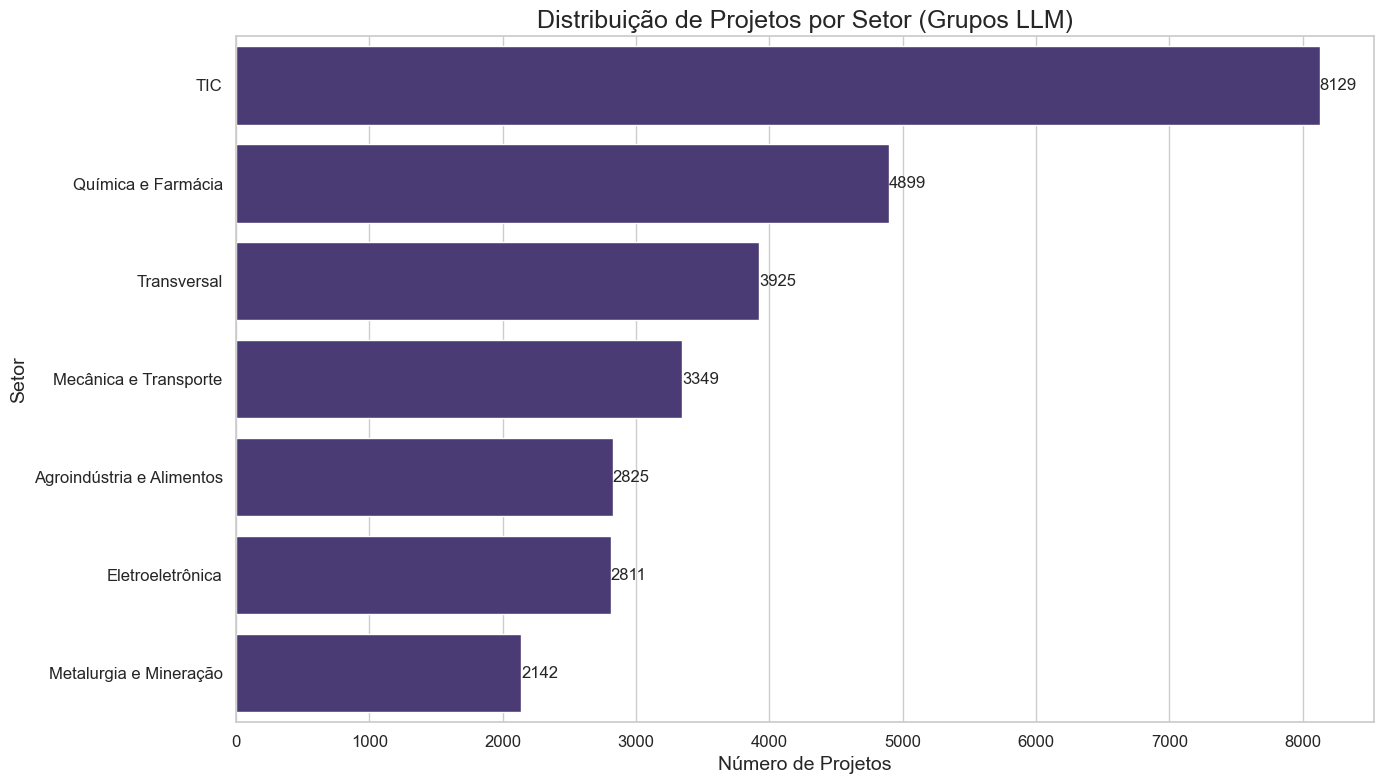
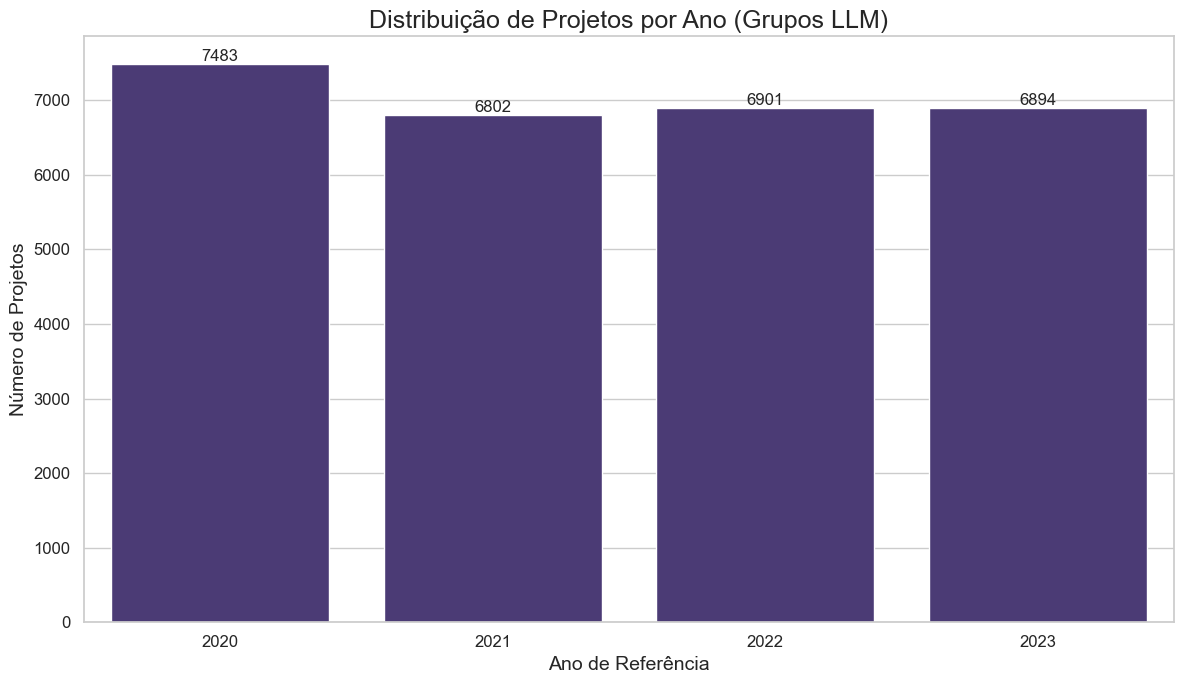
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from io import StringIO  
  
# Configurações visuais para os gráficos  
sns.set\_theme(style="whitegrid", palette="viridis")  
plt.rcParams['figure.figsize'] = (14, 7)  
plt.rcParams['axes.titlesize'] = 18  
plt.rcParams['axes.labelsize'] = 14  
plt.rcParams['xtick.labelsize'] = 12  
plt.rcParams['ytick.labelsize'] = 12  
plt.rcParams['legend.fontsize'] = 12  
  
# --- NOTA: Carregue seu arquivo CSV completo aqui ---  
# O código abaixo usa a amostra fornecida no prompt para demonstração.  
# Substitua esta seção pelo carregamento do seu arquivo real.  
csv\_data\_sample = """  
projeto\_id;grupo\_id\_final;sublote;similaridade\_score;justificativa\_agrupamento;origem\_agrupamento;ano\_referencia;setor;natureza;tipo\_pesquisa;empresa;projeto;do\_id\_at;do\_resultado\_analise  
49408.0;1;SL\_8;0.85;Melhorias incrementais em processos industriais de fabricação de produtos plásticos;LLM;2020.0;Transversal;Processo;DE - Desenvolvimento Experimental;CNPJ: 00385650000100;PROJETO A;301043772.0;Recomendado  
51621.0;1;SL\_8;0.82;Aprimoramento de processos de fabricação com foco em automação e qualidade;LLM;2020.0;Transversal;Processo;DE - Desenvolvimento Experimental;CNPJ: 84501873000178;PROJETO B;301043772.0;Não Recomendado  
62743.0;1;SL\_8;0.8;Desenvolvimento de tecnologias para evolução em processos de fabricação de filmes plásticos;LLM;2021.0;Transversal;Processo;DE - Desenvolvimento Experimental;CNPJ: 03071894000107;PROJETO C;300074422.0;Não Recomendado  
59124.0;1;SL\_8;0.78;Aperfeiçoamento de soluções tecnológicas em processos de fabricação de tampas plásticas;LLM;2021.0;Transversal;Processo;DE - Desenvolvimento Experimental;CNPJ: 11237579000127;PROJETO D;300804386.0;Não Recomendado  
63126.0;2;SL\_8;0.88;Soluções de tratamento de água com foco em filtragem e dosagem para estações de tratamento;LLM;2021.0;TIC;Processo;DE - Desenvolvimento Experimental;CNPJ: 15401489000180;PROJETO E;119427979.0;Não Recomendado  
68446.0;2;SL\_8;0.86;Desenvolvimento de sistemas de monitoramento e detecção de anomalias na qualidade da água;LLM;2021.0;TIC;Processo;DE - Desenvolvimento Experimental;CNPJ: 04089570000150;PROJETO F;301064213.0;Recomendado  
61196.0;2;SL\_8;0.85;Sistemas de controle de qualidade da água em redes de distribuição;LLM;2021.0;Química e Farmácia;Processo;DE - Desenvolvimento Experimental;CNPJ: 21716748000165;PROJETO G;127038059.0;Recomendado  
64884.0;3;SL\_8;0.9;Transformação digital e implementação de conceitos de Indústria 4.0 em processos industriais;LLM;2021.0;TIC;Processo;DE - Desenvolvimento Experimental;CNPJ: 61092037000181;PROJETO H;119427979.0;Recomendado  
61370.0;3;SL\_8;0.88;Plataformas de gestão integrada para automação de processos industriais;LLM;2021.0;TIC;Processo;DE - Desenvolvimento Experimental;CNPJ: 62258884000136;PROJETO I;121755221.0;Recomendado  
67638.0;3;SL\_8;0.85;Digitalização de rotinas de trabalho e implementação de sistemas automatizados;LLM;2021.0;Mecânica e Transporte;Processo;DE - Desenvolvimento Experimental;CNPJ: 00324107000195;PROJETO J;301040317.0;Não Recomendado  
47632.0;4;SL\_8;0.87;Desenvolvimento de embalagens sustentáveis e termoseláveis para produtos industriais;LLM;2020.0;Agroindústria e Alimentos;Processo;DE - Desenvolvimento Experimental;CNPJ: 52736949000158;PROJETO K;300074422.0;Não Recomendado  
47631.0;4;SL\_8;0.85;Inovações em embalagens laminadas para produtos industriais;LLM;2020.0;Agroindústria e Alimentos;Processo;DE - Desenvolvimento Experimental;CNPJ: 52736949000158;PROJETO L;300074422.0;Não Recomendado  
47621.0;4;SL\_8;0.82;Desenvolvimento de estruturas de embalagem monocamada para produtos de papel;LLM;2020.0;Metalurgia e Mineração;Processo;DE - Desenvolvimento Experimental;CNPJ: 52736949000158;PROJETO M;300074422.0;Não Recomendado  
82785.0;6;SL\_8;0.85;Desenvolvimento de soluções tecnológicas para injeção de solados;LLM;2022.0;Transversal;Processo;DE - Desenvolvimento Experimental;CNPJ: 01287588000179;PROJETO N;301010510.0;Não Recomendado  
51064.0;7;SL\_8;0.87;Desenvolvimento de estabilizantes térmicos para compostos de PVC;LLM;2020.0;Química e Farmácia;Processo;DE - Desenvolvimento Experimental;CNPJ: 89850341000160;PROJETO O;300074422.0;Recomendado  
66184.0;7;SL\_8;0.85;Pacotes de estabilizantes térmicos para PVC com foco em redução de custos;LLM;2021.0;Química e Farmácia;Processo;DE - Desenvolvimento Experimental;CNPJ: 89850341000160;PROJETO P;121755221.0;Recomendado  
51044.0;8;SL\_8;0.86;Impacto de copolímeros olefínicos em bloco nas propriedades de compostos de EVA;LLM;2020.0;Química e Farmácia;Processo;DE - Desenvolvimento Experimental;CNPJ: 89850341000160;PROJETO Q;300074422.0;Não Recomendado  
66157.0;8;SL\_8;0.84;Estudos de aditivos para melhorar desempenho de compostos de EVA;LLM;2021.0;Eletroeletrônica;Processo;DE - Desenvolvimento Experimental;CNPJ: 89850341000160;PROJETO R;121755221.0;Recomendado  
"""  
  
try:  
 # Tenta carregar o arquivo real  
 df\_original = pd.read\_csv('resultados\_agrupamento/GRUPOS\_FINAL\_FILTRADO.csv', sep=';')  
 print("Arquivo 'GRUPOS\_FINAL\_FILTRADO.csv' carregado com sucesso.")  
except FileNotFoundError:  
 print("AVISO: Arquivo real não encontrado. Utilizando dados de amostra para demonstração.")  
 # Se falhar, usa a amostra  
 df\_original = pd.read\_csv(StringIO(csv\_data\_sample), sep=';')  
  
# 1. Filtrar pela origem 'LLM'  
df = df\_original[df\_original['origem\_agrupamento'] == 'LLM'].copy()  
print(f"Total de registros carregados: {len(df\_original):,}")  
print(f"Registros de origem 'LLM' filtrados: {len(df):,}")  
  
# 2. Criar a coluna 'grupo\_id'  
df['grupo\_id'] = df['grupo\_id\_final'].astype(str) + '\_' + df['sublote'].astype(str)  
  
# 3. Limpeza e transformação de dados  
df['ano\_referencia'] = pd.to\_numeric(df['ano\_referencia'], errors='coerce').astype('Int64')  
df['do\_id\_at'] = df['do\_id\_at'].astype(str)  
df['do\_resultado\_analise'] = df['do\_resultado\_analise'].str.strip()  
  
# Remover grupos com apenas um projeto, pois não há como medir divergência  
group\_counts = df['grupo\_id'].value\_counts()  
single\_project\_groups = group\_counts[group\_counts <= 1].index  
df = df[~df['grupo\_id'].isin(single\_project\_groups)].copy()  
print(f"Registros após remoção de grupos com 1 projeto: {len(df):,}")  
  
# 4. Exibir informações do DataFrame processado  
print("\nInformações do DataFrame processado:")  
df.info()  
  
print("\nAmostra dos dados preparados:")  
print(df[['projeto\_id', 'grupo\_id', 'setor', 'do\_id\_at', 'do\_resultado\_analise']].head())

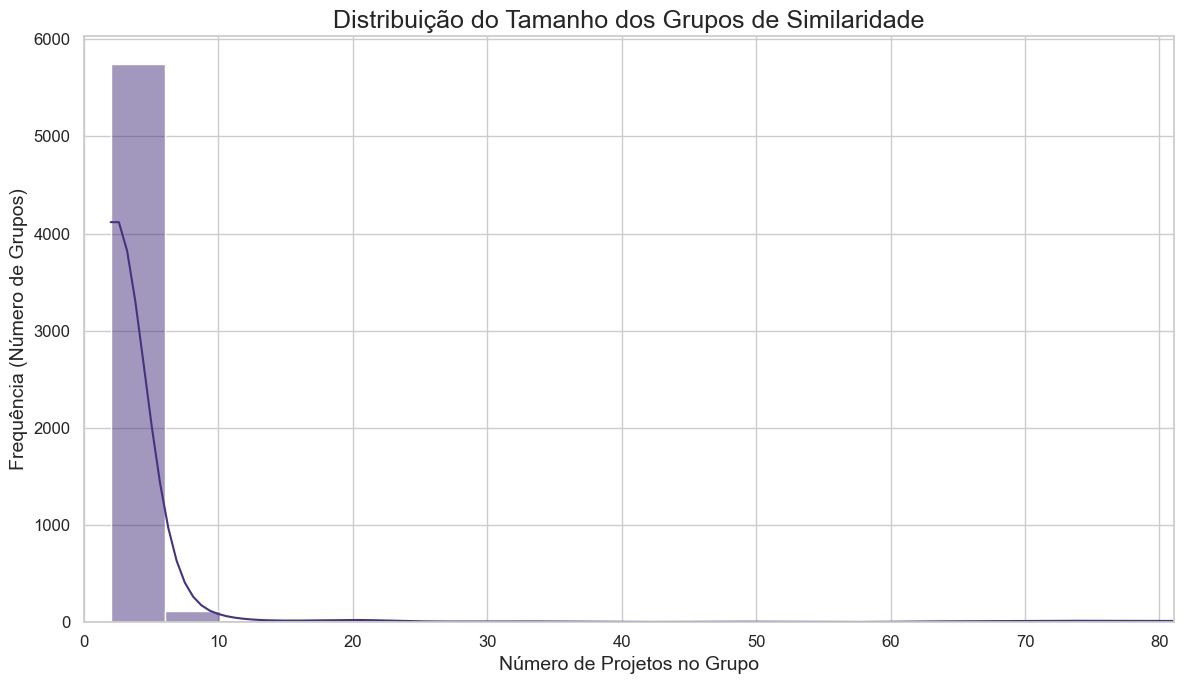
Arquivo 'GRUPOS\_FINAL\_FILTRADO.csv' carregado com sucesso.  
Total de registros carregados: 39,703  
Registros de origem 'LLM' filtrados: 28,499  
Registros após remoção de grupos com 1 projeto: 28,101  
  
Informações do DataFrame processado:  
<class 'pandas.core.frame.DataFrame'>  
Index: 28101 entries, 0 to 28498  
Data columns (total 31 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 projeto\_id 28082 non-null float64  
 1 grupo\_id\_final 28101 non-null int64   
 2 sublote 28087 non-null object   
 3 similaridade\_score 28074 non-null float64  
 4 justificativa\_agrupamento 28065 non-null object   
 5 origem\_agrupamento 28101 non-null object   
 6 ano\_referencia 28080 non-null Int64   
 7 setor 28080 non-null object   
 8 natureza 28080 non-null object   
 9 tipo\_pesquisa 28080 non-null object   
 10 empresa 28080 non-null object   
 11 projeto 28080 non-null object   
 12 do\_id\_at 28101 non-null object   
 13 do\_resultado\_analise 28080 non-null object   
 14 do\_justificativa\_padronizada 28080 non-null object   
 15 p\_resultado\_analise 24408 non-null object   
 16 p\_id\_analista\_mcti 24249 non-null float64  
 17 p\_justificativa\_padronizada 28080 non-null object   
 18 empresa\_do\_contestacao 5342 non-null object   
 19 do\_c\_id\_at 6483 non-null float64  
 20 do\_c\_resultado\_analise 8747 non-null object   
 21 do\_c\_justificativa\_padronizada 28080 non-null object   
 22 empresa\_parecer\_contestacao 5342 non-null object   
 23 p\_c\_id\_analista\_mcti 8929 non-null float64  
 24 p\_c\_resultado\_analise 8929 non-null object   
 25 p\_c\_justificativa\_padronizada 28080 non-null object   
 26 empresa\_recurso\_administrativo 0 non-null float64  
 27 ra\_id\_analista\_mcti 0 non-null float64  
 28 ra\_resultado\_analise 8929 non-null object   
 29 ra\_justificativa\_padronizada 28080 non-null object   
 30 grupo\_id 28101 non-null object   
dtypes: Int64(1), float64(7), int64(1), object(22)  
memory usage: 6.9+ MB  
  
Amostra dos dados preparados:  
 projeto\_id grupo\_id setor do\_id\_at do\_resultado\_analise  
0 49408.0 1\_SL\_8 Transversal 301043772.0 Recomendado  
1 51621.0 1\_SL\_8 Transversal 301043772.0 Não Recomendado  
2 62743.0 1\_SL\_8 Transversal 300074422.0 Não Recomendado  
3 59124.0 1\_SL\_8 Transversal 300804386.0 Não Recomendado  
4 63126.0 2\_SL\_8 Transversal 119427979.0 Não Recomendado

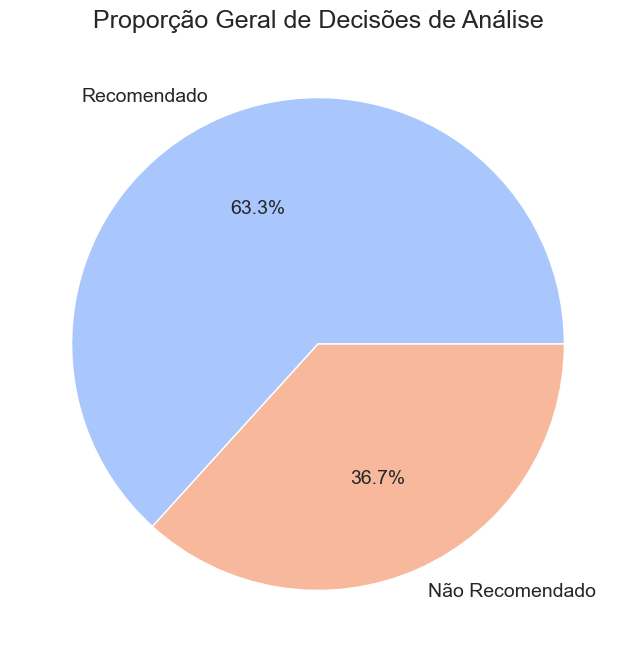
/tmp/ipykernel\_1155558/1237813749.py:43: DtypeWarning: Columns (2) have mixed types. Specify dtype option on import or set low\_memory=False.  
 df\_original = pd.read\_csv('resultados\_agrupamento/GRUPOS\_FINAL\_FILTRADO.csv', sep=';')

# Gráfico 1: Distribuição de Projetos por Setor  
plt.figure(figsize=(14, 8))  
sector\_order = df['setor'].value\_counts().index  
ax1 = sns.countplot(y='setor', data=df, order=sector\_order)  
ax1.set\_title('Distribuição de Projetos por Setor (Grupos LLM)')  
ax1.set\_xlabel('Número de Projetos')  
ax1.set\_ylabel('Setor')  
# Adiciona os valores nas barras  
for p in ax1.patches:  
 width = p.get\_width()  
 ax1.text(width + 0.1, p.get\_y() + p.get\_height() / 2,  
 f'{int(width)}', ha='left', va='center')  
plt.tight\_layout()  
plt.show()  
  
# Gráfico 2: Distribuição de Projetos por Ano de Referência  
plt.figure(figsize=(12, 7))  
year\_order = sorted(df['ano\_referencia'].dropna().astype(int).unique())  
ax2 = sns.countplot(x='ano\_referencia', data=df, order=year\_order)  
ax2.set\_title('Distribuição de Projetos por Ano (Grupos LLM)')  
ax2.set\_xlabel('Ano de Referência')  
ax2.set\_ylabel('Número de Projetos')  
# Adiciona os valores nas barras  
for p in ax2.patches:  
 height = p.get\_height()  
 ax2.text(p.get\_x() + p.get\_width() / 2., height + 0.1,  
 f'{int(height)}', ha='center', va='bottom')  
plt.tight\_layout()  
plt.show()  
  
# Gráfico 3: Histograma do Tamanho dos Grupos de Similaridade  
plt.figure(figsize=(12, 7))  
group\_sizes = df['grupo\_id'].value\_counts()  
ax3 = sns.histplot(group\_sizes, bins=30, kde=True)  
ax3.set\_title('Distribuição do Tamanho dos Grupos de Similaridade')  
ax3.set\_xlabel('Número de Projetos no Grupo')  
ax3.set\_ylabel('Frequência (Número de Grupos)')  
# Limitar o eixo x para melhor visualização, excluindo outliers extremos  
ax3.set\_xlim(0, group\_sizes.quantile(0.99))  
plt.tight\_layout()  
plt.show()  
  
# Gráfico 4: Proporção de Decisões de Análise  
plt.figure(figsize=(8, 8))  
decision\_counts = df['do\_resultado\_analise'].value\_counts()  
plt.pie(decision\_counts, labels=decision\_counts.index, autopct='%1.1f%%',  
 colors=sns.color\_palette("coolwarm", len(decision\_counts)),  
 wedgeprops={'edgecolor': 'white'}, textprops={'fontsize': 14})  
plt.title('Proporção Geral de Decisões de Análise')  
plt.show()



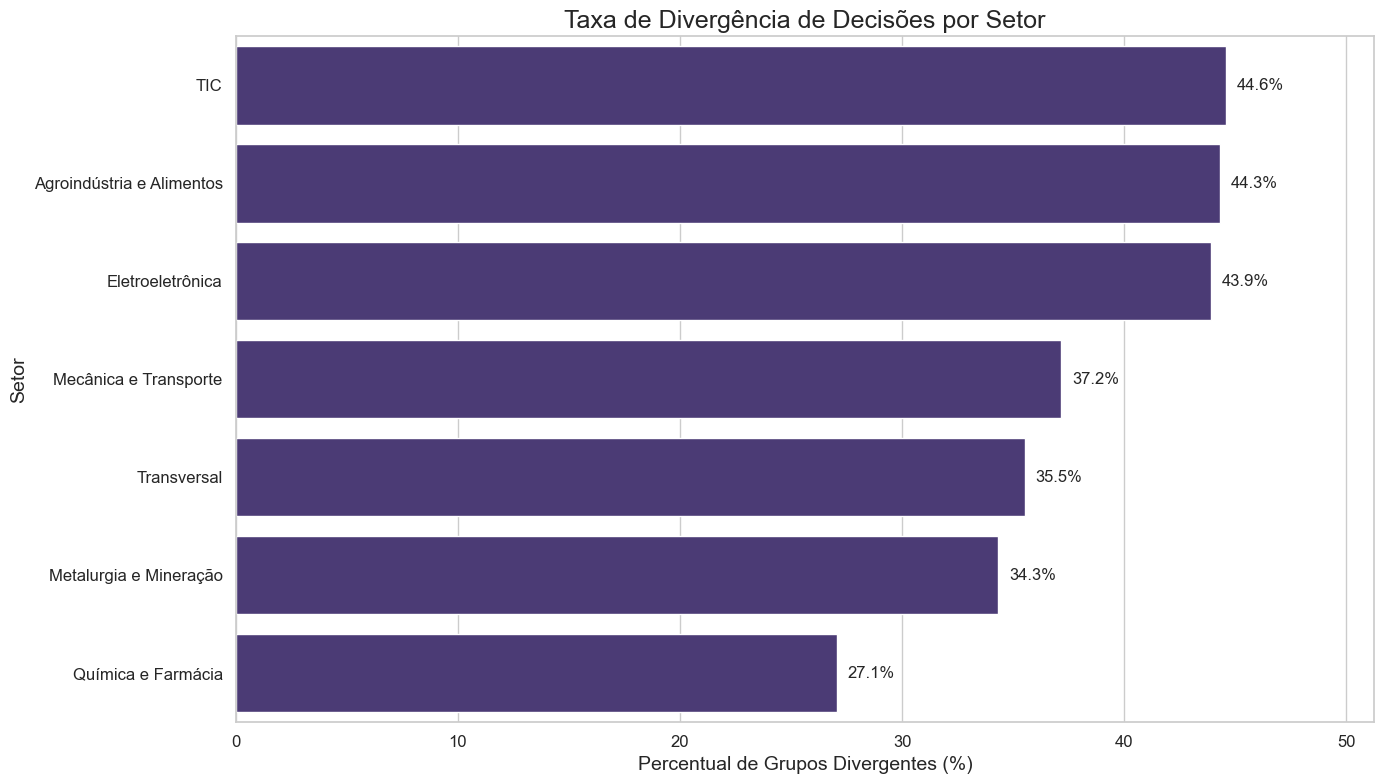






# Calcular a divergência para cada grupo  
divergence\_data = df.groupby('grupo\_id').agg(  
 total\_projetos=('projeto\_id', 'count'),  
 decisoes\_unicas=('do\_resultado\_analise', lambda x: x.nunique()),  
 setor=('setor', 'first')  
).reset\_index()  
  
# Identificar grupos divergentes  
divergence\_data['divergente'] = divergence\_data['decisoes\_unicas'] > 1  
  
# Calcular a taxa de divergência por setor  
sector\_divergence = divergence\_data.groupby('setor')['divergente'].value\_counts(normalize=True).unstack().fillna(0)  
if True in sector\_divergence.columns:  
 sector\_divergence['taxa\_divergencia'] = sector\_divergence[True] \* 100  
else:  
 sector\_divergence[True] = 0 # Adiciona a coluna se não existir  
 sector\_divergence['taxa\_divergencia'] = 0  
  
  
sector\_divergence = sector\_divergence.sort\_values('taxa\_divergencia', ascending=False)  
  
print("Taxa de Divergência por Setor (% de grupos com decisões conflitantes):")  
print(sector\_divergence[['taxa\_divergencia']].round(2))  
  
# Gráfico 5: Taxa de Divergência por Setor  
plt.figure(figsize=(14, 8))  
ax = sns.barplot(x=sector\_divergence['taxa\_divergencia'], y=sector\_divergence.index, orient='h')  
ax.set\_title('Taxa de Divergência de Decisões por Setor')  
ax.set\_xlabel('Percentual de Grupos Divergentes (%)')  
ax.set\_ylabel('Setor')  
# Adiciona os valores nas barras  
for p in ax.patches:  
 width = p.get\_width()  
 ax.text(width + 0.5, p.get\_y() + p.get\_height() / 2,  
 f'{width:.1f}%', ha='left', va='center')  
ax.set\_xlim(0, max(sector\_divergence['taxa\_divergencia']) \* 1.15) # Ajusta limite do eixo x  
plt.tight\_layout()  
plt.show()

Taxa de Divergência por Setor (% de grupos com decisões conflitantes):  
divergente taxa\_divergencia  
setor   
TIC 44.57  
Agroindústria e Alimentos 44.31  
Eletroeletrônica 43.91  
Mecânica e Transporte 37.18  
Transversal 35.52  
Metalurgia e Mineração 34.32  
Química e Farmácia 27.05



# Determinar a decisão majoritária para cada grupo  
majority\_decision = df.groupby('grupo\_id')['do\_resultado\_analise'].agg(lambda x: x.mode()[0] if not x.mode().empty else None).rename('decisao\_majoritaria')  
  
# Juntar a decisão majoritária de volta ao dataframe principal  
df\_majority = df.merge(majority\_decision, on='grupo\_id', how='left')  
  
# Calcular a consistência de cada avaliador  
df\_majority['consistente'] = df\_majority['do\_resultado\_analise'] == df\_majority['decisao\_majoritaria']  
  
# Contar avaliações e consistência por avaliador  
evaluator\_stats = df\_majority.groupby('do\_id\_at').agg(  
 total\_avaliacoes=('projeto\_id', 'count'),  
 avaliacoes\_consistentes=('consistente', 'sum')  
)  
  
# Calcular taxa de consistência  
evaluator\_stats['taxa\_consistencia'] = (evaluator\_stats['avaliacoes\_consistentes'] / evaluator\_stats['total\_avaliacoes']) \* 100  
  
# Calcular a taxa de aprovação por avaliador  
approval\_rate = df.groupby('do\_id\_at')['do\_resultado\_analise'].apply(lambda x: (x == 'Recomendado').mean() \* 100).rename('taxa\_aprovacao')  
evaluator\_stats = evaluator\_stats.join(approval\_rate)  
  
# Filtrar por avaliadores com um número mínimo de avaliações (ajustado para a amostra, idealmente >= 100)  
min\_avaliacoes = 2  
evaluators\_filtered = evaluator\_stats[evaluator\_stats['total\_avaliacoes'] >= min\_avaliacoes].sort\_values('taxa\_consistencia', ascending=False)  
  
print(f"Top Avaliadores por Taxa de Consistência (com >= {min\_avaliacoes} avaliações):")  
print(evaluators\_filtered.head(15).round(2))  
  
# Gráfico 6: Top 15 Avaliadores por Consistência  
plt.figure(figsize=(14, 9))  
# Usando o df completo para o top 15, mas mostrando apenas os disponíveis na amostra  
top\_15\_evaluators = evaluators\_filtered.head(15)  
ax1 = sns.barplot(x=top\_15\_evaluators['taxa\_consistencia'], y=top\_15\_evaluators.index, palette='crest\_r', orient='h')  
ax1.set\_title(f'Top Avaliadores por Taxa de Consistência Interna (mínimo {min\_avaliacoes} avaliações)')  
ax1.set\_xlabel('Taxa de Consistência (%)')  
ax1.set\_ylabel('ID do Avaliador')  
ax1.set\_xlim(min(50, top\_15\_evaluators['taxa\_consistencia'].min() - 5), 101)  
for p in ax1.patches:  
 ax1.text(p.get\_width() + 0.1, p.get\_y() + p.get\_height() / 2,  
 f'{p.get\_width():.1f}%', ha='left', va='center')  
plt.tight\_layout()  
plt.show()  
  
# Gráfico 7: Dispersão - Volume de Avaliações vs. Consistência  
plt.figure(figsize=(14, 9))  
ax2 = sns.scatterplot(  
 data=evaluators\_filtered,  
 x='total\_avaliacoes',  
 y='taxa\_consistencia',  
 size='total\_avaliacoes',  
 hue='taxa\_aprovacao',  
 palette='coolwarm\_r',  
 sizes=(100, 1500),  
 alpha=0.8  
)  
ax2.set\_title('Volume de Avaliações vs. Taxa de Consistência')  
ax2.set\_xlabel('Total de Avaliações')  
ax2.set\_ylabel('Taxa de Consistência (%)')  
ax2.legend(title='Taxa de Aprovação (%)', bbox\_to\_anchor=(1.05, 1), loc='upper left')  
plt.tight\_layout()  
plt.show()  
  
# Gráfico 8: Quadrantes de Perfil de Avaliador  
plt.figure(figsize=(14, 11))  
avg\_consistency = evaluators\_filtered['taxa\_consistencia'].mean()  
avg\_approval = evaluators\_filtered['taxa\_aprovacao'].mean()  
ax3 = sns.scatterplot(  
 data=evaluators\_filtered,  
 x='taxa\_aprovacao',  
 y='taxa\_consistencia',  
 size='total\_avaliacoes',  
 sizes=(100, 1500),  
 alpha=0.7,  
 hue='total\_avaliacoes',  
 palette='viridis\_r'  
)  
ax3.axvline(avg\_approval, color='red', linestyle='--', lw=1.5, label=f'Média Aprovação ({avg\_approval:.1f}%)')  
ax3.axhline(avg\_consistency, color='blue', linestyle='--', lw=1.5, label=f'Média Consistência ({avg\_consistency:.1f}%)')  
ax3.set\_title('Perfil dos Avaliadores: Consistência vs. Taxa de Aprovação')  
ax3.set\_xlabel('Taxa de Aprovação (%)')  
ax3.set\_ylabel('Taxa de Consistência (%)')  
# Anotações dos quadrantes  
plt.text(avg\_approval \* 1.02, avg\_consistency \* 1.01, 'Consistente e Aprovador', fontsize=12, color='darkgreen', alpha=0.9, ha='left')  
plt.text(avg\_approval \* 0.98, avg\_consistency \* 1.01, 'Consistente e Criterioso', fontsize=12, color='darkblue', alpha=0.9, ha='right')  
plt.text(avg\_approval \* 1.02, avg\_consistency \* 0.99, 'Inconsistente e Aprovador', fontsize=12, color='darkorange', alpha=0.9, ha='left', va='top')  
plt.text(avg\_approval \* 0.98, avg\_consistency \* 0.99, 'Inconsistente e Criterioso', fontsize=12, color='darkred', alpha=0.9, ha='right', va='top')  
plt.legend(title='Total de Avaliações', bbox\_to\_anchor=(1.05, 1), loc='upper left')  
plt.tight\_layout()  
plt.show()

Top Avaliadores por Taxa de Consistência (com >= 2 avaliações):  
 total\_avaliacoes avaliacoes\_consistentes taxa\_consistencia \  
do\_id\_at   
124343713.0 18 18 100.00   
129400658.0 5 5 100.00   
301364226.0 3 3 100.00   
300830730.0 153 147 96.08   
127789240.0 49 47 95.92   
300850231.0 288 276 95.83   
301643854.0 133 126 94.74   
300474232.0 36 34 94.44   
301643692.0 120 112 93.33   
107608291.0 88 82 93.18   
101611647.0 54 50 92.59   
301038133.0 73 67 91.78   
301282132.0 76 69 90.79   
123495530.0 204 185 90.69   
301099916.0 138 125 90.58   
  
 taxa\_aprovacao   
do\_id\_at   
124343713.0 100.00   
129400658.0 60.00   
301364226.0 100.00   
300830730.0 100.00   
127789240.0 95.92   
300850231.0 96.53   
301643854.0 95.49   
300474232.0 88.89   
301643692.0 99.17   
107608291.0 93.18   
101611647.0 98.15   
301038133.0 95.89   
301282132.0 100.00   
123495530.0 96.57   
301099916.0 86.96

/tmp/ipykernel\_1155558/89844228.py:34: FutureWarning:   
  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.  
  
 ax1 = sns.barplot(x=top\_15\_evaluators['taxa\_consistencia'], y=top\_15\_evaluators.index, palette='crest\_r', orient='h')

