```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import matplotlib as mpl
import numpy as np
from matplotlib.patches import PathPatch
from matplotlib.path import Path
```

```
# Configurações globais para o matplotlib
mpl.rcParams['font.family'] = 'Arial'  # Define a fonte como Arial
mpl.rcParams['axes.titlesize'] = 18    # Tamanho do título do gráfico
mpl.rcParams['axes.labelsize'] = 12    # Tamanho dos rótulos dos eixos
mpl.rcParams['xtick.labelsize'] = 10   # Tamanho dos rótulos do eixo x
mpl.rcParams['ytick.labelsize'] = 10   # Tamanho dos rótulos do eixo y

# Configurações globais para as legendas
mpl.rcParams['legend.fontsize'] = 14  # Tamanho da fonte da legenda
mpl.rcParams['legend.loc'] = 'upper left'  # Posição da legenda
mpl.rcParams['legend.frameon'] = True  # Exibir borda ao redor da legenda
mpl.rcParams['legend.title_fontsize'] = 16  # Tamanho da fonte do título da legenda
mpl.rcParams['legend.labelspacing'] = 1.2  # Espaçamento entre os itens da legenda
mpl.rcParams['legend.borderpad'] = 1.5  # Espaçamento ao redor da legenda
mpl.rcParams['legend.borderaxespad'] = 1.0  # Distância da legenda para o gráfico

# Configuração do estilo com seaborn e paleta de cores
sns.set(style="whitegrid")  # Define o estilo do gráfico como whitegrid
sns.set_palette("plasma")   # Define a paleta de cores padrão como "plasma"
```

```
# Carregar dataset
df = pd.read_csv('linkedin_postings_2023_2024.csv')
```

```
# Verificar tamanho do dataset|
df.shape
```

```
(123849, 31)
```

```
# Ver as primeiras linhas do dataset
df.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	med_salary
0	921716	Corcoran Sawyer Smith	Marketing Coordinator	Job descriptionA leading real estate firm in N...	20.0	HOURLY	Princeton, NJ	2774458.0	20.0	NaN
1	1829192	NaN	Mental Health Therapist/Counselor	At Aspen Therapy and Wellness , we are committ...	50.0	HOURLY	Fort Collins, CO	NaN	1.0	NaN

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	med_salary
2	10998357	The National Exemplar	Assitant Restaurant Manager	The National Exemplar is accepting application...	65000.0	YEARLY	Cincinnati, OH	64896719.0	8.0	NaN
3	23221523	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estates Associat...	Senior Associate Attorney - Elder Law / Trusts...	175000.0	YEARLY	New Hyde Park, NY	766262.0	16.0	NaN
4	35982263	NaN	Service Technician	Looking for HVAC service tech with experience ...	80000.0	YEARLY	Burlington, IA	NaN	3.0	NaN

5 rows × 31 columns

```
# Ver as colunas e tipos dados
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123849 entries, 0 to 123848
Data columns (total 31 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   job_id                      123849 non-null  int64  
 1   company_name                122130 non-null  object 
 2   title                       123849 non-null  object 
 3   description                 123842 non-null  object 
 4   max_salary                  29793 non-null   float64
 5   pay_period                  36073 non-null   object 
 6   location                    123849 non-null  object 
 7   company_id                  122132 non-null  float64
 8   views                       122160 non-null  float64
 9   med_salary                  6280 non-null    float64
 10  min_salary                  29793 non-null   float64
 11  formatted_work_type         123849 non-null  object 
 12  applies                     23320 non-null   float64
 13  original_listed_time        123849 non-null  float64
 14  remote_allowed              15246 non-null   float64
 15  job_posting_url             123849 non-null  object 
 16  application_url             87184 non-null   object 
 17  application_type            123849 non-null  object 
 18  expiry                      123849 non-null  float64
 19  closed_time                 1073 non-null    float64
 20  formatted_experience_level  94440 non-null   object 
 21  skills_desc                 2439 non-null    object 
 22  listed_time                 123849 non-null  float64
 23  posting_domain              83881 non-null   object 
 24  sponsored                   123849 non-null  int64  
 25  work_type                   123849 non-null  object 
 26  currency                    36073 non-null   object 
 27  compensation_type           36073 non-null   object 
 28  normalized_salary           36073 non-null   float64
 29  zip_code                    102977 non-null  float64
 30  fips                        96434 non-null   float64
dtypes: float64(14), int64(2), object(15)
memory usage: 29.3+ MB
```

```
# Limpeza de dados inicial, remover colunas sem relevância para o projeto.
df_cleaned = df.drop(columns=[
    'job_id',
    'company_name',
    'company_id',
    'original_listed_time',
```

```
    'application_type',
    'expiry',
    'closed_time',
    'pay_period',
    'job_posting_url',
    'application_url',
    'posting_domain',
    'sponsored',
    'currency',
    'compensation_type',
    'normalized_salary',
    'max_salary',
    'med_salary',
    'min_salary',
    'zip_code',
    'fips',
    'work_type'

])
```

```
# Verificar as colunas restantes
print(df_cleaned.columns)
```

```
Index(['title', 'description', 'location', 'views', 'formatted_work_type',
       'applies', 'remote_allowed', 'formatted_experience_level',
       'skills_desc', 'listed_time'],
      dtype='object')
```

```
# Contar valores ausentes em cada coluna
missing_values = df_cleaned.isnull().sum()
```

```
# Exibir apenas as colunas com valores ausentes
print(missing_values[missing_values > 0])
```

```
description                     7
views                        1689
applies                    100529
remote_allowed             108603
formatted_experience_level  29409
skills_desc                121410
dtype: int64
```

```
# Remover linha com valor ausente em 'description'
df_cleaned = df_cleaned.dropna(subset=['description'])
```

```
# Preencher valores ausentes em 'views' com 0
df_cleaned['views'] = df_cleaned['views'].fillna(0)
```

```
# Preencher valores ausentes em 'applies' com 0
df_cleaned['applies'] = df_cleaned['applies'].fillna(0)
```

```
# Preencher valores ausentes em 'formatted_experience_level' com "não especificado"
df_cleaned['formatted_experience_level'] = df_cleaned['formatted_experience_level'].fillna('Não especificado')
```

```
# Preencher valores ausentes em 'skills_desc' com "não especificado"
df_cleaned['skills_desc'] = df_cleaned['skills_desc'].fillna('Habilidades não especificadas')
```

```
# Verificar novamente há valores ausentes
missing_values_after = df_cleaned.isnull().sum()
print(missing_values_after[missing_values_after > 0])
```

```
remote_allowed    108599
dtype: int64
```

```
# Verificar linhas duplicadas
num_duplicates = df_cleaned.duplicated().sum()
print(f"Número de linhas duplicadas: {num_duplicates}")
```

```
Número de linhas duplicadas: 463
```

```
# Ver linhas duplicadas:
duplicates = df_cleaned[df_cleaned.duplicated(keep=False)]
display(duplicates)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	title	description	location	views	formatted_work_type	applies	remote_allowed	formatted_experience_level
1133	Assistant Claims Examiner (Onsite Irving, Texas)	Great companies need great teams to propel the...	Irving, TX	6.0	Full-time	0.0	NaN	Mid-Senior level
1537	Assistant Claims Examiner (Onsite Irving, Texas)	Great companies need great teams to propel the...	Irving, TX	6.0	Full-time	0.0	NaN	Mid-Senior level
2159	Laborer II-Drill Operator	Are you ready to join the A-Team and become a ...	Colorado Springs, CO	6.0	Full-time	0.0	NaN	Entry level
2284	Laborer II-Drill Operator	Are you ready to join the A-Team and become a ...	Colorado Springs, CO	6.0	Full-time	0.0	NaN	Entry level
3614	Café & Dining Aide, Extra On Call, Rotating	Employment Type\n\nPart time\n\nShift\n\nRotat...	Fresno, CA	5.0	Part-time	0.0	NaN	Entry level
...	...	...	...	...	...	...	...	...
123725	RN Progressive Care Unit	Onyx Health Care Staffing is seeking a qualifi...	Nashville, TN	4.0	Full-time	0.0	NaN	Mid-Senior level
123764	RN Progressive Care Unit	Onyx Health Care Staffing is seeking a qualifi...	Asheville, NC	5.0	Full-time	0.0	NaN	Mid-Senior level

	title	description	location	views	formatted_work_type	applies	remote_allowed	formatted_experience_level
123765	RN Progressive Care Unit	Onyx Health Care Staffing is seeking a qualifi...	Asheville, NC	5.0	Full-time	0.0	NaN	Mid-Senior level
123766	RN Progressive Care Unit	Onyx Health Care Staffing is seeking a qualifi...	Asheville, NC	5.0	Full-time	0.0	NaN	Mid-Senior level
123767	RN Progressive Care Unit	Onyx Health Care Staffing is seeking a qualifi...	Nashville, TN	4.0	Full-time	0.0	NaN	Mid-Senior level

833 rows × 10 columns

```
# Remover duplicatas com base nas colunas 'title', 'description' e 'location'
df_cleaned = df_cleaned.drop_duplicates(subset=['title', 'description', 'location'], keep='first')
```

```
# Verificar se ainda há duplicatas
duplicated_rows = df_cleaned[df_cleaned.duplicated(subset=['title', 'description', 'location'])]

# Exibir o número de duplicatas restantes
print(f"Número de duplicatas restantes: {duplicated_rows.shape[0]}")
```

```
Número de duplicatas restantes: 0
```

```
# Se 'listed_time' não estiver no tipo datetime, faça a conversão
if not pd.api.types.is_datetime64_any_dtype(df_cleaned['listed_time']):
    # Ajustar 'listed_time' de milissegundos
    df_cleaned['listed_time'] = df_cleaned['listed_time'] / 1000

    # Converter 'listed_time' para datetime
    df_cleaned['listed_time'] = pd.to_datetime(df_cleaned['listed_time'], unit='s', origin='unix')
```

```
# Verificar tamanho do df limpo
df_cleaned.shape
```

```
(119006, 10)
```

```
#########################################################################
################################## EDA ##################################
#########################################################################
```

```
# Extrair o mês de publicação das vagas
df_cleaned['month'] = df_cleaned['listed_time'].dt.to_period('M')

# Criar coluna do mês
df_cleaned['month'] = df_cleaned['listed_time'].dt.to_period('M')
```

```
# Agrupar dados pelo mês e contar número de vagas
monthly_counts = df_cleaned.groupby('month').size()
```

```
print(monthly_counts)
```

```
month
2024-03         1
2024-04    119005
Freq: M, dtype: int64
```

```
# Filtrar para manter apenas as vagas de abril de 2024
df_cleaned = df_cleaned.loc[df_cleaned['month'] == '2024-04']
```

```
print(df_cleaned['listed_time'].unique())
```

```
<DatetimeArray>
['2024-04-17 23:45:08', '2024-04-11 17:51:27', '2024-04-16 14:26:54',
 '2024-04-12 04:23:32', '2024-04-18 14:52:23', '2024-04-18 16:01:39',
 '2024-04-11 18:43:39', '2024-04-06 22:44:12', '2024-04-05 20:21:40',
 '2024-04-07 02:12:35',
 ...
 '2024-04-20 00:23:13', '2024-04-20 00:26:30', '2024-04-19 23:52:56',
 '2024-04-19 23:55:40', '2024-04-20 00:00:12', '2024-04-20 00:25:51',
 '2024-04-20 00:20:31', '2024-04-20 00:00:23', '2024-04-20 00:23:52',
 '2024-04-20 00:23:36']
Length: 52176, dtype: datetime64[ns]
```

```
# O resultado não é o que eu esperava, pois os posts de vagas referem-se apenas a um único mês de 2024. Sendo assim
# vou mudar o objetivo inicial do projeto para analisar a distribuição de vagas por tipo de trabalho, formato,
localização,
# por nível de experiencia requerido, correlação entre visualizações e aplicações e tendências em relação as
habilidades requeridas.
```

```
################################################################################
############## 1 - Análise de Vagas por Tipo de Contrato ##############
################################################################################
```

```
# CONTEXTO:
# Esta análise tem como objetivo entender a distribuição das vagas de emprego de acordo com os tipos de contrato
disponíveis.
# Sabemos que os tipos de contrato podem influenciar a estabilidade do cargo, a flexibilidade de horário e o perfil
do trabalhador.
# Ao analisar essas informações, podemos identificar quais formas de trabalho são mais predominantes no mercado.
```

```
work_type_counts = df_cleaned['formatted_work_type'].value_counts()
```

```
# Contar a quantidade de vagas por tipo de trabalho
work_type_counts = df_cleaned['formatted_work_type'].value_counts()

# Converter para DataFrame para melhor visualização
work_type_counts_df = work_type_counts.reset_index()
work_type_counts_df.columns = ['Tipo de Trabalho', 'Número de Vagas']

# Exibir a tabela
print(work_type_counts_df)
```

```
  Tipo de Trabalho  Número de Vagas
0        Full-time            86230
1         Contract             9602
2        Part-time             8925
3        Temporary             1079
4       Internship              759
```

```
5        Volunteer             426
6            Other             387
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import os
import matplotlib as mpl
import numpy as np
from matplotlib.patches import PathPatch
from matplotlib.path import Path

# Configurações globais para o matplotlib
mpl.rcParams['font.family'] = 'Arial'  # Define a fonte como Arial
mpl.rcParams['axes.titlesize'] = 18    # Tamanho do título do gráfico
mpl.rcParams['axes.labelsize'] = 12    # Tamanho dos rótulos dos eixos
mpl.rcParams['xtick.labelsize'] = 10   # Tamanho dos rótulos do eixo x
mpl.rcParams['ytick.labelsize'] = 10   # Tamanho dos rótulos do eixo

# Configuração do estilo com seaborn e paleta de cores
sns.set(style="whitegrid")  # Define o estilo do gráfico como whitegrid
sns.set_palette("plasma")   # Define a paleta de cores padrão como "plasma"

# Criar o gráfico de barras horizontais
plt.figure(figsize=(14, 8))  # Aumentando o tamanho da figura
ax = sns.barplot(
    x='Número de Vagas',
    y='Tipo de Trabalho',
    data=work_type_counts_df,
    errorbar=None,  # Substituir o `ci` por `errorbar=None` para evitar o aviso de depreciação
)

# Aplicar um gradiente de cor usando a paleta plasma
norm = mpl.colors.Normalize(vmin=0, vmax=len(ax.patches))
cmap = plt.get_cmap("plasma")

# Função para arredondar as barras
def round_bars(patch, color, rounding_size=10):
    # Obter a posição da barra
    x0, y0 = patch.get_xy()  # Posição de origem
    width, height = patch.get_width(), patch.get_height()  # Largura e altura da barra

    # Definir o caminho para a barra arredondada
    verts = [
        (x0 + rounding_size, y0),  # Começo arredondado
        (x0 + width - rounding_size, y0),
        (x0 + width, y0 + rounding_size),  # Arredondamento da borda superior
        (x0 + width, y0 + height - rounding_size),
        (x0 + width - rounding_size, y0 + height),
        (x0 + rounding_size, y0 + height),
        (x0, y0 + height - rounding_size),  # Arredondamento da borda inferior
        (x0, y0 + rounding_size)
    ]

    # Fechar o caminho (voltar ao início)
    verts.append(verts[0])

    # Criar o Path e o Patch
    path = Path(verts, closed=True)
    patch.set_facecolor(color)  # Aplica a cor gradiente
    patch.set_edgecolor("white")  # Define a cor da borda como branca
    patch.set_linewidth(2)  # Define a espessura da borda

    # Substituir o patch original pelo novo com bordas arredondadas
    path_patch = PathPatch(path, facecolor=color, edgecolor="white", lw=2)
    ax.add_patch(path_patch)
    patch.remove()  # Remove o patch original

# Arredondar as barras e aplicar o gradiente de cores
for i, patch in enumerate(ax.patches):
    # Calculando o valor para a cor gradiente
    color = cmap(norm(i))  # Aplica um gradiente na cor da barra
    round_bars(patch, color)

# Ajustar o visual do gráfico
plt.title('Distribuição de Vagas por Tipo de Contrato', fontsize=18, fontweight='bold', color='black', pad=30)
plt.xlabel('Número de Vagas', fontsize=14, color='darkred', labelpad=20)  # Cor personalizada e aumento do
espaçamento
plt.ylabel('Tipo de Trabalho', fontsize=14, color='darkblue', labelpad=20)  # Cor personalizada e aumento do
```

```
espaçamento

# Remover as linhas do fundo (grid e bordas)
ax.grid(False)  # Remove o grid do gráfico
for spine in ax.spines.values():
    spine.set_visible(False)  # Remove as bordas (spines) do gráfico

# Remover a legenda desnecessária
plt.legend([], [], frameon=False)

# Melhorar o layout para evitar cortes
plt.tight_layout()

# Gravar gráfico em png na pasta do website
output_dir = '/Users/elisadrummond/projeto-tendencias-empregos/static/graphs'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Salvar o gráfico
graph_path = os.path.join(output_dir, 'grafico1_tipo_contrato_moderno.png')
plt.savefig(graph_path, dpi=300, bbox_inches='tight')  # Aumentar a resolução para maior qualidade

# Exibir o gráfico
plt.show()

# Fechar a figura após salvar
plt.close()

# Mensagem de confirmação
print(f"Gráfico salvo em: {graph_path}")
```

```
Gráfico salvo em: /Users/elisadrummond/projeto-tendencias-empregos/static/graphs/grafico1_tipo_contrato_moderno.png
```

```
# INSIGHTS: O gráfico revela vagas mais comuns são para Full-time, seguidas por Contract e Part-time. Este padrão
indica que as empresas
# preferem ofertas de trabalho com contratos de tempo integral ou por prazo determinado, o que pode refletir uma
demanda maior por
# estabilidade e comprometimento de longo prazo. As vagas para Internship, Volunteer, e Temporary são
significativamente menores,
# sugerindo que o mercado oferece poucas oportunidades temporárias ou de estágio, o que pode limitar as opções para
iniciantes ou
# aqueles que buscam experiência em áreas específicas.
```

```
################################################################################
############ 2 - Análise de Vagas por Formato de Trabalho  #############
################################################################################
```

```
# CONTEXTO:
# A coluna remote_allowed descreve a possibilidade de trabalho remoto nas vagas de emprego.
# "Não Permitido" indica que o trabalho remoto não é uma opção para a vaga, enquanto "Permitido" significa que o
trabalho remoto é permitido.
```

```
# Exibir os valores únicos presentes na coluna 'remote_allowed'
formats = df_cleaned['remote_allowed'].unique()

# Mostrar os formatos
print("Formatos disponíveis em 'remote_allowed':")
print(formats)
```

```
Formatos disponíveis em 'remote_allowed':
[nan  1.]
```

```
# Contar a ocorrência de cada valor na coluna 'remote_allowed'
remote_allowed_counts = df_cleaned['remote_allowed'].value_counts(dropna=False)

# Exibir os resultados
print("Contagem de formatos em 'remote_allowed':")
print(remote_allowed_counts)
```

```
Contagem de formatos em 'remote_allowed':
remote_allowed
NaN    104479
1.0     14526
Name: count, dtype: int64
```

```
# Substituir 1 por 'Permitido' e NaN por 'Não Permitido'
df_cleaned['remote_allowed'] = df_cleaned['remote_allowed'].replace({1.0: 'Permitido', np.nan: 'Não Permitido'})

# Contar a quantidade de vagas por categoria
remote_allowed_counts = df_cleaned['remote_allowed'].value_counts()

# Converter para um DataFrame para melhor visualização
remote_allowed_table = remote_allowed_counts.reset_index()
remote_allowed_table.columns = ['Trabalho Remoto', 'Número de Vagas']

# Exibir a tabela
print(remote_allowed_table)
```

```
  Trabalho Remoto  Número de Vagas
0   Não Permitido           104479
1       Permitido            14526
```

```
# Criar o gráfico de pizza no padrão solicitado
plt.figure(figsize=(10, 6))  # Tamanho proporcional

# Normalizar os índices para aplicar o gradiente
norm = mpl.colors.Normalize(vmin=0, vmax=len(remote_allowed_counts))
cmap = plt.get_cmap("plasma")

# Preparar dados para o gráfico de pizza
patches, texts, autotexts = plt.pie(
    remote_allowed_counts,
    labels=None,  # Remove as legendas padrão
    autopct='%1.1f%%',
    startangle=90,
    colors=[cmap(norm(i)) for i in range(len(remote_allowed_counts))],
    wedgeprops=dict(edgecolor='white', linewidth=1.5)  # Bordas brancas mais sutis
)

# Criar a legenda separada
plt.legend(
    labels=remote_allowed_counts.index,
    loc='center left',  # Posicionamento à esquerda
    bbox_to_anchor=(1, 0.5),  # Afastar a legenda do gráfico
    fontsize=12  # Tamanho da fonte consistente com o gráfico de barras
)

# Ajustar os autotextos (percentuais dentro do gráfico)
for autotext in autotexts:
    autotext.set_fontsize(11)
    autotext.set_color('white')  # Percentuais em branco para contraste

# Ajustar o título do gráfico
plt.title('Permissão de Trabalho Remoto', fontsize=16, fontweight='bold', color='black', pad=20)

# Melhorar o layout para evitar cortes
plt.tight_layout()

# Gravar gráfico em png na pasta do website
output_dir = '/Users/elisadrummond/projeto-tendencias-empregos/static/graphs'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
```

```
# Salvar o gráfico
graph_path = os.path.join(output_dir, 'grafico2_remoto_permitido.png')
plt.savefig(graph_path, dpi=300, bbox_inches='tight')  # Aumentar a resolução para maior qualidade

# Exibir o gráfico
plt.show()

# Fechar a figura após salvar
plt.close()

# Mensagem de confirmação
print(f"Gráfico salvo em: {graph_path}")
```

```
Gráfico salvo em: /Users/elisadrummond/projeto-tendencias-empregos/static/graphs/grafico2_remoto_permitido.png
```

```
# INSIGHTS:
# A discrepância entre as vagas com "trabalho remoto permitido" e "não permitido" sugere que muitas empresas ainda
estão em processo
# de adaptação ao modelo remoto, ou até mesmo limitam a flexibilidade para certas posições.
# Isso pode refletir uma tendência que, embora crescente, ainda não é universal no mercado de trabalho, especialmente
em áreas que
# exigem presença física ou operações no local de trabalho.
```

```
################################################################################
################ 3 - Análise de Vagas por Localização  #################
################################################################################
```

```
# CONTEXTO:
# A análise examina a distribuição de vagas agrupadas por estados norte-americanos.
# Com esses dados, é possível identificar os estados que concentram mais oportunidades de trabalho e avaliar sua
participação relativa
# no total de vagas
```

```
# Contar a quantidade de vagas por localização
location_counts = df_cleaned['location'].value_counts()

# Converter para DataFrame para melhor visualização
location_counts_df = location_counts.reset_index()
location_counts_df.columns = ['Localização', 'Número de Vagas']

# Exibir a tabela
print(location_counts_df)
```

```
               Localização  Número de Vagas
0            United States             7818
1             New York, NY             2688
2              Chicago, IL             1799
3              Houston, TX             1678
4               Dallas, TX             1340
...                    ...              ...
8521       Kent County, RI                1
8522           Cusseta, AL                1
8523  Greater Columbus Area               1
8524          Lovelock, NV                1
8525    Carroll County, MD                1

[8526 rows x 2 columns]
```

```
# Explorar melhor a localização para ver se há mais limpeza a ser feita

# Extrair o estado da coluna location
df_cleaned['estado'] = df_cleaned['location'].str.extract(r',\s*([A-Z]{2})')[0]

# Contar a quantidade de vagas por estado
state_counts = df_cleaned['estado'].value_counts()

# Converter para DataFrame para melhor visualização
state_counts_df = state_counts.reset_index()
state_counts_df.columns = ['Estado', 'Número de Vagas']

# Exibir a tabela
print(state_counts_df)
```

```
   Estado  Número de Vagas
0      CA            11197
1      TX             9858
2      NY             5836
3      FL             5709
4      NC             4749
5      IL             4323
6      PA             3909
7      VA             3522
8      MA             3357
9      GA             3349
10     OH             3296
11     NJ             3178
12     MI             2665
13     WA             2638
14     AZ             2395
15     CO             2258
16     MD             1916
17     TN             1832
18     MN             1810
19     MO             1803
20     WI             1797
21     IN             1737
22     SC             1429
23     KY             1140
24     OR             1138
25     CT             1136
26     LA              954
27     IA              949
28     UT              946
29     AL              941
30     DC              859
31     NV              857
32     KS              855
33     OK              769
34     AR              634
35     NE              570
36     NH              545
37     NM              481
38     ID              410
39     HI              409
40     WV              396
41     MS              379
42     ME              364
43     DE              312
44     RI              301
45     MT              232
46     ND              232
47     AK              204
48     VT              175
49     SD              159
50     WY              118
51     ON                1
52     QC                1
```

```
# Somar o total de vagas
total_vagas = state_counts_df['Número de Vagas'].sum()
```

```
# Exibir o total de vagas
print(f'Total de Vagas: {total_vagas}')

# Calcular o percentual para cada estado
state_counts_df['Percentual'] = (state_counts_df['Número de Vagas'] / total_vagas) * 100

# Exibir a tabela com percentuais
print(state_counts_df)
```

```
Total de Vagas: 101030
   Estado  Número de Vagas  Percentual
0      CA            11197   11.082847
1      TX             9858    9.757498
2      NY             5836    5.776502
3      FL             5709    5.650797
4      NC             4749    4.700584
5      IL             4323    4.278927
6      PA             3909    3.869148
7      VA             3522    3.486093
8      MA             3357    3.322775
9      GA             3349    3.314857
10     OH             3296    3.262397
11     NJ             3178    3.145600
12     MI             2665    2.637830
13     WA             2638    2.611106
14     AZ             2395    2.370583
15     CO             2258    2.234980
16     MD             1916    1.896466
17     TN             1832    1.813323
18     MN             1810    1.791547
19     MO             1803    1.784618
20     WI             1797    1.778680
21     IN             1737    1.719291
22     SC             1429    1.414431
23     KY             1140    1.128378
24     OR             1138    1.126398
25     CT             1136    1.124418
26     LA              954    0.944274
27     IA              949    0.939325
28     UT              946    0.936356
29     AL              941    0.931407
30     DC              859    0.850243
31     NV              857    0.848263
32     KS              855    0.846283
33     OK              769    0.761160
34     AR              634    0.627536
35     NE              570    0.564189
36     NH              545    0.539444
37     NM              481    0.476096
38     ID              410    0.405820
39     HI              409    0.404830
40     WV              396    0.391963
41     MS              379    0.375136
42     ME              364    0.360289
43     DE              312    0.308819
44     RI              301    0.297931
45     MT              232    0.229635
46     ND              232    0.229635
47     AK              204    0.201920
48     VT              175    0.173216
49     SD              159    0.157379
50     WY              118    0.116797
51     ON                1    0.000990
52     QC                1    0.000990
```

```
# O número de vagas ficou diferente do número de linhas do dataset.

#Contar quantas linhas têm localização válida
localizacao_valida = df_cleaned[df_cleaned['location'].notnull()]

# Total de linhas com localização válida
total_localizacao_valida = localizacao_valida.shape[0]
```

```
print(f'Total de linhas com localização válida: {total_localizacao_valida}')
```

```
Total de linhas com localização válida: 119005
```

```
# Comparar total de linhas no DataFrame com o total de vagas por estado
print(f'Total de linhas no DataFrame: {df_cleaned.shape[0]}')
print(f'Total de vagas por estado: {state_counts_df["Número de Vagas"].sum()}')
```

```
Total de linhas no DataFrame: 119005
Total de vagas por estado: 101030
```

```
# Verificar quais registros não têm uma localização válida
registros_sem_localizacao = df_cleaned[df_cleaned['location'].isnull() | (df_cleaned['location'] == '')]

# Exibir alguns desses registros
print(registros_sem_localizacao)
print(f'Número de registros sem localização: {registros_sem_localizacao.shape[0]}')
```

```
Empty DataFrame
Columns: [title, description, location, views, formatted_work_type, applies, remote_allowed,
formatted_experience_level, skills_desc, listed_time, month, estado]
Index: []
Número de registros sem localização: 0
```

```
# Verificar a presença de valores nulos em todo o DataFrame
print(df_cleaned.isnull().sum())
```

```
title                           0
description                     0
location                        0
views                           0
formatted_work_type             0
applies                         0
remote_allowed                  0
formatted_experience_level      0
skills_desc                     0
listed_time                     0
month                           0
estado                      17975
dtype: int64
```

```
# Identificado que a coluna que possui erros: Estado.

# Exibir registros que não têm estado
registros_sem_estado = df_cleaned[df_cleaned['estado'].isnull() | (df_cleaned['estado'] == '')]

# Exibir alguns desses registros
print(registros_sem_estado[['location']])  # Apenas a coluna location para foco
print(f'Número de registros sem estado: {registros_sem_estado.shape[0]}')
```

```
                          location
6                    United States
14                   United States
18              Greater Philadelphia
22      Los Angeles Metropolitan Area
27          New Jersey, United States
```

```
...                                ...
123839           Greater Indianapolis
123841                  United States
123842                  United States
123845                  United States
123847           Texas, United States

[17975 rows x 1 columns]
Número de registros sem estado: 17975
```

```python
# Para resolver o problema, vou criar uma bibioteca para mapeamento dos estados e criar comparar o nome à sigla,
# e de pois atribuir a sigla do estado com uma funçao SE:
# Dicionário de mapeamento de estados
estado_mapping = {
    'Alabama': 'AL',
    'Alaska': 'AK',
    'Arizona': 'AZ',
    'Arkansas': 'AR',
    'California': 'CA',
    'Colorado': 'CO',
    'Connecticut': 'CT',
    'Delaware': 'DE',
    'Florida': 'FL',
    'Georgia': 'GA',
    'Hawaii': 'HI',
    'Idaho': 'ID',
    'Illinois': 'IL',
    'Indiana': 'IN',
    'Iowa': 'IA',
    'Kansas': 'KS',
    'Kentucky': 'KY',
    'Louisiana': 'LA',
    'Maine': 'ME',
    'Maryland': 'MD',
    'Massachusetts': 'MA',
    'Michigan': 'MI',
    'Minnesota': 'MN',
    'Mississippi': 'MS',
    'Missouri': 'MO',
    'Montana': 'MT',
    'Nebraska': 'NE',
    'Nevada': 'NV',
    'New Hampshire': 'NH',
    'New Jersey': 'NJ',
    'New Mexico': 'NM',
    'New York': 'NY',
    'North Carolina': 'NC',
    'North Dakota': 'ND',
    'Ohio': 'OH',
    'Oklahoma': 'OK',
    'Oregon': 'OR',
    'Pennsylvania': 'PA',
    'Rhode Island': 'RI',
    'South Carolina': 'SC',
    'South Dakota': 'SD',
    'Tennessee': 'TN',
    'Texas': 'TX',
    'Utah': 'UT',
    'Vermont': 'VT',
    'Virginia': 'VA',
    'Washington': 'WA',
    'West Virginia': 'WV',
    'Wisconsin': 'WI',
    'Wyoming': 'WY'
}
```

```python
# Função para mapear os estados
def map_estado(location):
    for estado, sigla in estado_mapping.items():
        if estado in location:
            return sigla
    return None  # Retorna None se não encontrar nenhum estado

# Aplicar a função para preencher a coluna 'estado'
df_cleaned['estado'] = df_cleaned.apply(lambda x: map_estado(x['location']) if pd.isnull(x['estado']) else
```

```
x['estado'], axis=1)

# Verificar a quantidade de estados preenchidos
print(f'Número de registros sem estado após a substituição: {df_cleaned["estado"].isnull().sum()}')
```

```
Número de registros sem estado após a substituição: 11597
```

```
# Exibir registros que ainda estão sem estado
registros_sem_estado = df_cleaned[df_cleaned['estado'].isnull() | (df_cleaned['estado'] == '')]

# Exibir algumas dessas localizações
print(registros_sem_estado[['location']])
print(f'Número de registros sem estado: {registros_sem_estado.shape[0]}')
```

```
                           location
6                     United States
14                    United States
18               Greater Philadelphia
22      Los Angeles Metropolitan Area
33                    United States
...                            ...
123784                United States
123790                United States
123841                United States
123842                United States
123845                United States

[11597 rows x 1 columns]
Número de registros sem estado: 11597
```

```
# Ainda há linhas sem registos de Estadom então vou remover registros sem estado
df_cleaned = df_cleaned[df_cleaned['estado'].notnull() & (df_cleaned['estado'] != '')]

# Verificar o novo total de registros
print(f'Total de registros após remoção: {df_cleaned.shape[0]}')
```

```
Total de registros após remoção: 107408
```

```
# Contar o número de vagas por estado novamente
state_counts_df = df_cleaned['estado'].value_counts().reset_index()
state_counts_df.columns = ['Estado', 'Número de Vagas']

# Somar o total de vagas
total_vagas = state_counts_df['Número de Vagas'].sum()
print(f'Total de Vagas após limpeza: {total_vagas}')

# Calcular o percentual para cada estado
state_counts_df['Percentual'] = (state_counts_df['Número de Vagas'] / total_vagas) * 100

# Exibir a tabela atualizada
print(state_counts_df)
```

```
Total de Vagas após limpeza: 107408
   Estado  Número de Vagas  Percentual
0      CA            11859   11.041077
1      TX            10528    9.801877
2      NY             7367    6.858893
3      FL             5987    5.574073
4      NC             4961    4.618837
5      IL             4498    4.187770
6      PA             4057    3.777186
7      VA             3656    3.403843
```

8	OH	3535	3.291189
9	MA	3496	3.254879
10	GA	3458	3.219499
11	NJ	3375	3.142224
12	WA	2997	2.790295
13	MI	2735	2.546365
14	AZ	2472	2.301505
15	CO	2340	2.178609
16	MD	1983	1.846231
17	TN	1888	1.757783
18	MN	1877	1.747542
19	IN	1870	1.741025
20	MO	1861	1.732646
21	WI	1859	1.730784
22	SC	1569	1.460785
23	OR	1253	1.166580
24	CT	1178	1.096753
25	KY	1167	1.086511
26	AL	992	0.923581
27	UT	985	0.917064
28	LA	976	0.908685
29	IA	965	0.898443
30	KS	955	0.889133
31	NV	876	0.815582
32	DC	859	0.799754
33	OK	825	0.768099
34	AR	654	0.608893
35	NE	592	0.551169
36	NH	567	0.527894
37	NM	493	0.458997
38	HI	429	0.399412
39	ID	426	0.396619
40	WV	396	0.368688
41	MS	394	0.366826
42	ME	380	0.353791
43	DE	323	0.300722
44	RI	316	0.294205
45	MT	251	0.233688
46	ND	239	0.222516
47	AK	215	0.200171
48	VT	178	0.165723
49	SD	170	0.158275
50	WY	124	0.115448
51	ON	1	0.000931
52	QC	1	0.000931

```
pip install plotly
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: plotly in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages
(from plotly) (9.0.0)
Requirement already satisfied: packaging in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages (from
plotly) (24.1)

 [1m[ [0m [34;49mnotice [0m [1;39;49m] [0m [39;49m A new release of pip is available:  [0m [31;49m24.2 [0m [39;49m ->
 [0m [32;49m24.3.1 [0m
 [1m[ [0m [34;49mnotice [0m [1;39;49m] [0m [39;49m To update, run:
 [0m [32;49m/Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip [0m
Note: you may need to restart the kernel to use updated packages.
```

```
import plotly.express as px

# Criar o gráfico coroplético com uma paleta de cores consistente
fig = px.choropleth(
    state_counts_df,
    locations='Estado',  # Coluna com os estados
    locationmode="USA-states",  # Localização para estados dos EUA
    color='Número de Vagas',  # Preencher com base no número de vagas
    hover_name='Estado',  # Nome do estado no hover
```

```
    color_continuous_scale=px.colors.sequential.Plasma,  # Paleta de cores próxima do padrão
    labels={'Número de Vagas': 'Número de Vagas'},  # Rótulos
    title="Distribuição de Vagas por Estado"
)

# Refinar o layout do mapa
fig.update_geos(
    showcoastlines=True,
    coastlinecolor="Black",
    projection_type="albers usa",  # Projeção apropriada para os EUA
)

# Refinar o layout geral
fig.update_layout(
    title_font_size=18,
    title_x=0.5,  # Centralizar o título
    coloraxis_colorbar=dict(
        title="Número de Vagas",
        ticks="outside"
    )
)

# Exibir o gráfico
fig.show()
```

var gd = document.getElementById('cd85d853-0b68-430b-90f7-36e8cff77abe'); var x = new MutationObserver(function (mutations, observer) {{ var display = window.getComputedStyle(gd).display; if (!display || display === 'none') {{ console.log([gd, 'removed!']); Plotly.purge(gd); observer.disconnect(); }} }});

// Listen for the removal of the full notebook cells var notebookContainer = gd.closest('#notebook-container'); if (notebookContainer) {{ x.observe(notebookContainer, {childList: true}); }}

// Listen for the clearing of the current output cell var outputEl = gd.closest('.output'); if (outputEl) {{ x.observe(outputEl, {childList: true}); }}

```
                })               };               });           </script>       </div>
```

```
# INSIGHTS:
# 1. Concentração de Vagas:
# O estado da Califórnia (CA) lidera a lista com 11.041% das vagas, totalizando 11.859 oportunidades. Isso reflete o
peso econômico e o
# dinamismo do estado, especialmente em setores como tecnologia e entretenimento.
# O Texas (TX) ocupa o segundo lugar, com 10.528 vagas (9.8%), seguido por Nova York (NY), com 7.367 vagas (6.85%).

# 2. Regiões com Baixa Participação:
# Estados como Vermont (VT), Dakota do Sul (SD) e Wyoming (WY) possuem menos de 200 vagas cada, indicando menos
demanda por mão de obra
# ou menor publicação de oportunidades no LinkedIn.
```

```
##########################################################################
############ 4 - Análise de Vagas por Nível de Experiência  ############
##########################################################################
```

```
# CONTEXTO:
# Essa análise examina a distribuição de vagas por nível de experiência, destacando as oportunidades disponíveis para
diferentes
# perfis de candidatos.
```

```
# Contar o número de vagas por nível de experiência
experience_counts_df = df_cleaned['formatted_experience_level'].value_counts().reset_index()
experience_counts_df.columns = ['Nível de Experiência', 'Número de Vagas']

# Ordenar por número de vagas
experience_counts_df = experience_counts_df.sort_values(by='Número de Vagas', ascending=False)

# Exibir a tabela
print(experience_counts_df)
```

```
  Nível de Experiência  Número de Vagas
0     Mid-Senior level            35324
1          Entry level            33135
2     Não especificado            25284
3            Associate             8344
4             Director             3062
5           Internship             1298
6            Executive              961
```

```
# Criar o gráfico de barras horizontais
plt.figure(figsize=(14, 8))  # Aumentando o tamanho da figura
ax = sns.barplot(
    x='Número de Vagas',
    y='Nível de Experiência',
    data=experience_counts_df,
    errorbar=None,  # Substituir o `ci` por `errorbar=None` para evitar o aviso de depreciação
)

# Aplicar um gradiente de cor usando a paleta plasma
norm = mpl.colors.Normalize(vmin=0, vmax=len(ax.patches))
cmap = plt.get_cmap("plasma")

# Função para arredondar as barras
def round_bars(patch, color, rounding_size=10):
    # Obter a posição da barra
    x0, y0 = patch.get_xy()  # Posição de origem
    width, height = patch.get_width(), patch.get_height()  # Largura e altura da barra

    # Definir o caminho para a barra arredondada
    verts = [
        (x0 + rounding_size, y0),  # Começo arredondado
        (x0 + width - rounding_size, y0),
        (x0 + width, y0 + rounding_size),  # Arredondamento da borda superior
        (x0 + width, y0 + height - rounding_size),
        (x0 + width - rounding_size, y0 + height),
        (x0 + rounding_size, y0 + height),
        (x0, y0 + height - rounding_size),  # Arredondamento da borda inferior
        (x0, y0 + rounding_size)
    ]

    # Fechar o caminho (voltar ao início)
    verts.append(verts[0])

    # Criar o Path e o Patch
    path = Path(verts, closed=True)
    patch.set_facecolor(color)  # Aplica a cor gradiente
    patch.set_edgecolor("white")  # Define a cor da borda como branca
    patch.set_linewidth(2)  # Define a espessura da borda
```

```
    # Substituir o patch original pelo novo com bordas arredondadas
    path_patch = PathPatch(path, facecolor=color, edgecolor="white", lw=2)
    ax.add_patch(path_patch)
    patch.remove()  # Remove o patch original

# Arredondar as barras e aplicar o gradiente de cores
for i, patch in enumerate(ax.patches):
    # Calculando o valor para a cor gradiente
    color = cmap(norm(i))  # Aplica um gradiente na cor da barra
    round_bars(patch, color)

# Ajustar o visual do gráfico
plt.title('Distribuição de Vagas por Nível de Experiência', fontsize=18, fontweight='bold', color='black', pad=30)
plt.xlabel('Número de Vagas', fontsize=14, color='darkred', labelpad=20)  # Cor personalizada e aumento do
espaçamento
plt.ylabel('Nível de Experiência', fontsize=14, color='darkblue', labelpad=20)  # Cor personalizada e aumento do
espaçamento

# Remover as linhas do fundo (grid e bordas)
ax.grid(False)  # Remove o grid do gráfico
for spine in ax.spines.values():
    spine.set_visible(False)  # Remove as bordas (spines) do gráfico

# Remover a legenda desnecessária
plt.legend([], [], frameon=False)

# Melhorar o layout para evitar cortes
plt.tight_layout()

# Gravar gráfico em png na pasta do website
output_dir = '/Users/elisadrummond/projeto-tendencias-empregos/static/graphs'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Salvar o gráfico
graph_path = os.path.join(output_dir, 'grafico3_nivel_experiencia.png')
plt.savefig(graph_path, dpi=300, bbox_inches='tight')  # Aumentar a resolução para maior qualidade

# Exibir o gráfico
plt.show()

# Fechar a figura após salvar
plt.close()

# Mensagem de confirmação
print(f"Gráfico salvo em: {graph_path}")
```

```
Gráfico salvo em: /Users/elisadrummond/projeto-tendencias-empregos/static/graphs/grafico3_nivel_experiencia.png
```

```
# INSIGHTS:
# O nível predominante nas vagas de emprego é o Mid-Senior Level, com 35.324 vagas (32,9%), evidenciando uma forte
demanda por
# profissionais com experiência intermediária a avançada. Em seguida, o Entry Level ocupa a segunda posição com
33.135 vagas (30,8%),
# oferecendo boas oportunidades para recém-formados e profissionais iniciando suas carreiras. No entanto, há uma
lacuna significativa,
# pois 25.284 vagas (23,5%) não especificam o nível de experiência, o que pode dificultar a segmentação precisa de
candidatos.
```

```
##########################################################################
############ 4 - Correlaçao entre visualização e aplicação #############
##########################################################################
```

```
# CONTEXTO:
# A correlação entre visualizações de vagas e candidaturas foi analisada, separando vagas que permitem trabalho
remoto e aquelas que
```

```
# não permitem. O coeficiente de correlação de Pearson foi calculado para entender a relação linear entre essas
variáveis para cada grupo.
```

```
# Calcula o coeficiente de correlação de Pearson entre as colunas 'views' e 'applies'
# A correlação de Pearson mede a força e a direção da relação linear entre as duas variáveis.
correlacao = df['views'].corr(df['applies'])

# Exibe o coeficiente de correlação no console.
# O coeficiente de correlação pode variar de -1 (correlação negativa perfeita) a 1 (correlação positiva perfeita).
# Se o valor estiver perto de 0, significa que não há uma relação linear forte entre as variáveis.
print(f'Coeficiente de correlação de Pearson: {correlacao}')
```

```
Coeficiente de correlação de Pearson: 0.49430495349631376
```

```
# Agora, podemos calcular as correlações para cada categoria:
df_remoto = df_cleaned[df_cleaned['remote_allowed'] == 'Permitido']
df_nao_remoto = df_cleaned[df_cleaned['remote_allowed'] == 'Não Permitido']

# Correlação para vagas remotas
correlacao_remoto = df_remoto['views'].corr(df_remoto['applies'])
print(f"Correlação (remoto permitido): {correlacao_remoto}")

# Correlação para vagas não remotas
correlacao_nao_remoto = df_nao_remoto['views'].corr(df_nao_remoto['applies'])
print(f"Correlação (remoto não permitido): {correlacao_nao_remoto}")
```

```
Correlação (remoto permitido): 0.7728109439719456
Correlação (remoto não permitido): 0.38354013638845524
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Definindo o tamanho do gráfico (largura, altura)
plt.figure(figsize=(20, 10))  # Aumentando o tamanho do gráfico

# Gráfico de dispersão com linha de tendência utilizando a paleta 'plasma'
sns.regplot(x='views', y='applies', data=df_cleaned[df_cleaned['remote_allowed'] == 'Permitido'],
            scatter_kws={'color': sns.color_palette("plasma", 2)[0], 's': 100},  # Cor para 'Remoto Permitido' e
tamanho dos pontos
            line_kws={'color': sns.color_palette("plasma", 2)[0], 'lw': 2}, label='Remoto Permitido', ci=None)

sns.regplot(x='views', y='applies', data=df_cleaned[df_cleaned['remote_allowed'] == 'Não Permitido'],
            scatter_kws={'color': sns.color_palette("plasma", 2)[1], 's': 100},  # Cor para 'Remoto Não Permitido' e
tamanho dos pontos
            line_kws={'color': sns.color_palette("plasma", 2)[1], 'lw': 2}, label='Remoto Não Permitido', ci=None)

# Adicionar título e rótulos
plt.title('Relação entre Views e Applies por Tipo de Trabalho', fontsize=22, fontweight='bold', color='black',
pad=30)
plt.xlabel('Views', fontsize=18, color='darkred', labelpad=20)
plt.ylabel('Applies', fontsize=18, color='darkblue', labelpad=20)

# Adicionar legenda
plt.legend(fontsize=16)

# Remover grid e bordas
plt.grid(False)

# Gravar gráfico em png na pasta do website
output_dir = '/Users/elisadrummond/projeto-tendencias-empregos/static/graphs'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Salvar o gráfico
graph_path = os.path.join(output_dir, 'grafico4_correlacao_views_applies.png')
plt.savefig(graph_path, dpi=300, bbox_inches='tight')
```

```
# Exibir o gráfico
plt.tight_layout()
plt.show()

# Fechar a figura após salvar
plt.close()

# Mensagem de confirmação
print(f"Gráfico salvo em: {graph_path}")
```

```
Gráfico salvo em: /Users/elisadrummond/projeto-tendencias-
empregos/static/graphs/grafico4_correlacao_views_applies.png
```

```
# INSIGHTS:
# A correlação mais forte para vagas com trabalho remoto permitido (0.77) indica que, quanto mais visualizações essas
vagas recebem,
# mais candidaturas são geradas, refletindo uma preferência dos candidatos por oportunidades remotas. Por outro lado,
as vagas sem
# trabalho remoto mostram uma correlação mais baixa (0.38), sugerindo que, embora também haja interesse, outros
fatores podem influenciar
# mais as candidaturas.
```

```
################################################################################
############### 5 - TOP 10 Títulos mais ofertados ################
################################################################################
```

```
# CONTEXTO:
# A análise compara os 10 títulos de vagas de emprego mais ofertadas com o número de aplicações para cada cargo.
# O objetivo é entender a relação entre a quantidade devagas abertas e o interesse dos candidatos, além de explorar a
correlação entre
# essas variáveis.
```

```
# 1. Contagem dos Títulos mais Ofertados
contagem_ofertas = df_cleaned['title'].value_counts().head(10)

# 2. Calcular o número de aplicações por título de vaga
aplicacoes_por_titulo = df_cleaned.groupby('title')['applies'].sum()

# 3. Combinar as duas informações (ofertas e aplicações) para comparação
vaga_aplicacao_comparacao = pd.DataFrame({
    'Número de Ofertas': contagem_ofertas,
    'Número de Aplicações': aplicacoes_por_titulo[contagem_ofertas.index]
})
```

```
import matplotlib.pyplot as plt
import numpy as np
import os

# Criar gráfico de barras agrupadas com gradiente e maior tamanho
fig, ax = plt.subplots(figsize=(20, 10))  # Gráfico maior

# Índice para as barras
index = np.arange(len(vaga_aplicacao_comparacao))

# Largura das barras
bar_width = 0.4

# Gradiente para as barras
cmap = plt.cm.plasma
colors1 = cmap(np.linspace(0.2, 0.5, len(vaga_aplicacao_comparacao)))
colors2 = cmap(np.linspace(0.7, 1.0, len(vaga_aplicacao_comparacao)))
```

```
# Definir as barras para o número de ofertas com gradiente
bars1 = ax.bar(
    index,
    vaga_aplicacao_comparacao['Número de Ofertas'],
    bar_width,
    color=colors1,
    label='Número de Ofertas'  # Adicionando label para a legenda
)

# Definir as barras para o número de aplicações com gradiente
bars2 = ax.bar(
    index + bar_width,
    vaga_aplicacao_comparacao['Número de Aplicações'],
    bar_width,
    color=colors2,
    label='Número de Aplicações'  # Adicionando label para a legenda
)

# Ajustar o título e rótulos
ax.set_title(
    'Top 10 Vagas mais Ofertadas vs. Número de Aplicações',
    fontsize=22,
    fontweight='bold',
    color='black',
    pad=30
)
ax.set_xlabel(
    'Título da Vaga',
    fontsize=18,
    color='darkred',
    labelpad=20
)
ax.set_ylabel(
    'Número de Ofertas e Aplicações',
    fontsize=18,
    color='darkblue',
    labelpad=20
)

# Ajustar rótulos do eixo X
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(vaga_aplicacao_comparacao.index, rotation=45, ha='right', fontsize=16)

# Adicionar a legenda
ax.legend(fontsize=16, loc='upper left')

# Remover grid e bordas
ax.grid(False)
for spine in ax.spines.values():
    spine.set_visible(False)

# Melhorar layout
plt.tight_layout()

# Salvar o gráfico
output_dir = '/Users/elisadrummond/projeto-tendencias-empregos/static/graphs'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

graph_path = os.path.join(output_dir, 'grafico5_top10_titulos_ofertados.png')
plt.savefig(graph_path, dpi=300, bbox_inches='tight')

# Exibir gráfico
plt.show()

# Fechar a figura
plt.close(fig)

# Mensagem de confirmação
print(f"Gráfico salvo em: {graph_path}")
```

```
Gráfico salvo em: /Users/elisadrummond/projeto-tendencias-empregos/static/graphs/grafico5_top10_titulos_ofertados.png
```

```
# INSIGHTS:
# As vagas mais ofertadas incluem cargos como Sales Manager, Customer Service Representative e Project Manager, mas a quantidade de
# aplicações varia significativamente entre elas. Enquanto algumas vagas com muitos anúncios, como Sales Manager, têm poucas candidaturas,
# outras, como Project Manager, geram uma grande quantidade de aplicações, o que indica uma demanda maior para essas posições.
```

```
##########################################################################
############### 5 - TOP 10 Títulos mais aplicados ################
##########################################################################
```

```
# CONTEXTO:
# A análise comparou as 10 vagas mais aplicadas com a quantidade de ofertas disponíveis para cada cargo. A correlação entre as duas
# variáveis foi calculada, e o gráfico gerado mostra, para cada vaga, a comparação entre o número de aplicações e de ofertas.
```

```
# 1. Contar as vagas com mais aplicações
top_10_aplicacoes = df_cleaned.groupby('title')['applies'].sum().sort_values(ascending=False).head(10)

# Exibir as 10 vagas com mais aplicações
print("Top 10 Vagas com Mais Aplicações:")
print(top_10_aplicacoes)

# 2. Calcular o número de ofertas por título de vaga
ofertas_por_titulo = df_cleaned.groupby('title')['title'].count()

# 3. Combinar as duas informações (aplicações e ofertas) para comparação
aplicacao_oferta_comparacao = pd.DataFrame({
    'Número de Aplicações': top_10_aplicacoes,
    'Número de Ofertas': ofertas_por_titulo[top_10_aplicacoes.index]
})

# Exibir a comparação
print("\nComparação entre o número de aplicações e o número de ofertas:")
print(aplicacao_oferta_comparacao)

# Calcular os percentuais
aplicacao_oferta_comparacao['Percentual de Aplicações (%)'] = (aplicacao_oferta_comparacao['Número de Aplicações'] / aplicacao_oferta_comparacao['Número de Ofertas']) * 100

# Exibir os percentuais
print("\nPercentual de Aplicações em relação às Ofertas:")
print(aplicacao_oferta_comparacao[['Percentual de Aplicações (%)']])

# Exibindo os resultados
print("\nResumo:")
print(f"Total de vagas analisadas: {len(aplicacao_oferta_comparacao)}")
```

```
Top 10 Vagas com Mais Aplicações:
title
Data Analyst                3682.0
Data Engineer               2897.0
Business Analyst            2122.0
Software Engineer           2072.0
Project Manager             1086.0
Cloud Engineer               890.0
DevOps Engineer              791.0
Senior Data Engineer         747.0
Senior Software Engineer     735.0
Data Scientist               676.0
Name: applies, dtype: float64

Comparação entre o número de aplicações e o número de ofertas:
                         Número de Aplicações  Número de Ofertas
title
Data Analyst                           3682.0                 96
Data Engineer                          2897.0                 75
```

```
Business Analyst                        2122.0               121
Software Engineer                       2072.0               122
Project Manager                         1086.0               309
Cloud Engineer                           890.0                26
DevOps Engineer                          791.0                41
Senior Data Engineer                     747.0                27
Senior Software Engineer                 735.0               101
Data Scientist                           676.0                45

Percentual de Aplicações em relação às Ofertas:
                          Percentual de Aplicações (%)
title
Data Analyst                                3835.416667
Data Engineer                               3862.666667
Business Analyst                            1753.719008
Software Engineer                           1698.360656
Project Manager                              351.456311
Cloud Engineer                              3423.076923
DevOps Engineer                             1929.268293
Senior Data Engineer                        2766.666667
Senior Software Engineer                     727.722772
Data Scientist                              1502.222222

Resumo:
Total de vagas analisadas: 10
```

```python
# Definindo o tamanho do gráfico
fig, ax = plt.subplots(figsize=(20, 10))  # Aumentando o tamanho do gráfico

# Número de barras
n = len(aplicacao_oferta_comparacao)

# Criando um índice para as barras
index = np.arange(n)

# Largura das barras
bar_width = 0.35

# Gradiente para as barras
cmap = plt.cm.plasma
colors1 = cmap(np.linspace(0.2, 0.5, n))  # Gradiente para o número de aplicações
colors2 = cmap(np.linspace(0.7, 1.0, n))  # Gradiente para o número de ofertas

# Plotando as barras para o número de aplicações
bars1 = ax.bar(index, aplicacao_oferta_comparacao['Número de Aplicações'], bar_width, color=colors1, label='Número de
Aplicações')

# Plotando as barras para o número de ofertas, deslocadas para o lado direito
bars2 = ax.bar(index + bar_width, aplicacao_oferta_comparacao['Número de Ofertas'], bar_width, color=colors2,
label='Número de Ofertas')

# Adicionando título e rótulos
ax.set_title('Top 10 Vagas Aplicadas: Comparação entre Aplicações e Ofertas', fontsize=22, fontweight='bold',
color='black', pad=30)
ax.set_xlabel('Título da Vaga', fontsize=18, color='darkred', labelpad=20)
ax.set_ylabel('Número de Ofertas e Aplicações', fontsize=18, color='darkblue', labelpad=20)

# Ajustando o eixo X para as vagas
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(aplicacao_oferta_comparacao.index, rotation=45, ha='right', fontsize=16)

# Adicionar a legenda
ax.legend(fontsize=16, loc='upper right')

# Remover grid e bordas
ax.grid(False)
for spine in ax.spines.values():
    spine.set_visible(False)

# Melhorar layout
plt.tight_layout()

# Gravar gráfico em png na pasta do website
output_dir = '/Users/elisadrummond/projeto-tendencias-empregos/static/graphs'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
```

```
# Salvar o gráfico
graph_path = os.path.join(output_dir, 'grafico6_top10_vagas_procuradas_candidatos.png')
plt.savefig(graph_path, dpi=300, bbox_inches='tight')

# Exibir gráfico
plt.show()

# Fechar a figura após exibição
plt.close(fig)

# Mensagem de confirmação
print(f"Gráfico salvo em: {graph_path}")
```

```
Gráfico salvo em: /Users/elisadrummond/projeto-tendencias-
empregos/static/graphs/grafico6_top10_vagas_procuradas_candidatos.png
```

```
# INSIGHTS:
# Os dados revelam que funções como Data Analyst e Data Engineer têm alta competição, com muitas aplicações para
poucas ofertas,
# indicando uma alta demanda e limitada oferta. Vagas como Business Analyst, Software Engineer e Project Manager
mostram um equilíbrio
# maior entre ofertas e aplicações, enquanto funções como Cloud Engineer e DevOps Engineer têm mais ofertas do que
aplicações,
# sugerindo menor concorrência. A vaga de Data Scientist tem poucas aplicações, apesar de ter algumas ofertas,
indicando uma demanda mais
# controlada, possivelmente devido à especialização exigida.
```

```
##########################################################################
############### 5 - SKILLS ################
##########################################################################
```

```
# CONTEXTO:
# Esta análise visa entender as habilidades mais demandadas nas vagas de emprego e identificar as competências mais
valorizadas pelas
# empresas, e o que isso pode revelar sobre as tendências do mercado de trabalho atual.
```

```
!pip install wordcloud
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: wordcloud in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages (from
wordcloud) (2.0.2)
Requirement already satisfied: pillow in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages (from
wordcloud) (10.4.0)
Requirement already satisfied: matplotlib in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages (from
wordcloud) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages
(from matplotlib->wordcloud) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages (from
matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages
(from matplotlib->wordcloud) (4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages
(from matplotlib->wordcloud) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages
(from matplotlib->wordcloud) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages
(from matplotlib->wordcloud) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-
packages (from matplotlib->wordcloud) (2.9.0.post0)
Requirement already satisfied: importlib-resources>=3.2.0 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-
packages (from matplotlib->wordcloud) (6.4.5)
```

```
Requirement already satisfied: zipp>=3.1.0 in /Users/elisadrummond/Library/Python/3.9/lib/python/site-packages (from
importlib-resources>=3.2.0->matplotlib->wordcloud) (3.20.2)
Requirement already satisfied: six>=1.5 in
/Library/Developer/CommandLineTools/Library/Frameworks/Python3.framework/Versions/3.9/lib/python3.9/site-packages
(from python-dateutil>=2.7->matplotlib->wordcloud) (1.15.0)

 [1m[ [0m [34;49mnotice [0m [1;39;49m] [0m [39;49m A new release of pip is available:  [0m [31;49m24.2 [0m [39;49m ->
 [0m [32;49m24.3.1 [0m
 [1m[ [0m [34;49mnotice [0m [1;39;49m] [0m [39;49m To update, run:
 [0m [32;49m/Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade pip [0m
```

```python
# Limpar as habilidades e separar em listas
df_cleaned['skills_desc'] = df_cleaned['skills_desc'].str.replace(r'This position requires the following skills: ',
'', regex=True)

# Dividir as habilidades em listas separadas
df_cleaned['skills_list'] = df_cleaned['skills_desc'].str.split(',')

# Explodir as listas para que cada habilidade fique em uma linha separada
habilidades_expandidas = df_cleaned.explode('skills_list')

# Limpar espaços extras e normalizar as habilidades
habilidades_expandidas['skills_list'] = habilidades_expandidas['skills_list'].str.strip()

# Remover 'Habilidades não especificadas' da lista
habilidades_expandidas = habilidades_expandidas[habilidades_expandidas['skills_list'] != 'Habilidades não
especificadas']

# Recontar as habilidades
habilidades_frequentes = habilidades_expandidas['skills_list'].value_counts()

# Selecionar as 15 habilidades mais frequentes
top_15_habilidades = habilidades_frequentes.head(15)

# Ajustes de fonte e tamanho conforme definido
font_properties = {
    'family': 'Arial',  # Definindo a fonte
    'weight': 'normal',
    'size': 14  # Tamanho de texto padrão
}

# Gerar a nuvem de palavras com ajustes
wordcloud = WordCloud(
    width=800,
    height=400,
    background_color='white',
    colormap='plasma',
    font_path=None,  # Caso queira usar uma fonte específica, adicione o caminho para o arquivo
    stopwords=None,  # Define que as palavras mais comuns não sejam ignoradas
    max_words=150
).generate_from_frequencies(top_15_habilidades)

# Exibindo a nuvem de palavras
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Remove os eixos para uma melhor visualização

# Ajustar o título com mais espaço entre a nuvem de palavras
plt.title('As 15 Habilidades mais requisitadas', fontsize=16, fontdict=font_properties, pad=20)

# Gravar a imagem em PNG na pasta do website
output_dir = '/Users/elisadrummond/projeto-tendencias-empregos/static/graphs'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Salvar o gráfico
graph_path = os.path.join(output_dir, 'nuvem_palavras_habilidades.png')
plt.savefig(graph_path)

# Fechar a figura após salvar
plt.close()

# Mensagem de confirmação
print(f"Nuvem de palavras salva em: {graph_path}")

# Exibir o gráfico
plt.show()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

Cell In[266], line 30
     23 font_properties = {
     24     'family': 'Arial',  # Definindo a fonte
     25     'weight': 'normal',
     26     'size': 14  # Tamanho de texto padrão
     27 }
     29 # Gerar a nuvem de palavras com ajustes
---> 30 wordcloud = WordCloud(
     31     width=800,
     32     height=400,
     33     background_color='white',
     34     colormap='plasma',
     35     font_path=None,  # Caso queira usar uma fonte específica, adicione o caminho para o arquivo
     36     stopwords=None,  # Define que as palavras mais comuns não sejam ignoradas
     37     max_words=150
     38 ).generate_from_frequencies(top_15_habilidades)
     40 # Exibindo a nuvem de palavras
     41 plt.figure(figsize=(10, 6))


NameError: name 'WordCloud' is not defined
```

```
# INSIGHTS:
# As habilidades mais frequentes indicam que empresas valorizam tanto habilidades técnicas quanto comportamentais,
como comunicação eficaz
# e empatia. O aumento da ênfase em habilidades sociais, como "Networking" e "Community Outreach", reflete a
crescente importância da
# construção de relações e da responsabilidade social nas organizações.
```

```
!pip freeze > requirements.txt
```