Parte 1 - Análise das métricas das 15 IFs

In [1]:
```
import pandas as pd
import numpy as np
from google_play_scraper import Sort, reviews_all, reviews
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
from google_play_scraper import app

lista_bancos = ['br.com.gabba.Caixa',
'com.bradesco',
'com.itau',
'com.nu.production',
'br.com.bb.android',
'com.santander.app',
'br.com.original.bank',
'com.mercadopago.wallet',
'com.b2winc.amedigital',
'br.com.uol.ps.myaccount',
'br.com.intermedium',
'com.c6bank.app',
'com.btg.pactual.banking',
'br.com.neon',
'com.votorantim.bvpd']

df = list()

for banco in lista_bancos:
    result = app(banco,
    lang='pt', # definindo a linguagem que queremos pegar
    country='br' # Google Play de qual país?'
    )
    df.append(result)
```

In [3]:
```
df_new = pd.json_normalize(df)
```

In [4]:
```
# Função para expandir listas em novas colunas
def expand_list(row):
    for i, value in enumerate(row['histogram']):
        row[f'Nota_{i+1}'] = value
    return row

# Aplicando a função em cada linha do DataFrame
df_expanded = df_new.apply(expand_list, axis=1)

# Removendo a coluna original 'Itens'
df_final = df_expanded.drop('histogram', axis=1)

# Dicionário de substituições
substituicoes = {
    'Neon conta digital, cartão de crédito, empréstimo': 'Banco Neon',
    'americanas s.a.': 'Ame Digital',
    'Mercado Libre': 'Mercado Pago'
}

# Aplicando as substituições na coluna 'developer' do DataFrame
df_final['developer'] = df_final['developer'].replace(substituicoes)

print("\nDataFrame com listas expandidas em novas colunas:")
df_final.head()
```

```
DataFrame com listas expandidas em novas colunas:
```

Out[4]:

	title	description	descriptionHTML	summary	installs	minInstalls	realInstalls	score	rati
0	CAIXA	O aplicativo Caixa facilita seu dia a dia! Voc...	O aplicativo Caixa facilita seu dia a dia! Voc...	Quer mais de uma centena de possibilidades em ...	100.000.000+	100000000	171947748	4.673539	4551
1	Bradesco: Conta, Cartão e Pix!	Abra sua conta em poucos passos e comece já a ...	Abra sua conta em poucos passos e comece já a ...	Cartão, compras com cashback, empréstimos e ma...	100.000.000+	100000000	106018224	4.574296	4870
2	Banco Itaú: Conta, Cartão e +	Tudo o que você pode fazer pelo app:\r\n\r\n◉ ...	Tudo o que você pode fazer pelo app: ◉ ...	Cartão de crédito, cashback, investimentos, em...	50.000.000+	50000000	89462660	4.568687	4056
3	Nubank: conta, cartão e mais	O Nubank é a instituição financeira mais recom...	O Nubank é a instituição financeira mais recom...	Cartão de crédito, Empréstimos, Investimentos,...	100.000.000+	100000000	186374871	4.768120	3963
4	Banco do Brasil: Conta Digital	😜 Quer abrir uma conta digital grátis e acessa...	😜 Quer abrir uma conta digital grátis e acessa...	Cartão de Crédito, Pix, Empréstimo, Investimen...	100.000.000+	100000000	103238184	4.587459	6927

5 rows × 49 columns

In [5]:
```
# Total de votantes
df_final['Total votantes'] = df_final[['Nota_1', 'Nota_2', 'Nota_3', 'Nota_4', 'Nota_5']].sum(axis=1)

# O total de promotores
df_final['Total promotores'] = df_final['Nota_5']

# O total de neutros
df_final['Total neutros'] = df_final['Nota_4']

# Soma os valores das colunas de notas detratoras (Nota_1, Nota_2 e Nota_3) por linha
df_final['Total detratores'] = df_final[['Nota_1', 'Nota_2', 'Nota_3']].sum(axis=1)

# Calcula a Nota NPS Ajustada
df_final['Nota NPS Ajustada'] = (df_final['Total promotores'] - df_final['Total detratores']) / df_final

# Salva os resultados em um arquivo Excel
df_final.to_excel('resultados_app.xlsx')
```

In [6]:
```
df_graf_1 = df_final.sort_values(by='score', ascending=False)

df_graf_1 = df_graf_1[['score', 'developer']].reset_index()
```

In [ ]:
```
# Configurando o estilo do gráfico
sns.set(style="whitegrid")

# Criando o gráfico de barras
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=df_graf_1, x='developer', y='score', palette='viridis')

# Adicionando rótulos de dados com ajuste de posição
for index, row in df_graf_1.iterrows():
    ax.text(index, row['score'] + 0.05,  # Aumentando a altura para evitar sobreposição
            f"{round(row['score'], 2)}",
            ha='center', va='bottom', fontsize=10)

# Personalizando o gráfico
ax.set_title('Notas das Instituições Financeiras na Google Play', fontsize=16)
ax.set_xlabel('IF', fontsize=12)
ax.set_ylabel('Nota Google Play', fontsize=12)
```

```
# Rotacionando os rótulos do eixo X
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right', fontsize=10)

# Ajustando os limites do eixo Y para dar mais espaço
ax.set_ylim(0, df_graf_1['score'].max() + 0.5)

# Removendo a legenda
ax.legend_.remove() if ax.get_legend() else None

# Exibindo o gráfico
plt.show()
```

In [8]:
```
df_graf_1[['developer','score']]
```

Out[8]:

	developer	score
0	PagBank	4.842176
1	Mercado Pago	4.827662
2	Banco Inter SA	4.816691
3	Nu	4.768120
4	Banco Santander (Brasil) S.A.	4.765653
5	Ame Digital	4.765561
6	Banco BTG Pactual S.A.	4.751824
7	Caixa Econômica Federal	4.673539
8	Banco Neon	4.638668
9	Banco do Brasil SA	4.587459
10	Banco Bradesco SA	4.574296
11	Itaú Unibanco S. A.	4.568687
12	Banco Votorantim S/A	4.520145
13	Banco Original	3.938062
14	C6 Bank	3.573557

In [9]:
```
df_graf_2 = df_final.sort_values(by='realInstalls', ascending=False)
df_graf_2 = df_graf_2[['realInstalls', 'developer']].reset_index()
```

In [ ]:
```
# Convertendo as instalações para milhões
df_graf_2['realInstalls_milhoes'] = df_graf_2['realInstalls'] / 1_000_000

# Configurando o estilo do gráfico
sns.set(style="whitegrid")

# Criando o gráfico de barras
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=df_graf_2, x='developer', y='realInstalls_milhoes', palette='viridis')

# Adicionando rótulos de dados com ajuste de posição
for index, row in df_graf_2.iterrows():
    ax.text(index, row['realInstalls_milhoes'] + 0.1,  # Ajustando a posição do rótulo
            f"{round(row['realInstalls_milhoes'], 1)}MM",
            ha='center', va='bottom', fontsize=9)

# Personalizando o gráfico
ax.set_title('Instalações em milhões de usuários', fontsize=16)
ax.set_xlabel('IF', fontsize=12)
ax.set_ylabel('Instalações (milhões)', fontsize=12)

# Rotacionando os rótulos do eixo X
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right', fontsize=10)

# Ajustando os limites do eixo Y para dar mais espaço
ax.set_ylim(0, df_graf_2['realInstalls_milhoes'].max() + 10)
```

```
# Removendo a legenda (caso exista)
ax.legend_.remove() if ax.get_legend() else None

# Exibindo o gráfico
plt.show()
```

In [12]:
```
df_graf_3 = df_final.sort_values(by='Nota NPS Ajustada', ascending=False)
df_graf_3 = df_graf_3[['Nota NPS Ajustada', 'developer']].reset_index()
```

In [ ]:
```
# Configurando o estilo do gráfico
sns.set(style="whitegrid")

# Criando o gráfico de barras
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=df_graf_3, x='developer', y='Nota NPS Ajustada', palette='viridis')

# Adicionando rótulos de dados com ajuste de posição
for index, row in df_graf_3.iterrows():
    ax.text(index, row['Nota NPS Ajustada'] + 0.025,  # Ajustando a posição do rótulo
            f"{round(row['Nota NPS Ajustada'] * 100, 1)}%",  # Arredondando para 1 casa decimal e exibin
            ha='center', va='bottom', fontsize=9)

# Personalizando o gráfico
ax.set_title('Nota NPS Ajustada das IFs', fontsize=16)
ax.set_xlabel('IF', fontsize=12)
ax.set_ylabel('Nota NPS Ajustada (%)', fontsize=12)

# Rotacionando os rótulos do eixo X
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right', fontsize=10)

# Ajustando os limites do eixo Y para dar mais espaço
ax.set_ylim(0, df_graf_3['Nota NPS Ajustada'].max() + 0.15)

# Removendo a legenda (caso exista)
ax.legend_.remove() if ax.get_legend() else None

# Exibindo o gráfico
plt.show()
```

In [14]:
```
df_graf_3[['developer','Nota NPS Ajustada']]
```

Out[14]:

	developer	Nota NPS Ajustada
0	PagBank	0.876840
1	Mercado Pago	0.864618
2	Banco Inter SA	0.863272
3	Ame Digital	0.834709
4	Nu	0.830367
5	Banco BTG Pactual S.A.	0.828880
6	Banco Santander (Brasil) S.A.	0.828836
7	Caixa Econômica Federal	0.756135
8	Banco Neon	0.739217
9	Banco do Brasil SA	0.720485
10	Itaú Unibanco S. A.	0.712146
11	Banco Bradesco SA	0.705829
12	Banco Votorantim S/A	0.666248
13	Banco Original	0.387496
14	C6 Bank	0.181305

Parte 2 - Criação de modelos BERT, TD-IDF e Dicionarização dos comentários da google play

In [22]:
```python
# Função para coletar até 1000 comentários detratores (1-3 estrelas) mais relevantes
def coletar_comentarios_detratores(app_id, max_comentarios=1000):
    print(f"Coletando até {max_comentarios} comentários detratores para {app_id}...")

    comentarios_totais = []
    continuacao_token = None  # Inicializando o token de continuação para reviews paginadas

    try:
        while True:
            comentarios, continuacao_token = reviews(
                app_id,
                lang='pt',  # Idioma dos comentários
                country='br',  # País
                sort=Sort.MOST_RELEVANT,  # Ordenar pelos mais relevantes
                count=100,  # Número de comentários por lote (máximo de 100 por vez)
                continuation_token=continuacao_token  # Token para paginação
            )

            # Filtrar apenas comentários com notas de 1 a 3 (detratores)
            comentarios_detratores = [comentario for comentario in comentarios if comentario['score'] <=
            comentarios_totais.extend(comentarios_detratores)

            # Verificar se já temos 1000 comentários ou se não há mais comentários para buscar
            if len(comentarios_totais) >= max_comentarios or continuacao_token is None:
                break

    except Exception as e:
        print(f"Erro ao coletar comentários para {app_id}: {e}")
        return pd.DataFrame()  # Retorna DataFrame vazio em caso de erro

    # Limitar o número de comentários ao máximo solicitado (1000)
    comentarios_totais = comentarios_totais[:max_comentarios]

    # Convertendo a lista de dicionários em DataFrame do pandas
    df = pd.DataFrame(comentarios_totais)
    return df
```

In [244…
```python
# Função para coletar até 30 comentários detratores (1-3 estrelas) com mais de 5 palavras
def coletar_comentarios_detratores_teste(app_id, max_comentarios=30):
    print(f"Coletando até {max_comentarios} comentários detratores com mais de 5 palavras para {app_id}.

    comentarios_totais = []
    continuacao_token = None  # Inicializando o token de continuação para reviews paginadas

    try:
        while True:
            comentarios, continuacao_token = reviews(
                app_id,
                lang='pt',  # Idioma dos comentários
                country='br',  # País
                sort=Sort.NEWEST,  # Ordenar pelos mais novos
                count=100,  # Número de comentários por lote (máximo permitido)
                continuation_token=continuacao_token  # Token para paginação
            )

            # Filtrar apenas comentários com notas de 1 a 3 (detratores)
            comentarios_detratores = [
                comentario for comentario in comentarios
                if comentario['score'] <= 3 and len(comentario['content'].split()) > 5  # Mais de 5 pala
            ]
            comentarios_totais.extend(comentarios_detratores)

            # Verificar se já temos o número desejado de comentários ou se não há mais para buscar
            if len(comentarios_totais) >= max_comentarios or continuacao_token is None:
                break

    except Exception as e:
        print(f"Erro ao coletar comentários para {app_id}: {e}")
        return pd.DataFrame()  # Retorna DataFrame vazio em caso de erro
```

```
    # Limitar o número de comentários ao máximo solicitado (30)
    comentarios_totais = comentarios_totais[:max_comentarios]

    # Convertendo a lista de dicionários em DataFrame do pandas
    df = pd.DataFrame(comentarios_totais)
    return df
```

In [24]:
```
# Criar um DataFrame vazio para armazenar todos os comentários
df_comentarios = pd.DataFrame()
```

In [26]:
```
# Coletar comentários para cada aplicativo
for banco in lista_bancos:
    df_banco = coletar_comentarios_detratores(banco)

    if not df_banco.empty:
        df_banco['app_id'] = banco  # Adicionar uma coluna com o ID do app
        df_comentarios = pd.concat([df_comentarios, df_banco], ignore_index=True)
```

```
Coletando até 1000 comentários detratores para br.com.gabba.Caixa...
Coletando até 1000 comentários detratores para com.bradesco...
Coletando até 1000 comentários detratores para com.itau...
Coletando até 1000 comentários detratores para com.nu.production...
Coletando até 1000 comentários detratores para br.com.bb.android...
Coletando até 1000 comentários detratores para com.santander.app...
Coletando até 1000 comentários detratores para br.com.original.bank...
Coletando até 1000 comentários detratores para com.mercadopago.wallet...
Coletando até 1000 comentários detratores para com.b2winc.amedigital...
Coletando até 1000 comentários detratores para br.com.uol.ps.myaccount...
Coletando até 1000 comentários detratores para br.com.intermedium...
Coletando até 1000 comentários detratores para com.c6bank.app...
Coletando até 1000 comentários detratores para com.btg.pactual.banking...
Coletando até 1000 comentários detratores para br.com.neon...
Coletando até 1000 comentários detratores para com.votorantim.bvpd...
```

In [27]:
```
# Salvar os comentários em um arquivo Excel
df_comentarios.to_excel('comentarios_bancos_google_play.xlsx')
```

In [28]:
```
# Criar um DataFrame vazio para armazenar todos os comentários
df_comentarios_teste = pd.DataFrame()
```

In [245…
```
# Coletar comentários para cada aplicativo
for banco in lista_bancos:
    df_banco = coletar_comentarios_detratores_teste(banco)

    if not df_banco.empty:
        df_banco['app_id'] = banco  # Adicionar uma coluna com o ID do app
        df_comentarios_teste = pd.concat([df_comentarios_teste, df_banco], ignore_index=True)
```

```
Coletando até 30 comentários detratores com mais de 5 palavras para br.com.gabba.Caixa...
Coletando até 30 comentários detratores com mais de 5 palavras para com.bradesco...
Coletando até 30 comentários detratores com mais de 5 palavras para com.itau...
Coletando até 30 comentários detratores com mais de 5 palavras para com.nu.production...
Coletando até 30 comentários detratores com mais de 5 palavras para br.com.bb.android...
Coletando até 30 comentários detratores com mais de 5 palavras para com.santander.app...
Coletando até 30 comentários detratores com mais de 5 palavras para br.com.original.bank...
C:\Users\caio_\AppData\Local\Temp\ipykernel_21804\1506589469.py:7: FutureWarning: The behavior of DataFra
me concatenation with empty or all-NA entries is deprecated. In a future version, this will no longer exc
lude empty or all-NA columns when determining the result dtypes. To retain the old behavior, exclude the
relevant entries before the concat operation.
  df_comentarios_teste = pd.concat([df_comentarios_teste, df_banco], ignore_index=True)
Coletando até 30 comentários detratores com mais de 5 palavras para com.mercadopago.wallet...
C:\Users\caio_\AppData\Local\Temp\ipykernel_21804\1506589469.py:7: FutureWarning: The behavior of DataFra
me concatenation with empty or all-NA entries is deprecated. In a future version, this will no longer exc
lude empty or all-NA columns when determining the result dtypes. To retain the old behavior, exclude the
relevant entries before the concat operation.
  df_comentarios_teste = pd.concat([df_comentarios_teste, df_banco], ignore_index=True)
Coletando até 30 comentários detratores com mais de 5 palavras para com.b2winc.amedigital...
Coletando até 30 comentários detratores com mais de 5 palavras para br.com.uol.ps.myaccount...
Coletando até 30 comentários detratores com mais de 5 palavras para br.com.intermedium...
Coletando até 30 comentários detratores com mais de 5 palavras para com.c6bank.app...
Coletando até 30 comentários detratores com mais de 5 palavras para com.btg.pactual.banking...
Coletando até 30 comentários detratores com mais de 5 palavras para br.com.neon...
Coletando até 30 comentários detratores com mais de 5 palavras para com.votorantim.bvpd...
```

In [246…
```
# Salvar os comentários em um arquivo Excel
df_comentarios_teste.to_excel('comentarios_bancos_teste_new.xlsx')
```

In [ ]:
```
print("Coleta de comentários finalizada e salva no arquivo 'comentarios_bancos_teste.xlsx'.")
```

In [15]:
```
from sklearn.model_selection import train_test_split
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification, AdamW
from sklearn.metrics import confusion_matrix
from sklearn.utils.class_weight import compute_class_weight
from torch.utils.data import DataLoader, TensorDataset
from torch.nn import CrossEntropyLoss
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
import torch
import unicodedata
import emoji
import re
```

In [16]:
```
def remover_emojis(texto):
    return emoji.replace_emoji(texto, replace='')
```

In [17]:
```
# Função para remover acentuação
def remover_acentos(texto):
    return ''.join(c for c in unicodedata.normalize('NFKD', texto) if unicodedata.category(c) != 'Mn')
```

In [18]:
```
# Função para pré-processar os comentários
def preprocessar_texto(texto):

    # Remover emojis
    texto = remover_emojis(texto)

    # Remover URLs
    texto = re.sub(r'http\S+|www\S+|https\S+', '', texto)

    # Remover menções (@usuario)
    texto = re.sub(r'@\w+', '', texto)

    return texto
```

In [19]:
```
# Carregar o conjunto de dados rotulado (supondo que temos um dataset com categorias definidas)
df_rotulado = pd.read_excel(r"C:\Users\caio_\OneDrive\Documentos\Python Scripts\comentarios_bancos_teste
df_completo = pd.read_excel(r'C:\Users\caio_\OneDrive\Documentos\Python Scripts\comentarios_bancos_googl
```

In [22]:
```
df_rotulado['texto_processado'] = df_rotulado['content']
df_completo['texto_processado'] = df_completo['content']
```

In [23]:
```
# Pré-processar o texto dataframe rotulado
df_rotulado['texto_processado'] = df_rotulado['texto_processado'].apply(preprocessar_texto)
df_rotulado['texto_processado'] = df_rotulado['texto_processado'].apply(remover_emojis)

# Pré-processar o texto dataframe completo
df_completo['texto_processado'] = df_completo['texto_processado'].apply(preprocessar_texto)
df_completo['texto_processado'] = df_completo['texto_processado'].apply(remover_emojis)
```

In [24]:
```
df_rotulado['texto_processado']
```

Out[24]:
```
0      Esses dias ficou o dia inteiro fora do ar........
1      Muito bom o aplicativo só gostaria de saber co...
2      Dei 1 estrela pq não tem como dar 0. Do nada n...
3                Por enquanto nenhuma muito difícil usar
4      Não funciona. Simplesmente trava e não há quem...
                             ...
445    Falta alguns detalhes,como exemplo email com d...
446    Lixo de aplicativo, tento pagar a parcela do c...
447    Infelizmente a última atualização deixou a des...
448    Orrivel tenho uma chave Pix nesse banco que eu...
449    Não sei como entrar no app coloco a senha mais...
Name: texto_processado, Length: 450, dtype: object
```

In [25]:
```
df_completo['texto_processado']
```

Out[25]:
```
0        PÉSSIMO Erros constantes do app dizendo que nã...
1        Muitos erros na aba de acessar os contratos de...
2        O aplicativo da Caixa Econômica Federal deixa ...
3        Como pode um banco tão grande oscilar tanto em...
4        O aplicativo era ótimo, aí inventaram essa últ...
                              ...
22995    Simplesmente não funciona, celular Samsung And...
22996    Estou 4 dias que o aplicativo não funciona tod...
22997    Aplicativo trava MUITO e/ou NÃO funciona (fiz ...
22998    Horrível. Quando recebi o e-mail para instalar...
22999    Na tem um suporte bom é péssimo para soluciona...
Name: texto_processado, Length: 23000, dtype: object
```

In [26]:
```
df_rotulado.dropna(subset=['avaliacao humana'], inplace=True)
```

In [27]:
```
df_rotulado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 435 entries, 0 to 449
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   Unnamed: 0            435 non-null    int64         
 1   reviewId              435 non-null    object        
 2   userName              435 non-null    object        
 3   userImage             435 non-null    object        
 4   content               435 non-null    object        
 5   score                 435 non-null    int64         
 6   thumbsUpCount         435 non-null    int64         
 7   reviewCreatedVersion  358 non-null    object        
 8   at                    435 non-null    datetime64[ns]
 9   replyContent          255 non-null    object        
 10  repliedAt             255 non-null    datetime64[ns]
 11  appVersion            358 non-null    object        
 12  app_id                435 non-null    object        
 13  avaliacao humana      435 non-null    object        
 14  texto_processado      435 non-null    object        
dtypes: datetime64[ns](2), int64(3), object(10)
memory usage: 54.4+ KB
```

In [28]:
```
df_graf_cat = pd.pivot_table(df_rotulado, index='avaliacao humana', aggfunc='count')
```

In [29]:
```
df_graf_cat = pd.pivot_table(df_rotulado, index='avaliacao humana', aggfunc='count')
df_graf_cat = df_graf_cat.reset_index()
df_graf_cat = df_graf_cat.sort_values(by='app_id', ascending=False)
```

In [30]:
```
df_graf_cat[['avaliacao humana', 'app_id']]
```

Out[30]:

	avaliacao humana	app_id
5	Serviços e produtos	145
1	Erros de entrada e autenticação	91
6	Usabilidade e acessibilidade	75
7	Velocidade e desempenho	45
2	Erros transacionais	35
3	Funcionalidade e integração	21
0	Design visual e personalização	17
4	Segurança	6

In [ ]:
```
sns.set(style="whitegrid")

# Criar gráfico de barras horizontais com ajustes nos rótulos de dados
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=df_graf_cat,
    y="avaliacao humana",
```

```
    x="app_id",
    palette="viridis"
)

# Adicionar rótulos de dados sem casas decimais
for container in ax.containers:
    ax.bar_label(container, fmt="%.0f", label_type="edge", padding=3, fontsize=9)

# Configurar título e rótulos
plt.title("Categorias avaliadas no teste", fontsize=14, pad=15)
plt.xlabel("Quantidade", fontsize=12)
plt.ylabel("Categoria", fontsize=12)
plt.tight_layout()

# Mostrar o gráfico
plt.show()
```

In [32]:
```
# Conversão da categoria para valores numéricos
label_mapping = {label: idx for idx, label in enumerate(df_rotulado['avaliacao humana'].unique())}
df_rotulado['num_categoria'] = df_rotulado['avaliacao humana'].map(label_mapping)
```

In [ ]:
```
# Inicializar o tokenizer
tokenizer = DistilBertTokenizer.from_pretrained("neuralmind/bert-base-portuguese-cased")

# Divisão dos dados
train_texts, val_texts, train_labels, val_labels = train_test_split(
    df_rotulado["texto_processado"].tolist(),
    df_rotulado["num_categoria"].tolist(),
    test_size=0.3,
    random_state=77
)

# Tokenizar os textos
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=128, return_tensors="
val_encodings = tokenizer(val_texts, truncation=True, padding=True, max_length=128, return_tensors="pt")
train_labels = torch.tensor(train_labels)
val_labels = torch.tensor(val_labels)
```

In [34]:
```
# Calcular ponderação das classes
class_weights = compute_class_weight(
    class_weight="balanced",
    classes=list(range(len(label_mapping))),
    y=df_rotulado["num_categoria"].tolist()
)
class_weights = torch.tensor(class_weights, dtype=torch.float).to("cuda" if torch.cuda.is_available() el

# Inicializar o modelo
model_bert = DistilBertForSequenceClassification.from_pretrained(
    "neuralmind/bert-base-portuguese-cased",
    num_labels=len(label_mapping)
)

# Configurar otimizador
optimizer = AdamW(model_bert.parameters(), lr=1e-5)

# Definir dispositivo
device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
model_bert.to(device)

# Preparar os dados
train_dataset = TensorDataset(train_encodings['input_ids'], train_encodings['attention_mask'], train_lab
val_dataset = TensorDataset(val_encodings['input_ids'], val_encodings['attention_mask'], val_labels)

train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=16)

# Loop de treinamento
epochs = 5
loss_fn = CrossEntropyLoss(weight=class_weights)  # Incorporar os pesos das classes

for epoch in range(epochs):
    model_bert.train()
    total_loss = 0
```

```
    for batch in train_loader:
        inputs, masks, labels = [x.to(device) for x in batch]
        optimizer.zero_grad()

        outputs = model_bert(input_ids=inputs, attention_mask=masks, labels=labels)
        loss = loss_fn(outputs.logits, labels)  # Usar loss ponderada
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    avg_loss = total_loss / len(train_loader)
    print(f"Epoch {epoch + 1}, Loss média: {avg_loss:.4f}")
```

```
You are using a model of type bert to instantiate a model of type distilbert. This is not supported for a
ll configurations of models and can yield errors.
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at neu
ralmind/bert-base-portuguese-cased and are newly initialized: ['classifier.bias', 'classifier.weight', 'e
mbeddings.LayerNorm.bias', 'embeddings.LayerNorm.weight', 'embeddings.position_embeddings.weight', 'embed
dings.word_embeddings.weight', 'pre_classifier.bias', 'pre_classifier.weight', 'transformer.layer.0.atten
tion.k_lin.bias', 'transformer.layer.0.attention.k_lin.weight', 'transformer.layer.0.attention.out_lin.bi
as', 'transformer.layer.0.attention.out_lin.weight', 'transformer.layer.0.attention.q_lin.bias', 'transfo
rmer.layer.0.attention.q_lin.weight', 'transformer.layer.0.attention.v_lin.bias', 'transformer.layer.0.at
tention.v_lin.weight', 'transformer.layer.0.ffn.lin1.bias', 'transformer.layer.0.ffn.lin1.weight', 'trans
former.layer.0.ffn.lin2.bias', 'transformer.layer.0.ffn.lin2.weight', 'transformer.layer.0.output_layer_n
orm.bias', 'transformer.layer.0.output_layer_norm.weight', 'transformer.layer.0.sa_layer_norm.bias', 'tra
nsformer.layer.0.sa_layer_norm.weight', 'transformer.layer.1.attention.k_lin.bias', 'transformer.layer.1.
attention.k_lin.weight', 'transformer.layer.1.attention.out_lin.bias', 'transformer.layer.1.attention.out
_lin.weight', 'transformer.layer.1.attention.q_lin.bias', 'transformer.layer.1.attention.q_lin.weight',
'transformer.layer.1.attention.v_lin.bias', 'transformer.layer.1.attention.v_lin.weight', 'transformer.la
yer.1.ffn.lin1.bias', 'transformer.layer.1.ffn.lin1.weight', 'transformer.layer.1.ffn.lin2.bias', 'transf
ormer.layer.1.ffn.lin2.weight', 'transformer.layer.1.output_layer_norm.bias', 'transformer.layer.1.output
_layer_norm.weight', 'transformer.layer.1.sa_layer_norm.bias', 'transformer.layer.1.sa_layer_norm.weigh
t', 'transformer.layer.10.attention.k_lin.bias', 'transformer.layer.10.attention.k_lin.weight', 'transfor
mer.layer.10.attention.out_lin.bias', 'transformer.layer.10.attention.out_lin.weight', 'transformer.laye
r.10.attention.q_lin.bias', 'transformer.layer.10.attention.q_lin.weight', 'transformer.layer.10.attentio
n.v_lin.bias', 'transformer.layer.10.attention.v_lin.weight', 'transformer.layer.10.ffn.lin1.bias', 'tran
sformer.layer.10.ffn.lin1.weight', 'transformer.layer.10.ffn.lin2.bias', 'transformer.layer.10.ffn.lin2.w
eight', 'transformer.layer.10.output_layer_norm.bias', 'transformer.layer.10.output_layer_norm.weight',
'transformer.layer.10.sa_layer_norm.bias', 'transformer.layer.10.sa_layer_norm.weight', 'transformer.laye
r.11.attention.k_lin.bias', 'transformer.layer.11.attention.k_lin.weight', 'transformer.layer.11.attentio
n.out_lin.bias', 'transformer.layer.11.attention.out_lin.weight', 'transformer.layer.11.attention.q_lin.b
ias', 'transformer.layer.11.attention.q_lin.weight', 'transformer.layer.11.attention.v_lin.bias', 'transf
ormer.layer.11.attention.v_lin.weight', 'transformer.layer.11.ffn.lin1.bias', 'transformer.layer.11.ffn.l
in1.weight', 'transformer.layer.11.ffn.lin2.bias', 'transformer.layer.11.ffn.lin2.weight', 'transformer.l
ayer.11.output_layer_norm.bias', 'transformer.layer.11.output_layer_norm.weight', 'transformer.layer.11.s
a_layer_norm.bias', 'transformer.layer.11.sa_layer_norm.weight', 'transformer.layer.2.attention.k_lin.bia
s', 'transformer.layer.2.attention.k_lin.weight', 'transformer.layer.2.attention.out_lin.bias', 'transfor
mer.layer.2.attention.out_lin.weight', 'transformer.layer.2.attention.q_lin.bias', 'transformer.layer.2.a
ttention.q_lin.weight', 'transformer.layer.2.attention.v_lin.bias', 'transformer.layer.2.attention.v_lin.
weight', 'transformer.layer.2.ffn.lin1.bias', 'transformer.layer.2.ffn.lin1.weight', 'transformer.layer.
2.ffn.lin2.bias', 'transformer.layer.2.ffn.lin2.weight', 'transformer.layer.2.output_layer_norm.bias', 't
ransformer.layer.2.output_layer_norm.weight', 'transformer.layer.2.sa_layer_norm.bias', 'transformer.laye
r.2.sa_layer_norm.weight', 'transformer.layer.3.attention.k_lin.bias', 'transformer.layer.3.attention.k_l
in.weight', 'transformer.layer.3.attention.out_lin.bias', 'transformer.layer.3.attention.out_lin.weight',
'transformer.layer.3.attention.q_lin.bias', 'transformer.layer.3.attention.q_lin.weight', 'transformer.la
yer.3.attention.v_lin.bias', 'transformer.layer.3.attention.v_lin.weight', 'transformer.layer.3.ffn.lin1.
bias', 'transformer.layer.3.ffn.lin1.weight', 'transformer.layer.3.ffn.lin2.bias', 'transformer.layer.3.f
fn.lin2.weight', 'transformer.layer.3.output_layer_norm.bias', 'transformer.layer.3.output_layer_norm.wei
ght', 'transformer.layer.3.sa_layer_norm.bias', 'transformer.layer.3.sa_layer_norm.weight', 'transformer.
layer.4.attention.k_lin.bias', 'transformer.layer.4.attention.k_lin.weight', 'transformer.layer.4.attenti
on.out_lin.bias', 'transformer.layer.4.attention.out_lin.weight', 'transformer.layer.4.attention.q_lin.bi
as', 'transformer.layer.4.attention.q_lin.weight', 'transformer.layer.4.attention.v_lin.bias', 'transform
er.layer.4.attention.v_lin.weight', 'transformer.layer.4.ffn.lin1.bias', 'transformer.layer.4.ffn.lin1.we
ight', 'transformer.layer.4.ffn.lin2.bias', 'transformer.layer.4.ffn.lin2.weight', 'transformer.layer.4.o
utput_layer_norm.bias', 'transformer.layer.4.output_layer_norm.weight', 'transformer.layer.4.sa_layer_nor
m.bias', 'transformer.layer.4.sa_layer_norm.weight', 'transformer.layer.5.attention.k_lin.bias', 'transfo
rmer.layer.5.attention.k_lin.weight', 'transformer.layer.5.attention.out_lin.bias', 'transformer.layer.5.
attention.out_lin.weight', 'transformer.layer.5.attention.q_lin.bias', 'transformer.layer.5.attention.q_l
in.weight', 'transformer.layer.5.attention.v_lin.bias', 'transformer.layer.5.attention.v_lin.weight', 'tr
ansformer.layer.5.ffn.lin1.bias', 'transformer.layer.5.ffn.lin1.weight', 'transformer.layer.5.ffn.lin2.bi
as', 'transformer.layer.5.ffn.lin2.weight', 'transformer.layer.5.output_layer_norm.bias', 'transformer.la
yer.5.output_layer_norm.weight', 'transformer.layer.5.sa_layer_norm.bias', 'transformer.layer.5.sa_layer_
norm.weight', 'transformer.layer.6.attention.k_lin.bias', 'transformer.layer.6.attention.k_lin.weight',
'transformer.layer.6.attention.out_lin.bias', 'transformer.layer.6.attention.out_lin.weight', 'transforme
r.layer.6.attention.q_lin.bias', 'transformer.layer.6.attention.q_lin.weight', 'transformer.layer.6.atten
tion.v_lin.bias', 'transformer.layer.6.attention.v_lin.weight', 'transformer.layer.6.ffn.lin1.bias', 'tra
nsformer.layer.6.ffn.lin1.weight', 'transformer.layer.6.ffn.lin2.bias', 'transformer.layer.6.ffn.lin2.wei
ght', 'transformer.layer.6.output_layer_norm.bias', 'transformer.layer.6.output_layer_norm.weight', 'tran
sformer.layer.6.sa_layer_norm.bias', 'transformer.layer.6.sa_layer_norm.weight', 'transformer.layer.7.att
ention.k_lin.bias', 'transformer.layer.7.attention.k_lin.weight', 'transformer.layer.7.attention.out_lin.
bias', 'transformer.layer.7.attention.out_lin.weight', 'transformer.layer.7.attention.q_lin.bias', 'trans
former.layer.7.attention.q_lin.weight', 'transformer.layer.7.attention.v_lin.bias', 'transformer.layer.7.
attention.v_lin.weight', 'transformer.layer.7.ffn.lin1.bias', 'transformer.layer.7.ffn.lin1.weight', 'tra
nsformer.layer.7.ffn.lin2.bias', 'transformer.layer.7.ffn.lin2.weight', 'transformer.layer.7.output_layer
_norm.bias', 'transformer.layer.7.output_layer_norm.weight', 'transformer.layer.7.sa_layer_norm.bias', 't
ransformer.layer.7.sa_layer_norm.weight', 'transformer.layer.8.attention.k_lin.bias', 'transformer.layer.
8.attention.k_lin.weight', 'transformer.layer.8.attention.out_lin.bias', 'transformer.layer.8.attention.o
ut_lin.weight', 'transformer.layer.8.attention.q_lin.bias', 'transformer.layer.8.attention.q_lin.weight',
```

```
'transformer.layer.8.attention.v_lin.bias', 'transformer.layer.8.attention.v_lin.weight', 'transformer.layer.8.ffn.lin1.bias', 'transformer.layer.8.ffn.lin1.weight', 'transformer.layer.8.ffn.lin2.bias', 'transformer.layer.8.ffn.lin2.weight', 'transformer.layer.8.output_layer_norm.bias', 'transformer.layer.8.output_layer_norm.weight', 'transformer.layer.8.sa_layer_norm.bias', 'transformer.layer.8.sa_layer_norm.weight', 'transformer.layer.9.attention.k_lin.bias', 'transformer.layer.9.attention.k_lin.weight', 'transformer.layer.9.attention.out_lin.bias', 'transformer.layer.9.attention.out_lin.weight', 'transformer.layer.9.attention.q_lin.bias', 'transformer.layer.9.attention.q_lin.weight', 'transformer.layer.9.attention.v_lin.bias', 'transformer.layer.9.attention.v_lin.weight', 'transformer.layer.9.ffn.lin1.bias', 'transformer.layer.9.ffn.lin1.weight', 'transformer.layer.9.ffn.lin2.bias', 'transformer.layer.9.ffn.lin2.weight', 'transformer.layer.9.output_layer_norm.bias', 'transformer.layer.9.output_layer_norm.weight', 'transformer.layer.9.sa_layer_norm.bias', 'transformer.layer.9.sa_layer_norm.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
c:\Users\caio_\anaconda3\Lib\site-packages\transformers\optimization.py:591: FutureWarning: This implementation of AdamW is deprecated and will be removed in a future version. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warning=True` to disable this warning
  warnings.warn(
Epoch 1, Loss média: 2.1544
Epoch 2, Loss média: 2.0599
Epoch 3, Loss média: 2.0757
Epoch 4, Loss média: 2.0310
Epoch 5, Loss média: 2.0273
```

In [65]:

```python
# Avaliação do modelo
model_bert.eval()
predictions, true_labels = [], []

for batch in val_loader:
    inputs, masks, labels = [x.to(device) for x in batch]
    with torch.no_grad():
        outputs = model_bert(input_ids=inputs, attention_mask=masks)
        logits = outputs.logits
        predictions.extend(torch.argmax(logits, dim=1).cpu().tolist())
        true_labels.extend(labels.cpu().tolist())

# Calculando a matriz de confusão
conf_matrix = confusion_matrix(true_labels, predictions)
print(conf_matrix)
print(classification_report(true_labels, predictions, target_names=label_mapping.keys()))

# Calculando o classification report
report = classification_report(true_labels, predictions, target_names=label_mapping.keys(), output_dict=

# Convertendo o classification report em um DataFrame
report_df = pd.DataFrame(report).transpose()

# Exibindo a tabela
print(report_df)

# Se quiser salvar a tabela em um arquivo CSV
report_df.to_csv("classification_report_DistilBERT.csv", index=True)

# Salvar o modelo e o tokenizer
model_bert.save_pretrained("modelo_distilbert")
tokenizer.save_pretrained("modelo_distilbert")
```

```
[[ 0 13  0  0  3  0  0  0]
 [ 0 29  4  0  6  0  0  0]
 [ 0 11  6  0  9  0  0  0]
 [ 0 13  1  2  7  0  0  0]
 [ 0  6  0  0  7  0  0  1]
 [ 0  5  0  0  0  0  0  0]
 [ 0  3  0  0  3  0  0  0]
 [ 0  1  0  0  1  0  0  0]]
                                 precision    recall  f1-score   support

            Erros transacionais       0.00      0.00      0.00        16
            Serviços e produtos       0.36      0.74      0.48        39
Erros de entrada e autenticação       0.55      0.23      0.32        26
   Usabilidade e acessibilidade       1.00      0.09      0.16        23
        Velocidade e desempenho       0.19      0.50      0.28        14
 Design visual e personalização       0.00      0.00      0.00         5
    Funcionalidade e integração       0.00      0.00      0.00         6
                      Segurança       0.00      0.00      0.00         2

                       accuracy                           0.34       131
                      macro avg       0.26      0.20      0.16       131
                   weighted avg       0.41      0.34      0.27       131

                                 precision    recall  f1-score     support
Erros transacionais               0.000000  0.000000  0.000000   16.000000
Serviços e produtos               0.358025  0.743590  0.483333   39.000000
Erros de entrada e autenticação   0.545455  0.230769  0.324324   26.000000
Usabilidade e acessibilidade      1.000000  0.086957  0.160000   23.000000
Velocidade e desempenho           0.194444  0.500000  0.280000   14.000000
Design visual e personalização    0.000000  0.000000  0.000000    5.000000
Funcionalidade e integração       0.000000  0.000000  0.000000    6.000000
Segurança                         0.000000  0.000000  0.000000    2.000000
accuracy                          0.335878  0.335878  0.335878    0.335878
macro avg                         0.262240  0.195164  0.155957  131.000000
weighted avg                      0.411198  0.335878  0.266278  131.000000
```

```
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[65]:
```
('modelo_distilbert\\tokenizer_config.json',
 'modelo_distilbert\\special_tokens_map.json',
 'modelo_distilbert\\vocab.txt',
 'modelo_distilbert\\added_tokens.json')
```

In [43]:
```
import matplotlib
print(matplotlib.__version__)
```

```
3.8.0
```

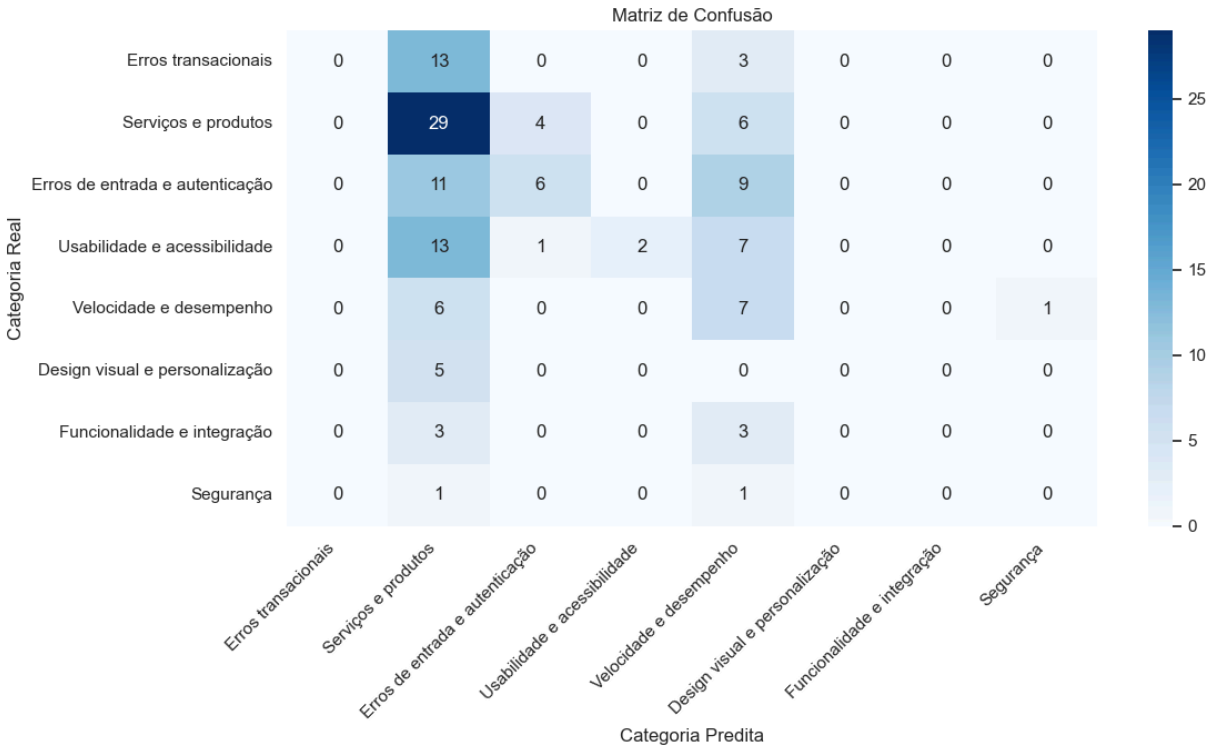
In [45]:
```
import matplotlib
```

In [48]:
```
# Calculando a matriz de confusão
conf_matrix = confusion_matrix(true_labels, predictions)

# Criando o gráfico
```

```
plt.figure(figsize=(12, 7))
sns.heatmap(conf_matrix,
            annot= True,
            fmt="d",
            cmap="Blues",
            xticklabels=label_mapping.keys(),
            yticklabels=label_mapping.keys()
            )
plt.title("Matriz de Confusão")
plt.xlabel("Categoria Predita")
plt.ylabel("Categoria Real")
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)

# Exibir a matriz de confusão
plt.tight_layout()
plt.show()
```



In [49]:
```
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

In [ ]:
```
# Baixar recursos necessários do NLTK
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Inicializando ferramentas
stop_words = set(stopwords.words('portuguese'))  # Stopwords em português
lemmatizer = WordNetLemmatizer()
```

In [51]:
```
# Função de pré-processamento
def preprocess_text(text):
    # Converter para minúsculas
    text = text.lower()

    # Remover URLs
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)

    # Remover números
    text = re.sub(r'\d+', '', text)

    # Remover pontuações
```

```
    text = text.translate(str.maketrans('', '', string.punctuation))

    # Tokenização
    tokens = word_tokenize(text)

    # Remover stopwords e lematizar
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]

    # Reunir palavras em um texto novamente
    processed_text = ' '.join(tokens)

    return processed_text
```

In [52]:
```
df_rotulado['texto_processado_tfidf'] = df_rotulado['content']
df_completo['texto_processado_tfidf'] = df_completo['content']
```

In [53]:
```
# Pré-processar o texto dataframe rotulado
df_rotulado['texto_processado_tfidf'] = df_rotulado['texto_processado'].apply(preprocess_text)
df_rotulado['texto_processado_tfidf'] = df_rotulado['texto_processado'].apply(remover_emojis)

# Pré-processar o texto dataframe completo
df_completo['texto_processado_tfidf'] = df_completo['texto_processado_tfidf'].apply(preprocess_text)
df_completo['texto_processado_tfidf'] = df_completo['texto_processado_tfidf'].apply(remover_emojis)
```

In [ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report, accuracy_score

# Vetorização com TF-IDF
vectorizer = TfidfVectorizer(max_features=500, ngram_range=(1, 3))
X = vectorizer.fit_transform(df_rotulado['texto_processado_tfidf'])
y = df_rotulado['avaliacao humana']

# Divisão de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=77)

# Dicionário para armazenar modelos e seus resultados
modelos = {
    "Regressão Logística": LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=500, r
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=77),
    "Naive Bayes": MultinomialNB(),
    "SVM (Linear Kernel)": LinearSVC(max_iter=3000, random_state=77)
}

# Loop para treinar e avaliar cada modelo
resultados = {}

for nome, modelo in modelos.items():
    print(f"Treinando e avaliando o modelo: {nome}")
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)

    # Calcular métricas
    acuracia = accuracy_score(y_test, y_pred)
    print(f"Acurácia de {nome}: {acuracia:.4f}")
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))

    # Salvar resultados
    resultados[nome] = {
        "modelo": modelo,
        "acuracia": acuracia,
        "classification_report": classification_report(y_test, y_pred, output_dict=True)
    }

# Selecionar o modelo com maior precisão
melhor_modelo = max(resultados, key=lambda x: resultados[x]['acuracia'])

print(f"\nO melhor modelo é: {melhor_modelo} com uma acurácia de {resultados[melhor_modelo]['acuracia']:
```

```python
# Exemplo de uso do modelo vencedor:
modelo_vencedor = resultados[melhor_modelo]["modelo"]
melhor_modelo = modelo

# Calculando o classification report
report = classification_report(y_test, y_pred, output_dict=True)

# Convertendo o classification report em um DataFrame
report_df = pd.DataFrame(report).transpose()

# Exibindo a tabela
print(report_df)

# Se quiser salvar a tabela em um arquivo CSV
report_df.to_csv("classification_report_TF-IDF.csv", index=True)

import pickle

# Salvar o modelo vencedor e o vetor TF-IDF
with open("melhor_modelo.pkl", "wb") as f:
    pickle.dump(melhor_modelo, f)

with open("vetor_tfidf.pkl", "wb") as f:
    pickle.dump(vectorizer, f)

print("Melhor modelo e vetor TF-IDF salvos com sucesso!")
```

```
Treinando e avaliando o modelo: Regressão Logística
Acurácia de Regressão Logística: 0.4885
                               precision    recall  f1-score   support

 Design visual e personalização       0.00      0.00      0.00         5
Erros de entrada e autenticação       0.54      0.54      0.54        26
            Erros transacionais       0.00      0.00      0.00        16
    Funcionalidade e integração       0.00      0.00      0.00         6
                      Segurança       0.00      0.00      0.00         2
            Serviços e produtos       0.46      0.92      0.61        39
   Usabilidade e acessibilidade       0.45      0.43      0.44        23
        Velocidade e desempenho       1.00      0.29      0.44        14

                       accuracy                           0.49       131
                      macro avg       0.31      0.27      0.25       131
                   weighted avg       0.43      0.49      0.41       131

[[ 0  0  0  0  0  4  1  0]
 [ 0 14  0  0  0  9  3  0]
 [ 0  4  0  0  0 10  2  0]
 [ 0  0  0  0  0  5  1  0]
 [ 0  0  0  0  0  2  0  0]
 [ 0  3  0  0  0 36  0  0]
 [ 0  4  0  0  0  9 10  0]
 [ 0  1  0  0  0  4  5  4]]
Treinando e avaliando o modelo: Random Forest
```

```
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
Acurácia de Random Forest: 0.5191
                                precision    recall  f1-score   support

 Design visual e personalização       0.00      0.00      0.00         5
Erros de entrada e autenticação       0.50      0.69      0.58        26
            Erros transacionais       0.00      0.00      0.00        16
    Funcionalidade e integração       0.00      0.00      0.00         6
                      Segurança       0.00      0.00      0.00         2
            Serviços e produtos       0.54      0.87      0.67        39
   Usabilidade e acessibilidade       0.38      0.39      0.38        23
        Velocidade e desempenho       0.88      0.50      0.64        14

                       accuracy                           0.52       131
                      macro avg       0.29      0.31      0.28       131
                   weighted avg       0.42      0.52      0.45       131

[[ 0  1  0  0  0  0  3  1]
 [ 0 18  0  0  0  5  3  0]
 [ 0  7  0  0  0  6  3  0]
 [ 0  1  0  0  0  4  1  0]
 [ 0  1  0  0  0  1  0  0]
 [ 0  4  0  0  0 34  1  0]
 [ 0  4  0  0  0 10  9  0]
 [ 0  0  0  0  0  3  4  7]]
Treinando e avaliando o modelo: Naive Bayes
Acurácia de Naive Bayes: 0.4198
                                precision    recall  f1-score   support

 Design visual e personalização       0.00      0.00      0.00         5
Erros de entrada e autenticação       0.52      0.50      0.51        26
            Erros transacionais       0.00      0.00      0.00        16
    Funcionalidade e integração       0.00      0.00      0.00         6
                      Segurança       0.00      0.00      0.00         2
            Serviços e produtos       0.38      0.92      0.53        39
   Usabilidade e acessibilidade       0.50      0.17      0.26        23
        Velocidade e desempenho       1.00      0.14      0.25        14

                       accuracy                           0.42       131
                      macro avg       0.30      0.22      0.19       131
                   weighted avg       0.41      0.42      0.33       131

[[ 0  0  0  0  0  5  0  0]
 [ 0 13  0  0  0 13  0  0]
 [ 0  3  0  0  0 12  1  0]
 [ 0  0  0  0  0  6  0  0]
 [ 0  0  0  0  0  2  0  0]
 [ 0  3  0  0  0 36  0  0]
 [ 0  5  0  0  0 14  4  0]
 [ 0  1  0  0  0  8  3  2]]
Treinando e avaliando o modelo: SVM (Linear Kernel)
Acurácia de SVM (Linear Kernel): 0.5496
                                precision    recall  f1-score   support

 Design visual e personalização       0.50      0.20      0.29         5
Erros de entrada e autenticação       0.48      0.46      0.47        26
            Erros transacionais       0.40      0.12      0.19        16
    Funcionalidade e integração       0.00      0.00      0.00         6
                      Segurança       0.00      0.00      0.00         2
            Serviços e produtos       0.63      0.92      0.75        39
   Usabilidade e acessibilidade       0.48      0.61      0.54        23
        Velocidade e desempenho       0.64      0.50      0.56        14

                       accuracy                           0.55       131
                      macro avg       0.39      0.35      0.35       131
                   weighted avg       0.50      0.55      0.51       131

[[ 1  1  0  0  0  1  1  1]
 [ 0 12  1  2  0  6  3  2]
 [ 1  3  2  0  0  6  3  1]
 [ 0  1  0  0  0  2  3  0]
 [ 0  0  0  0  0  1  1  0]
 [ 0  3  0  0  0 36  0  0]
 [ 0  4  2  0  0  3 14  0]
 [ 0  1  0  0  0  2  4  7]]
```

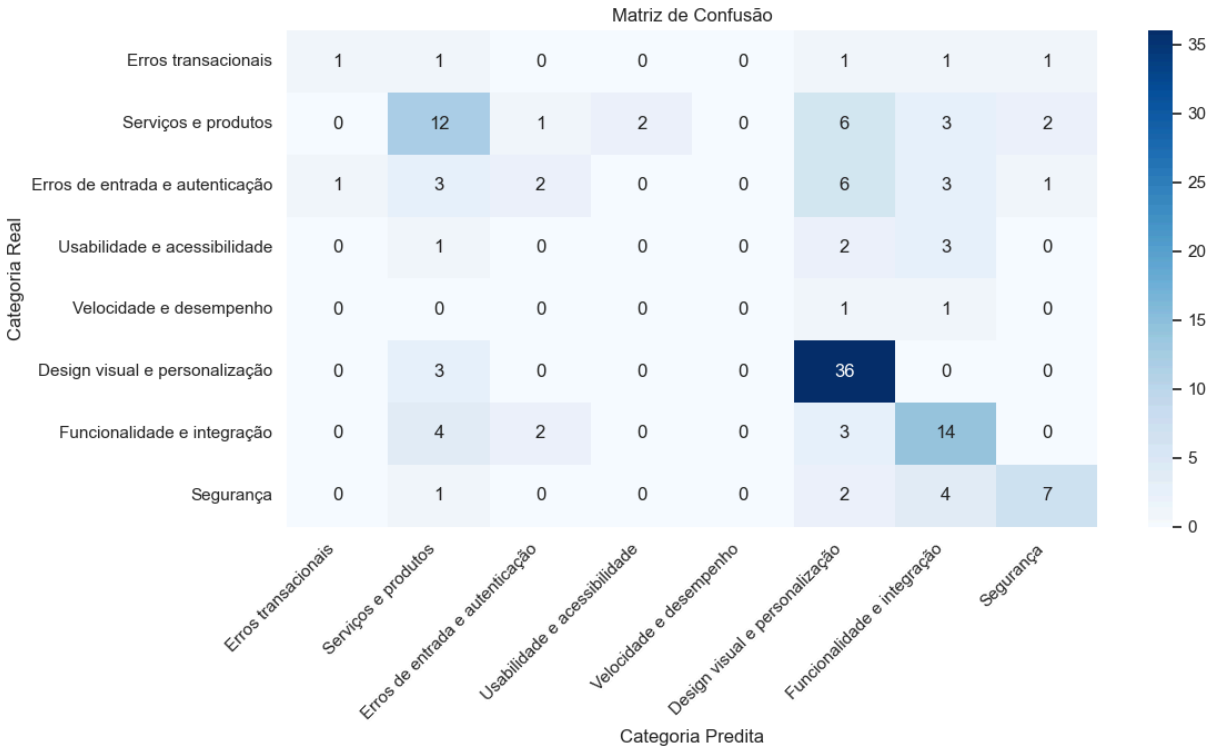
```
O melhor modelo é: SVM (Linear Kernel) com uma acurácia de 0.5496
Melhor modelo e vetor TF-IDF salvos com sucesso!
```

```
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\caio_\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarnin
g: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `z
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [55]:
```
# Calculando a matriz de confusão
conf_matrix = confusion_matrix(y_test, y_pred)

# Criando o gráfico
plt.figure(figsize=(12, 7))
sns.heatmap(conf_matrix,
            annot= True,
            fmt="d",
            cmap="Blues",
            xticklabels=label_mapping.keys(),
            yticklabels=label_mapping.keys()
            )
plt.title("Matriz de Confusão")
plt.xlabel("Categoria Predita")
plt.ylabel("Categoria Real")
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)

# Exibir a matriz de confusão
plt.tight_layout()
plt.show()
```



In [56]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

# Carregar o modelo
with open("melhor_modelo.pkl", "rb") as f:
    modelo_carregado = pickle.load(f)

# Carregar o vetor TF-IDF
with open("vetor_tfidf.pkl", "rb") as f:
    vetor_carregado = pickle.load(f)
```

In [57]:
```
# Verificar o tipo do modelo carregado
print(type(modelo_carregado))
```

```
<class 'sklearn.svm._classes.LinearSVC'>
```

In [ ]:
```
# Dicionário de palavras específicas
dicionario_categorias = {
    "Segurança": ["fraude", "golpe", "invasão", "senha", "segurança", "inseguro", "insegura", "clonado"]
    "Usabilidade e Acessibilidade": ["menu", "dificuldade", "acessível", "confuso", "navegação"],
    "Velocidade e Desempenho": ["lento", "travado", "demora", "carregamento", "bugado", "bug", "bugs", "
    "Design e Personalização": ["layout", "visual", "cores", "aparência", "confusão", "poluído", "design
    "Erros Transacionais": ["erro", "pagamento", "boleto", "cancelado", "transação", " fazer pix", "tran
    "Serviços e Produtos": ["tarifa", "taxa", "atendimento", "produto", "serviço", "crédito", "limite",
```

```
    "Erros de entrada e autenticação": ["login", "senha", "acesso", "autenticar", "erro", "biometria", "
    "Funcionalidade e Integração": ["pix", "apple pay", "google pay", "compatibilidade", "funcionalidade
}
```

In [59]:
```
# Tokenizar a base real
real_encodings = tokenizer(df_completo["texto_processado"].tolist(), truncation=True, padding=True, max_
real_dataset = torch.utils.data.TensorDataset(real_encodings['input_ids'], real_encodings['attention_mas
real_loader = torch.utils.data.DataLoader(real_dataset, batch_size=16)
```

In [ ]:
```
# Função para prever categoria usando o dicionário
def predict_with_dictionary(text, dicionario_categorias):
    for categoria, palavras in dicionario_categorias.items():
        for palavra in palavras:
            if palavra in text.lower():  # Verifica se a palavra chave está no texto
                return categoria
    return "Outros"  # Caso não encontre nenhuma palavra chave

# Mapeamento de índices para nomes de categorias
index_to_category = {
    0: "Design e Personalização",
    1: "Funcionalidade e integração",
    2: "Erros de entrada e autenticação",
    3: "Erros transacionais",
    4: "Segurança",
    5: "Serviços e produtos",
    6: "Usabilidade e Acessibilidade",
    7: "Velocidade e Desempenho"
}
```

In [61]:
```
# 4. Função para prever usando o modelo BERT
def predict_with_bert(text, tokenizer, model_bert):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=128)
    outputs = model_bert(**inputs)
    logits = outputs.logits
    prediction_index = torch.argmax(logits, dim=1).item()
    prediction_category = index_to_category.get(prediction_index, "Categoria desconhecida")  # Obter o n
    return prediction_category
```

In [62]:
```
# 5. Função para prever com o modelo TF-IDF
def predict_with_tfidf(text, vetor_carregado, modelo_carregado):
    X = vetor_carregado.transform([text])
    prediction = modelo_carregado.predict(X)
    return prediction[0]
```

In [63]:
```
# 6. Processamento e previsão para a base real
def processar_base_real(df_completo):
    # Inicialize as listas de previsões
    bert_predictions = []
    dict_predictions = []
    tfidf_predictions = []

    for text in df_completo['texto_processado']:
        # 6.1 Previsão com BERT
        bert_pred = predict_with_bert(text, tokenizer, model_bert)
        bert_predictions.append(bert_pred)

        # 6.2 Previsão com Dicionário
        dict_pred = predict_with_dictionary(text, dicionario_categorias)
        dict_predictions.append(dict_pred)

    for text in df_completo['texto_processado_tfidf']:
        # 6.3 Previsão com TF-IDF
        tfidf_pred = predict_with_tfidf(text, vetor_carregado, modelo_carregado)
        tfidf_predictions.append(tfidf_pred)

    # 7. Adicionar as previsões no DataFrame
    df_completo['Previsão_BERT'] = bert_predictions
    df_completo['Previsão_Dicionário'] = dict_predictions
    df_completo['Previsão_TFIDF'] = tfidf_predictions

    return df_completo
```

In [64]:

```python
# Processar a base real e adicionar as previsões
df_completo_processado = processar_base_real(df_completo)

# Exibir o DataFrame final com as previsões
print(df_completo_processado.head())

# 9. Salvando o DataFrame com as previsões
df_completo_processado.to_excel('base_real_com_previsoes.xlsx')
```

```
   Unnamed: 0                              reviewId                 userName  \
0           0  0209e7e1-5393-4771-936b-9fb568fa3340             ROGER CARDOSO
1           1  82bb8b06-6d42-478b-b834-e736c6b26ca5              Pietra Elias
2           2  471a1b12-f8a2-4eb6-9a7f-cd33bf37ee7a  Douglas Ramos dos Santos
3           3  10105335-3acd-46aa-8fc3-429a00e451aa      Jsrodrigues20 araujo
4           4  e5317cb0-3190-48c4-9545-2003f18633ae                Kleiton LR

                                      userImage  \
0  https://play-lh.googleusercontent.com/a-/ALV-U...
1  https://play-lh.googleusercontent.com/a-/ALV-U...
2  https://play-lh.googleusercontent.com/a-/ALV-U...
3  https://play-lh.googleusercontent.com/a/ACg8oc...
4  https://play-lh.googleusercontent.com/a-/ALV-U...

                                        content  score  thumbsUpCount  \
0  PÉSSIMO Erros constantes do app dizendo que nã...      1            131
1  Muitos erros na aba de acessar os contratos de...      1             16
2  O aplicativo da Caixa Econômica Federal deixa ...      1             94
3  Como pode um banco tão grande oscilar tanto em...      1              6
4  O aplicativo era ótimo, aí inventaram essa últ...      1            547

  reviewCreatedVersion                   at  \
0                5.6.0 2024-11-11 16:26:53
1                5.6.0 2024-11-19 23:53:51
2                5.6.0 2024-11-13 19:21:27
3                5.6.0 2024-11-20 06:26:23
4                5.6.0 2024-11-02 20:33:00

                                     replyContent           repliedAt  \
0  Olá! Gostaríamos de entender melhor o que ocor... 2024-11-11 16:27:11
1  Olá! Gostaríamos de entender melhor o que ocor... 2024-11-19 23:52:11
2  Olá! Gostaríamos de entender melhor o que ocor... 2024-11-13 19:21:43
3  Olá! Gostaríamos de entender melhor o que ocor... 2024-11-20 06:25:11
4  Olá! Gostaríamos de entender melhor o que ocor... 2024-11-02 20:33:41

  appVersion              app_id  \
0      5.6.0  br.com.gabba.Caixa
1      5.6.0  br.com.gabba.Caixa
2      5.6.0  br.com.gabba.Caixa
3      5.6.0  br.com.gabba.Caixa
4      5.6.0  br.com.gabba.Caixa

                                 texto_processado  \
0  PÉSSIMO Erros constantes do app dizendo que nã...
1  Muitos erros na aba de acessar os contratos de...
2  O aplicativo da Caixa Econômica Federal deixa ...
3  Como pode um banco tão grande oscilar tanto em...
4  O aplicativo era ótimo, aí inventaram essa últ...

                            texto_processado_tfidf  \
0  péssimo erros constantes app dizendo internet ...
1  muitos erros aba acessar contratos habitação c...
2  aplicativo caixa econômica federal deixa basta...
3  pode banco tão grande oscilar tanto algo simpl...
4  aplicativo ótimo aí inventaram última atualiza...

                Previsão_BERT           Previsão_Dicionário  \
0  Funcionalidade e integração       Problemas Transacionais
1  Funcionalidade e integração       Problemas Transacionais
2  Funcionalidade e integração  Usabilidade e Acessibilidade
3  Funcionalidade e integração       Problemas Transacionais
4  Funcionalidade e integração       Problemas Transacionais

                   Previsão_TFIDF
0  Erros de entrada e autenticação
1     Usabilidade e acessibilidade
2      Funcionalidade e integração
3  Erros de entrada e autenticação
4     Usabilidade e acessibilidade
```

In [105…

```
df_completo_processado
df_distintos = df_completo_processado.drop_duplicates(['reviewId']).reset_index(drop=True)

df_distintos
```

Out[105...

	Unnamed: 0	reviewId	userName	userImage	content	score	thumbsU
0	0	0209e7e1-5393-4771-936b-9fb568fa3340	ROGER CARDOSO	https://play-lh.googleusercontent.com/a-/ALV-U...	PÉSSIMO Erros constantes do app dizendo que nã...	1	
1	1	82bb8b06-6d42-478b-b834-e736c6b26ca5	Pietra Elias	https://play-lh.googleusercontent.com/a-/ALV-U...	Muitos erros na aba de acessar os contratos de...	1	
2	2	471a1b12-f8a2-4eb6-9a7f-cd33bf37ee7a	Douglas Ramos dos Santos	https://play-lh.googleusercontent.com/a-/ALV-U...	O aplicativo da Caixa Econômica Federal deixa ...	1	
3	3	10105335-3acd-46aa-8fc3-429a00e451aa	Jsrodrigues20 araujo	https://play-lh.googleusercontent.com/a/ACg8oc...	Como pode um banco tão grande oscilar tanto em...	1	
4	4	e5317cb0-3190-48c4-9545-2003f18633ae	Kleiton LR	https://play-lh.googleusercontent.com/a-/ALV-U...	O aplicativo era ótimo, aí inventaram essa últ...	1	
...	...	...	...	...	...	...	...
14998	22995	f138016a-6f42-4877-9203-56c867bb8ea9	Renato Lima	https://play-lh.googleusercontent.com/a/ACg8oc...	Simplesmente não funciona, celular Samsung And...	1	
14999	22996	5a4359ba-a839-4375-9183-173b20f322b5	Ivonete Gouveia	https://play-lh.googleusercontent.com/a/ACg8oc...	Estou 4 dias que o aplicativo não funciona tod...	1	
15000	22997	cbd8e5bb-f6c3-4df1-a55b-980574fbf537	marco aurelio c s	https://play-lh.googleusercontent.com/a/ACg8oc...	Aplicativo trava MUITO e/ou NÃO funciona (fiz ...	1	
15001	22998	153631b1-4f66-40d9-8d02-c67bcb337651	Amilde Moreira	https://play-lh.googleusercontent.com/a/ACg8oc...	Horrível. Quando recebi o e-mail para instalar...	1	
15002	22999	2ed70070-a3f3-4177-9e64-7fe0c9de6cc6	Gilberto Vieira Dos Santos	https://play-lh.googleusercontent.com/a-/ALV-U...	Na tem um suporte bom é péssimo para soluciona...	1	

15003 rows × 18 columns

In [75]:
```
df_graf_cat_final_BERT = pd.pivot_table(df_distintos, index='Previsão_BERT', aggfunc='count')
df_graf_cat_final_BERT = df_graf_cat_final_BERT.reset_index()
df_graf_cat_final_BERT = df_graf_cat_final_BERT.sort_values(by='app_id', ascending=False)
```

In [78]:
```
df_graf_cat_final_BERT = df_graf_cat_final_BERT[['Previsão_BERT', 'app_id']]
```

In [79]:
```
df_graf_cat_final_BERT
```

Out[79]:

	Previsão_BERT	app_id
0	Funcionalidade e integração	14116
3	Segurança	812
1	Problemas de entrada e autenticação	61
2	Problemas transacionais	12
4	Velocidade e Desempenho	2

In [ ]:
```
sns.set(style="whitegrid")

# Renomeando as categorias conforme solicitado
df_graf_cat_final_BERT["Previsão_BERT"] = df_graf_cat_final_BERT["Previsão_BERT"].replace({
    "Problemas de entrada e autenticação": "Erros de entrada e autenticação",
    "Problemas Transacionais": "Erros transacionais"
})

# Criar gráfico de barras horizontais com ajustes nos rótulos de dados
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=df_graf_cat_final_BERT,
    y="Previsão_BERT",
    x="app_id",
    palette="viridis"
)

# Adicionar rótulos de dados sem casas decimais
for container in ax.containers:
    ax.bar_label(container, fmt="%.0f", label_type="edge", padding=3, fontsize=9)

# Configurar título e rótulos
plt.title("Categorias avaliadas no modelo DistilBERT", fontsize=14, pad=15)
plt.xlabel("Quantidade", fontsize=12)
plt.ylabel("Categoria", fontsize=12)
plt.tight_layout()

# Mostrar o gráfico
plt.show()
```

In [ ]:
```
df_graf_cat_final_tf = pd.pivot_table(df_distintos, index='Previsão_TFIDF', aggfunc='count')
df_graf_cat_final_tf = df_graf_cat_final_tf.reset_index()
df_graf_cat_final_tf = df_graf_cat_final_tf.sort_values(by='app_id', ascending=False)
df_graf_cat_final_tf = df_graf_cat_final_tf[['Previsão_TFIDF', 'app_id']]
df_graf_cat_final_tf
sns.set(style="whitegrid")

# Criar gráfico de barras horizontais com ajustes nos rótulos de dados
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=df_graf_cat_final_tf,
    y="Previsão_TFIDF",
    x="app_id",
    palette="viridis"
)

# Adicionar rótulos de dados sem casas decimais
for container in ax.containers:
    ax.bar_label(container, fmt="%.0f", label_type="edge", padding=3, fontsize=9)

# Configurar título e rótulos
plt.title("Categorias avaliadas no modelo de TF-IDF", fontsize=14, pad=15)
plt.xlabel("Quantidade", fontsize=12)
plt.ylabel("Categoria", fontsize=12)
plt.tight_layout()

# Mostrar o gráfico
plt.show()
```

In [111…
```
df_graf_cat_final_tf
```

Out[111...

	Previsão_TFIDF	app_id
5	Serviços e produtos	5513
1	Erros de entrada e autenticação	4007
6	Usabilidade e acessibilidade	3320
7	Velocidade e desempenho	1711
2	Erros transacionais	278
3	Funcionalidade e integração	104
0	Design visual e personalização	67
4	Segurança	3

In [ ]:
```
df_graf_cat_final_dic = pd.pivot_table(df_distintos, index='Previsão_Dicionário', aggfunc='count')

df_graf_cat_final_dic = df_graf_cat_final_dic.reset_index()
df_graf_cat_final_dic = df_graf_cat_final_dic.sort_values(by='app_id', ascending=False)

# Renomeando as categorias conforme solicitado
df_graf_cat_final_dic["Previsão_Dicionário"] = df_graf_cat_final_dic["Previsão_Dicionário"].replace({
    "Problemas de entrada e autenticação": "Erros de entrada e autenticação",
    "Problemas Transacionais": "Erros transacionais"
})

df_graf_cat_final_dic = df_graf_cat_final_dic[['Previsão_Dicionário', 'app_id']]
df_graf_cat_final_dic
sns.set(style="whitegrid")

# Criar gráfico de barras horizontais com ajustes nos rótulos de dados
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=df_graf_cat_final_dic,
    y="Previsão_Dicionário",
    x="app_id",
    palette="viridis"
)

# Adicionar rótulos de dados sem casas decimais
for container in ax.containers:
    ax.bar_label(container, fmt="%.0f", label_type="edge", padding=3, fontsize=9)

# Configurar título e rótulos
plt.title("Categorias avaliadas no modelo de Dicionarização", fontsize=14, pad=15)
plt.xlabel("Quantidade", fontsize=12)
plt.ylabel("Categoria", fontsize=12)
plt.tight_layout()

# Mostrar o gráfico
plt.show()
```

In [87]:
```
df_graf_cat_final_dic
```

Out[87]:

	Previsão_Dicionário	app_id
2	Outros	3227
3	Problemas Transacionais	2992
6	Serviços e Produtos	2432
8	Velocidade e Desempenho	2192
5	Segurança	2028
4	Problemas de entrada e autenticação	711
1	Funcionalidade e Integração	605
7	Usabilidade e Acessibilidade	539
0	Design e Personalização	277

In [108…]
```
# Cálculo do total geral
total_geral = df_graf_cat_final_dic["app_id"].sum()

# Cálculo do percentual de "Outros"
quantidade_outros = df_graf_cat_final_dic.loc[df_graf_cat_final_dic["Previsão_Dicionário"] == "Outros",
percentual_outros = (quantidade_outros / total_geral) * 100

# Criando a coluna de percentual
df_graf_cat_final_dic["Percentual (%)"] = (df_graf_cat_final_dic["app_id"] / total_geral) * 100
df_graf_cat_final_dic["Percentual (%)"] = df_graf_cat_final_dic["Percentual (%)"].round(1)

# Exibindo a tabela
df_graf_cat_final_dic
```

Out[108…]

	Previsão_Dicionário	app_id	Percentual (%)
2	Outros	3227	21.5
3	Erros transacionais	2992	19.9
6	Serviços e Produtos	2432	16.2
8	Velocidade e Desempenho	2192	14.6
5	Segurança	2028	13.5
4	Erros de entrada e autenticação	711	4.7
1	Funcionalidade e Integração	605	4.0
7	Usabilidade e Acessibilidade	539	3.6
0	Design e Personalização	277	1.8

In [109…]
```
# Removendo a categoria "Outros"
df_sem_outros = df_graf_cat_final_dic[df_graf_cat_final_dic["Previsão_Dicionário"] != "Outros"]

# Recalculando o total geral (sem "Outros")
total_geral_sem_outros = df_sem_outros["app_id"].sum()

# Recalculando os percentuais
df_sem_outros["Percentual (%)"] = (df_sem_outros["app_id"] / total_geral_sem_outros) * 100

# Exibindo a tabela final
df_sem_outros
```

```
C:\Users\caio_\AppData\Local\Temp\ipykernel_17976\309878343.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
  df_sem_outros["Percentual (%)"] = (df_sem_outros["app_id"] / total_geral_sem_outros) * 100
```

Out[109…]

	Previsão_Dicionário	app_id	Percentual (%)
3	Erros transacionais	2992	25.407609
6	Serviços e Produtos	2432	20.652174
8	Velocidade e Desempenho	2192	18.614130
5	Segurança	2028	17.221467
4	Erros de entrada e autenticação	711	6.037704
1	Funcionalidade e Integração	605	5.137568
7	Usabilidade e Acessibilidade	539	4.577106
0	Design e Personalização	277	2.352242

In [ ]:
```
df_sem_outros = df_sem_outros[['Previsão_Dicionário', 'app_id']]
df_sem_outros
sns.set(style="whitegrid")

# Criar gráfico de barras horizontais com ajustes nos rótulos de dados
```

```
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=df_sem_outros,
    y="Previsão_Dicionário",
    x="app_id",
    palette="viridis"
)

# Adicionar rótulos de dados sem casas decimais
for container in ax.containers:
    ax.bar_label(container, fmt="%.0f", label_type="edge", padding=3, fontsize=9)

# Configurar título e rótulos
plt.title("Categorias avaliadas no modelo de Dicionarização - Desconsiderando Outros", fontsize=14, pad=
plt.xlabel("Quantidade", fontsize=12)
plt.ylabel("Categoria", fontsize=12)
plt.tight_layout()

# Mostrar o gráfico
plt.show()
```