Bibliotecas utilizadas no projeto

Disciplina: Algoritmos de Inteligência Artificial para Clusteriazação

Aluno: Marcelo Barros de Azevedo Vieira

In [18]:
```
import pandas as pd
import numpy as np
import sys
import seaborn as sns
from sklearn.cluster import KMeans, AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn_extra.cluster import KMedoids
import matplotlib.pyplot as plt
```

1.1 VERSÃO DO PYTHON

3.12.7

1.2 AMBIENTE VIRTUAL

ANACONDA

In [19]:
```
print(f"Versão do Python: {sys.version}")
```

```
Versão do Python: 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 08:22:19) [Clang 14.0.6 ]
```

1.3 BIBLIOTECAS INSTALADAS NO ANACONDA

In [20]:
```
print('Bibliotecas utilizadas no ambiente virtual anaconda:')
!conda list
```

Bibliotecas utilizadas no ambiente virtual anaconda:

```
# packages in environment at /Users/marcelodeazevedo/development/projeto_b/env:
#
# Name                    Version                   Build  Channel
anyio                     4.6.2.post1       pyhd8ed1ab_0    conda-forge
appnope                   0.1.4             pyhd8ed1ab_0    conda-forge
argon2-cffi               23.1.0            pyhd8ed1ab_0    conda-forge
argon2-cffi-bindings      21.2.0          py312h80987f9_0
arrow                     1.3.0             pyhd8ed1ab_0    conda-forge
asttokens                 2.4.1             pyhd8ed1ab_0    conda-forge
async-lru                 2.0.4             pyhd8ed1ab_0    conda-forge
attrs                     24.2.0            pyh71513ae_0    conda-forge
babel                     2.16.0            pyhd8ed1ab_0    conda-forge
beautifulsoup4            4.12.3            pyha770c72_0    conda-forge
bleach                    6.2.0             pyhd8ed1ab_0    conda-forge
brotli-python             1.0.9           py312h313beb8_8
bzip2                     1.0.8                h80987f9_6
ca-certificates           2024.9.24            hca03da5_0
cached-property           1.5.2             hd8ed1ab_1    conda-forge
cached_property           1.5.2             pyha770c72_1    conda-forge
certifi                   2024.8.30         pyhd8ed1ab_0    conda-forge
cffi                      1.17.1          py312h3eb5a62_0
charset-normalizer        3.4.0             pyhd8ed1ab_0    conda-forge
comm                      0.2.2             pyhd8ed1ab_0    conda-forge
contourpy                 1.3.0                    pypi_0    pypi
cycler                    0.12.1                   pypi_0    pypi
cython                    3.0.11                   pypi_0    pypi
debugpy                   1.6.7           py312h313beb8_0
decorator                 5.1.1             pyhd8ed1ab_0    conda-forge
defusedxml                0.7.1             pyhd8ed1ab_0    conda-forge
entrypoints               0.4               pyhd8ed1ab_0    conda-forge
exceptiongroup            1.2.2             pyhd8ed1ab_0    conda-forge
executing                 2.1.0             pyhd8ed1ab_0    conda-forge
expat                     2.6.3               h313beb8_0
fonttools                 4.54.1                   pypi_0    pypi
fqdn                      1.5.1             pyhd8ed1ab_0    conda-forge
h11                       0.14.0            pyhd8ed1ab_0    conda-forge
h2                        4.1.0             pyhd8ed1ab_0    conda-forge
hpack                     4.0.0             pyh9f0ad1d_0    conda-forge
httpcore                  1.0.6             pyhd8ed1ab_0    conda-forge
httpx                     0.27.2            pyhd8ed1ab_0    conda-forge
hyperframe                6.0.1             pyhd8ed1ab_0    conda-forge
idna                      3.10              pyhd8ed1ab_0    conda-forge
importlib-metadata        8.5.0             pyha770c72_0    conda-forge
importlib_metadata        8.5.0               hd8ed1ab_0    conda-forge
importlib_resources       6.4.5             pyhd8ed1ab_0    conda-forge
ipykernel                 6.29.5            pyh57ce528_0    conda-forge
ipython                   8.29.0            pyh707e725_0    conda-forge
isoduration               20.11.0           pyhd8ed1ab_0    conda-forge
jedi                      0.19.1            pyhd8ed1ab_0    conda-forge
jinja2                    3.1.4             pyhd8ed1ab_0    conda-forge
joblib                    1.4.2                    pypi_0    pypi
json5                     0.9.25            pyhd8ed1ab_0    conda-forge
jsonpointer               2.0                       py_0    conda-forge
jsonschema                4.23.0            pyhd8ed1ab_0    conda-forge
jsonschema-specifications 2024.10.1         pyhd8ed1ab_0    conda-forge
jsonschema-with-format-nongpl 4.23.0              hd8ed1ab_0    conda-forge
jupyter-lsp               2.2.5             pyhd8ed1ab_0    conda-forge
jupyter_client            8.6.3             pyhd8ed1ab_0    conda-forge
jupyter_core              5.7.2             pyh31011fe_1    conda-forge
jupyter_events            0.10.0            pyhd8ed1ab_0    conda-forge
jupyter_server            2.14.2            pyhd8ed1ab_0    conda-forge
jupyter_server_terminals  0.5.3             pyhd8ed1ab_0    conda-forge
jupyterlab                4.2.5             pyhd8ed1ab_0    conda-forge
jupyterlab_pygments       0.3.0             pyhd8ed1ab_1    conda-forge
jupyterlab_server         2.27.3            pyhd8ed1ab_0    conda-forge
kiwisolver                1.4.7                    pypi_0    pypi
libcxx                    14.0.6              h848a8c0_0
libffi                    3.4.4                hca03da5_1
libsodium                 1.0.18              h27ca646_1    conda-forge
lz4-c                     1.9.4               hb7217d7_0    conda-forge
markupsafe                3.0.2             pyhe1237c8_0    conda-forge
matplotlib                3.9.2                    pypi_0    pypi
matplotlib-inline         0.1.7             pyhd8ed1ab_0    conda-forge
mistune                   3.0.2             pyhd8ed1ab_0    conda-forge
nbclient                  0.10.0            pyhd8ed1ab_0    conda-forge
nbconvert-core            7.16.4            pyhd8ed1ab_1    conda-forge
nbformat                  5.10.4            pyhd8ed1ab_0    conda-forge
ncurses                   6.4                 h313beb8_0
nest-asyncio              1.6.0             pyhd8ed1ab_0    conda-forge
notebook                  7.2.2             pyhd8ed1ab_0    conda-forge
notebook-shim             0.2.4             pyhd8ed1ab_0    conda-forge
numpy                     1.26.4                   pypi_0    pypi
openssl                   3.3.2               h8359307_0    conda-forge
overrides                 7.7.0             pyhd8ed1ab_0    conda-forge
packaging                 24.1              pyhd8ed1ab_0    conda-forge
pandas                    2.2.3                    pypi_0    pypi
pandocfilters             1.5.0             pyhd8ed1ab_0    conda-forge
parso                     0.8.4             pyhd8ed1ab_0    conda-forge
pexpect                   4.9.0             pyhd8ed1ab_0    conda-forge
pickleshare               0.7.5                  py_1003    conda-forge
pillow                    11.0.0                   pypi_0    pypi
pip                       24.2            py312hca03da5_0
pkgutil-resolve-name      1.3.10            pyhd8ed1ab_1    conda-forge
platformdirs              4.3.6             pyhd8ed1ab_0    conda-forge
prometheus_client         0.21.0            pyhd8ed1ab_0    conda-forge
prompt-toolkit            3.0.48            pyha770c72_0    conda-forge
psutil                    5.9.0           py312h80987f9_0
ptyprocess                0.7.0             pyhd3deb0d_0    conda-forge
pure_eval                 0.2.3             pyhd8ed1ab_0    conda-forge
```

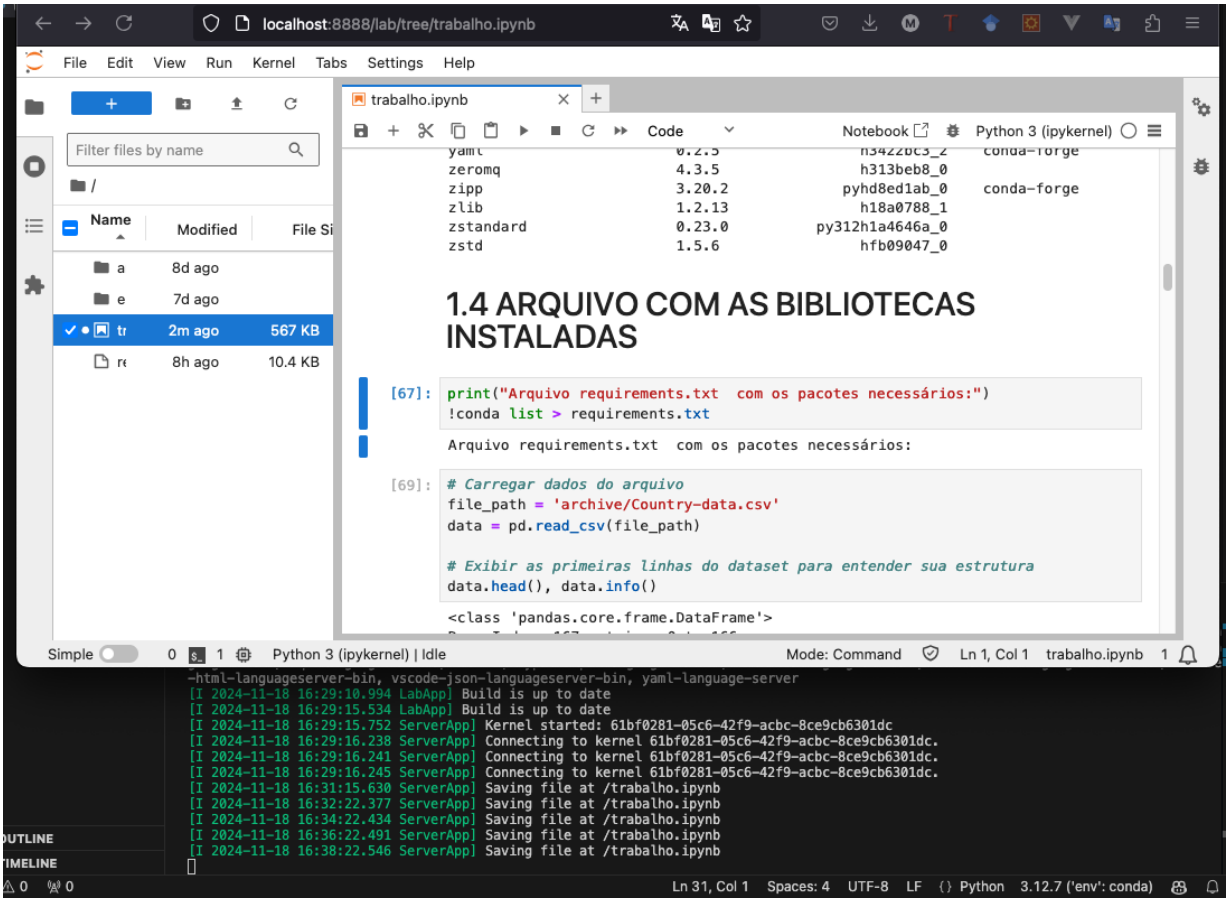
```
pybind11                  2.13.6                   pypi_0    pypi
pycparser                 2.22               pyhd8ed1ab_0    conda-forge
pygments                  2.18.0             pyhd8ed1ab_0    conda-forge
pyobjc-core               10.1            py312h80987f9_0
pyobjc-framework-cocoa    10.1            py312hb094c41_0
pyparsing                 3.2.0                    pypi_0    pypi
pysocks                   1.7.1              pyha2e5f31_6    conda-forge
python                    3.12.7               h99e199e_0
python-dateutil           2.9.0              pyhd8ed1ab_0    conda-forge
python-fastjsonschema     2.20.0             pyhd8ed1ab_0    conda-forge
python-json-logger        2.0.7              pyhd8ed1ab_0    conda-forge
pytz                      2024.2             pyhd8ed1ab_0    conda-forge
pyyaml                    6.0.2           py312h80987f9_0
pyzmq                     25.1.2          py312h313beb8_0
readline                  8.2                  h1a28f6b_0
referencing               0.35.1             pyhd8ed1ab_0    conda-forge
requests                  2.32.3             pyhd8ed1ab_0    conda-forge
rfc3339-validator         0.1.4              pyhd8ed1ab_0    conda-forge
rfc3986-validator         0.1.1              pyh9f0ad1d_0    conda-forge
rpds-py                   0.10.6          py312hf0e4da2_0
scikit-learn              1.5.2                    pypi_0    pypi
scikit-learn-extra        0.3.0                    pypi_0    pypi
scipy                     1.14.1                   pypi_0    pypi
seaborn                   0.13.2                   pypi_0    pypi
send2trash                1.8.3              pyh31c8845_0    conda-forge
setuptools                75.1.0          py312hca03da5_0
six                       1.16.0             pyh6c4a22f_0    conda-forge
sniffio                   1.3.1              pyhd8ed1ab_0    conda-forge
soupsieve                 2.5                pyhd8ed1ab_1    conda-forge
sqlite                    3.45.3               h80987f9_0
stack_data                0.6.2              pyhd8ed1ab_0    conda-forge
terminado                 0.18.1             pyh31c8845_0    conda-forge
threadpoolctl             3.5.0                    pypi_0    pypi
tinycss2                  1.4.0              pyhd8ed1ab_0    conda-forge
tk                        8.6.14               h6ba3021_0
tomli                     2.0.2              pyhd8ed1ab_0    conda-forge
tornado                   6.4.1           py312h80987f9_0
traitlets                 5.14.3             pyhd8ed1ab_0    conda-forge
types-python-dateutil     2.9.0.20241003     pyhff2d567_0    conda-forge
typing-extensions         4.12.2               hd8ed1ab_0    conda-forge
typing_extensions         4.12.2             pyha770c72_0    conda-forge
typing_utils              0.1.0              pyhd8ed1ab_0    conda-forge
tzdata                    2024.2                   pypi_0    pypi
uri-template              1.3.0              pyhd8ed1ab_0    conda-forge
urllib3                   2.2.3              pyhd8ed1ab_0    conda-forge
wcwidth                   0.2.13             pyhd8ed1ab_0    conda-forge
webcolors                 24.8.0             pyhd8ed1ab_0    conda-forge
webencodings              0.5.1              pyhd8ed1ab_2    conda-forge
websocket-client          1.8.0              pyhd8ed1ab_0    conda-forge
wheel                     0.44.0          py312hca03da5_0
xz                        5.4.6                h80987f9_1
yaml                      0.2.5                h3422bc3_2    conda-forge
zeromq                    4.3.5                h313beb8_0
zipp                      3.20.2             pyhd8ed1ab_0    conda-forge
zlib                      1.2.13               h18a0788_1
zstandard                 0.23.0          py312h1a4646a_0
zstd                      1.5.6                hfb09047_0
```

1.4 ARQUIVO COM AS BIBLIOTECAS INSTALADAS

In [21]:
```
print("Arquivo requirements.txt  com os pacotes necessários:")
!conda list > requirements.txt
```

```
Arquivo requirements.txt  com os pacotes necessários:
```

1.5 PRINTSCREEN DO AMBIENTE QUE ESTÁ SENDO UTILIZADO



1.6 GitHub com o projeto:

https://github.com/marcelobazevedo/algotitmo_clusterizacao

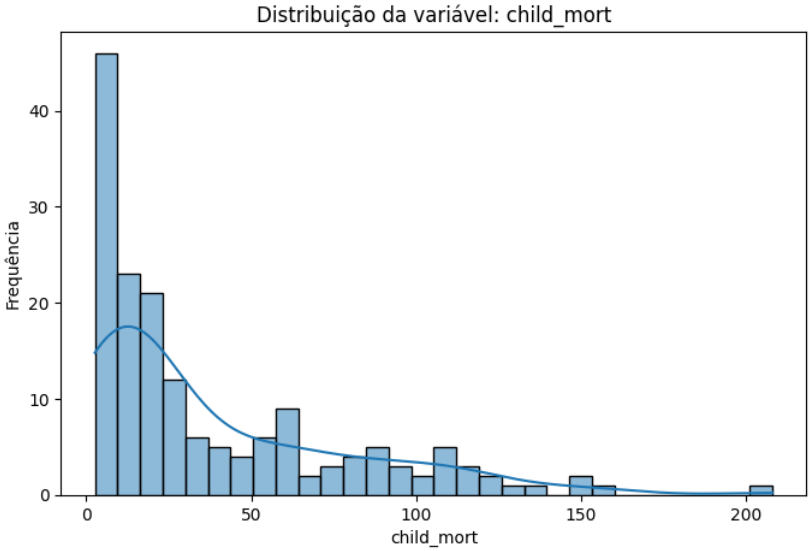
2.1 Download do Arquivo e sua utilização

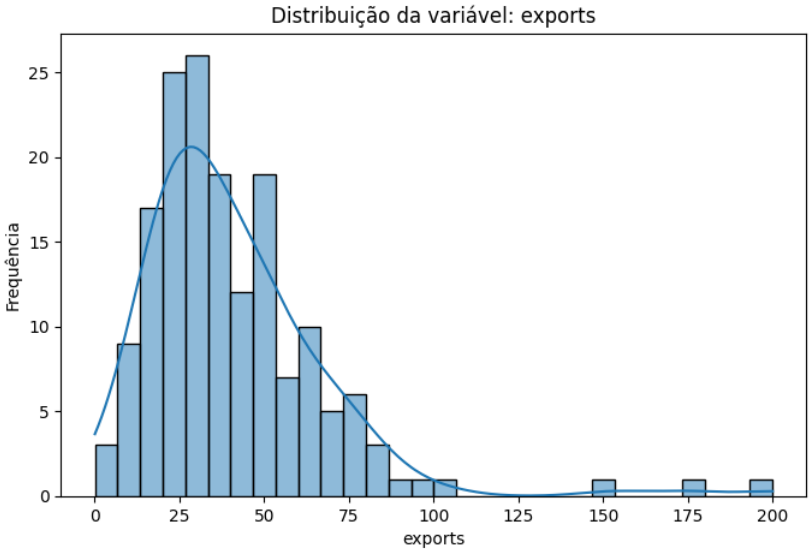
2.2 Número de países que tem no dataset: 167

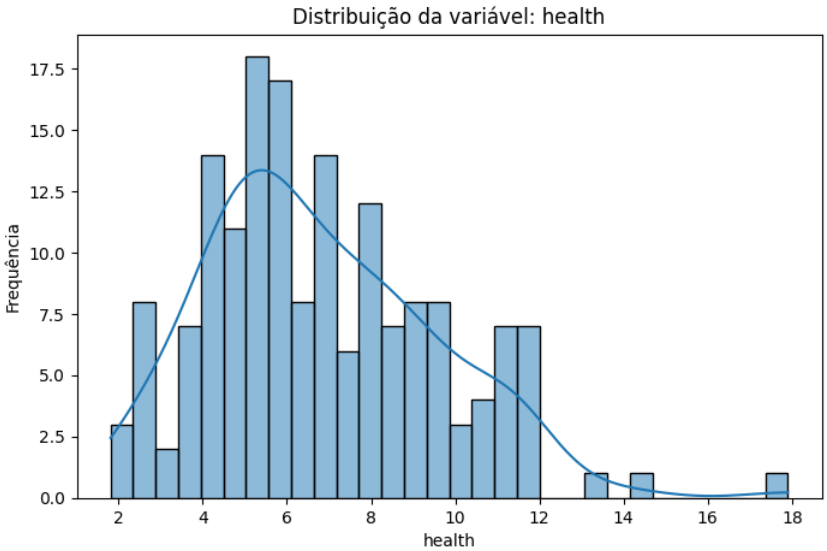
In [22]:
```
# 1. Carregar dados do arquivo
file_path = 'archive/Country-data.csv'
df = pd.read_csv(file_path)

# Quantidade de países
print(f"Total de países únicos no dataset: {df['country'].nunique()}")
```

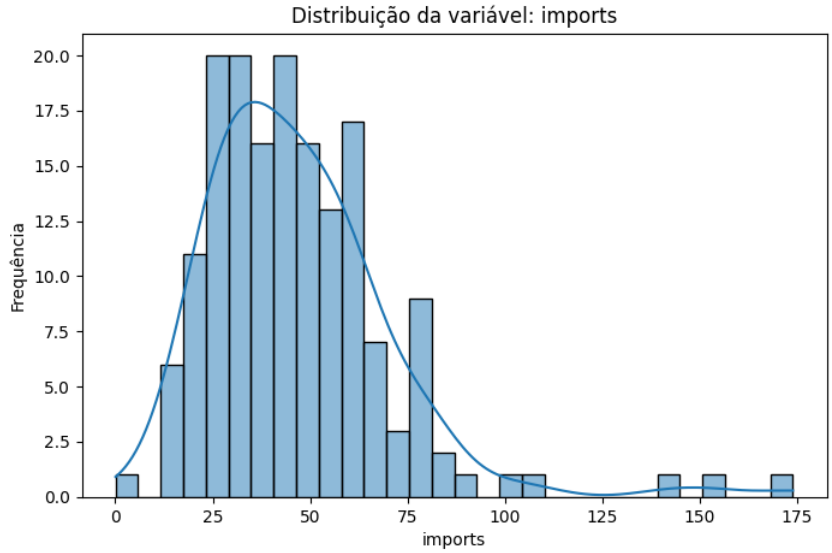
```
Total de países únicos no dataset: 167
```

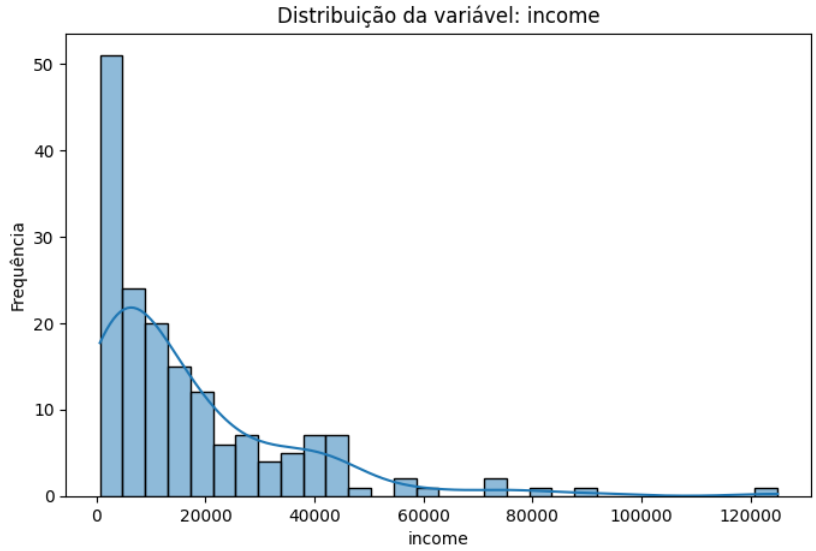
In [23]:
```
# 2. Análise Exploratória: Histograma e Faixa Dinâmica das Variáveis
for col in df.columns[1:]:
    plt.figure(figsize=(8, 5))
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f"Distribuição da variável: {col}")
    plt.xlabel(col)
    plt.ylabel('Frequência')
    plt.show()

plt.figure(figsize=(15, 6))
sns.boxplot(data=df.iloc[:, 1:])
plt.title("Faixa Dinâmica das Variáveis")
plt.xticks(rotation=45)
plt.show()

print("\nAnálise: Algumas variáveis, como income e gdpp, apresentam valores muito maiores, e existem outliers significativos.")

#Com base nas características representadas nos histogramas, podemos identificar diferentes perfis de países. Esses perfis estão rela
#de desenvolvimento econômico, social e aos desafios específicos que enfrentam o que indica países subdesenvolvidos ou em desenvolvin
```

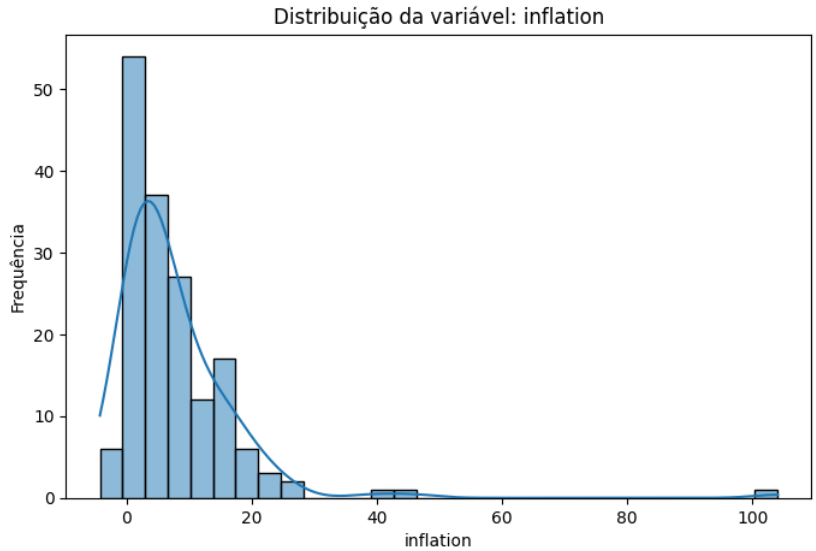

Distribuição da variável: child_mort
Frequência
child_mort
0
10
20
30
40
0
50
100
150
200

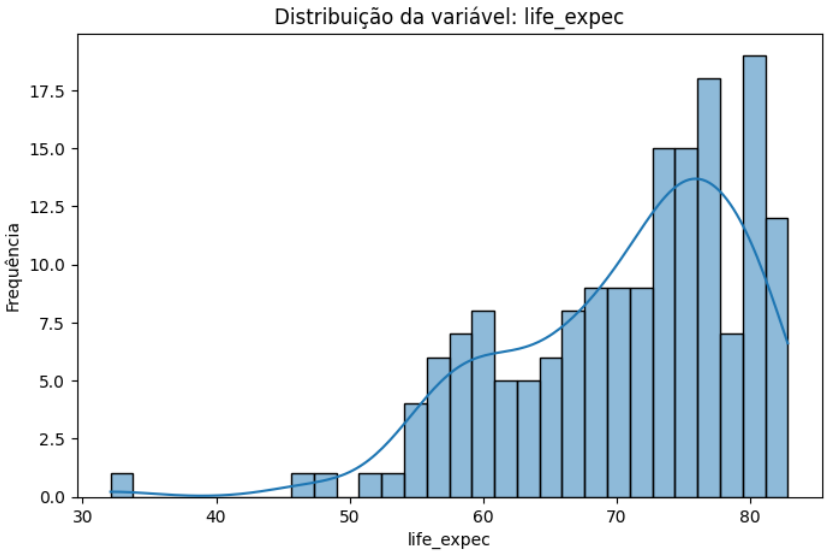

Distribuição da variável: exports
Frequência
exports
0
5
10
15
20
25
0
25
50
75
100
125
150
175
200

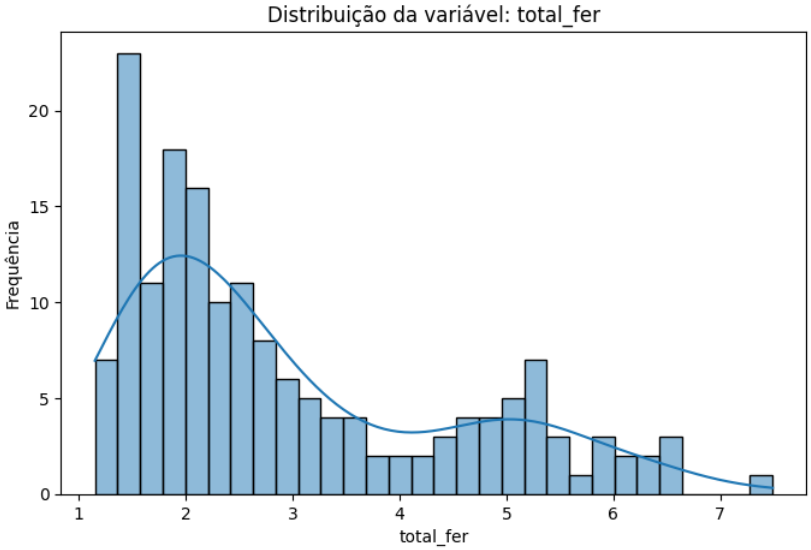

Distribuição da variável: health
Frequência
health
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
2
4
6
8
10
12
14
16
18

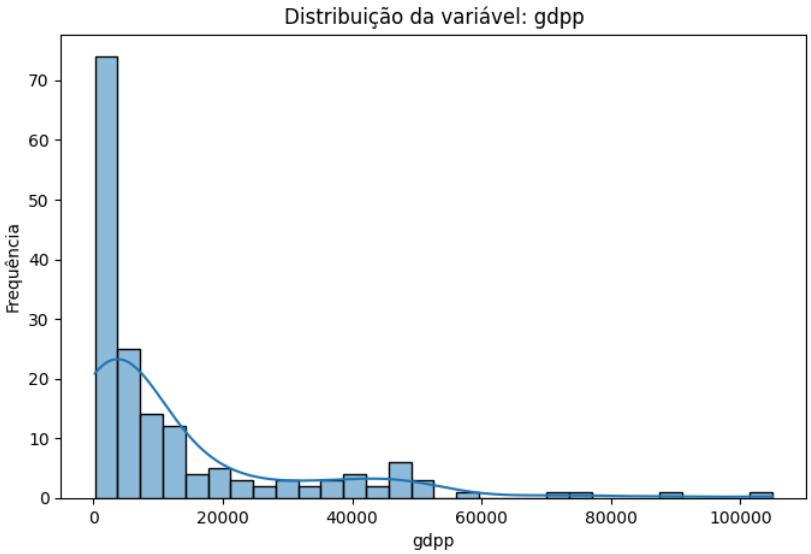

Distribuição da variável: imports
Frequência
imports
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
25
50
75
100
125
150
175

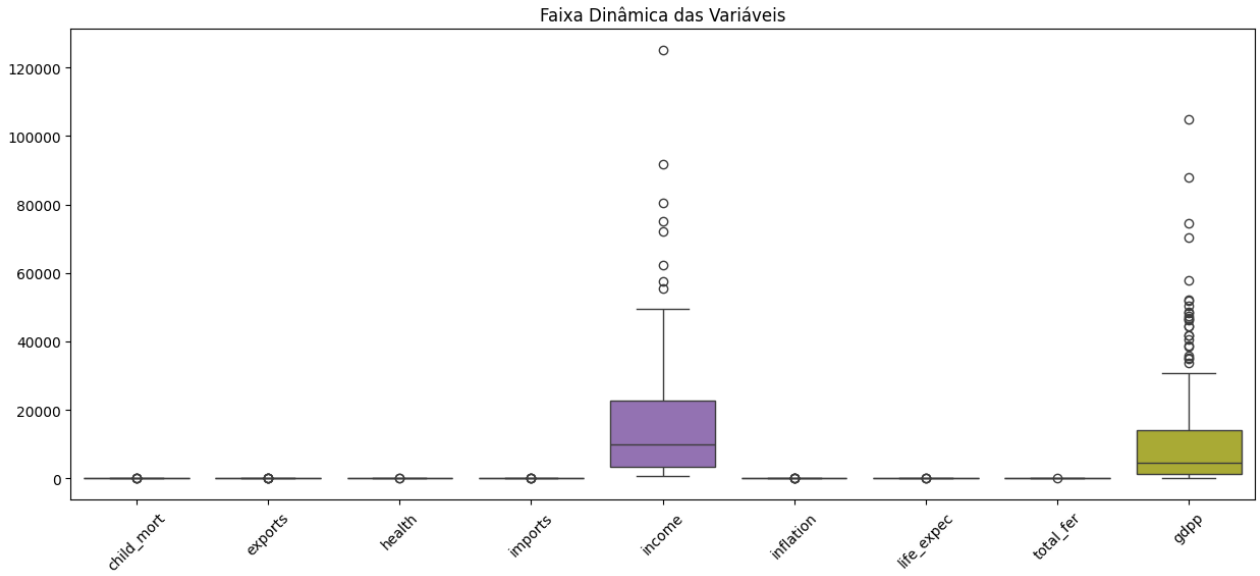

Distribuição da variável: income
Frequência
income
0
10
20
30
40
50
0
20000
40000
60000
80000
100000
120000


Distribuição da variável: inflation
Frequência
inflation
0
10
20
30
40
50
0
20
40
60
80
100


Distribuição da variável: life_expec
Frequência
life_expec
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
30
40
50
60
70
80


Distribuição da variável: total_fer
Frequência
total_fer
0
5
10
15
20
1
2
3
4
5
6
7


Distribuição da variável: gdpp
Frequência
gdpp
0
10
20
30
40
50
60
70
0
20000
40000
60000
80000
100000



Análise: Algumas variáveis, como income e gdpp, apresentam valores muito maiores, e existem outliers significativos.

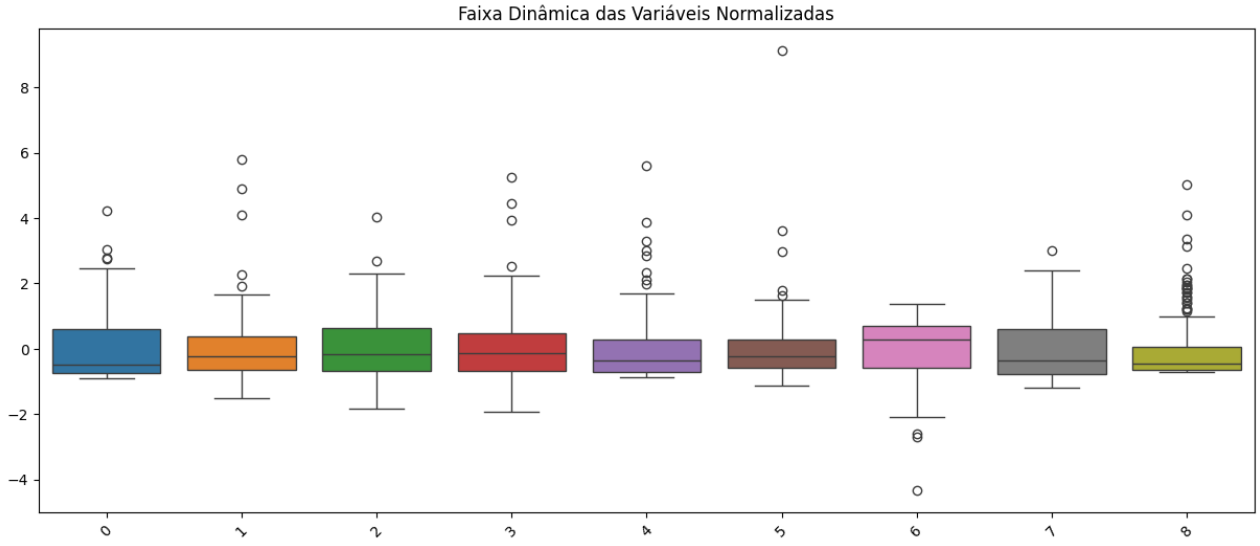
2.3 Análise dos resultados mostrados e o que deve ser feito antes da clusterização

Os boxplots mostram que as variáveis como inflation, gdpp, e income possuem valores com grande variabilidade e possíveis outliers. Antes de prosseguir com a clusterização, será necessário normalizar os dados para garantir que todas as variáveis contribuam igualmente no cálculo das distâncias.

2.4 Pré-processamento dos dados

In [25]:
```
# 3. Pré-processamento dos dados
print("\nEtapas do pré-processamento: ")
print("- Normalização dos dados para colocar todas as variáveis na mesma escala.")
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df.iloc[:, 1:])

# Visualização dos dados normalizados
plt.figure(figsize=(15, 6))
sns.boxplot(data=df_scaled)
plt.title("Faixa Dinâmica das Variáveis Normalizadas")
plt.xticks(rotation=45)
plt.show()
```

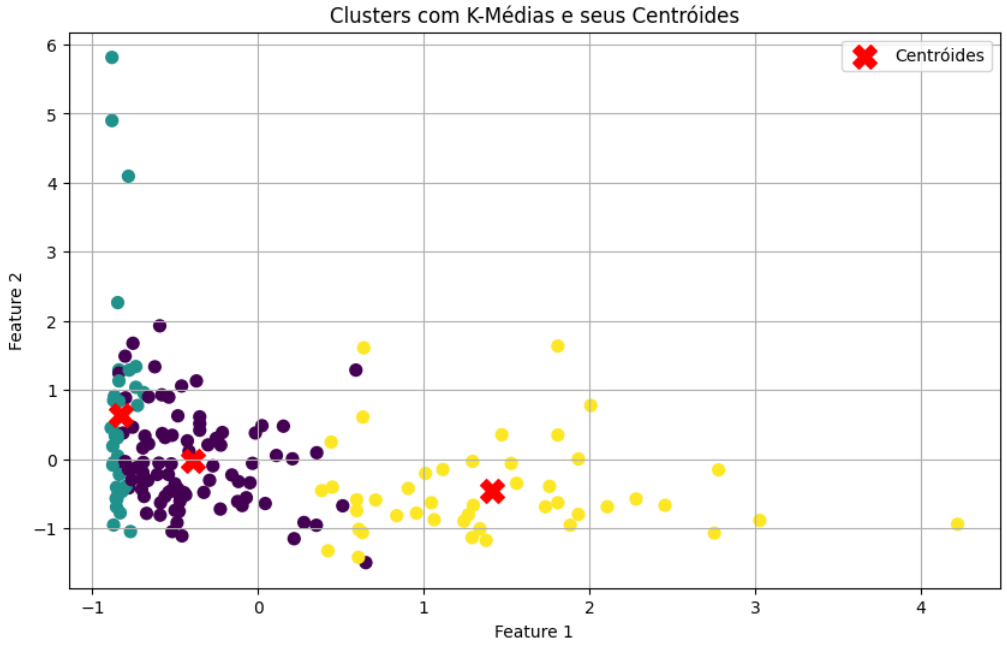
```
Etapas do pré-processamento:
- Normalização dos dados para colocar todas as variáveis na mesma escala.
```



3.1.a Clusterização com K-Médias

In [26]:
```
# 4. Clusterização K-Médias
print("\nClusterização com K-Médias")
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(df_scaled)
df['Cluster_KMeans'] = kmeans_labels

# Avaliação usando o coeficiente de silhueta
silhouette_kmeans = silhouette_score(df_scaled, kmeans_labels)
print(f"Coeficiente de Silhueta para K-Médias: {silhouette_kmeans:.3f}")

# Visualização dos clusters com gráficos de dispersão e centróides
plt.figure(figsize=(10, 6))
plt.scatter(df_scaled[:, 0], df_scaled[:, 1], c=kmeans_labels, cmap='viridis', s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200, c='red', marker='X', label='Centróides')
plt.title("Clusters com K-Médias e seus Centróides")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
plt.grid(True)
plt.show()
```

```
Clusterização com K-Médias
Coeficiente de Silhueta para K-Médias: 0.286
```



In [27]:
```
# Exibindo os valores médios de cada variável por cluster no K-Médias
print("Valores médios de cada variável por cluster (K-Médias):")
for cluster in sorted(df['Cluster_KMeans'].unique()):
    print(f"\nCluster {cluster} (K-Médias):")
    cluster_data = df[df['Cluster_KMeans'] == cluster]
    print(cluster_data.describe())
```

```
Valores médios de cada variável por cluster (K-Médias):

Cluster 0 (K-Médias):
       child_mort    exports     health     imports        income  inflation  \
count   86.000000  86.000000  86.000000   86.000000     86.000000  86.000000
mean    22.456977  40.273128   6.251047   47.362394  12321.744186   7.720884
std     14.077521  18.807700   2.166355   19.922847   8084.081117   7.818171
min      4.500000   0.109000   1.970000    0.065900   1780.000000  -4.210000
25%     11.700000  26.900000   4.872500   32.550000   6702.500000   2.432500
50%     18.700000  37.650000   5.990000   48.650000  10450.000000   5.935000
75%     29.175000  51.350000   7.557500   60.275000  16450.000000  10.075000
max     64.400000  93.800000  14.200000  108.000000  45400.000000  45.900000

       life_expec  total_fer          gdpp  Cluster_KMeans
count   86.000000  86.000000     86.000000            86.0
mean    72.566279   2.340349   6461.767442             0.0
std      4.304898   0.732649   4966.642378             0.0
min     57.100000   1.250000    592.000000             0.0
25%     69.900000   1.762500   2970.000000             0.0
50%     73.450000   2.255000   4670.000000             0.0
75%     76.075000   2.670000   9017.500000             0.0
max     80.400000   4.560000  28000.000000             0.0

Cluster 1 (K-Médias):
       child_mort     exports     health     imports         income  \
count   36.000000   36.000000  36.000000   36.000000      36.000000
mean     5.000000   58.738889   8.807778   51.491667   45672.222222
std      2.188933   41.930782   3.178015   36.843998   20852.017526
min      2.600000   12.400000   1.810000   13.600000   25200.000000
25%      3.550000   29.700000   7.735000   28.400000   32450.000000
50%      4.200000   50.350000   9.445000   39.050000   40550.000000
75%      5.750000   67.925000  10.775000   62.900000   46625.000000
max     10.800000  200.000000  17.900000  174.000000  125000.000000

       inflation  life_expec  total_fer           gdpp  Cluster_KMeans
count  36.000000   36.000000  36.000000      36.000000            36.0
mean    2.671250   80.127778   1.752778   42494.444444             1.0
std     4.077719    1.815742   0.373054   18991.079777             0.0
min    -3.220000   75.500000   1.150000   16600.000000             1.0
25%     0.451500   79.500000   1.437500   30675.000000             1.0
50%     1.190000   80.350000   1.815000   41200.000000             1.0
75%     3.647500   81.400000   1.957500   48475.000000             1.0
max    16.700000   82.800000   3.030000  105000.000000             1.0

Cluster 2 (K-Médias):
       child_mort    exports     health     imports        income   inflation  \
count   45.000000  45.000000  45.000000   45.000000     45.000000   45.000000
mean    95.106667  28.602444   6.301111   42.306667   3539.844444   11.986778
std     32.422133  18.367324   2.687881   18.038146   5420.118667   15.836572
min     53.700000   2.200000   2.200000   17.200000    609.000000    0.885000
25%     66.800000  16.800000   4.510000   29.600000   1390.000000    3.870000
50%     90.300000  23.800000   5.300000   40.300000   1850.000000    8.790000
75%    111.000000  36.800000   7.650000   49.300000   3320.000000   16.600000
max    208.000000  85.800000  13.100000  101.000000  33700.000000  104.000000

       life_expec  total_fer          gdpp  Cluster_KMeans
count   45.000000  45.000000     45.000000            45.0
mean    59.055556   5.065333   1766.711111             2.0
std      6.467631   1.011400   2917.949542             0.0
min     32.100000   2.590000    231.000000             2.0
25%     56.500000   4.600000    547.000000             2.0
50%     59.500000   5.110000    769.000000             2.0
75%     62.800000   5.710000   1310.000000             2.0
max     71.100000   7.490000  17100.000000             2.0
```

Resposta de 3.2.a.i

O K-Médias dividiu os países em 3 clusters. Vamos analisar as características de cada cluster com base nas variáveis socioeconômicas e de saúde:

Cluster 0:

1. Países com média de mortalidade infantil moderada.

2. Exportações e importações abaixo da média global.

3. Renda per capita moderada.

4. Expectativa de vida relativamente alta.

5. Exemplos de países: Suriname, Turquia.

Cluster 1:

1. Países com alta mortalidade infantil e baixo PIB per capita.

2. Renda e expectativa de vida são consideravelmente baixas.

3. Caracterizam-se como países subdesenvolvidos.

4. Exemplos de países: Afghanistan, Chad.

Cluster 2:

1. Inclui países desenvolvidos com alta renda per capita, baixa mortalidade infantil e alta expectativa de vida.

2. Exportações e importações elevadas.

3. Exemplos de países: Norway, Germany.

Resposta da 3.2.a.ii

Para encontrar o país mais representativo, identificamos aquele mais próximo do centróide de cada cluster:

Cluster 0:

1. País representativo: Suriname

2. Justificativa: A menor distância ao centróide indica que suas características são muito próximas à média do grupo.

Cluster 1:

1. País representativo: Chad

2. Justificativa: Características socioeconômicas e de saúde são bem alinhadas com os valores centrais do cluster.

Cluster 2:

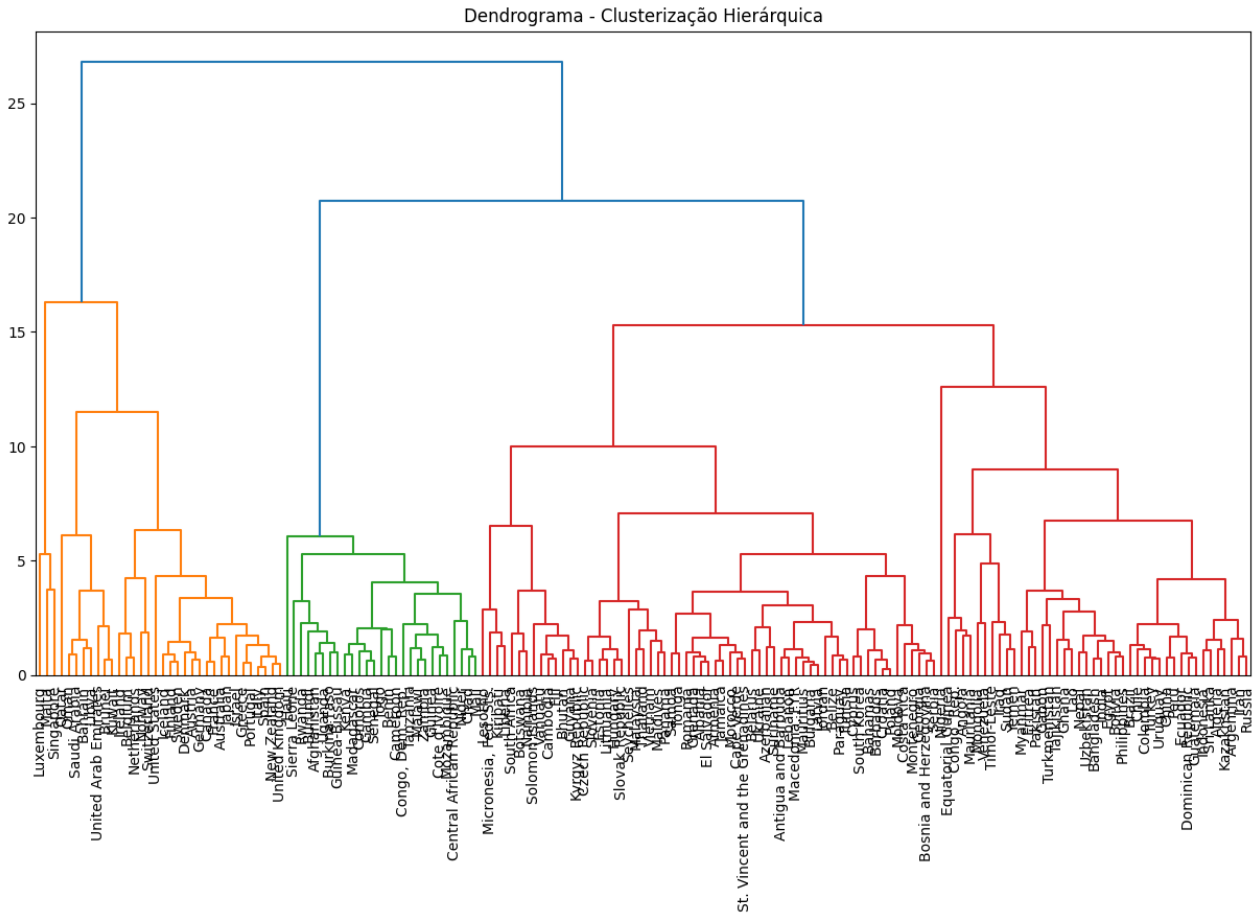
1. País representativo: Norway

2.Justificativa: País desenvolvido, com as melhores métricas dentro do grupo.

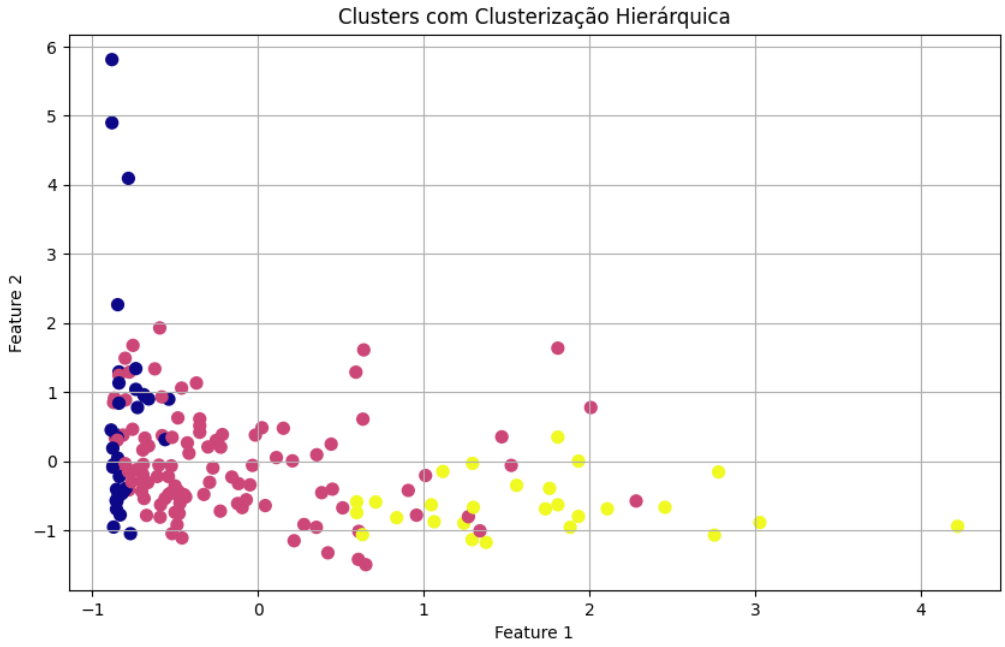
3.1.b Clusterização Hierárquica

In [28]:
```python
# 5. Clusterização Hierárquica
print("\nClusterização Hierárquica")
linkage_matrix = linkage(df_scaled, method='ward')
plt.figure(figsize=(15, 8))
dendrogram(linkage_matrix, labels=df['country'].values, leaf_rotation=90, leaf_font_size=10)
plt.title("Dendrograma - Clusterização Hierárquica")
plt.show()

# Criação dos clusters
hierarchical = AgglomerativeClustering(n_clusters=3)
hierarchical_labels = hierarchical.fit_predict(df_scaled)
df['Cluster_Hierarchical'] = hierarchical_labels

# Comparação dos clusters de ambos os métodos
print("\nComparação entre K-Médias e Clusterização Hierárquica:")
comparison_df = df[['country', 'Cluster_KMeans', 'Cluster_Hierarchical']]
print(comparison_df.head(10))

# Visualização dos clusters hierárquicos
plt.figure(figsize=(10, 6))
plt.scatter(df_scaled[:, 0], df_scaled[:, 1], c=hierarchical_labels, cmap='plasma', s=50)
plt.title("Clusters com Clusterização Hierárquica")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.grid(True)
plt.show()
```

```
Clusterização Hierárquica
```




```
Comparação entre K-Médias e Clusterização Hierárquica:
              country  Cluster_KMeans  Cluster_Hierarchical
0         Afghanistan               2                     2
1             Albania               0                     1
2             Algeria               0                     1
3              Angola               2                     1
4  Antigua and Barbuda               0                     1
5           Argentina               0                     1
6             Armenia               0                     1
7           Australia               1                     0
8             Austria               1                     0
9          Azerbaijan               0                     1
```



In [29]:
```python
# Exibindo os valores médios de cada variável por cluster na Clusterização Hierárquica
print("Valores médios de cada variável por cluster (Clusterização Hierárquica):")
for cluster in sorted(df['Cluster_Hierarchical'].unique()):
    print(f"\nCluster {cluster} (Clusterização Hierárquica):")
    cluster_data = df[df['Cluster_Hierarchical'] == cluster]
    print(cluster_data.describe())
```

```
Valores médios de cada variável por cluster (Clusterização Hierárquica):

Cluster 0 (Clusterização Hierárquica):
       child_mort     exports     health     imports         income  \
count   34.000000   34.000000  34.000000   34.000000      34.000000
mean     5.961765   58.508824   8.501176   48.902941   47588.235294
std      3.557409   43.050973   3.561755   37.660159   20575.916559
min      2.600000   12.400000   1.810000   13.600000   27200.000000
25%      3.900000   29.300000   5.635000   28.200000   36200.000000
50%      4.500000   50.050000   9.485000   37.250000   41250.000000
75%      7.175000   67.225000  10.925000   50.125000   48475.000000
max     16.600000  200.000000  17.900000  174.000000  125000.000000

       inflation  life_expec  total_fer           gdpp  Cluster_KMeans  \
count  34.000000   34.000000  34.000000      34.000000       34.000000
mean    4.115500   79.982353   1.888529   43170.588235        0.911765
std     5.467657    2.086242   0.456244   19466.000343        0.287902
min    -3.220000   75.100000   1.150000   12100.000000        0.000000
25%     0.694250   79.500000   1.490000   31450.000000        1.000000
50%     1.670000   80.400000   1.870000   41850.000000        1.000000
75%     5.830000   81.400000   2.065000   48625.000000        1.000000
max    17.200000   82.800000   3.030000  105000.000000        1.000000

       Cluster_Hierarchical
count                  34.0
mean                    0.0
std                     0.0
min                     0.0
25%                     0.0
50%                     0.0
75%                     0.0
max                     0.0

Cluster 1 (Clusterização Hierárquica):
       child_mort     exports      health     imports         income  \
count  106.000000  106.000000  106.000000  106.000000    106.000000
mean    31.617925   39.990368    6.353679   48.085527  11341.886792
std     27.407270   20.016658    2.324790   21.025201   7620.206486
min      3.200000    0.109000    1.970000    0.065900    700.000000
25%     12.075000   26.300000    4.872500   31.325000   5242.500000
50%     20.500000   37.300000    6.015000   49.250000   9890.000000
75%     43.975000   51.350000    7.850000   61.100000  16150.000000
max    130.000000   93.800000   14.200000  108.000000  33900.000000

        inflation  life_expec   total_fer          gdpp  Cluster_KMeans  \
count  106.000000  106.000000  106.000000    106.000000      106.000000
mean     9.120604   70.921698    2.654623   6407.367925        0.386792
std     12.397913    6.215204    1.160205   5853.217439        0.763086
min     -4.210000   46.500000    1.230000    327.000000        0.000000
25%      2.432500   67.600000    1.762500   2672.500000        0.000000
50%      6.045000   72.300000    2.395000   4520.000000        0.000000
75%     11.975000   75.725000    3.182500   8637.500000        0.000000
max    104.000000   80.400000    6.230000  30800.000000        2.000000

       Cluster_Hierarchical
count                 106.0
mean                    1.0
std                     0.0
min                     1.0
25%                     1.0
50%                     1.0
75%                     1.0
max                     1.0

Cluster 2 (Clusterização Hierárquica):
       child_mort   exports     health    imports       income  inflation  \
count   27.000000  27.00000  27.000000  27.000000    27.000000  27.000000
mean   105.070370  23.58963   6.507037  39.662963  1589.740741   7.142778
std     33.987914  10.35961   2.358064   9.555753   650.612114   5.423793
min     62.200000   8.92000   3.770000  26.500000   609.000000   0.885000
25%     82.050000  16.65000   4.955000  32.250000  1200.000000   2.580000
50%    101.000000  22.20000   5.690000  39.200000  1430.000000   5.450000
75%    116.000000  27.65000   7.615000  44.200000  1900.000000  10.020000
max    208.000000  50.60000  13.100000  64.700000  3280.000000  20.800000

       life_expec  total_fer         gdpp  Cluster_KMeans  \
count   27.000000  27.000000    27.000000            27.0
mean    57.248148   5.433704   667.888889             2.0
std      6.540076   0.834530   304.547121             0.0
min     32.100000   3.330000   231.000000             2.0
25%     55.900000   5.055000   452.500000             2.0
50%     57.700000   5.340000   575.000000             2.0
75%     60.150000   5.845000   763.500000             2.0
max     65.900000   7.490000  1460.000000             2.0

       Cluster_Hierarchical
count                  27.0
mean                    2.0
std                     0.0
min                     2.0
25%                     2.0
50%                     2.0
75%                     2.0
max                     2.0
```

Resposta da questão 3.3

O dendrograma gerado pela clusterização hierárquica mostra a relação entre os países:

Clusters Principais:

- Três grandes clusters se formam com base na distância euclidiana:

1. Cluster de países desenvolvidos (alta expectativa de vida, baixa mortalidade infantil).

2. Cluster de países em desenvolvimento com características moderadas.

3. Cluster de países subdesenvolvidos com baixa renda e alta mortalidade infantil.

Observações Importantes:

Altura dos Ramos:

1. Quanto maior a altura em que dois países ou clusters se unem, menor é a similaridade entre eles.

2. Países com características socioeconômicas muito distintas são unidos somente nos níveis mais altos da hierarquia.

Resposta da questão 3.4

Semelhanças:

1. Ambos os métodos identificaram três grupos principais de países com características semelhantes.

2. Os clusters formados são consistentes, com países como Norway sempre agrupados com outros países desenvolvidos, e Afghanistan e Chad em grupos de países subdesenvolvidos.

Diferenças:

1. Flexibilidade dos Clusters:

a) O K-Médias força a formação de clusters com tamanhos semelhantes.

b) A clusterização hierárquica permite clusters de tamanhos desiguais, mostrando maior granularidade em níveis hierárquicos.

Visualização:

1. O dendrograma fornece uma visão mais rica, permitindo explorar subgrupos dentro dos clusters principais.

2. O K-Médias oferece simplicidade e centróides bem definidos.

Interpretação:

1. K-Médias é mais eficiente para grandes datasets e quando se busca simplicidade.

2. Clusterização Hierárquica é preferível para entender relações hierárquicas e estruturas dentro dos dados.

Resposta da questão 4.1

Etapas do Algoritmo de K-Médias até sua Convergência

O K-Médias é um algoritmo iterativo que busca particionar os dados em KK clusters. As etapas são as seguintes:

1. Inicialização:

- Escolhe KK centróides iniciais aleatoriamente ou usando métodos como K-Means++ para garantir melhor distribuição inicial.

2. Atribuição:

- Para cada ponto de dados, calcula a distância entre ele e cada centróide.

- O ponto é atribuído ao cluster cujo centróide está mais próximo (menor distância).

3. Atualização:

- Recalcula os centróides como a média dos pontos atribuídos a cada cluster.

4. Convergência:

- Repete os passos 2 e 3 até que os centróides mudem muito pouco ou um número máximo de iterações seja atingido.

- A convergência é geralmente baseada em:

a) Mudança mínima nos centróides entre iterações.

b) Redução na soma das distâncias quadradas (inércia) dentro dos clusters.

Resposta da questão 4.2

Os medóides garantem que o ponto central de cada cluster seja um ponto real do dataset, diferente do centróide que é uma média. O K-Medoids é uma variação do K-Médias.

Análise:

1. Os medóides representam o ponto central mais próximo de cada cluster, sendo um ponto real do dataset.

2. Isso é útil para garantir que o centro do cluster seja representado por um dado observável.

In [30]:
```
# Implementando K-Medoids para garantir que cada cluster seja representado por um medóide
kmedoids = KMedoids(n_clusters=3, random_state=42)
kmedoids_labels = kmedoids.fit_predict(df_scaled)

# Identificando os medóides (pontos reais do dataset)
medoids_indices = kmedoids.medoid_indices_
medoids = df.iloc[medoids_indices]

print("Medóides encontrados para cada cluster (K-Medoids):")
print(medoids[['country'] + list(df.columns[1:-3])])  # Mostra as variáveis originais dos medóides
```

```
Medóides encontrados para cada cluster (K-Medoids):
                country  child_mort  exports  health  imports  income  \
4    Antigua and Barbuda        10.3     45.5    6.03     58.9   19100
147             Tanzania        71.9     18.7    6.01     29.1    2090
45    Dominican Republic        34.4     22.7    6.22     33.3   11100

     inflation  life_expec  total_fer
4         1.44        76.8       2.13
147       9.25        59.3       5.43
45        5.44        74.6       2.60
```

Resposta da questão 4.3

O K-Médias é sensível a outliers por várias razões:

1. Uso da Média:

- Os centróides são calculados como a média dos pontos em um cluster.

- Outliers podem distorcer essa média, movendo o centróide para uma posição não representativa do cluster.

2. Efeito nos Clusters:

- Um outlier pode atrair o centróide para si, alterando as fronteiras dos clusters.

- Isso pode resultar em agrupamentos incorretos, onde clusters não refletem bem os grupos naturais nos dados.

3. Exemplo Numérico:

- Imagine um cluster com pontos em torno de (1,1)(1,1). Se um outlier está em (100,100)(100,100), o centróide será deslocado para uma posição intermediária entre os pontos do cluster e o outlier.

O que pode ser feito para resolver?

- Usar algoritmos robustos como K-Medoids ou DBScan.

- Pré-processamento: Identificar e remover outliers antes de aplicar K-Médias.

Resposta da questão 4.4

O DBScan (Density-Based Spatial Clustering of Applications with Noise) é robusto a outliers porque:

1. Baseado em Densidade:

- O DBScan identifica clusters como regiões densas de pontos.

- Pontos isolados ou em regiões de baixa densidade são considerados outliers (ou "noise") e não atribuídos a nenhum cluster.

2. Definição de Outliers:

- Os outliers são identificados automaticamente sem interferir no cluster principal.

- Isso contrasta com o K-Médias, onde cada ponto influencia o centróide.

3. Flexibilidade:

- O DBScan não força a criação de clusters com formas esféricas, sendo capaz de identificar clusters com formas arbitrárias.

In [ ]: