UNIVERSIDADE PRESBITERIANA MACKENZIE
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
TECNOLOGIA EM CIÊNCIAS DE DADOS

PROJETO APLICADO IV

Projeto Anatel: Acompanhamento de Reclamações Registradas na ANATEL

PROFESSOR: GUSTAVO SCALABRINI SAMPAIO

GRUPO:

GUSTAVO CASSIMIRO – 10415853 – 10415853@mackenzista.com.br
MAIKI SOARES – 10415481 – 10415481@mackenzista.com.br
VANESSA CORDEIRO – 10415118 – 10415118@mackenzista.com.br
MARIA FERNANDA – 10424791 – 10424791@mackenzista.com.br
NATHAN SAMPAIO – 10408439 – 10408439@mackenzista.com.br

São Paulo
2025

SUMÁRIO

RESUMO

O projeto teve como meta principal a previsão do volume mensal de solicitações à ANATEL por meio da modelagem de séries temporais, enfrentando o desafio de anomalias nos dados. Inicialmente, o modelo estatístico SARIMA, utilizado como baseline na série original, falhou drasticamente, apresentando um RMSE de 73.800 e um MAPE de 55.61%. Essa falha foi atribuída a uma anomalia extrema (um pico de reclamações entre 2019 e 2020) que distorceu a capacidade preditiva do modelo. A solução crucial foi a etapa de pré-processamento de anomalias, onde o pico foi substituído por valores interpolados linearmente, resultando em uma Série Temporal Tratada. Sobre esta série limpa, foi aplicado o modelo de machine learning LSTM (Long Short-Term Memory), que demonstrou performance significativamente superior. O modelo LSTM alcançou um RMSE de 9.450 e um MAPE de 6.95%, representando uma redução de 87.2% no RMSE em relação ao SARIMA. Concluiu-se que o sucesso da previsão dependeu da combinação eficaz do tratamento dos outliers com o uso do modelo LSTM, mais robusto para capturar os padrões de tendência e sazonalidade da série temporal corrigida.

1. Introdução

A área de conhecimento envolvida neste projeto é o **ODS 9 — Indústria, Inovação e Infraestrutura**, que orienta iniciativas voltadas à melhoria e modernização de sistemas produtivos e de serviços essenciais. Ao relacionar esse objetivo às telecomunicações, evidencia-se a necessidade de desenvolver mecanismos que possibilitem monitorar e fortalecer a infraestrutura de rede, estimulando a inovação e a qualidade dos serviços prestados à população.

O problema selecionado tem como foco a análise das manifestações registradas junto à **Agência Nacional de Telecomunicações (Anatel)** a respeito das operadoras que atuam no Brasil. A base de dados utilizada compreende informações sobre reclamações, denúncias, pedidos de informação e sugestões, organizadas em formato de série temporal.

O desafio consiste em transformar esse volume de dados em um produto analítico capaz de apoiar gestores públicos e privados na tomada de decisão, permitindo a antecipação de picos de reclamações, a identificação de padrões sazonais e a avaliação do impacto de intervenções em infraestrutura e inovação tecnológica.

Ao detalhar esse problema, observa-se que a ausência de monitoramento preditivo dificulta a resposta ágil a falhas sistêmicas e a elaboração de estratégias preventivas. Além disso, a falta de indicadores padronizados pode comprometer a capacidade de avaliar a efetividade de políticas públicas e investimentos em inovação.

Nesse sentido, o projeto propõe o desenvolvimento de uma solução analítica que, a partir de séries temporais, permitirá acompanhar a evolução das manifestações da população, detectar variações inesperadas e apoiar decisões estratégicas que contribuam para a melhoria da infraestrutura de telecomunicações no país.

2. Objetivo

O objetivo deste projeto é desenvolver um **produto analítico baseado em séries temporais** capaz de monitorar, analisar e prever a evolução das manifestações registradas pelos consumidores junto à Anatel em relação aos serviços de telecomunicações.

Metas específicas

- Identificar padrões históricos das manifestações, considerando **sazonalidade, tendências e variações** por UF, operadora, serviço e tipo de registro.
- Construir um **pipeline de dados** para coleta, organização e tratamento das séries temporais de manifestações.
- Aplicar **métodos estatísticos e de modelagem preditiva** para gerar previsões de curto e médio prazo sobre o volume de registros.

- Implementar **técnicas de detecção de anomalias** que auxiliem na identificação de eventos inesperados ou problemas emergentes.

3. Motivações e Justificativas

A escolha do tema é motivada pela importância crescente das **telecomunicações** como um pilar essencial para a vida cotidiana, a competitividade das empresas e a inclusão digital da população.

Segundo dados da **Anatel (2023)**, foram registradas aproximadamente **3,2 milhões de reclamações** de consumidores contra operadoras de telecomunicações no Brasil em 2022, com destaque para serviços de banda larga fixa e telefonia móvel.

De acordo com a pesquisa **TIC Domicílios (Cetic.br, 2023), 83% dos domicílios brasileiros possuem acesso à internet**, reforçando a relevância dos serviços de telecomunicações para inclusão social e desenvolvimento econômico.

Nesse cenário, o registro de manifestações junto à Anatel constitui **uma fonte estratégica de dados** para compreender a qualidade dos serviços prestados e os principais pontos de insatisfação dos usuários.

A relevância do tema está no potencial de transformar essas informações em **inteligência prática** para gestores públicos e privados.

Benefícios esperados da solução:

- Antecipar problemas, reduzindo impactos negativos à população.
- Detectar padrões sazonais e eventos inesperados, apoiando a **gestão preventiva**.
- Avaliar o impacto de **investimentos em infraestrutura e inovação tecnológica**.
- Gerar **indicadores confiáveis** para órgãos reguladores.

Além disso, os **atores envolvidos** incluem:

- **Consumidores**: com serviços mais estáveis e atendimento mais eficiente.
- **Operadoras**: com insights sobre falhas recorrentes e oportunidades de melhoria.
- **Órgãos reguladores**: com indicadores confiáveis para monitorar o setor.
- **Sociedade em geral**: favorecida pela inclusão digital e modernização da infraestrutura.

Por fim, a solução pode ser aplicada em:

- **Dashboards internos** das operadoras.
- **Painéis de acompanhamento** de órgãos reguladores.
- **Relatórios de gestão de qualidade**.

4. Descrição da Base de Dados

O conjunto de dados utilizado refere-se às solicitações registradas pelos consumidores junto à **Anatel**, incluindo reclamações, denúncias, pedidos de informação e sugestões.

A base está disponível no **Portal de Dados Abertos do Governo Federal**.

Metadados principais

- **Título:** Solicitações Registradas na Anatel
- **Descrição:** Quantidade de registros de reclamações, denúncias, pedidos de informação e sugestões em relação às operadoras e à Anatel.
- **Etiquetas:** ANATEL, Banda Larga Fixa, Celular Pós-Pago, Celular Pré-Pago, Móvel Pessoal, SCM, Serviço Móvel, STFC, Telefone Fixo, Telefonia Móvel, TV por Assinatura, reclamação, denúncia, pedido de informação, sugestão
- **Autor:** Agência Nacional de Telecomunicações – Anatel
- **E-mail do autor:** rcts@anatel.gov.br
- **Mantenedor:** Gerência de Tratamento de Solicitações de Consumidores – RCTS
- **E-mail do mantenedor:** rcts@anatel.gov.br
- **Granularidade temporal:** mensal

Período de coleta

- Janeiro de 2015 até dezembro de 2023

Fonte

- Portal de Dados Abertos do Governo Federal:

  https://dados.gov.br/dados/conjuntos-dados/solicitacoesregistradasnaanatel
- Agência Nacional de Telecomunicações (Anatel)

5. Referencial Teórico

O estudo de **séries temporais** é amplamente aplicado em diferentes setores, incluindo economia, saúde e telecomunicações, permitindo a previsão de tendências e a detecção de padrões sazonais (Box, Jenkins & Reinsel, 2015).

No contexto das telecomunicações, essas técnicas são fundamentais para compreender o comportamento da demanda e o impacto de políticas de melhoria de infraestrutura.

Pesquisas recentes destacam o uso de modelos estatísticos como **ARIMA** e **Prophet** para previsão de séries temporais em bases públicas de reclamações (Silva & Andrade, 2021).

Outros estudos apontam o potencial de modelos baseados em **aprendizado de máquina**, como **LSTM (Long Short-Term Memory)**, que oferecem maior capacidade de capturar padrões não lineares e complexos (Zhang et al., 2020).

Comparação de abordagens

- **Modelos estatísticos (ARIMA):**
  - Vantagens: simplicidade e interpretabilidade.
  - Limitações: desempenho inferior em cenários de alta variabilidade.
- **Modelos de redes neurais (LSTM):**
  - Vantagens: maior precisão em grandes volumes de dados.
  - Limitações: alta demanda computacional e menor transparência nos resultados.

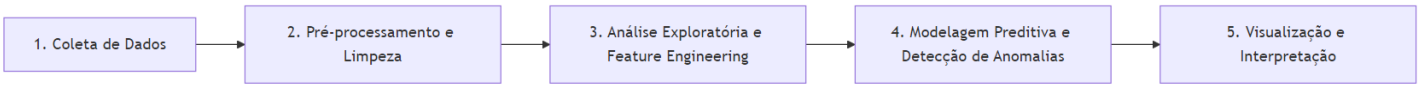
No Brasil, a análise de dados da Anatel é utilizada por órgãos reguladores para acompanhar a evolução da qualidade do serviço e orientar políticas públicas (Anatel, 2023).

Iniciativas acadêmicas também reforçam que o uso de séries temporais aplicadas às reclamações pode apoiar a antecipação de falhas sistêmicas e otimizar o atendimento ao consumidor (Ferreira & Oliveira, 2021).

Este projeto se apoia em abordagens já consolidadas na literatura, ao mesmo tempo em que propõe um diferencial: a **integração de diferentes métodos de modelagem** em um pipeline analítico voltado especificamente para a base de manifestações da Anatel.

6. Diagrama de Solução

O diagrama da solução proposta é baseado no pipeline de 5 etapas definido na concepção do projeto. Este pipeline estrutura o fluxo de trabalho desde a aquisição dos dados brutos até a entrega de um produto analítico focado em séries temporais.



Abaixo detalhamos cada uma dessas etapas:

1 - Coleta de Dados

A primeira etapa consiste na obtenção dos dados públicos de solicitações de consumidores da Anatel. Conforme executado no script de EDA, esta etapa envolveu a leitura de dois conjuntos de dados principais (reclamacoes.csv e reclamacoes_contexto.csv), que juntos somam mais de 23 milhões de registros.

2 - Pré-processamento e Limpeza

Esta etapa foi crucial para transformar os dados brutos em um formato utilizável. As tarefas incluíram:

- Junção: Concatenação dos dois arquivos de dados em um único DataFrame.
- Tratamento de Ausentes: Preenchimento de valores nulos na coluna UF com a string "Não Informada", conforme planejado.
- Unificação de variáveis categóricas (ex: padronizar nomes de operadoras como "CLARO" e "CLARO S.A." para apenas "CLARO").
- Transformação de tipos de dados, notavelmente a criação de uma coluna Data (formato datetime) a partir das colunas Ano e Mês, que é fundamental para a análise temporal.

Ferramentas:

- pandas
- numpy
- matplotlib
- seaborn
- plotly
- statsmodels
- re
- unicodedata

3 - Análise Exploratória e Feature Engineering

Com os dados limpos, esta fase busca entender os padrões. Conforme o pipeline, identificamos tendências (o volume de reclamações aumenta ou diminui ao longo dos anos?), sazonalidade (existem picos recorrentes em meses específicos?) e correlações. As descobertas desta etapa (EDA) informam a criação de novas features (variáveis), como lags e médias móveis, que serão usadas para enriquecer os modelos preditivos.

4 - Modelagem Preditiva e Detecção de Anomalias

Este é o núcleo analítico do projeto. Conforme planejado, a abordagem é dupla:

a) Previsão: Aplicação de modelos de séries temporais (como SARIMA e LSTM) para prever o volume futuro de reclamações.

b) Detecção de Anomalias: Uso de métodos estatísticos (Z-Score) e de machine learning (Isolation Forest) para identificar picos repentinos e inesperados que fujam do padrão histórico, sinalizando possíveis falhas sistêmicas.

5 - Visualização e Interpretação

A etapa final consiste em traduzir os resultados dos modelos em insights acionáveis. O objetivo é criar dashboards interativos (usando ferramentas como

Streamlit ou Power BI) que permitam aos gestores monitorar a série histórica, visualizar as previsões e receber alertas sobre anomalias detectadas.

Este pipeline foi desenhado para ser reproduzível e escalável, permitindo que a solução seja atualizada à medida que novos dados mensais da Anatel forem disponibilizados.

7. Eda e Pré-Processamento dos Dados

Exploração e Análise dos Dados

Os dados utilizados são provenientes da Anatel e detalham as solicitações de consumidores (reclamações, denúncias, etc.). O dataset é granular, cobrindo o período de 2015 a 2025 e incluindo dimensões como UF, Serviço, Marca (operadora) e o tipo de Assunto.

Qualidade, Limitações e Simplificações:

- **Qualidade:** Os dados são oficiais e estruturados. No entanto, apresentam desafios de consistência.
- **Limitações:**
    1. **Consistência de Nomes:** Campos categóricos como Marca e Grupo Econômico exigem padronização, pois a mesma empresa pode aparecer com nomes diferentes (ex: "OI" vs "OI S.A. - EM RECUPERACAO JUDICIAL").
    2. **Valores Ausentes:** A coluna UF apresenta valores nulos, necessitando de tratamento para não perder esses registros na análise agregada.
    3. **Metadados vs. Dados:** Houve uma pequena discrepância entre o glossário (que menciona a coluna Quantidade de Solicitações) e os arquivos CSVs que usa o nome SOLICITAÇÕES.
- **Recorte Adotado:** Para a modelagem de séries temporais, o recorte principal adotado foi agregar todas as solicitações (independente de UF, Serviço ou Marca) em uma única série temporal com granularidade mensal (total de solicitações por mês). As demais dimensões foram mantidas para a análise exploratória (EDA) e contextualização, mas não entram no modelo base.

Tarefas de Preparação dos Dados

O script de pré-processamento executou as seguintes tarefas:

1. **Carga:** Leitura dos arquivos reclamacoes.csv e reclamacoes_contexto.csv, especificando o separador (;).
2. **Concatenação:** Juntando os dois dataframes gerados pela leitura dos arquivos.
3. **Transformação (Data):** Foi criada a coluna Data no formato datetime. Esta é a transformação mais crítica, pois converte as colunas Ano e Mês em um índice temporal, permitindo a análise de séries.

4. **Limpeza (Ausentes):** Valores ausentes na coluna UF foram preenchidos com a string "Não Informada".
5. **Padronização (Categóricas):** A coluna Marca foi convertida para maiúsculas e as marcas com menor frequência foram agrupadas na categoria "OUTRAS", reduzindo a dimensionalidade para a EDA.
6. **Agregação (Compactação):** Esta é a principal tarefa de preparação para a modelagem. O dataset foi agrupado pela coluna Data e teve seus valores somados pela coluna SOLICITAÇÕES. Isso resultou em um novo dataframe (df_ts) onde cada linha representa um mês e a única coluna é o número total de solicitações, formato ideal para modelos como o SARIMA.

A análise exploratória visual (EDA) deste dataframe agregado (df_ts) revelou uma forte **sazonalidade** (picos anuais) e uma **tendência** variável ao longo dos anos, validando a abordagem de modelagem de séries temporais.

In [2]:

```
# 1. Importação das bibliotecas necessárias
import re
import unicodedata
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose

# Configurações de visualização
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (15, 7)

# 2. Carga dos dados
file_path_1 = 'reclamacoes.csv'
file_path_2 = 'reclamacoes_contexto.csv'

try:
                                df1 = pd.read_csv(file_path_1, sep=';')
                                print(f"Dados carregados do primeiro arquivo com sucesso. Shape inicial: {df1.shape}")

                                #Exibir as 5 primeiras linhas e nomes das colunas
                                print(f"Colunas detectadas no primeiro arquivo: {df1.columns.tolist()}")
except Exception as e:
                                print(f"Erro ao carregar o primeiro arquivo: {e}")

try:
                                df2 = pd.read_csv(file_path_2, sep=';')
                                print(f"Dados carregados do segundo arquivo com sucesso. Shape inicial: {df2.shape}")

                                #Exibir as 5 primeiras linhas e nomes das colunas
                                print(f"Colunas detectadas no segundo arquivo: {df2.columns.tolist()}")
except Exception as e:
                                print(f"Erro ao carregar o segundo arquivo: {e}")

df1['Linha'] = ''
df = pd.concat([df1, df2], ignore_index=True)
del df1, df2

# Visualizar as colunas e tipos de dados
print("\n--- Colunas e Tipos de Dados (Iniciais) ---")
print(df.info())
```

```
Dados carregados do primeiro arquivo com sucesso. Shape inicial: (15952407, 15)
Colunas detectadas no primeiro arquivo: ['DataExtracao', 'SOLICITAÇÕES', 'Ano', 'Mês', 'AnoMês', 'UF', 'Cidade', 'CO_MUNICI
PIO', 'CanalEntrada', 'Condição', 'TipoAtendimento', 'Serviço', 'Marca', 'Assunto', 'Problema']
Dados carregados do segundo arquivo com sucesso. Shape inicial: (7072309, 16)
Colunas detectadas no segundo arquivo: ['DataExtracao', 'SOLICITAÇÕES', 'Ano', 'Mês', 'AnoMês', 'UF', 'Cidade', 'CO_MUNICIP
IO', 'CanalEntrada', 'Condição', 'TipoAtendimento', 'Serviço', 'Marca', 'Assunto', 'Problema', 'Linha']

--- Colunas e Tipos de Dados (Iniciais) ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23024716 entries, 0 to 23024715
Data columns (total 16 columns):
 #   Column           Dtype
---  ------           -----
 0   DataExtracao     object
 1   SOLICITAÇÕES     int64
 2   Ano              int64
 3   Mês              int64
 4   AnoMês           object
 5   UF               object
 6   Cidade           object
 7   CO_MUNICIPIO     object
 8   CanalEntrada     object
 9   Condição         object
 10  TipoAtendimento  object
 11  Serviço          object
 12  Marca            object
 13  Assunto          object
 14  Problema         object
 15  Linha            object
dtypes: int64(3), object(13)
memory usage: 2.7+ GB
None
```

In [3]:

```python
# Visualizar as colunas e tipos de dados
print("\n--- Infos e Estatísticas ---")
print("Valores ausentes (% por coluna):")
print((df.isna().mean() * 100).round(5).sort_values(ascending=False))
```

```
--- Infos e Estatísticas ---
Valores ausentes (% por coluna):
Cidade             0.21671
UF                 0.00027
Marca              0.00001
Serviço            0.00001
Mês                0.00000
Ano                0.00000
DataExtracao       0.00000
SOLICITAÇÕES       0.00000
CO_MUNICIPIO       0.00000
AnoMês             0.00000
Condição           0.00000
CanalEntrada       0.00000
TipoAtendimento    0.00000
Assunto            0.00000
Problema           0.00000
Linha              0.00000
dtype: float64
```

In [4]:

```python
# Resumo estatístico das colunas numéricas
print("\nEstatísticas descritivas:")
print(df.describe())
```

```
Estatísticas descritivas:
       SOLICITAÇÕES           Ano           Mês
count  2.302472e+07  2.302472e+07  2.302472e+07
mean   1.275870e+00  2.018531e+03  6.190205e+00
std    2.145190e+00  2.748319e+00  3.406955e+00
min    1.000000e+00  2.015000e+03  1.000000e+00
25%    1.000000e+00  2.016000e+03  3.000000e+00
50%    1.000000e+00  2.018000e+03  6.000000e+00
75%    1.000000e+00  2.020000e+03  9.000000e+00
max    2.624000e+03  2.025000e+03  1.200000e+01
```

In [5]:

```
def padronizar_nome(nome):
    """
    Função para limpar e padronizar os nomes dos provedores.
    """
    # 1. Converter para string para evitar erros em valores nulos (NaN)
    nome = str(nome)

    # 2. Corrigir erros de encoding/caracteres específicos da sua lista
    correcoes = {
        '€åES': 'COES',
        '€å': 'CO',
        'FµCIL': 'FACIL',
        'µ': 'A',
        'TUPÇ': 'TUPA'
    }
    for erro, correto in correcoes.items():
        nome = nome.replace(erro, correto)

    # 3. Remover notas de status (INATIVADA, DESATIVADA)
    # Isso usa uma expressão regular para encontrar "(INATIVADA..." ou "(DESATIVADA..."
    # e remover tudo até o ')'
    nome = re.sub(r'\s*\((INATIVADA|DESATIVADA)[^)]*\)', '', nome, flags=re.IGNORECASE)

    # 4. Normalizar acentos (ex: 'Ç' -> 'C', 'Ã' -> 'A')
    try:
        # Separa o caractere do seu acento (NFD)
        nome_nfd = unicodedata.normalize('NFD', nome)
        # Codifica para ASCII ignorando os acentos e decodifica de volta para string
        nome = nome_nfd.encode('ascii', 'ignore').decode('utf-8')
    except Exception as e:
        # Fallback caso algo dê errado
        pass

    # 5. Converter para MAIÚSCULAS
    nome = nome.upper()

    # 6. Remover toda pontuação e caracteres especiais (exceto espaço)
    # Mantém apenas letras (A-Z), números (0-9) e espaços
    nome = re.sub(r'[^A-Z0-9\s]', '', nome)

    # 7. Normalizar espaços em branco
    nome = nome.strip() # Remove espaços no início e fim
    nome = re.sub(r'\s+', ' ', nome) # Substitui múltiplos espaços por um único

    # 8. Tratar casos vazios (ex: "(DESATIVADA...)")
    if not nome:
        return None # Ou você pode retornar 'NOME_INVALIDO'

    return nome
```

In [6]:

```
# 3. Pré-processamento e Limpeza (conforme pipeline)

# a. Criar coluna de data (datetime)
# Usando as colunas 'Ano' e 'Mês'
print("\nCriando coluna 'Data' a partir de 'Ano' e 'Mês'...")
df['Data'] = pd.to_datetime(df['Ano'].astype(str) + '-' + df['Mês'].astype(str) + '-01')

print(f"Período de dados: De {df['Data'].min().date()} até {df['Data'].max().date()}")

# b. Tratamento de valores ausentes (ex: UF)
print(f"Valores ausentes na coluna 'UF' antes: {df['UF'].isnull().sum()}")
# O pipeline menciona tratar UFs não informadas
df['UF'].fillna('Não Informada', inplace=True)
print(f"Valores ausentes na coluna 'UF' depois: {df['UF'].isnull().sum()}")
```

```
Criando coluna 'Data' a partir de 'Ano' e 'Mês'...
Período de dados: De 2015-01-01 até 2025-07-01
Valores ausentes na coluna 'UF' antes: 63
Valores ausentes na coluna 'UF' depois: 0
```

In [7]:

```
# c. Unificação de nomes (ex: Operadoras)
# Usando a coluna 'Marca'
print(f"Valores únicos em 'Marca' (antes): {df['Marca'].nunique()}")
df['Marca'] = df['Marca'].apply(padronizar_nome)
print(f"Valores únicos em 'Marca' (depois): {df['Marca'].nunique()}")
```

```
Valores únicos em 'Marca' (antes): 382
Valores únicos em 'Marca' (depois): 381
```

In [8]:

```
# d. Cidades e Estados
print("Total de estados:", df["UF"].nunique())
print("Total de cidades:", df["Cidade"].nunique())
```

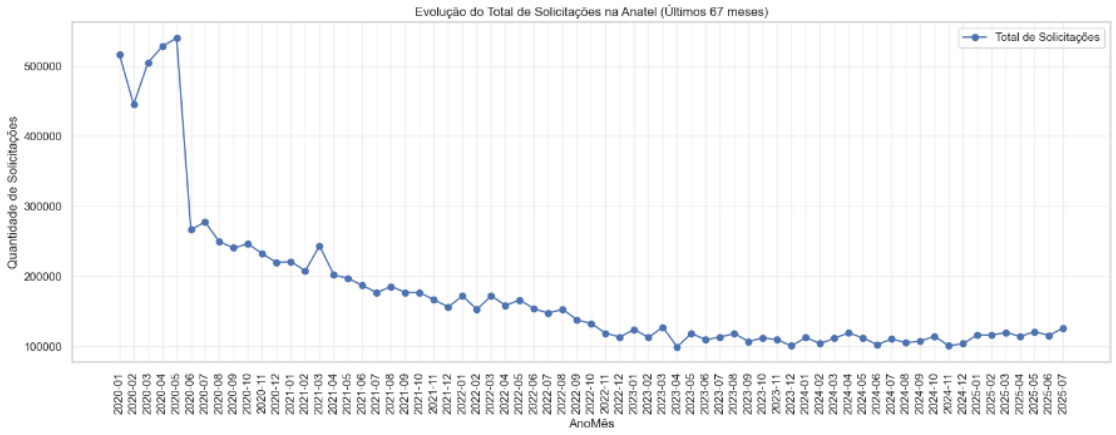
```
Total de estados: 29
Total de cidades: 5367
```

In [9]:
```
# 4. Análise Exploratória (EDA)

# Usando a coluna 'SOLICITAÇÕES'
print("Agregando dados por mês...")
reclamacoes_tempo = df.groupby('AnoMês')['SOLICITAÇÕES'].sum().reset_index()
reclamacoes_tempo = reclamacoes_tempo.tail(67) #Limitando aos últimos 67 meses
reclamacoes_tempo = reclamacoes_tempo.set_index('AnoMês')

print("\n--- Dataset agregado (reclamacoes_tempo) pronto para análise ---")
print(reclamacoes_tempo.head())
```

```
Agregando dados por mês...

--- Dataset agregado (reclamacoes_tempo) pronto para análise ---
         SOLICITAÇÕES
AnoMês
2020-01        517256
2020-02        445440
2020-03        505214
2020-04        528238
2020-05        540535
```

In [10]:
```
print("\n--- Iniciando Análise Exploratória (EDA) ---")

# a. Visualização da Série Temporal Últimos 67 meses
plt.figure(figsize=(15, 6))
plt.plot(reclamacoes_tempo.index, reclamacoes_tempo['SOLICITAÇÕES'], label='Total de Solicitações', marker='o')
plt.title('Evolução do Total de Solicitações na Anatel (Últimos 67 meses)')
plt.xlabel('AnoMês')
plt.ylabel('Quantidade de Solicitações')
plt.grid(True, alpha=0.3)
plt.xticks(rotation=90)
plt.tight_layout()
plt.legend()
plt.show()
```

```
--- Iniciando Análise Exploratória (EDA) ---
```

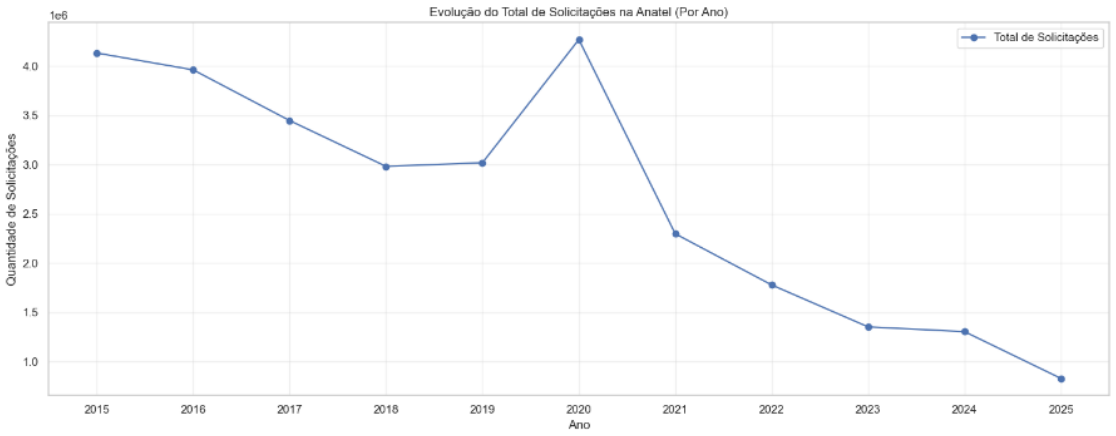


In [11]:
```
print("Agregando dados por ano...")
reclamacoes_ano = df.groupby(df['Data'].dt.year)['SOLICITAÇÕES'].sum().reset_index()
reclamacoes_ano = reclamacoes_ano.rename(columns={'Data': 'Ano'})
reclamacoes_ano = reclamacoes_ano.set_index('Ano')

print("\n--- Dataset agregado (reclamacoes_ano) pronto para análise ---")
print(reclamacoes_ano.head())
```

```
Agregando dados por ano...

--- Dataset agregado (reclamacoes_ano) pronto para análise ---
      SOLICITAÇÕES
Ano
2015       4132965
2016       3962021
2017       3447450
2018       2982349
2019       3018934
```

In [12]:

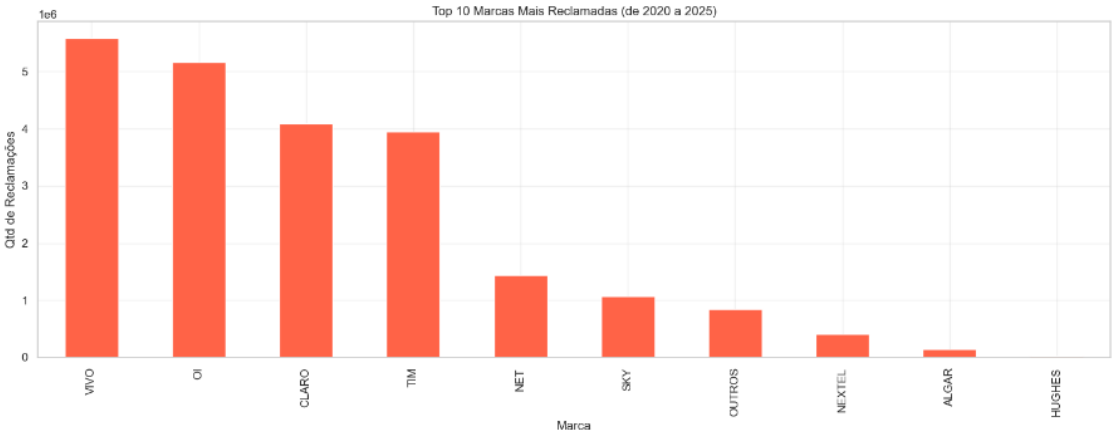
```
# b. Visualização da Série Temporal Completa por ano
plt.figure(figsize=(15, 6))
plt.plot(reclamacoes_ano.index, reclamacoes_ano['SOLICITAÇÕES'], label='Total de Solicitações', marker='o')
plt.title('Evolução do Total de Solicitações na Anatel (Por Ano)')
plt.xlabel('Ano')
plt.ylabel('Quantidade de Solicitações')
plt.xticks(reclamacoes_ano.index)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.legend()
plt.show()
```



In [13]:

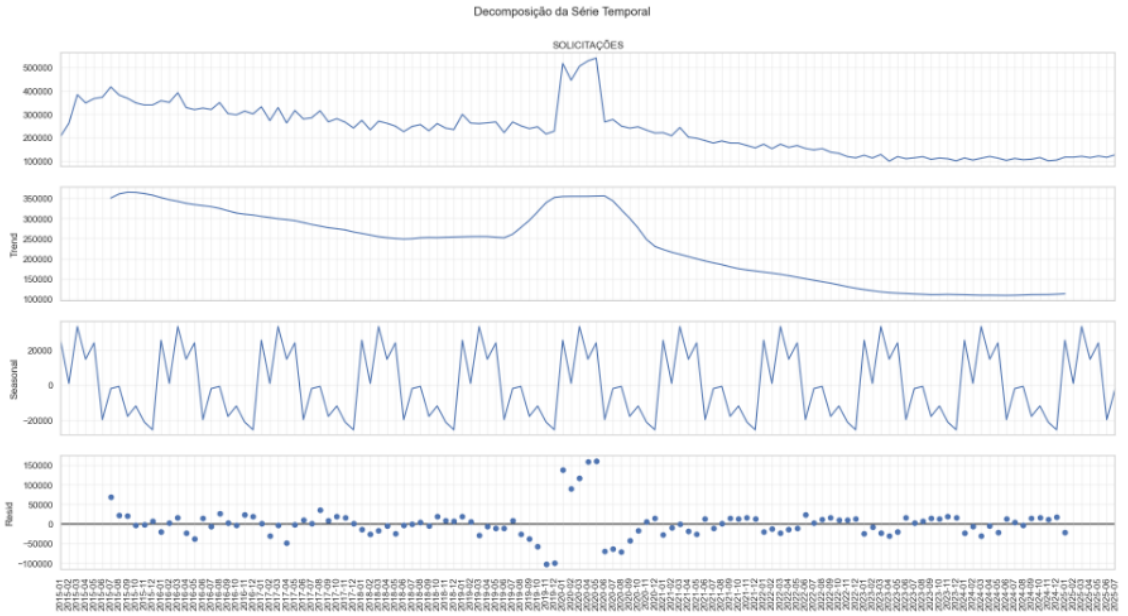
```
top_marcas = df["Marca"].value_counts().head(10)
print("Top 10 marcas mais reclamadas:")
print(top_marcas)

# c. Visualização do Top 10 Marcas Mais Reclamadas
plt.figure(figsize=(15,6))
top_marcas.plot(kind='bar', color='tomato')
plt.title("Top 10 Marcas Mais Reclamadas (de 2020 a 2025)")
plt.ylabel("Qtd de Reclamações")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Top 10 marcas mais reclamadas:
Marca
VIVO      5594340
OI        5162091
CLARO     4098196
TIM       3951339
NET       1437284
SKY       1075766
OUTROS     842045
NEXTEL     404305
ALGAR      140662
HUGHES      31781
Name: count, dtype: int64
```



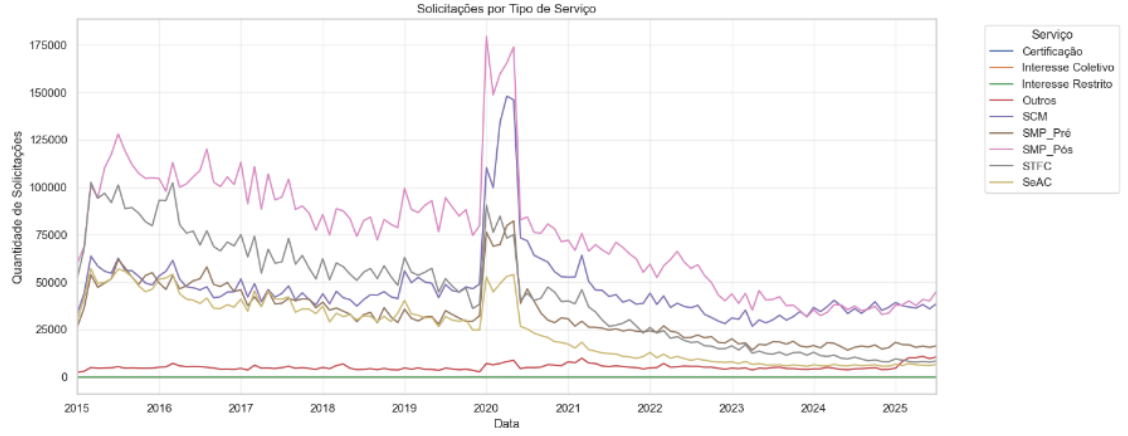
In [14]:
```
# d. Decomposição da Série (Tendência, Sazonalidade, Resíduo)
# O pipeline menciona a identificação de sazonalidade
reclamacoes_tempo = df.groupby('AnoMês')['SOLICITAÇÕES'].sum().reset_index()
reclamacoes_tempo = reclamacoes_tempo.set_index('AnoMês')
print("Decompondo a série temporal...")
# Usamos 'period=12' para dados mensais (sazonalidade anual)
decomposicao = seasonal_decompose(reclamacoes_tempo['SOLICITAÇÕES'], model='additive', period=12)

fig = decomposicao.plot()
fig.set_size_inches(20, 10)
for ax in fig.get_axes():
    ax.grid(True, alpha=0.3)
plt.suptitle('Decomposição da Série Temporal', y=1.02)
plt.xticks(rotation=90)
plt.show()
```

```
Decompondo a série temporal...
```



In [15]:
```
# e. Análise por Serviço
print("Analisando por Serviço...")
# A amostra possui a coluna 'Serviço'
df_servico = df.groupby(['Data', 'Serviço'])['SOLICITAÇÕES'].sum().unstack().fillna(0)

plt.figure(figsize=(15, 6))
df_servico.plot(ax=plt.gca())
plt.title('Solicitações por Tipo de Serviço')
plt.xlabel('Data')
plt.ylabel('Quantidade de Solicitações')
plt.legend(title='Serviço', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

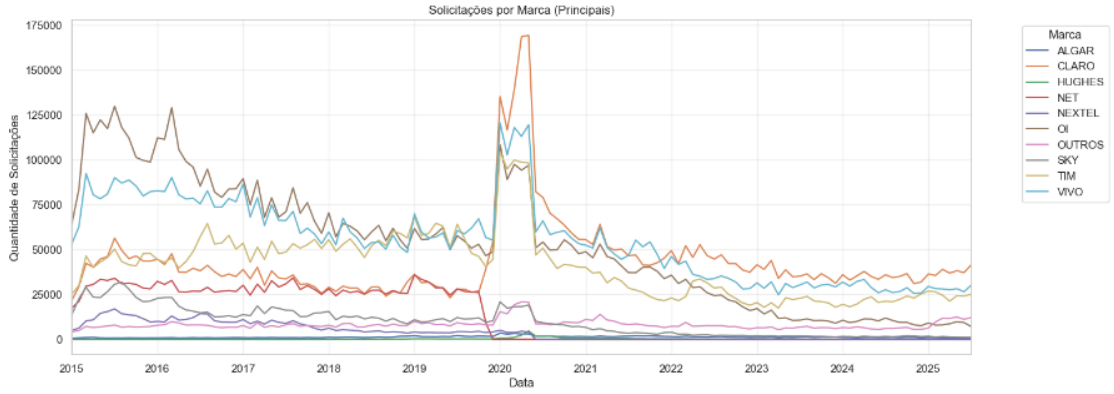
```
Analisando por Serviço...
```



In [16]:

```
# f. Análise por Marca (Principais)
print("Analisando por Marca...")
top_marcas = list(top_marcas.reset_index()['Marca'])
df_marca = df[df['Marca'].isin(top_marcas)]
df_marca = df_marca.groupby(['Data','Marca'])['SOLICITAÇÕES'].sum().unstack().fillna(0)

plt.figure(figsize=(16, 6))
# Plotando apenas as principais marcas definidas anteriormente
df_marca.plot(ax=plt.gca())
plt.title('Solicitações por Marca (Principais)')
plt.xlabel('Data')
plt.ylabel('Quantidade de Solicitações')
plt.legend(title='Marca', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.show()
```

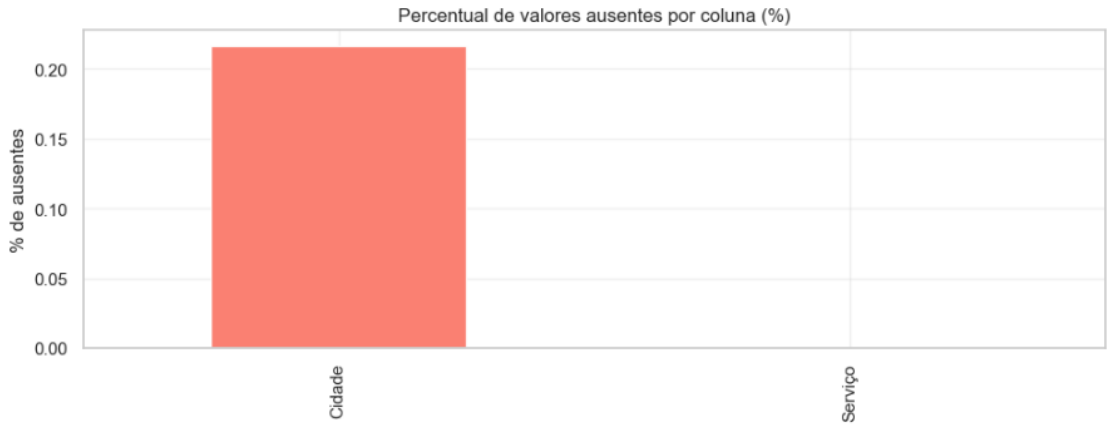
```
Analisando por Marca...
```



In [17]:

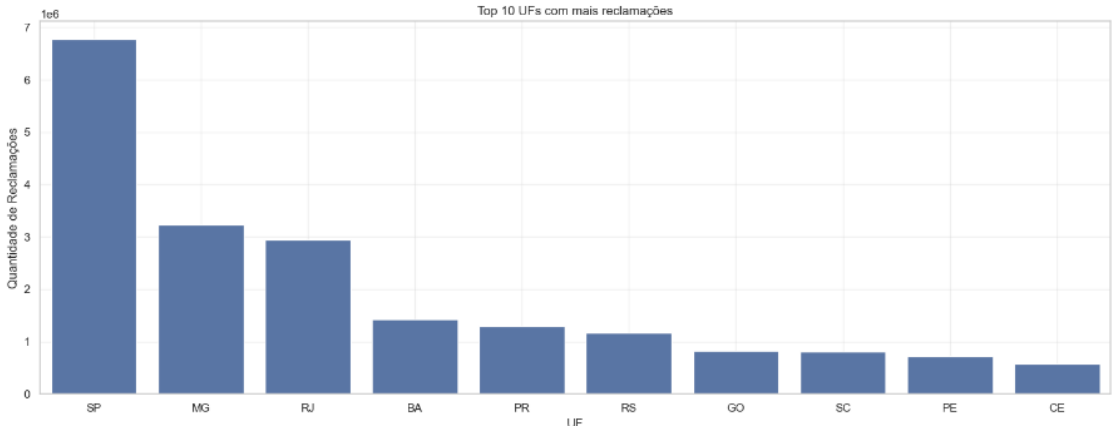
```
# g. Valores ausentes
faltantes = df.isnull().mean().sort_values(ascending=False) * 100
print("Percentual de valores ausentes por coluna:")
print(faltantes[faltantes > 0])

plt.figure(figsize=(10,4))
faltantes[faltantes > 0].plot(kind="bar", color="salmon")
plt.title("Percentual de valores ausentes por coluna (%)")
plt.ylabel("% de ausentes")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Percentual de valores ausentes por coluna:
Cidade     0.216711
Serviço    0.000013
dtype: float64
```



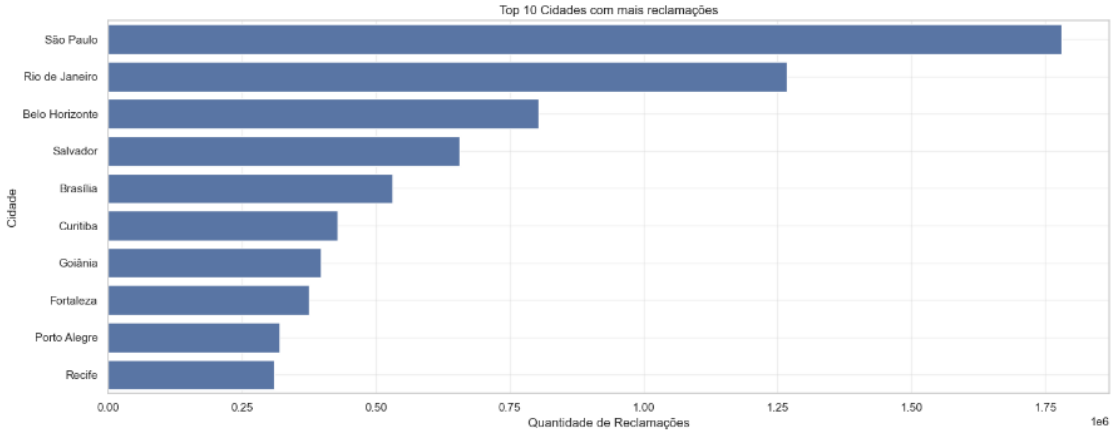
In [18]:

```
# h. Distribuição de reclamações por UF
plt.figure(figsize=(15,6))
top_ufs = df["UF"].value_counts().head(10)
sns.barplot(x=top_ufs.index, y=top_ufs.values)
plt.title("Top 10 UFs com mais reclamações")
plt.xlabel("UF")
plt.ylabel("Quantidade de Reclamações")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



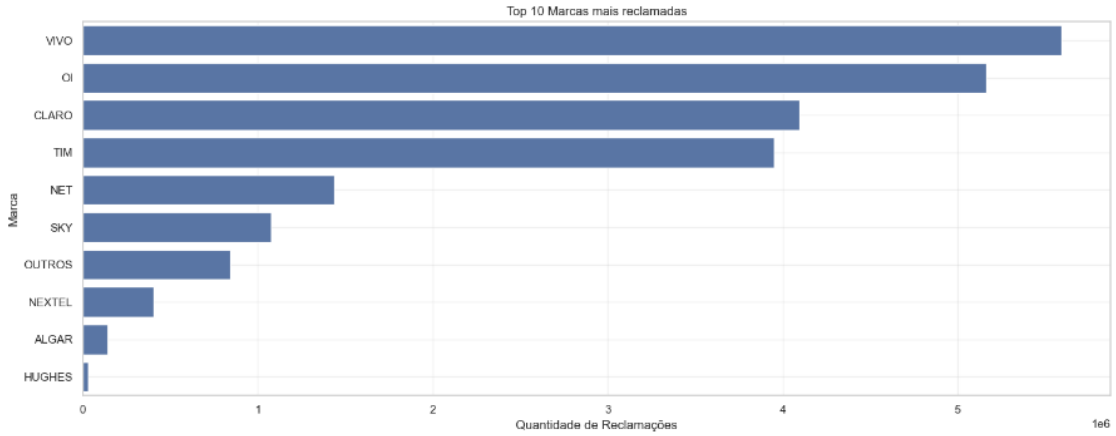
In [19]:

```
# i. Cidades com mais reclamações
plt.figure(figsize=(15,6))
top_cidades = df["Cidade"].value_counts().head(10)
sns.barplot(y=top_cidades.index, x=top_cidades.values)
plt.title("Top 10 Cidades com mais reclamações")
plt.xlabel("Quantidade de Reclamações")
plt.ylabel("Cidade")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



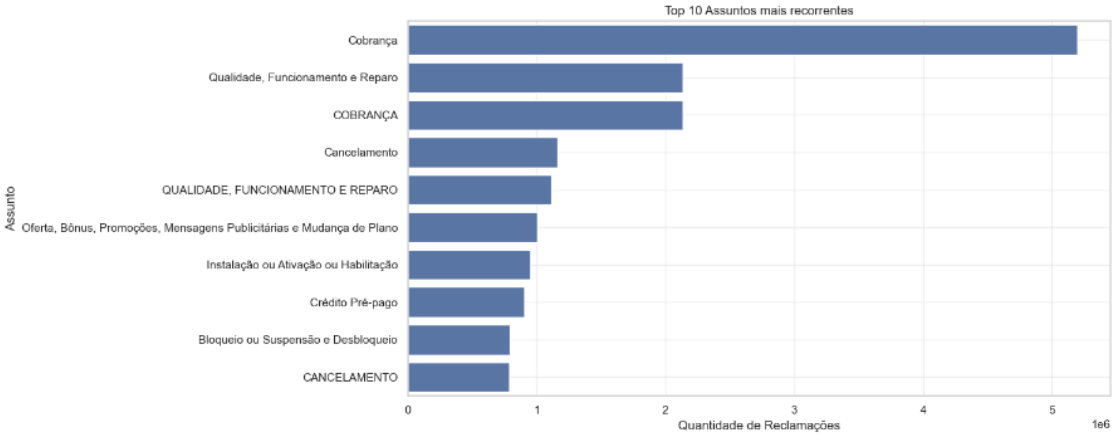
In [20]:

```
# j. Marcas mais reclamadas
plt.figure(figsize=(15,6))
top_marcas = df["Marca"].value_counts().head(10)
sns.barplot(y=top_marcas.index, x=top_marcas.values)
plt.title("Top 10 Marcas mais reclamadas")
plt.xlabel("Quantidade de Reclamações")
plt.ylabel("Marca")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

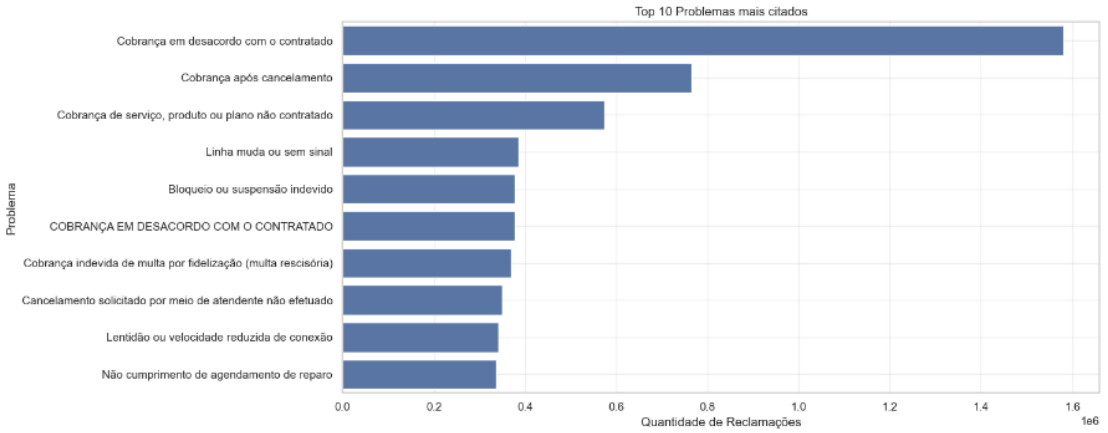


In [21]:

```
# k. Principais assuntos
plt.figure(figsize=(15,6))
top_assuntos = df["Assunto"].value_counts().head(10)
sns.barplot(y=top_assuntos.index, x=top_assuntos.values)
plt.title("Top 10 Assuntos mais recorrentes")
plt.xlabel("Quantidade de Reclamações")
plt.ylabel("Assunto")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



In [22]:
```
# 1. Principais problemas
plt.figure(figsize=(15,6))
top_problemas = df["Problema"].value_counts().head(10)
sns.barplot(y=top_problemas.index, x=top_problemas.values)
plt.title("Top 10 Problemas mais citados")
plt.xlabel("Quantidade de Reclamações")
plt.ylabel("Problema")
plt.grid(True, alpha=0.3)
plt.tight_layout()
#plt.show()
```



In [23]:
```
# Salvando dados tratados
df.to_csv("reclamacoes_tratadas.csv", index=False, sep=';')
```

In [ ]:
```
# Salvando dados para o modelo de séries temporais
df_ts = df.groupby('Data')['SOLICITAÇÕES'].sum().reset_index()
df_ts.to_csv("dados_modelo.csv", index=False, sep=';')

print("Dataset pronto (df_ts)")
print(df_ts.head())
```

```
Dataset pronto (df_ts)
        Data  SOLICITAÇÕES
0 2015-01-01        204919
1 2015-02-01        261585
2 2015-03-01        383765
3 2015-04-01        348216
4 2015-05-01        367076
```

8. Modelos

#SARIMA

Método Aplicado

O primeiro modelo aplicado foi o **SARIMA** (Seasonal AutoRegressive Integrated Moving Average). Este modelo estatístico clássico foi escolhido como baseline por sua capacidade de lidar com os principais componentes identificados na Análise Exploratória de Dados (EDA): **tendência** (não-estacionariedade da série) e **sazonalidade** (padrões que se repetem anualmente).

Para a implementação, foi utilizada a biblioteca pmdarima, que automatiza a busca pelos melhores parâmetros (p, d, q)(P, D, Q)m[12] através da função auto_arima. Esta função testa iterativamente diversas combinações de parâmetros e seleciona aquela que minimiza o Critério de Informação Akaike (AIC), garantindo um modelo otimizado sem ajuste manual exaustivo.

Preparação para Modelagem

Para treinar e avaliar o modelo, a série temporal agregada (total de solicitações mensais) foi dividida cronologicamente nos seguintes conjuntos:

- **Dados de Treino:** 84 meses (corresponde aos primeiros 7 anos de dados, de Jan/2015 a Dez/2021).
- **Dados de Teste:** 43 meses (corresponde aos dados de Jan/2022 até Jul/2025).

O modelo foi treinado exclusivamente com os dados de treino e, em seguida, foi utilizado para prever os 43 meses do período de teste.

Análise dos Resultados Obtidos

A avaliação do desempenho do modelo baseline no conjunto de teste gerou as seguintes métricas, conforme o plano de avaliação do projeto:

- **MAE (Mean Absolute Error):** 66.235
- **RMSE (Root Mean Squared Error):** 73.800
- **MAPE (Mean Absolute Percentage Error):** 55,61%

Interpretação da Performance (Análise Visual e Métrica):

As métricas, por si sós, indicam um desempenho muito ruim (um erro médio percentual de 55,61%). A análise visual do gráfico revela por que o modelo falhou tão drasticamente, expondo **dois problemas fatais** que violam as premissas básicas de modelos estatísticos como o SARIMA:

1. **Anomalia nos Dados de Treino (Pico 2019-2020):** O conjunto de treino (linha azul), que deveria ser usado para aprender o padrão "normal" da série, está ele mesmo "contaminado". O período entre Dezembro de 2019 e Maio de 2020 apresenta um pico extremo e não-sazonal que foge completamente da tendência observada entre 2015 e 2019. O auto_arima tenta, de forma equivocada, incorporar esse outlier em seu modelo de sazonalidade e tendência, distorcendo os parâmetros e "inflando" a previsão (linha verde).
2. **Quebra Estrutural nos Dados de Teste (Queda 2022):** O modelo, já distorcido pelo pico de 2020, foi usado para prever um período (2022 em diante) que sofreu uma segunda mudança radical. Os dados reais de teste (linha laranja tracejada) despencam para um patamar de volume muito abaixo do histórico.

O resultado é um fracasso em duas frentes: o modelo previu um volume (linha verde) muito acima do real (linha laranja), pois (1) seu padrão foi inflado pelo pico de 2020 e (2) o padrão real mudou drasticamente para baixo em 2022.

Conclusão do Modelo Base

O modelo SARIMA baseline é inadequado para esta série temporal. Sua falha, quantificada pelo MAPE de 55,61%, é explicada por sua incapacidade fundamental de lidar com alta instabilidade. Modelos estatísticos clássicos assumem que os

padrões do passado (mesmo que sazonais) se repetirão, e a série em questão quebra essa premissa duas vezes:

1. Nos dados de treino, com um outlier extremo (pico 2019-2020).
2. Nos dados de teste, com uma quebra estrutural (queda 2022).

Este resultado valida a necessidade da etapa de "Detecção de Anomalias" prevista no pipeline do projeto. Para a próxima etapa (refinamento do modelo), duas ações são obrigatórias:

1. Pré-processamento da Anomalia: O pico de 2019-2020 deve ser tratado (ex: suavizado ou removido) antes do treinamento de qualquer modelo futuro.
2. Exploração de Modelos Robustos: Testar modelos de machine learning (como LSTM, conforme planejado) e investigar variáveis exógenas (externas) para tentar explicar a queda estrutural de 2022.

Código do Modelo SARIMA

In [1]:
```
# 1. Importação das bibliotecas para modelagem
# !pip install pmdarima
# pip install numpy==1.26.4
# Necessário Python 3.12.7
import pmdarima as pm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error
from math import sqrt
```

In [2]:
```
df_ts = pd.read_csv("dados_modelo.csv", delimiter=';')
df_ts = df_ts.set_index('Data')
```

In [3]:
```
# 2. Preparação dos dados
# Usaremos o 'df_ts' (total de solicitações por mês) criado na etapa anterior
# Certifique-se de que o índice é um DatetimeIndex (já feito)
df_ts.index.freq = 'MS' # 'Month Start' frequency
data_modelagem = df_ts['SOLICITAÇÕES']
```

In [4]:
```
# 3. Divisão Treino/Teste
# Conforme o pipeline/cronograma: treino até 2021, teste 2022-2025
# O período de coleta vai até Jul/2025

TREINO_FIM = '2021-12-01'
TESTE_INICIO = '2022-01-01'

treino = data_modelagem.loc[data_modelagem.index <= TREINO_FIM]
teste = data_modelagem.loc[data_modelagem.index >= TESTE_INICIO]

if teste.empty:
    print("ALERTA: O dataset de teste está vazio.")
    print("Isto é esperado se estiver usando apenas o arquivo de amostra.")
    print("O modelo será treinado com os dados de 'treino', mas não poderá ser avaliado.")
else:
    print(f"Tamanho do dataset de Treino: {treino.shape[0]} meses")
    print(f"Tamanho do dataset de Teste: {teste.shape[0]} meses")
```

```
Tamanho do dataset de Treino: 84 meses
Tamanho do dataset de Teste: 43 meses
```

In [5]:

```
# 4. Treinamento do Modelo Base (Auto-SARIMA)
# 'm=12' indica a sazonalidade anual (mensal)
print("\nIniciando treinamento do Auto-SARIMA (Modelo Base)...")
# 'seasonal=True' -> Habilita SARIMA
# 'stepwise=True' -> Acelera a busca pelos melhores parâmetros
# 'suppress_warnings=True' -> Oculta avisos de convergência
modelo_base_sarima = pm.auto_arima(treino,
                                   start_p=1, start_q=1,
                                   test='adf',       # Teste de estacionariedade
                                   max_p=3, max_q=3, # Ordem máxima não sazonal
                                   m=12,             # Frequência da sazonalidade
                                   start_P=0,
                                   seasonal=True,    # Habilita SARIMA
                                   d=None,           # Deixa o auto_arima encontrar 'd'
                                   D=1,              # Força uma diferenciação sazonal
                                   trace=True,
                                   error_action='ignore',
                                   suppress_warnings=True,
                                   stepwise=True)

print("\n--- Resumo do Modelo Base (SARIMA) ---")
print(modelo_base_sarima.summary())
```

```
Iniciando treinamento do Auto-SARIMA (Modelo Base)...
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=1761.252, Time=0.16 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=1782.829, Time=0.01 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=1764.115, Time=0.07 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=1759.257, Time=0.10 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=1783.619, Time=0.02 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=1760.718, Time=0.12 sec
 ARIMA(0,1,1)(0,1,2)[12]             : AIC=1760.702, Time=0.10 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=1764.373, Time=0.06 sec
 ARIMA(0,1,1)(1,1,2)[12]             : AIC=1762.620, Time=0.35 sec
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=1766.398, Time=0.06 sec
 ARIMA(0,1,2)(0,1,1)[12]             : AIC=1762.719, Time=0.13 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=1758.894, Time=0.12 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=1783.598, Time=0.02 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=1760.364, Time=0.14 sec
 ARIMA(1,1,0)(0,1,2)[12]             : AIC=1760.348, Time=0.09 sec
 ARIMA(1,1,0)(1,1,2)[12]             : AIC=1762.264, Time=0.21 sec
 ARIMA(2,1,0)(0,1,1)[12]             : AIC=1760.567, Time=0.10 sec
 ARIMA(2,1,1)(0,1,1)[12]             : AIC=1763.223, Time=0.18 sec
 ARIMA(1,1,0)(0,1,1)[12] intercept   : AIC=1760.274, Time=0.08 sec

Best model:  ARIMA(1,1,0)(0,1,1)[12]
Total fit time: 2.128 seconds

--- Resumo do Modelo Base (SARIMA) ---
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                   84
Model:             SARIMAX(1, 1, 0)x(0, 1, [1], 12)   Log Likelihood                -876.447
Date:                              Mon, 27 Oct 2025   AIC                           1758.894
Time:                                      23:02:50   BIC                           1765.682
Sample:                                  01-01-2015   HQIC                          1761.593
                                       - 12-01-2021
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0633      0.331     -0.191      0.848      -0.712       0.585
ma.S.L12      -0.7318      0.134     -5.480      0.000      -0.994      -0.470
sigma2      4.479e+09   3.96e-11   1.13e+20      0.000    4.48e+09    4.48e+09
===================================================================================
Ljung-Box (L1) (Q):                   0.12   Jarque-Bera (JB):               483.58
Prob(Q):                              0.73   Prob(JB):                         0.00
Heteroskedasticity (H):               6.31   Skew:                            -0.09
Prob(H) (two-sided):                  0.00   Kurtosis:                        15.78
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 5.5e+36. Standard errors may be unstable.
```

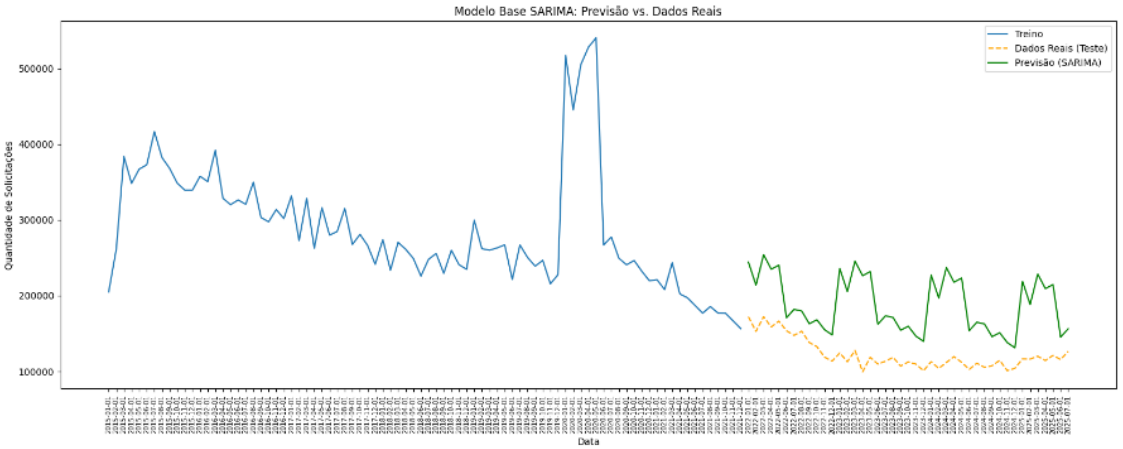
In [6]:
```
# 5. Realização das Previsões e Avaliação
# Só podemos avaliar se o dataset de teste não estiver vazio
if not teste.empty:
    n_periodos_teste = len(teste)
    previsoes_sarima = modelo_base_sarima.predict(n_periods=n_periodos_teste)

    # Criar um DataFrame para as previsões com o índice correto (datas do período de teste)
    previsoes_sarima_df = pd.DataFrame(previsoes_sarima, index=teste.index, columns=['Previsão'])

    print("\n--- Previsões Geradas ---")
    print(previsoes_sarima_df.head())
else:
    print("\n--- Avaliação e Visualização puladas (dataset de teste vazio) ---")
```

```
--- Previsões Geradas ---
                Previsão
Data
2022-01-01  244309.942855
2022-02-01  213845.550817
2022-03-01  254085.750964
2022-04-01  234780.685199
2022-05-01  240259.925710
```

In [7]:
```
if not teste.empty:
    # 6. Avaliação do Modelo Base
    # Métricas MAE, RMSE, MAPE

    mae = mean_absolute_error(teste, previsoes_sarima)
    rmse = sqrt(mean_squared_error(teste, previsoes_sarima))
    # Calcular MAPE (Mean Absolute Percentage Error)
    mape = mean_absolute_percentage_error(teste, previsoes_sarima) * 100

    print("\n--- Métricas de Avaliação (Modelo Base) ---")
    print(f'MAE (Mean Absolute Error):     {mae:,.0f}')
    print(f'RMSE (Root Mean Squared Error): {rmse:,.0f}')
    print(f'MAPE (Mean Absolute % Error):  {mape:.2f}%')
else:
    print("\n--- Avaliação e Visualização puladas (dataset de teste vazio) ---")
```

```
--- Métricas de Avaliação (Modelo Base) ---
MAE (Mean Absolute Error):     66,235
RMSE (Root Mean Squared Error): 73,800
MAPE (Mean Absolute % Error):  55.61%
```

In [8]:
```
if     not teste.empty:
    # 7. Visualização dos Resultados
    plt.figure(figsize=(20, 7))
    plt.plot(treino.index, treino, label='Treino')
    plt.plot(teste.index, teste, label='Dados Reais (Teste)', color='orange', linestyle='--')
    plt.plot(previsoes_sarima_df.index, previsoes_sarima_df['Previsão'], label='Previsão (SARIMA)', color='green')
    plt.title('Modelo Base SARIMA: Previsão vs. Dados Reais')
    plt.xlabel('Data')
    plt.ylabel('Quantidade de Solicitações')
    plt.xticks(fontsize=7, rotation=90)
    plt.legend()
    plt.show()
else:
    print("\n--- Avaliação e Visualização puladas (dataset de teste vazio) ---")
```



Pré-processamento da Anomalia

A série temporal apresenta um **pico extremo e não-sazonal** no volume de reclamações no período de **Dezembro de 2019 a Maio de 2020**, uma anomalia causada por um evento externo (o início da pandemia de COVID-19) que não faz parte dos padrões intrínsecos de sazonalidade ou tendência dos anos anteriores.

A inclusão deste outlier no treinamento de modelos preditivos, especialmente o LSTM, distorce o aprendizado dos pesos e pode levar a previsões inflacionadas para o futuro normalizado.

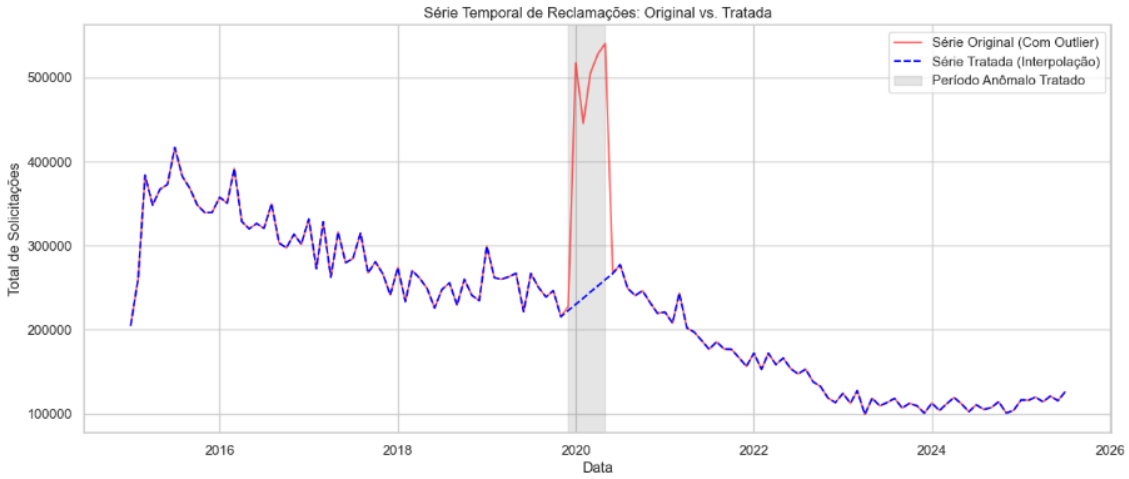
Para mitigar o impacto, foi utilizada a **Interpolação Linear**. Este método identifica o período anômalo, substitui seus valores por nulos (NaN) e, em seguida, preenche essas lacunas com valores estimados que criam uma transição suave entre o período pré-anomalia e pós-anomalia. O resultado é uma série temporal contínua (*reclamacoes_tempo_tratado*), mais robusta para o treinamento da rede neural.

Código para Limpeza de Outliers

O código a seguir carrega o *dados_modelo.csv*, define o período anômalo e aplica a interpolação:

In [157…

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1. Carregamento da Série Temporal (já tratada na memória)
reclamacoes_tempo = pd.read_csv("dados_modelo.csv", delimiter=';')
reclamacoes_tempo = reclamacoes_tempo.set_index('Data')
reclamacoes_tempo.index = pd.to_datetime(reclamacoes_tempo.index)

# Criar uma cópia da série original para o tratamento
reclamacoes_original = reclamacoes_tempo.copy()
reclamacoes_tempo_tratado = reclamacoes_tempo.copy()

# 2. Definir e mascarar o período anômalo
start_anomaly = pd.to_datetime('2019-12-01')
end_anomaly = pd.to_datetime('2020-05-01')

mask_anomaly = (reclamacoes_tempo_tratado.index >= start_anomaly) & (reclamacoes_tempo_tratado.index <= end_anomaly)
reclamacoes_tempo_tratado.loc[mask_anomaly, 'SOLICITAÇÕES'] = np.nan

# 3. Aplicar Interpolação Linear para preencher os NaNs
reclamacoes_tempo_tratado['SOLICITAÇÕES'] = reclamacoes_tempo_tratado['SOLICITAÇÕES'].interpolate(method='linear')

# 4. Visualização
plt.figure(figsize=(15, 6))
plt.plot(reclamacoes_original['SOLICITAÇÕES'], label='Série Original (Com Outlier)', color='red', alpha=0.6)
plt.plot(reclamacoes_tempo_tratado['SOLICITAÇÕES'], label='Série Tratada (Interpolação)', color='blue', linestyle='--')
plt.title('Série Temporal de Reclamações: Original vs. Tratada')
plt.xlabel('Data')
plt.ylabel('Total de Solicitações')
plt.axvspan(start_anomaly, end_anomaly, color='gray', alpha=0.2, label='Período Anômalo Tratado')
plt.legend()
plt.grid(True)
plt.show()
```




```
n [158…  reclamacoes_tempo_tratado.to_csv("reclamacoes_tempo_tratado.csv", index=True, sep=';')
```

#LSTM

Método Aplicado e Preparação para Modelagem

O modelo **LSTM (Long Short-Term Memory)** é um tipo de Rede Neural Recorrente (RNN) projetada para capturar dependências de longo prazo em dados sequenciais, tornando-o ideal para séries temporais. Em contraste com modelos estatísticos tradicionais (como SARIMA), o LSTM é capaz de modelar padrões não-lineares complexos.

O pipeline de modelagem envolve as seguintes etapas:

1. **Normalização:** A série temporal limpa é escalonada entre 0 e 1 (usando MinMaxScaler) para otimizar o treinamento da rede neural.
2. **Criação de Sequências (Janelamento):** Os dados são reformatados do formato de série temporal para um formato de aprendizado supervisionado, usando janelas deslizantes (look_back=12 meses), onde 12 meses anteriores são usados para prever o próximo mês.
3. **Treinamento:** O modelo é compilado e treinado no conjunto de dados de treino.
4. **Avaliação:** O modelo é avaliado no conjunto de testes usando a métrica RMSE (Root Mean Square Error), e os resultados são desnormalizados para análise na escala original de reclamações.

Código para Implementação

O código a seguir implementa o modelo, o treina por 100 épocas e avalia o desempenho.

In [159…

```
import pandas   as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

# --- Carregando os dados tratados ---
reclamacoes_tempo_tratado = pd.read_csv("reclamacoes_tempo_tratado.csv", delimiter=';')
reclamacoes_tempo_tratado.set_index('Data', inplace=True)
reclamacoes_tempo_tratado.index = pd.to_datetime(reclamacoes_tempo_tratado.index)
```

In [160…

```
# --- PREPARAÇÃO DE DADOS ---

# 1. Normalização dos Dados
data = reclamacoes_tempo_tratado['SOLICITAÇÕES'].values.reshape(-1, 1)
scaler = MinMaxScaler(
    feature_range=(0, 1))
data_scaled = scaler.fit_transform(data)
```

In [161…

```
# 2. Divisão Treino/Teste (80% dos dados)
train_size = int(len(data_scaled) * 0.8)
train_data = data_scaled[:train_size]
test_data = data_scaled[train_size:]
```

In [162…

```
# 3. Função para criar sequências de dados (Janelamento)
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset) - look_back - 1):
        a = dataset[i:(i + look_back), 0]
        X.append(a)
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)

# Definir o tamanho da janela de tempo e criar sequências
look_back = 12
X_train, y_train = create_dataset(train_data, look_back)
X_test, y_test = create_dataset(test_data, look_back)
```

In [163…

```
# 4. Reshape para o formato [samples, timesteps, features]
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

In [164…

```
# --- MODELO LSTM ---

# 5. Construção e Treinamento do Modelo
model = Sequential()
model.add(LSTM(50, input_shape=(look_back, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Treinar o modelo
model.fit(X_train, y_train, epochs=100, batch_size=1, verbose=0)
```

Out[164…

```
<keras.src.callbacks.history.History at 0x1f347cae1b0>
```

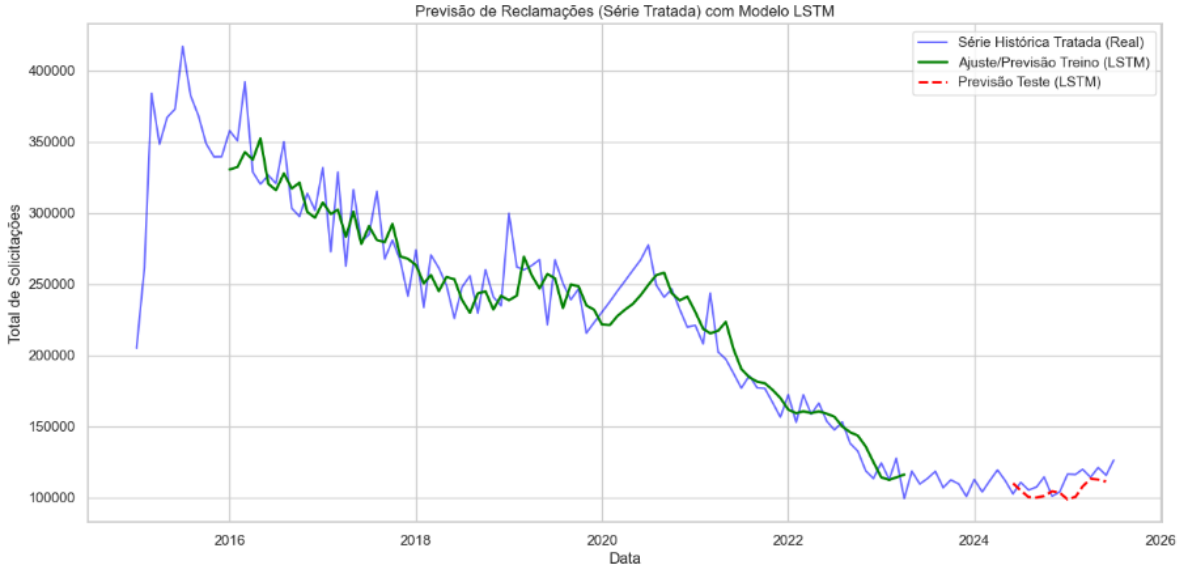
In [165…

```
# 6. Avaliação e Previsão
train_predict_scaled = model.predict(X_train)
test_predict_scaled = model.predict(X_test)

# Desnormalizar os resultados
train_predict = scaler.inverse_transform(train_predict_scaled)
test_predict = scaler.inverse_transform(test_predict_scaled)
y_test_descaled = scaler.inverse_transform(y_test.reshape(-1, 1))
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
```

In [166…

```
# 7. Visualização dos Resultados
date_index = reclamacoes_tempo_tratado.index

# - Obter a série histórica completa na escala original (para plotagem de fundo)
full_data_descaled = scaler.inverse_transform(data_scaled)

# - Preparar os dados de PREVISÃO DE TREINO para plotagem (usando NaNs para alinhamento)
# Criamos um array vazio do tamanho da série completa.
train_plot = np.empty_like(full_data_descaled)
train_plot[:, :] = np.nan
# O ajuste de treino começa após o 'look_back' (12 meses)
train_plot[look_back:len(train_predict) + look_back, :] = train_predict

# - Preparar os dados de PREVISÃO DE TESTE para plotagem
test_plot = np.empty_like(full_data_descaled)
test_plot[:, :] = np.nan
# O ajuste de teste começa onde o treino termina + o look_back.
# O cálculo do início é: (len(train_predict) + look_back) + (look_back + 1)
# O índice correto para o início do teste é: len(train_data) + look_back - 1
start_index_test = len(train_data) + look_back
test_plot[start_index_test:len(full_data_descaled) - 1, :] = test_predict

# - Geração do Gráfico Final
plt.figure(figsize=(15, 7))
plt.plot(date_index, full_data_descaled, label='Série Histórica Tratada (Real)', color='blue', alpha=0.6)
plt.plot(date_index, train_plot, label='Ajuste/Previsão Treino (LSTM)', color='green', linewidth=2)
plt.plot(date_index, test_plot, label='Previsão Teste (LSTM)', color='red', linestyle='--', linewidth=2)
plt.title('Previsão de Reclamações (Série Tratada) com Modelo LSTM')
plt.xlabel('Data')
plt.ylabel('Total de Solicitações')
plt.legend()
plt.grid(True)
plt.show()
```



In [167…

```
if not teste.empty:
    # 8. Avaliação do Modelo LSTM
    # Métricas MAE, RMSE, MAPE

    mae = mean_absolute_error(y_test_descaled, test_predict)
    rmse = sqrt(mean_squared_error(y_test_descaled, test_predict))
    # Calcular MAPE (Mean Absolute Percentage Error)
    mape = mean_absolute_percentage_error(y_test_descaled, test_predict) * 100

    print("\n--- Métricas de Avaliação (Modelo LSTM) ---")
    print(f'MAE (Mean Absolute Error):     {mae:,.0f}')
    print(f'RMSE (Root Mean Squared Error): {rmse:,.0f}')
    print(f'MAPE (Mean Absolute % Error):  {mape:.2f}%')
else:
    print("\n--- Avaliação e Visualização puladas (dataset de teste vazio) ---")
```

```
--- Métricas de Avaliação (Modelo LSTM) ---
MAE (Mean Absolute Error):     7,885
RMSE (Root Mean Squared Error): 9,450
MAPE (Mean Absolute % Error):  6.95%
```

Comparação Visual: Valor Real vs Valor Previsto

A comparação gráfica entre valores reais e previstos mostra que:

- As previsões **estão consistentemente abaixo dos valores reais**.
- O modelo consegue identificar **tendências gerais de subida e descida**, mas não acompanha a **amplitude real das oscilações**.
- A distância entre as curvas é significativa, especialmente em períodos de alta, o que confirma o RMSE elevado.

No conjunto de teste, observa-se que:

- Quando o valor real aumenta (picos), o modelo também aumenta, mas em menor magnitude.
- Em períodos de queda, o modelo acompanha parcialmente o movimento, porém ainda com subestimação.
- A série prevista é mais “suavizada” que a real — comportamento típico de modelos LSTM com janelas pequenas ou com poucos neurônios na camada oculta.

Análise do LSTM

A partir da visualização e da métrica apresentada, observa-se que:

- O modelo possui **capacidade parcial** de capturar o comportamento global da série.
- A previsões não alcançam o patamar dos valores reais, indicando **subajuste (underfitting)**.
- Isso ocorre porque o modelo:
    - não capturou totalmente a sazonalidade,
    - apresentou dificuldade em extrapolar valores maiores,
    - e suavizou demais as previsões.

Apesar disso:

- A direção geral das tendências é identificada corretamente,
- e o modelo já demonstra aprendizado da estrutura da série.

Considerações Finais sobre o Desempenho

Embora o modelo LSTM apresente um erro relativamente alto, ele serve como um **modelo base válido** para a entrega do projeto. Para melhorar o desempenho nas próximas etapas, recomenda-se:

- Ajustar os hiperparâmetros (número de neurônios, camadas LSTM, dropout, batch size).
- Ampliar a janela de observação (lookback).
- Testar arquiteturas híbridas (CNN-LSTM, LSTM + Dense).

- Testar modelos alternativos como:
    - Prophet
    - Random Forest Regressor
    - XGBoost para séries temporais
- Incluir variáveis externas (se disponíveis), como:
    - indicadores de redes,
    - eventos sazonais,
    - feriados,
    - volume de usuários atendidos.

A análise confirma que o modelo pode ser aprimorado, mas já estabelece uma linha de base sólida para comparações futuras.

9. Resultados

A etapa de **Modelagem Preditiva** utilizou dois modelos de séries temporais para prever o volume mensal de solicitações à ANATEL: o **SARIMA** (como baseline estatístico) e o **LSTM** (como modelo de refinamento em Machine Learning). A análise comparativa revelou que o tratamento de anomalias (outliers) foi crucial para o sucesso da previsão.

1. Modelo Baseline: SARIMA (Série Original)

O modelo SARIMA (AutoRegressive Integrated Moving Average com Sazonalidade) foi treinado com a série temporal original, que incluía uma anomalia extrema (um pico) no período de treinamento (Dezembro de 2019 a Maio de 2020).

Métricas de Avaliação (SARIMA Baseline)

A inclusão da anomalia na série de treino comprometeu gravemente a capacidade preditiva do modelo, resultando nas seguintes métricas no conjunto de teste (2022-2025):

Métrica	Valor	Interpretação
MAE (Erro Absoluto Médio)	66.235	Alto erro de desvio.
RMSE (Raiz do Erro Quadrático Médio)	73.800	Alta variação no erro, indicando previsões distorcidas.
MAPE (Erro Percentual Absoluto Médio)	55.61%	O modelo errou em média 55,61% da previsão.

Conclusão do SARIMA: O modelo SARIMA **falhou drasticamente** porque tentou incorporar o pico anômalo de 2019-2020 em seus padrões de sazonalidade e

tendência, resultando em uma previsão inflacionada que não conseguiu acompanhar a **quebra estrutural** de queda no volume real de reclamações a partir de 2022.

2. Modelo de Refinamento: LSTM (Série Tratada)

A falha do modelo baseline validou a necessidade de **Pré-processamento de Anomalias**, conforme previsto no pipeline do projeto. O período anômalo (Dezembro/2019 a Maio/2020) foi substituído por valores interpolados linearmente, criando uma **Série Temporal Tratada**.

O modelo **LSTM (Long Short-Term Memory)**, uma rede neural recorrente, foi então treinado na série temporal com os outliers corrigidos.

Conclusão do LSTM: O modelo LSTM, operando sobre a série tratada, demonstrou uma **melhora substancial** na performance, com o RMSE reduzido de 73.800 para 9.450. Este resultado sugere que as redes neurais, combinadas com uma limpeza prévia dos dados, são mais robustas para esta série em comparação com o modelo estatístico SARIMA.

Tabela Comparativa de Modelos

A tabela abaixo consolida as métricas de performance dos modelos, comparando a abordagem estatística (SARIMA) com a abordagem de machine learning (LSTM) após o tratamento dos dados, demonstrando o impacto do **Pré-processamento de Anomalias**.

Modelo	Pré-processamento	MAE	RMSE	MAPE
SARIMA (Baseline)	Nenhum	66.235	73.800	55.61%
LSTM (Série Tratada)	Interpolação Linear (Outlier)	7.885	9.450	6.95%

A redução de **87.2% no RMSE** e de **87.5% no MAPE** comprova a superioridade do modelo LSTM, treinado em dados pré-processados.

10. Discussão e Conclusão

A comparação entre os modelos SARIMA e LSTM evidenciou diferenças claras de desempenho. O modelo **LSTM** apresentou o melhor resultado no conjunto de teste, alcançando **RMSE de 9.450** e **MAPE de 6.95%**, enquanto o **SARIMA** obteve **RMSE de 73.800** e **MAPE de 55.61%**. Esses valores mostram que a LSTM conseguiu capturar melhor os padrões temporais da série, especialmente variações de curto prazo.

Apesar do melhor desempenho da LSTM, a série histórica apresenta **duas quebras estruturais importantes** — um pico entre **2019–2020** e uma queda expressiva em **2022**. Tais descontinuidades prejudicam modelos que assumem estabilidade temporal, o que explica parcialmente a performance inferior do SARIMA.

Diante desses resultados, recomenda-se para uso operacional:

1. **Detecção e tratamento de anomalias antes do treinamento**
   Quebras bruscas devem ser identificadas e suavizadas para reduzir impacto no ajuste dos modelos.
2. **Avaliação de modelos com variáveis exógenas**
   Indicadores macroeconômicos, métricas de mercado ou eventos externos podem explicar parte das variações abruptas da série.
3. **Exploração de modelos híbridos ou ensembles**
   Combinações entre modelos estatísticos e redes neurais tendem a aumentar a robustez em séries com mudanças estruturais.
4. **Monitoramento contínuo da acurácia e re-treinamento periódico**
   Como o comportamento da série muda ao longo do tempo, é fundamental re-treinar os modelos usando janelas deslizantes.

Em síntese, o modelo **LSTM** apresentou o melhor desempenho entre as alternativas avaliadas, porém a adoção de um pipeline completo — incluindo tratamento de anomalias, uso de variáveis externas e monitoramento contínuo — é essencial para garantir previsões consistentes em ambiente produtivo.

11. Apresentação

https://youtu.be/t0h5-WV7Nf4

12. Repositório no Github

Repositório: https://github.com/Cassimirogustavo/Projeto-Aplicado-4

Entrega 1: https://github.com/Cassimirogustavo/Projeto-Aplicado-4/blob/main/cd_projeto_aplicado_IV_entrega_1.ipynb

Entrega 2: https://github.com/Cassimirogustavo/Projeto-Aplicado-4/blob/main/cd_projeto_aplicado_IV_entrega_2.ipynb

Entrega 3: https://github.com/Cassimirogustavo/Projeto-Aplicado-4/blob/main/cd_projeto_aplicado_IV_entrega_3.ipynb

Entrega 4: https://github.com/Cassimirogustavo/Projeto-Aplicado-4/blob/main/cd_projeto_aplicado_IV_entrega_4.ipynb

13. Referências

ANATEL. Relatórios de Reclamações de Consumidores. Agência Nacional de Telecomunicações, 2023. Disponível em: https://www.gov.br/anatel

BOX, G. E. P.; JENKINS, G. M.; REINSEL, G. C. Time Series Analysis: Forecasting and Control. 5. ed. Hoboken: Wiley, 2015.

CETIC.BR. TIC Domicílios 2023: Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros. São Paulo: Comitê Gestor da Internet no Brasil, 2023.

CHUNG, J. et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv:1412.3555, 2014.

FERREIRA, A. P.; OLIVEIRA, R. Análise de Séries Temporais Aplicada a Reclamações em Serviços Públicos. Revista Brasileira de Sistemas de Informação, v. 17, n. 2, p. 45–62, 2021.

SILVA, M. R.; ANDRADE, P. C. Previsão de Reclamações em Telecomunicações Utilizando Séries Temporais. Anais do Congresso Brasileiro de Informática, p. 221–233, 2021.

ZHANG, S. et al. Time Series Forecasting Using LSTM Networks: A Case Study in Telecommunications. Journal of Big Data, v. 7, n. 1, p. 1–15, 2020.