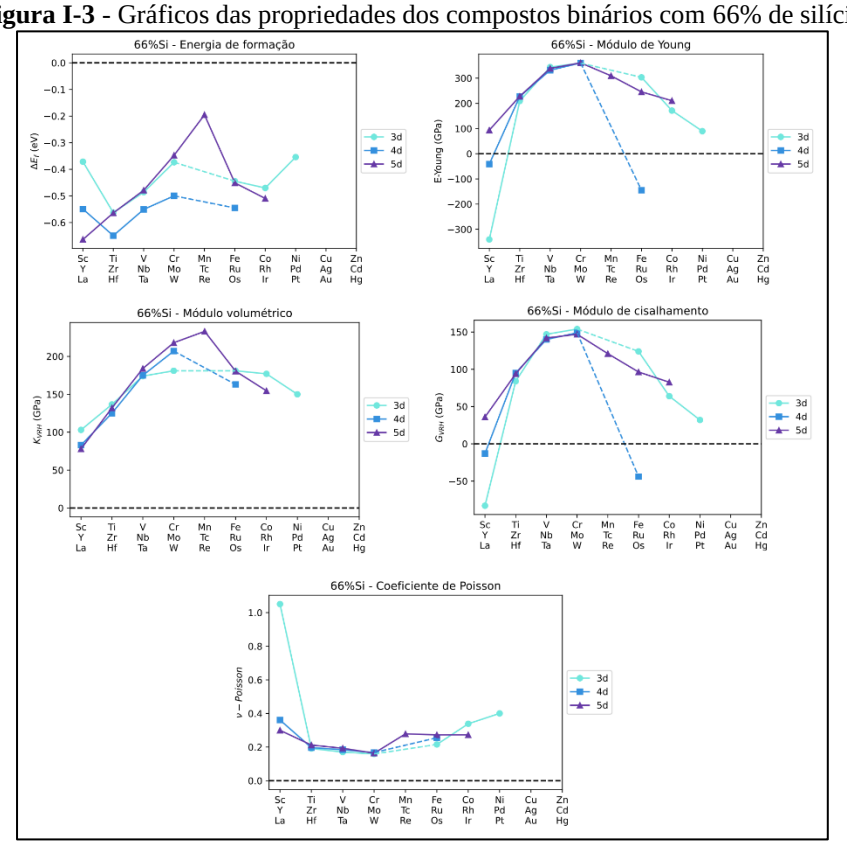
Fonte: Autoria própria

Figura I-2 - Gráficos das propriedades dos compostos binários com 50% de silício



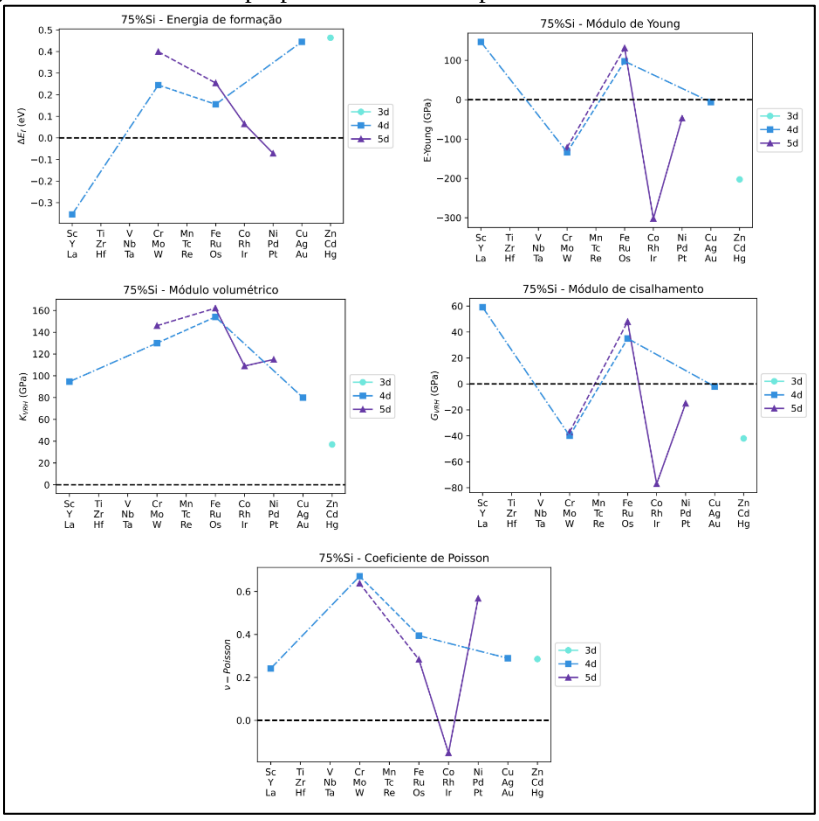
Fonte: Autoria própria

Figura I-3 - Gráficos das propriedades dos compostos binários com 66% de silício



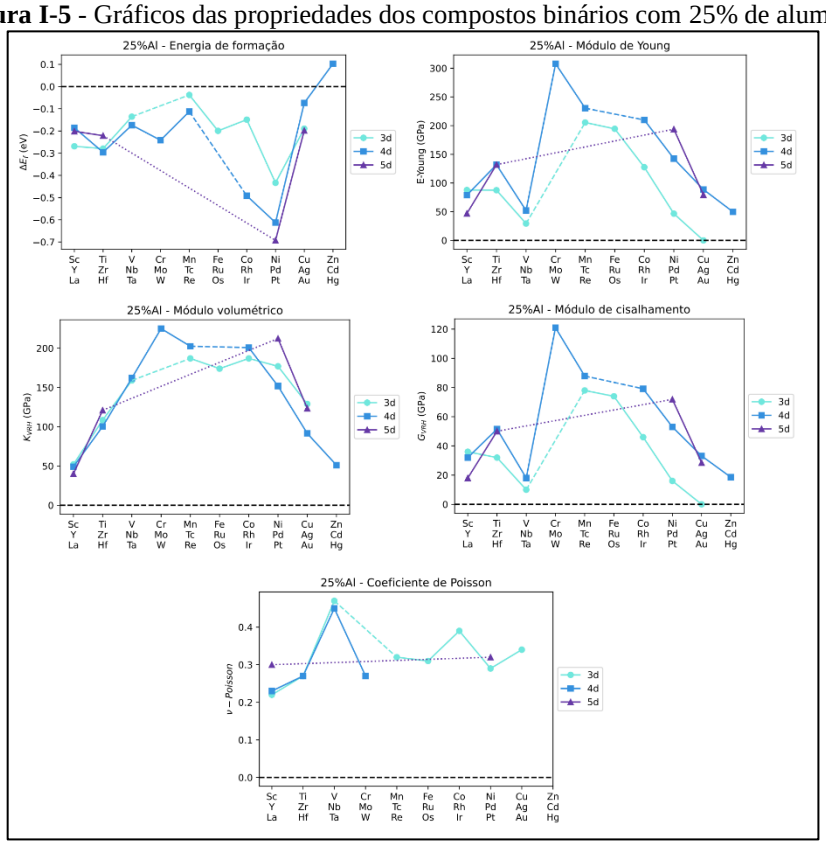
Fonte: Autoria própria

Figura I-4 - Gráficos das propriedades dos compostos binários com 75% de silício



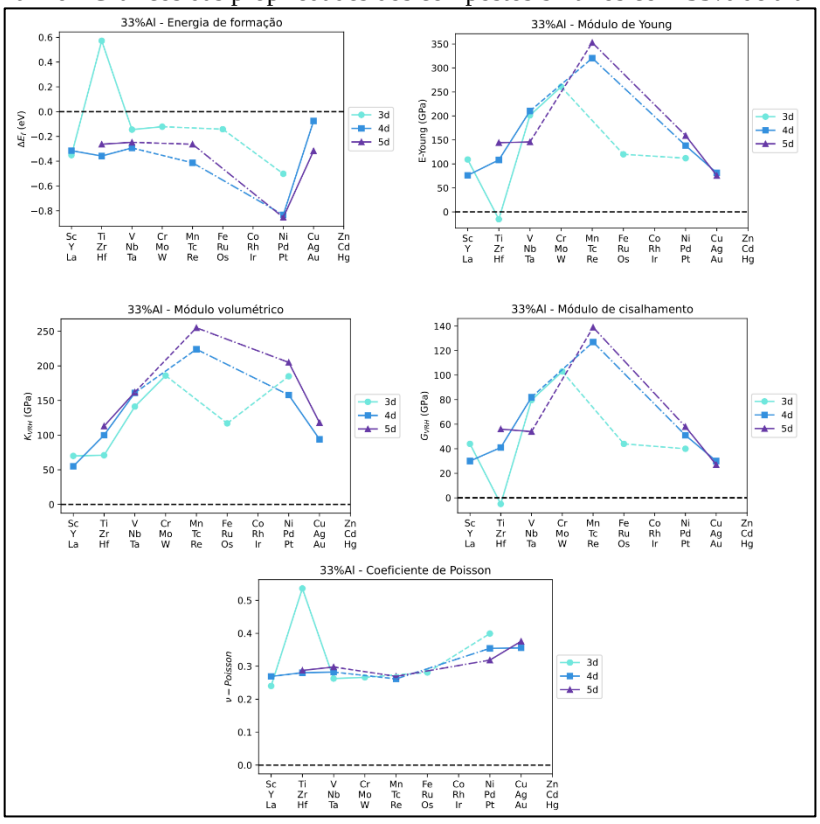
Fonte: Autoria própria

Figura I-5 - Gráficos das propriedades dos compostos binários com 25% de alumínio



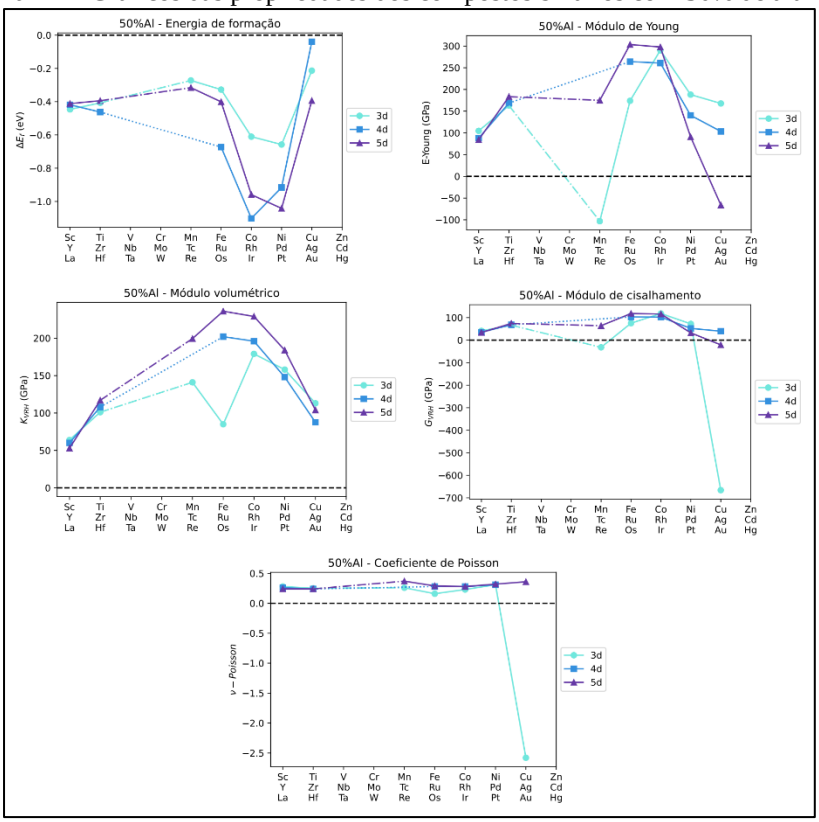
Fonte: Autoria própria

Figura I-6 - Gráficos das propriedades dos compostos binários com 33% de alumínio



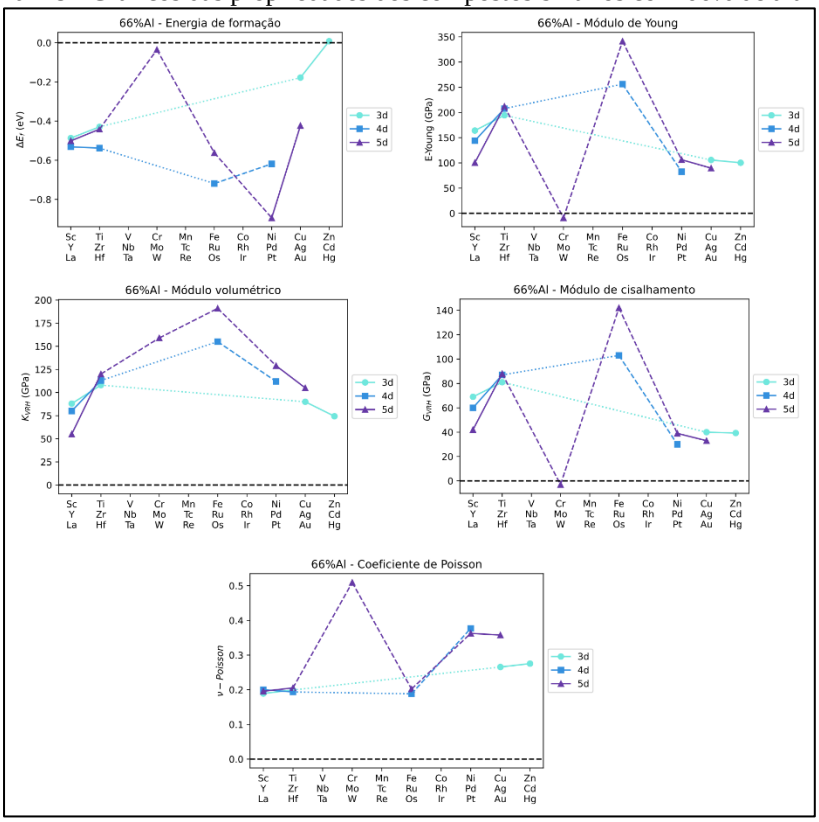
Fonte: Autoria própria

Figura I-7 - Gráficos das propriedades dos compostos binários com 50% de alumínio



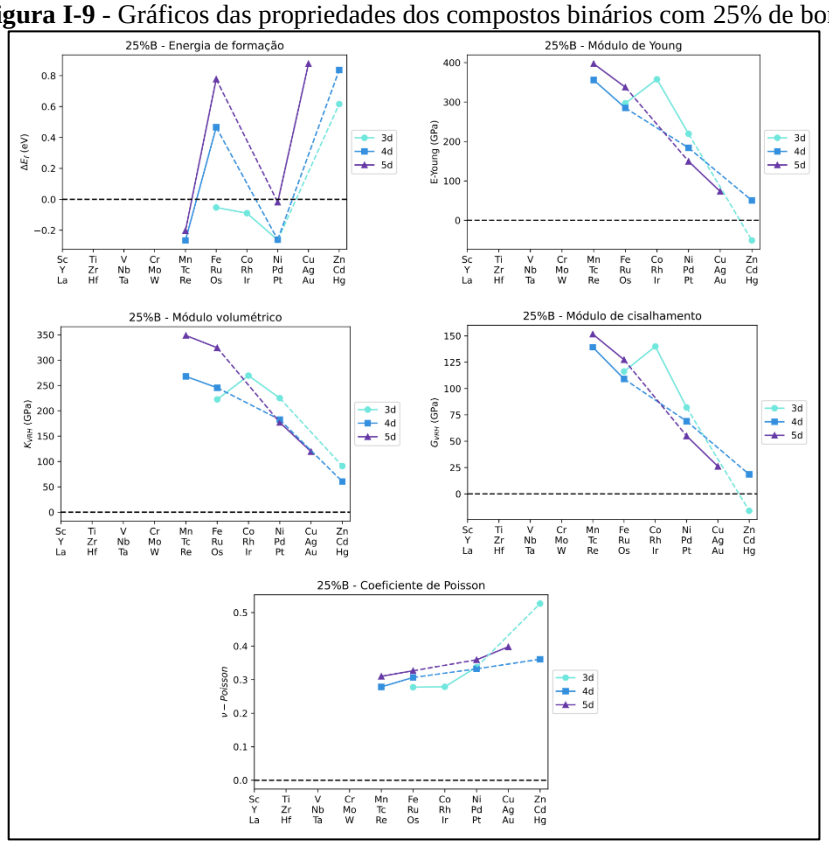
Fonte: Autoria própria

Figura I-8 - Gráficos das propriedades dos compostos binários com 66% de alumínio



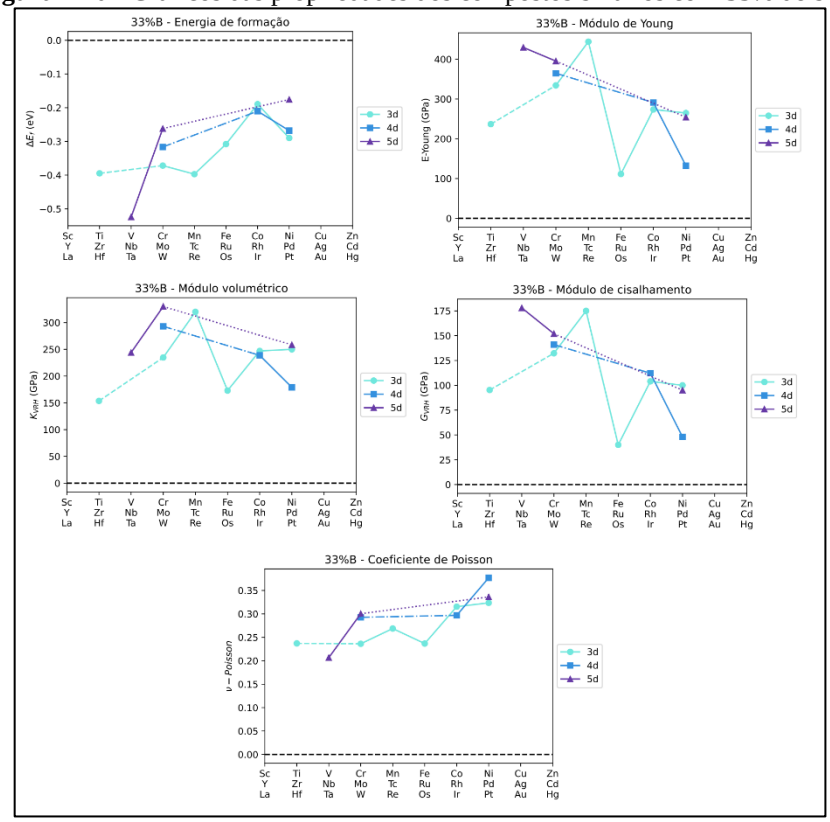
Fonte: Autoria própria

Figura I-9 - Gráficos das propriedades dos compostos binários com 25% de boro



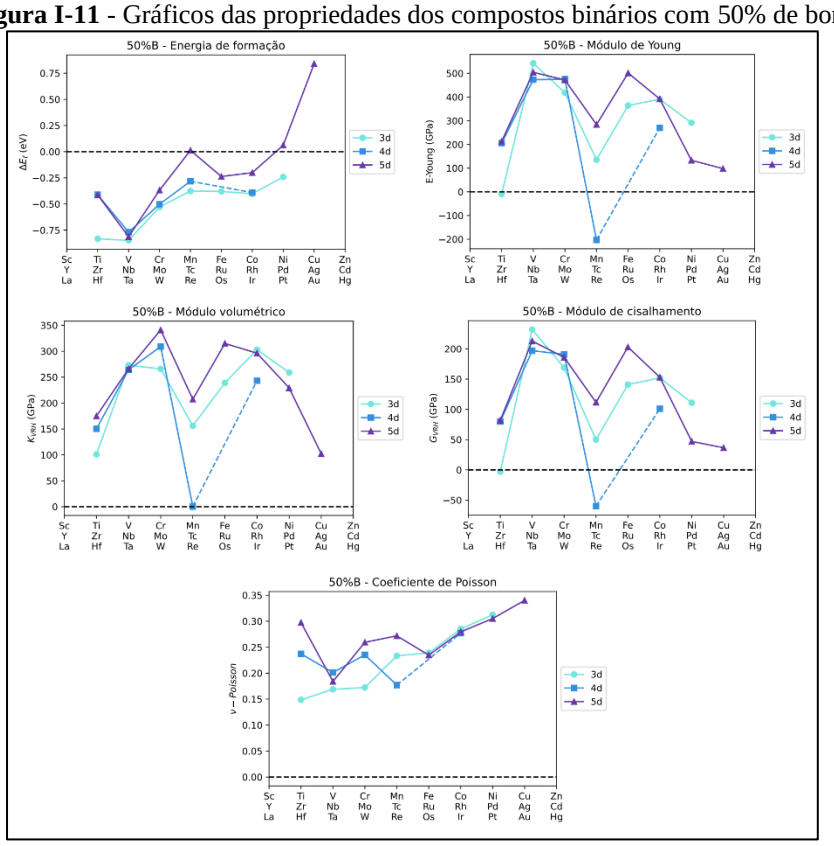
Fonte: Autoria própria

Figura I-10 - Gráficos das propriedades dos compostos binários com 33% de boro



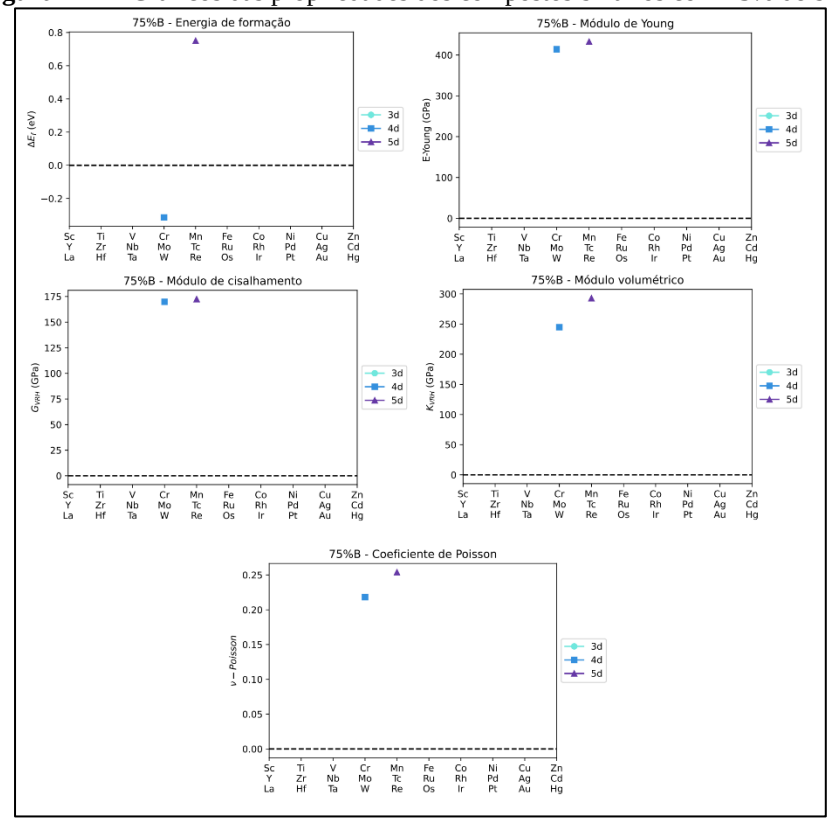
Fonte: Autoria própria

Figura I-11 - Gráficos das propriedades dos compostos binários com 50% de boro



Fonte: Autoria própria

Figura I-12 - Gráficos das propriedades dos compostos binários com 75% de boro



Fonte: Autoria própria