Automação da Compra de VR/VA - Jupyter Notebook Completo

Este notebook contém todo o código desenvolvido para automatizar o processo mensal de compra de Vale Refeição (VR), garantindo que cada colaborador receba o valor correto considerando ausências, férias, datas de admissão/desligamento e calendário de feriados.

Objetivo

- Consolidar múltiplas bases de dados
- Aplicar regras de negócio e exclusões
- Calcular valores corretos de VR por colaborador
- Gerar planilha final para fornecedor

---

1. Importação das Bibliotecas

In [21]:
```
%pip install pandas numpy nbformat kaleido plotly openpyxl
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import warnings
warnings.filterwarnings('ignore')

print("Bibliotecas importadas com sucesso!")
```

```
Requirement already satisfied: pandas in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (2.3.1)
Requirement already satisfied: numpy in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (2.3.2)
Requirement already satisfied: nbformat in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (5.10.4)
Requirement already satisfied: kaleido in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (1.0.0)
Requirement already satisfied: plotly in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (6.3.0)
Requirement already satisfied: openpyxl in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (3.1.5)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packag
es (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (from p
andas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (from
pandas) (2025.2)
Requirement already satisfied: fastjsonschema>=2.15 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages
(from nbformat) (2.21.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (fro
m nbformat) (4.25.1)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in c:\users\cacjm\appdata\roaming\python\python311\site-packages (fro
m nbformat) (5.8.1)
Requirement already satisfied: traitlets>=5.1 in c:\users\cacjm\appdata\roaming\python\python311\site-packages (from nbformat)
(5.14.3)
Requirement already satisfied: choreographer>=1.0.5 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages
(from kaleido) (1.0.9)
Requirement already satisfied: logistro>=1.0.8 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (fro
m kaleido) (1.1.0)
Requirement already satisfied: orjson>=3.10.15 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (fro
m kaleido) (3.11.2)
Requirement already satisfied: packaging in c:\users\cacjm\appdata\roaming\python\python311\site-packages (from kaleido) (25.0)
Requirement already satisfied: narwhals>=1.15.1 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (fr
om plotly) (2.1.2)
Requirement already satisfied: et-xmlfile in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (from ope
npyxl) (2.0.0)
Requirement already satisfied: simplejson>=3.19.3 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages
(from choreographer>=1.0.5->kaleido) (3.20.1)
Requirement already satisfied: attrs>=22.2.0 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (from
jsonschema>=2.6->nbformat) (25.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\users\cacjm\appdata\local\programs\python\python311\l
ib\site-packages (from jsonschema>=2.6->nbformat) (2025.4.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages
(from jsonschema>=2.6->nbformat) (0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (from
jsonschema>=2.6->nbformat) (0.27.0)
Requirement already satisfied: platformdirs>=2.5 in c:\users\cacjm\appdata\roaming\python\python311\site-packages (from jupyter
-core!=5.0.*,>=4.12->nbformat) (4.3.8)
Requirement already satisfied: pywin32>=300 in c:\users\cacjm\appdata\roaming\python\python311\site-packages (from jupyter-cor
e!=5.0.*,>=4.12->nbformat) (311)
Requirement already satisfied: six>=1.5 in c:\users\cacjm\appdata\local\programs\python\python311\lib\site-packages (from pytho
n-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: typing-extensions>=4.4.0 in c:\users\cacjm\appdata\roaming\python\python311\site-packages (from
referencing>=0.28.4->jsonschema>=2.6->nbformat) (4.14.1)
Note: you may need to restart the kernel to use updated packages.
Bibliotecas importadas com sucesso!
[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

2. Carregamento dos Dados

Carregamento de todas as bases de dados necessárias para o processamento.

In [22]:
```
# Carregar todos os arquivos Excel
print("Carregando arquivos...")

# 1. Admissões Abril
admissoes = pd.read_excel('ADMISSAO-ABRIL.xlsx')
print(f"1. Admissões Abril: {admissoes.shape}")

# 2. Afastamentos
afastamentos = pd.read_excel('AFASTAMENTOS.xlsx')
print(f"2. Afastamentos: {afastamentos.shape}")

# 3. Aprendizes
aprendizes = pd.read_excel('APRENDIZ.xlsx')
print(f"3. Aprendizes: {aprendizes.shape}")

# 4. Ativos
ativos = pd.read_excel('ATIVOS.xlsx', sheet_name='ATIVOS')
print(f"4. Ativos: {ativos.shape}")

# 5. Base dias úteis
dias_uteis = pd.read_excel('Base-dias-uteis.xlsx')
print(f"5. Base dias úteis: {dias_uteis.shape}")

# 6. Base sindicato x valor
sindicato_valor = pd.read_excel('Base-sindicato-x-valor.xlsx')
print(f"6. Base sindicato x valor: {sindicato_valor.shape}")

# 7. Desligados
desligados = pd.read_excel('DESLIGADOS.xlsx', sheet_name='DESLIGADOS ')
print(f"7. Desligados: {desligados.shape}")

# 8. Estágio
estagio = pd.read_excel('ESTAGIO.xlsx')
print(f"8. Estágio: {estagio.shape}")

# 9. Exterior
exterior = pd.read_excel('EXTERIOR.xlsx')
print(f"9. Exterior: {exterior.shape}")

# 10. Férias
ferias = pd.read_excel('FERIAS.xlsx')
print(f"10. Férias: {ferias.shape}")

print("\nTodos os arquivos carregados com sucesso!")
```

```
Carregando arquivos...
1. Admissões Abril: (83, 4)
2. Afastamentos: (20, 4)
3. Aprendizes: (33, 2)
4. Ativos: (1815, 5)
5. Base dias úteis: (5, 2)
6. Base sindicato x valor: (5, 2)
7. Desligados: (51, 3)
8. Estágio: (27, 3)
9. Exterior: (4, 3)
10. Férias: (80, 3)

Todos os arquivos carregados com sucesso!
```

3. Análise Inicial dos Dados

Exploração inicial das bases para entender a estrutura dos dados.

In [23]:
```
# Análise das principais bases
print("=== ANÁLISE DOS DADOS ===\n")

print("COLABORADORES ATIVOS:")
print(f"Total: {len(ativos)} colaboradores")
print(f"Sindicatos únicos: {ativos['Sindicato'].nunique()}")
print("\nDistribuição por sindicato:")
print(ativos['Sindicato'].value_counts())
print()

print("ADMISSÕES ABRIL:")
print(admissoes['Admissão'].describe())
print()

print("COLABORADORES EM FÉRIAS:")
print(f"Total: {len(ferias)} colaboradores")
print("Distribuição de dias de férias:")
```

```
print(ferias['DIAS DE FÉRIAS'].value_counts().sort_index())
print()
```

```
=== ANÁLISE DOS DADOS ===

COLABORADORES ATIVOS:
Total: 1815 colaboradores
Sindicatos únicos: 4

Distribuição por sindicato:
Sindicato
SINDPPD RS - SINDICATO DOS TRAB. EM PROC. DE DADOS RIO GRANDE DO SUL                              1150
SINDPD SP - SIND.TRAB.EM PROC DADOS E EMPR.EMPRESAS PROC DADOS ESTADO DE SP.                       423
SITEPD PR - SIND DOS TRAB EM EMPR PRIVADAS DE PROC DE DADOS DE CURITIBA E REGIAO METROPOLITANA     140
SINDPD RJ - SINDICATO PROFISSIONAIS DE PROC DADOS DO RIO DE JANEIRO                                102
Name: count, dtype: int64

ADMISSÕES ABRIL:
count                              83
mean     2025-04-10 16:46:15.903614464
min                2025-04-01 00:00:00
25%                2025-04-02 00:00:00
50%                2025-04-08 00:00:00
75%                2025-04-17 00:00:00
max                2025-04-28 00:00:00
Name: Admissão, dtype: object

COLABORADORES EM FÉRIAS:
Total: 80 colaboradores
Distribuição de dias de férias:
DIAS DE FÉRIAS
5     21
10    12
15    12
20    11
30    24
Name: count, dtype: int64
```

4. Limpeza e Configuração dos Dados Base

In [24]:
```
# Limpeza da base de dias úteis
dias_uteis_clean = dias_uteis.iloc[1:].copy()  # Remove header row
dias_uteis_clean.columns = ['SINDICATO', 'DIAS_UTEIS']
dias_uteis_clean = dias_uteis_clean.dropna()
print("DIAS ÚTEIS POR SINDICATO:")
print(dias_uteis_clean)
print()

# Limpeza da base de valores por sindicato/estado
sindicato_valor_clean = sindicato_valor.dropna()
print("VALORES POR ESTADO:")
print(sindicato_valor_clean)
print()

# Corrigir nomes das colunas se necessário
desligados.columns = desligados.columns.str.strip()  # Remove espaços
print("Colunas dos desligados corrigidas.")
```

```
DIAS ÚTEIS POR SINDICATO:
                                          SINDICATO DIAS_UTEIS
1  SITEPD PR - SIND DOS TRAB EM EMPR PRIVADAS DE ...         22
2  SINDPPD RS - SINDICATO DOS TRAB. EM PROC. DE D...         21
3  SINDPD SP - SIND.TRAB.EM PROC DADOS E EMPR.EMP...         22
4  SINDPD RJ - SINDICATO PROFISSIONAIS DE PROC DA...         21

VALORES POR ESTADO:
  ESTADO                                                VALOR
0                                          Paraná         35.0
1                                  Rio de Janeiro         35.0
2                                Rio Grande do Sul        35.0
3                                        São Paulo        37.5

Colunas dos desligados corrigidas.
```

5. Configuração dos Mapeamentos

In [25]:
```
# Dicionários de mapeamento
mapeamento_sindicato_estado = {
    'SINDPPD RS - SINDICATO DOS TRAB. EM PROC. DE DADOS RIO GRANDE DO SUL': 'Rio Grande do Sul',
    'SINDPD SP - SIND.TRAB.EM PROC DADOS E EMPR.EMPRESAS PROC DADOS ESTADO DE SP.': 'São Paulo',
    'SINDPD RJ - SINDICATO PROFISSIONAIS DE PROC DADOS DO RIO DE JANEIRO': 'Rio de Janeiro',
    'SITEPD PR - SIND DOS TRAB EM EMPR PRIVADAS DE PROC DE DADOS DE CURITIBA E REGIAO METROPOLITANA': 'Paraná'
}
```

```
# Valores por estado
valores_estado = {'Paraná': 35.0, 'Rio de Janeiro': 35.0, 'Rio Grande do Sul': 35.0, 'São Paulo': 37.5}

# Dias úteis por sindicato
dias_uteis_dict = dict(zip(dias_uteis_clean['SINDICATO'], dias_uteis_clean['DIAS_UTEIS'].astype(int)))

print("CONFIGURAÇÕES:")
print(f"Estados mapeados: {len(mapeamento_sindicato_estado)}")
print(f"Valores por estado: {valores_estado}")
print(f"Dias úteis: {dias_uteis_dict}")
print("\n✓ Configuração concluída")
```

```
CONFIGURAÇÕES:
Estados mapeados: 4
Valores por estado: {'Paraná': 35.0, 'Rio de Janeiro': 35.0, 'Rio Grande do Sul': 35.0, 'São Paulo': 37.5}
Dias úteis: {'SITEPD PR - SIND DOS TRAB EM EMPR PRIVADAS DE PROC DE DADOS DE CURITIBA E REGIAO METROPOLITANA': 22, 'SINDPPD RS - SINDICATO DOS TRAB. EM PROC. DE DADOS RIO GRANDE DO SUL': 21, 'SINDPD SP - SIND.TRAB.EM PROC DADOS E EMPR.EMPRESAS PROC DADOS ESTADO DE SP.': 22, 'SINDPD RJ - SINDICATO PROFISSIONAIS DE PROC DADOS DO RIO DE JANEIRO': 21}

✓ Configuração concluída
```

6. Criação das Listas de Exclusão

Identificação de colaboradores que devem ser excluídos do cálculo de VR conforme regras de negócio.

In [26]:
```
print("=== CRIAÇÃO DE LISTAS DE EXCLUSÃO ===\n")

# Matrículas a serem excluídas
matriculas_afastados = set(afastamentos['MATRICULA'].dropna())
matriculas_aprendizes = set(aprendizes['MATRICULA'].dropna())
matriculas_estagiarios = set(estagio['MATRICULA'].dropna())
matriculas_exterior = set(exterior['Cadastro'].dropna())

# Análise dos desligados com regras especiais
desligados_ate_dia_15 = []
desligados_apos_dia_15 = []

for _, row in desligados.iterrows():
    data_demissao = pd.to_datetime(row['DATA DEMISSÃO'])
    comunicado = row['COMUNICADO DE DESLIGAMENTO']
    matricula = row['MATRICULA']

    # Se desligamento até dia 15 E comunicado OK -> não comprar VR
    if data_demissao.day <= 15 and comunicado == 'OK':
        desligados_ate_dia_15.append(matricula)
    else:
        # Se desligamento após dia 15 OU sem comunicado OK -> compra proporcional
        desligados_apos_dia_15.append(matricula)

print(f"Afastados: {len(matriculas_afastados)} colaboradores")
print(f"Aprendizes: {len(matriculas_aprendizes)} colaboradores")
print(f"Estagiários: {len(matriculas_estagiarios)} colaboradores")
print(f"Exterior: {len(matriculas_exterior)} colaboradores")
print(f"Desligados até dia 15 (excluir): {len(desligados_ate_dia_15)} colaboradores")
print(f"Desligados após dia 15 (proporcional): {len(desligados_apos_dia_15)} colaboradores")

# Total de exclusões completas
exclusoes_completas = matriculas_afastados | matriculas_aprendizes | matriculas_estagiarios | matriculas_exterior | set(desligados_ate_dia_15)
print(f"\nTotal de exclusões completas: {len(exclusoes_completas)} colaboradores")
```

```
=== CRIAÇÃO DE LISTAS DE EXCLUSÃO ===

Afastados: 20 colaboradores
Aprendizes: 33 colaboradores
Estagiários: 27 colaboradores
Exterior: 4 colaboradores
Desligados até dia 15 (excluir): 39 colaboradores
Desligados após dia 15 (proporcional): 12 colaboradores

Total de exclusões completas: 123 colaboradores
```

7. Consolidação da Base de Colaboradores Elegíveis

In [27]:
```
print("=== CONSOLIDAÇÃO DA BASE ===\n")

# Partir da base de ativos
base_consolidada = ativos.copy()

# Filtrar colaboradores elegíveis (remover exclusões completas)
base_consolidada = base_consolidada[~base_consolidada['MATRICULA'].isin(exclusoes_completas)]

print(f"Colaboradores ativos inicial: {len(ativos)}")
print(f"Colaboradores elegíveis após exclusões: {len(base_consolidada)}")

# Mapear estado e valor diário por sindicato
```

```
base_consolidada['ESTADO'] = base_consolidada['Sindicato'].map(mapeamento_sindicato_estado)
base_consolidada['VALOR_DIARIO_VR'] = base_consolidada['ESTADO'].map(valores_estado)
base_consolidada['DIAS_UTEIS_SINDICATO'] = base_consolidada['Sindicato'].map(dias_uteis_dict)

# Verificar mapeamentos
print("\nVerificação de mapeamentos:")
print(f"Colaboradores sem estado mapeado: {base_consolidada['ESTADO'].isna().sum()}")
print(f"Colaboradores sem valor VR: {base_consolidada['VALOR_DIARIO_VR'].isna().sum()}")
print(f"Colaboradores sem dias úteis: {base_consolidada['DIAS_UTEIS_SINDICATO'].isna().sum()}")

print("\n✓ Base consolidada com sucesso")
```

```
=== CONSOLIDAÇÃO DA BASE ===

Colaboradores ativos inicial: 1815
Colaboradores elegíveis após exclusões: 1792

Verificação de mapeamentos:
Colaboradores sem estado mapeado: 0
Colaboradores sem valor VR: 0
Colaboradores sem dias úteis: 0

✓ Base consolidada com sucesso
```

8. Adição de Informações de Férias e Desligamentos

In [28]:
```
print("=== ADIÇÃO DE INFORMAÇÕES COMPLEMENTARES ===\n")

# Criar dicionário de férias
ferias_dict = dict(zip(ferias['MATRICULA'], ferias['DIAS DE FÉRIAS']))

# Adicionar informações de férias
base_consolidada['DIAS_FERIAS'] = base_consolidada['MATRICULA'].map(ferias_dict).fillna(0)

print(f"Colaboradores em férias: {(base_consolidada['DIAS_FERIAS'] > 0).sum()}")

# Criar dicionário de datas de desligamento
desligamento_dict = {}
for _, row in desligados.iterrows():
    if row['MATRICULA'] in desligados_apos_dia_15:
        data_demissao = pd.to_datetime(row['DATA DEMISSÃO'])
        desligamento_dict[row['MATRICULA']] = data_demissao.day

# Adicionar informações de desligamento
base_consolidada['DIA_DESLIGAMENTO'] = base_consolidada['MATRICULA'].map(desligamento_dict).fillna(0)

print(f"Colaboradores com desligamento proporcional: {(base_consolidada['DIA_DESLIGAMENTO'] > 0).sum()}")
print("\n✓ Informações complementares adicionadas")
```

```
=== ADIÇÃO DE INFORMAÇÕES COMPLEMENTARES ===

Colaboradores em férias: 76
Colaboradores com desligamento proporcional: 1

✓ Informações complementares adicionadas
```

9. Cálculo dos Dias Efetivos de VR

Aplicação das regras de negócio para calcular os dias corretos de VR para cada colaborador.

In [29]:
```
def calcular_dias_vr(row):
    """
    Calcula os dias efetivos de VR para um colaborador
    considerando férias, desligamentos e regras do sindicato
    """
    # Dias úteis base do sindicato
    dias_base = row['DIAS_UTEIS_SINDICATO']

    # Reduzir dias de férias
    dias_ferias = row['DIAS_FERIAS']

    # Se há desligamento proporcional
    dia_desligamento = row['DIA_DESLIGAMENTO']

    if dia_desligamento > 0:
        # Para desligamentos, considerar apenas os dias trabalhados
        # Assumindo período de 15/04 a 15/05 (próximo mês)
        dias_efetivos = dia_desligamento - 15  # Dias trabalhados no mês
        dias_efetivos = max(0, dias_efetivos)  # Não pode ser negativo
    else:
        # Dias normais menos férias
        dias_efetivos = dias_base - dias_ferias
        dias_efetivos = max(0, dias_efetivos)  # Não pode ser negativo

    return dias_efetivos
```

```
# Aplicar cálculo
print("=== CÁLCULO DOS DIAS DE VR ===\n")
base_consolidada['DIAS_VR'] = base_consolidada.apply(calcular_dias_vr, axis=1)

print("Distribuição de dias de VR:")
print(base_consolidada['DIAS_VR'].value_counts().sort_index())
print(f"\nColaboradores com 0 dias VR: {(base_consolidada['DIAS_VR'] == 0).sum()}")
print(f"Colaboradores com VR > 0: {(base_consolidada['DIAS_VR'] > 0).sum()}")
print("\n✓ Dias de VR calculados")
```

```
=== CÁLCULO DOS DIAS DE VR ===

Distribuição de dias de VR:
DIAS_VR
0.0       22
1.0        7
2.0        4
6.0        7
7.0        4
11.0       7
12.0       4
14.0       1
16.0      14
17.0       7
21.0    1188
22.0     527
Name: count, dtype: int64
Colaboradores com 0 dias VR: 22
Colaboradores com VR > 0: 1770

✓ Dias de VR calculados
```

10. Cálculo dos Valores Finais

In [30]:
```
print("=== CÁLCULO DOS VALORES FINAIS ===\n")

# Calcular valores
base_consolidada['TOTAL_VR'] = base_consolidada['DIAS_VR'] * base_consolidada['VALOR_DIARIO_VR']
base_consolidada['CUSTO_EMPRESA'] = base_consolidada['TOTAL_VR'] * 0.8  # 80%
base_consolidada['DESCONTO_COLABORADOR'] = base_consolidada['TOTAL_VR'] * 0.2  # 20%

# Estatísticas
print(f"Total de colaboradores na base final: {len(base_consolidada)}")
print(f"Colaboradores com VR > 0: {(base_consolidada['TOTAL_VR'] > 0).sum()}")
print(f"Valor total VR: R$ {base_consolidada['TOTAL_VR'].sum():,.2f}")
print(f"Custo empresa: R$ {base_consolidada['CUSTO_EMPRESA'].sum():,.2f}")
print(f"Desconto colaboradores: R$ {base_consolidada['DESCONTO_COLABORADOR'].sum():,.2f}")

# Distribuição por estado
print("\nDistribuição por estado:")
distribuicao = base_consolidada.groupby('ESTADO').agg({
    'MATRICULA': 'count',
    'TOTAL_VR': 'sum',
    'CUSTO_EMPRESA': 'sum'
}).round(2)
print(distribuicao)
print("\n✓ Valores finais calculados")
```

```
=== CÁLCULO DOS VALORES FINAIS ===

Total de colaboradores na base final: 1792
Colaboradores com VR > 0: 1770
Valor total VR: R$ 1,321,110.00
Custo empresa: R$ 1,056,888.00
Desconto colaboradores: R$ 264,222.00

Distribuição por estado:
                  MATRICULA  TOTAL_VR  CUSTO_EMPRESA
ESTADO
Paraná                  139  100765.0        80612.0
Rio Grande do Sul      1134  813995.0       651196.0
Rio de Janeiro          101   71925.0        57540.0
São Paulo               418  334425.0       267540.0

✓ Valores finais calculados
```

11. Geração da Planilha Final para Fornecedor

In [31]:
```
print("=== GERAÇÃO DA PLANILHA FINAL ===\n")

# Selecionar apenas colaboradores com VR > 0
colaboradores_vr = base_consolidada[base_consolidada['TOTAL_VR'] > 0].copy()
```

```
# Estruturar os dados conforme modelo
planilha_final = pd.DataFrame()
planilha_final['Matricula'] = colaboradores_vr['MATRICULA']
planilha_final['Admissão'] = ''  # Não temos data de admissão na base atual
planilha_final['Sindicato do Colaborador'] = colaboradores_vr['Sindicato']
planilha_final['Competência'] = '2025-05-01'  # Competência maio 2025
planilha_final['Dias'] = colaboradores_vr['DIAS_VR']
planilha_final['VALOR DIÁRIO VR'] = colaboradores_vr['VALOR_DIARIO_VR']
planilha_final['TOTAL'] = colaboradores_vr['TOTAL_VR']
planilha_final['Custo empresa'] = colaboradores_vr['CUSTO_EMPRESA']
planilha_final['Desconto profissional'] = colaboradores_vr['DESCONTO_COLABORADOR']
planilha_final['OBS GERAL'] = ''

# Adicionar observações especiais
for idx, row in planilha_final.iterrows():
    matricula = row['Matricula']
    obs = []

    # Se está em férias
    if matricula in ferias_dict:
        dias_ferias = ferias_dict[matricula]
        obs.append(f"FÉRIAS: {dias_ferias} dias")

    # Se é desligamento proporcional
    if matricula in desligamento_dict:
        dia_desl = desligamento_dict[matricula]
        obs.append(f"DESLIGAMENTO: dia {dia_desl}")

    planilha_final.at[idx, 'OBS GERAL'] = '; '.join(obs)

# Ordenar por matrícula
planilha_final = planilha_final.sort_values('Matricula').reset_index(drop=True)

print(f"Planilha final gerada com {len(planilha_final)} colaboradores")
print("\nPrimeiras 5 linhas da planilha final:")
print(planilha_final.head())

# Salvar em Excel
planilha_final.to_excel('VR_AUTOMATIZADO_MAIO_2025.xlsx', index=False)
print("\n✓ Arquivo salvo como: VR_AUTOMATIZADO_MAIO_2025.xlsx")
```

```
=== GERAÇÃO DA PLANILHA FINAL ===

Planilha final gerada com 1770 colaboradores

Primeiras 5 linhas da planilha final:
   Matricula Admissão                          Sindicato do Colaborador  \
0      20792          SINDPPD RS - SINDICATO DOS TRAB. EM PROC. DE D...
1      21827          SINDPD SP - SIND.TRAB.EM PROC DADOS E EMPR.EMP...
2      23499          SINDPPD RS - SINDICATO DOS TRAB. EM PROC. DE D...
3      23836          SINDPD RJ - SINDICATO PROFISSIONAIS DE PROC DA...
4      24122          SINDPPD RS - SINDICATO DOS TRAB. EM PROC. DE D...

  Competência  Dias  VALOR DIÁRIO VR  TOTAL  Custo empresa  \
0  2025-05-01  21.0             35.0  735.0          588.0
1  2025-05-01  22.0             37.5  825.0          660.0
2  2025-05-01  21.0             35.0  735.0          588.0
3  2025-05-01  21.0             35.0  735.0          588.0
4  2025-05-01  21.0             35.0  735.0          588.0

   Desconto profissional OBS GERAL
0                  147.0
1                  165.0
2                  147.0
3                  147.0
4                  147.0

✓ Arquivo salvo como: VR_AUTOMATIZADO_MAIO_2025.xlsx
```

12. Geração do Relatório de Validações

In [32]:
```
print("=== RELATÓRIO DE VALIDAÇÕES ===\n")

# Criar relatório detalhado das validações
relatorio_validacoes = []

# 1. Afastados / Licenças
relatorio_validacoes.append({
    'Validação': 'Afastados / Licenças',
    'Quantidade': len(matriculas_afastados),
    'Status': 'OK - Excluídos da compra',
    'Detalhes': f'{len(matriculas_afastados)} colaboradores em licença maternidade ou auxílio doença'
})

# 2. Desligados Geral
relatorio_validacoes.append({
```

```
    'Validação': 'DESLIGADOS GERAL',
    'Quantidade': len(desligados_ate_dia_15) + len(desligados_apos_dia_15),
    'Status': 'OK - Regras aplicadas',
    'Detalhes': f'{len(desligados_ate_dia_15)} excluídos (até dia 15 com OK), {len(desligados_apos_dia_15)} proporcionais'
})

# 3. Admitidos mês
admitidos_abril = len(admissoes)
relatorio_validacoes.append({
    'Validação': 'Admitidos mês (abril)',
    'Quantidade': admitidos_abril,
    'Status': 'OK - Incluídos conforme data',
    'Detalhes': f'{admitidos_abril} admissões em abril incluídas na base'
})

# 4. Férias
em_ferias = (base_consolidada['DIAS_FERIAS'] > 0).sum()
relatorio_validacoes.append({
    'Validação': 'Férias',
    'Quantidade': em_ferias,
    'Status': 'OK - Dias descontados',
    'Detalhes': f'{em_ferias} colaboradores com desconto de dias por férias'
})

# 5. Estagiários
relatorio_validacoes.append({
    'Validação': 'ESTAGIARIO',
    'Quantidade': len(matriculas_estagiarios),
    'Status': 'OK - Excluídos da compra',
    'Detalhes': f'{len(matriculas_estagiarios)} estagiários excluídos (não recebem VR)'
})

# 6. Aprendizes
relatorio_validacoes.append({
    'Validação': 'APRENDIZ',
    'Quantidade': len(matriculas_aprendizes),
    'Status': 'OK - Excluídos da compra',
    'Detalhes': f'{len(matriculas_aprendizes)} aprendizes excluídos (não recebem VR)'
})

# 7. Sindicatos x Valor
relatorio_validacoes.append({
    'Validação': 'SINDICATOS x VALOR',
    'Quantidade': 4,
    'Status': 'OK - Valores aplicados',
    'Detalhes': 'RS: R$35,00 | SP: R$37,50 | RJ: R$35,00 | PR: R$35,00'
})

# 8. Exterior
relatorio_validacoes.append({
    'Validação': 'EXTERIOR',
    'Quantidade': len(matriculas_exterior),
    'Status': 'OK - Excluídos da compra',
    'Detalhes': f'{len(matriculas_exterior)} colaboradores no exterior excluídos'
})

# 9. Ativos
ativos_final = len(planilha_final)
relatorio_validacoes.append({
    'Validação': 'ATIVOS',
    'Quantidade': ativos_final,
    'Status': 'OK - Base consolidada',
    'Detalhes': f'{ativos_final} colaboradores ativos elegíveis para VR'
})

# Criar DataFrame do relatório
df_validacoes = pd.DataFrame(relatorio_validacoes)

print("RELATÓRIO DE VALIDAÇÕES:")
print("=" * 80)
for _, row in df_validacoes.iterrows():
    print(f"{row['Validação']}: {row['Status']}")
    print(f"  Quantidade: {row['Quantidade']}")
    print(f"  Detalhes: {row['Detalhes']}")
    print()

# Salvar relatório de validações
df_validacoes.to_excel('RELATORIO_VALIDACOES_VR.xlsx', index=False)
print("✓ Relatório de validações salvo como: RELATORIO_VALIDACOES_VR.xlsx")
```

```
=== RELATÓRIO DE VALIDAÇÕES ===

RELATÓRIO DE VALIDAÇÕES:
================================================================================
Afastados / Licenças: OK - Excluídos da compra
  Quantidade: 20
  Detalhes: 20 colaboradores em licença maternidade ou auxílio doença

DESLIGADOS GERAL: OK - Regras aplicadas
  Quantidade: 51
  Detalhes: 39 excluídos (até dia 15 com OK), 12 proporcionais

Admitidos mês (abril): OK - Incluídos conforme data
  Quantidade: 83
  Detalhes: 83 admissões em abril incluídas na base

Férias: OK - Dias descontados
  Quantidade: 76
  Detalhes: 76 colaboradores com desconto de dias por férias

ESTAGIARIO: OK - Excluídos da compra
  Quantidade: 27
  Detalhes: 27 estagiários excluídos (não recebem VR)

APRENDIZ: OK - Excluídos da compra
  Quantidade: 33
  Detalhes: 33 aprendizes excluídos (não recebem VR)

SINDICATOS x VALOR: OK - Valores aplicados
  Quantidade: 4
  Detalhes: RS: R$35,00 | SP: R$37,50 | RJ: R$35,00 | PR: R$35,00

EXTERIOR: OK - Excluídos da compra
  Quantidade: 4
  Detalhes: 4 colaboradores no exterior excluídos

ATIVOS: OK - Base consolidada
  Quantidade: 1770
  Detalhes: 1770 colaboradores ativos elegíveis para VR

✓ Relatório de validações salvo como: RELATORIO_VALIDACOES_VR.xlsx
```

13. Resumo Executivo Final

In [33]:

```
print("=== RESUMO EXECUTIVO FINAL ===\n")

# Calcular distribuição correta por estado
estados_resumo = planilha_final.groupby(
    planilha_final['Sindicato do Colaborador'].map(mapeamento_sindicato_estado)
).agg({
    'Matricula': 'count',
    'TOTAL': 'sum'
}).round(2)

print("DISTRIBUIÇÃO POR ESTADO:")
print(estados_resumo)
print()

print("RESUMO FINANCEIRO FINAL:")
print("=" * 40)
for estado, dados in estados_resumo.iterrows():
    print(f"{estado}:")
    print(f"  Colaboradores: {dados['Matricula']}")
    print(f"  Valor total: R$ {dados['TOTAL']:,.2f}")
print()

print("TOTAIS GERAIS:")
print(f"Total colaboradores: {estados_resumo['Matricula'].sum()}")
print(f"Total VR: R$ {estados_resumo['TOTAL'].sum():,.2f}")
print(f"Custo empresa (80%): R$ {estados_resumo['TOTAL'].sum() * 0.8:,.2f}")
print(f"Desconto colaborador (20%): R$ {estados_resumo['TOTAL'].sum() * 0.2:,.2f}")
print()

print("=" * 60)
print("PROCESSAMENTO CONCLUÍDO COM SUCESSO!")
print("=" * 60)
print("ARQUIVOS GERADOS:")
print("✓ VR_AUTOMATIZADO_MAIO_2025.xlsx - Planilha final para fornecedor")
print("✓ RELATORIO_VALIDACOES_VR.xlsx - Relatório de validações")
print("=" * 60)
```

```
=== RESUMO EXECUTIVO FINAL ===

DISTRIBUIÇÃO POR ESTADO:
                          Matricula     TOTAL
Sindicato do Colaborador
Paraná                          132  100765.0
Rio Grande do Sul              1124  813995.0
Rio de Janeiro                  100   71925.0
São Paulo                       414  334425.0

RESUMO FINANCEIRO FINAL:
========================================
Paraná:
  Colaboradores: 132.0
  Valor total: R$ 100,765.00
Rio Grande do Sul:
  Colaboradores: 1124.0
  Valor total: R$ 813,995.00
Rio de Janeiro:
  Colaboradores: 100.0
  Valor total: R$ 71,925.00
São Paulo:
  Colaboradores: 414.0
  Valor total: R$ 334,425.00

TOTAIS GERAIS:
Total colaboradores: 1770
Total VR: R$ 1,321,110.00
Custo empresa (80%): R$ 1,056,888.00
Desconto colaborador (20%): R$ 264,222.00

============================================================
PROCESSAMENTO CONCLUÍDO COM SUCESSO!
============================================================
ARQUIVOS GERADOS:
✓ VR_AUTOMATIZADO_MAIO_2025.xlsx - Planilha final para fornecedor
✓ RELATORIO_VALIDACOES_VR.xlsx - Relatório de validações
============================================================
```

14. Função de Automação Reutilizável

Função completa para uso mensal automatizado.

In [34]:

```python
def automatizar_vr_mensal():
    """
    Função principal para automatizar o cálculo mensal de VR/VA
    """
    print("=== AUTOMAÇÃO DA COMPRA DE VR/VA ===\n")

    # STEP 1: Carregar dados
    print("STEP 1: Carregando dados...")

    # Carregar todas as bases
    ativos = pd.read_excel('ATIVOS.xlsx', sheet_name='ATIVOS')
    admissoes = pd.read_excel('ADMISSAO-ABRIL.xlsx')
    afastamentos = pd.read_excel('AFASTAMENTOS.xlsx')
    aprendizes = pd.read_excel('APRENDIZ.xlsx')
    desligados = pd.read_excel('DESLIGADOS.xlsx', sheet_name='DESLIGADOS ')
    estagio = pd.read_excel('ESTAGIO.xlsx')
    exterior = pd.read_excel('EXTERIOR.xlsx')
    ferias = pd.read_excel('FERIAS.xlsx')
    dias_uteis = pd.read_excel('Base-dias-uteis.xlsx')
    sindicato_valor = pd.read_excel('Base-sindicato-x-valor.xlsx')

    # Limpar dados
    dias_uteis_clean = dias_uteis.iloc[1:].copy()
    dias_uteis_clean.columns = ['SINDICATO', 'DIAS_UTEIS']
    dias_uteis_clean = dias_uteis_clean.dropna()

    sindicato_valor_clean = sindicato_valor.dropna()
    desligados.columns = desligados.columns.str.strip()

    # STEP 2: Configurar mapeamentos
    mapeamento_sindicato_estado = {
        'SINDPPD RS - SINDICATO DOS TRAB. EM PROC. DE DADOS RIO GRANDE DO SUL': 'Rio Grande do Sul',
        'SINDPD SP - SIND.TRAB.EM PROC DADOS E EMPR.EMPRESAS PROC DADOS ESTADO DE SP.': 'São Paulo',
        'SINDPD RJ - SINDICATO PROFISSIONAIS DE PROC DADOS DO RIO DE JANEIRO': 'Rio de Janeiro',
        'SITEPD PR - SIND DOS TRAB EM EMPR PRIVADAS DE PROC DE DADOS DE CURITIBA E REGIAO METROPOLITANA': 'Paraná'
    }

    valores_estado = {'Paraná': 35.0, 'Rio de Janeiro': 35.0, 'Rio Grande do Sul': 35.0, 'São Paulo': 37.5}
    dias_uteis_dict = dict(zip(dias_uteis_clean['SINDICATO'], dias_uteis_clean['DIAS_UTEIS'].astype(int)))

    # STEP 3: Criar exclusões
    matriculas_afastados = set(afastamentos['MATRICULA'].dropna())
    matriculas_aprendizes = set(aprendizes['MATRICULA'].dropna())
```

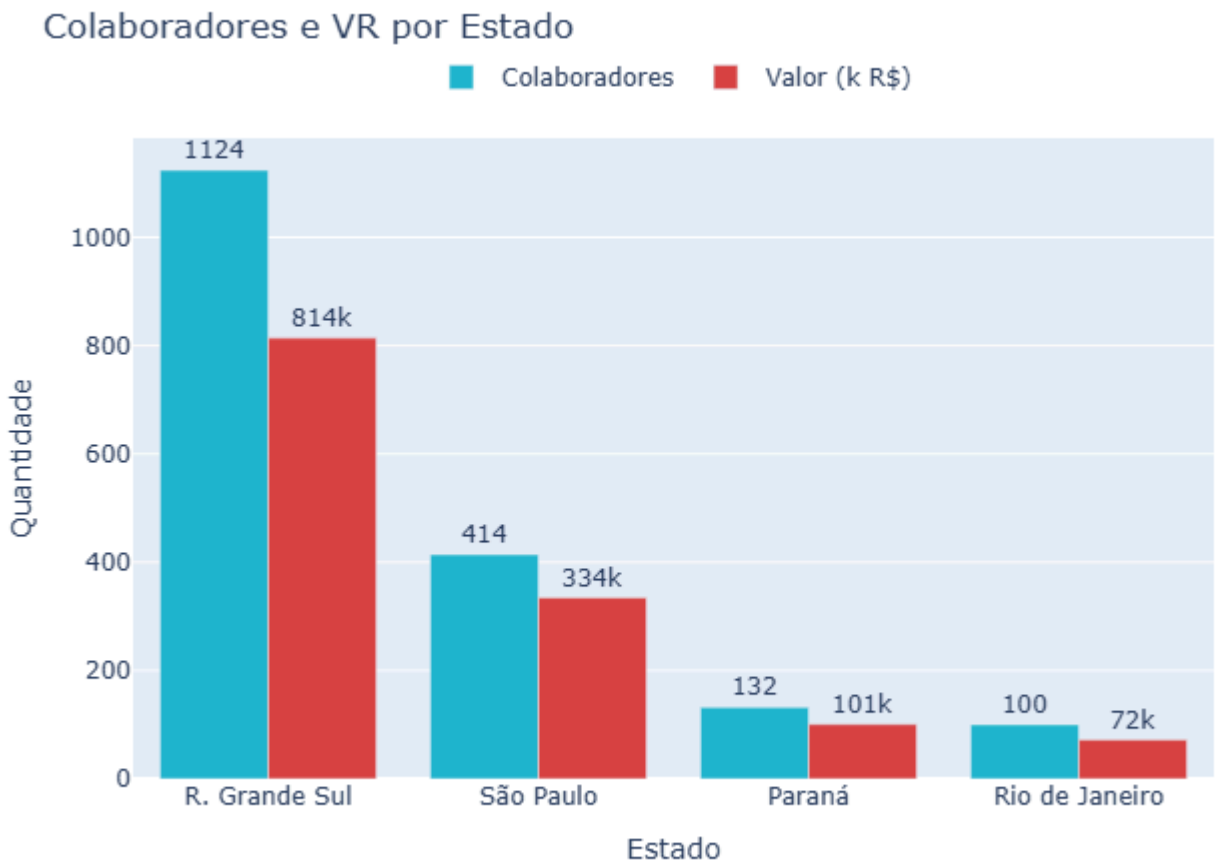
```
matriculas_estagiarios = set(estagio['MATRICULA'].dropna())
matriculas_exterior = set(exterior['Cadastro'].dropna())

# Processar desligados
desligados_ate_dia_15 = []
desligados_apos_dia_15 = []

for _, row in desligados.iterrows():
    data_demissao = pd.to_datetime(row['DATA DEMISSÃO'])
    comunicado = row['COMUNICADO DE DESLIGAMENTO']
    matricula = row['MATRICULA']

    if data_demissao.day <= 15 and comunicado == 'OK':
        desligados_ate_dia_15.append(matricula)
    else:
        desligados_apos_dia_15.append(matricula)

exclusoes_completas = matriculas_afastados | matriculas_aprendizes | matriculas_estagiarios | matriculas_exterior | set(de

# STEP 4: Consolidar base
base_consolidada = ativos[~ativos['MATRICULA'].isin(exclusoes_completas)].copy()

# Adicionar informações
base_consolidada['ESTADO'] = base_consolidada['Sindicato'].map(mapeamento_sindicato_estado)
base_consolidada['VALOR_DIARIO_VR'] = base_consolidada['ESTADO'].map(valores_estado)
base_consolidada['DIAS_UTEIS_SINDICATO'] = base_consolidada['Sindicato'].map(dias_uteis_dict)

# Férias e desligamentos
ferias_dict = dict(zip(ferias['MATRICULA'], ferias['DIAS DE FÉRIAS']))
base_consolidada['DIAS_FERIAS'] = base_consolidada['MATRICULA'].map(ferias_dict).fillna(0)

desligamento_dict = {}
for _, row in desligados.iterrows():
    if row['MATRICULA'] in desligados_apos_dia_15:
        data_demissao = pd.to_datetime(row['DATA DEMISSÃO'])
        desligamento_dict[row['MATRICULA']] = data_demissao.day

base_consolidada['DIA_DESLIGAMENTO'] = base_consolidada['MATRICULA'].map(desligamento_dict).fillna(0)

# STEP 5: Calcular dias VR
def calcular_dias_vr(row):
    dias_base = row['DIAS_UTEIS_SINDICATO']
    dias_ferias = row['DIAS_FERIAS']
    dia_desligamento = row['DIA_DESLIGAMENTO']

    if dia_desligamento > 0:
        dias_efetivos = dia_desligamento - 15
        dias_efetivos = max(0, dias_efetivos)
    else:
        dias_efetivos = dias_base - dias_ferias
        dias_efetivos = max(0, dias_efetivos)

    return dias_efetivos

base_consolidada['DIAS_VR'] = base_consolidada.apply(calcular_dias_vr, axis=1)

# STEP 6: Calcular valores
base_consolidada['TOTAL_VR'] = base_consolidada['DIAS_VR'] * base_consolidada['VALOR_DIARIO_VR']
base_consolidada['CUSTO_EMPRESA'] = base_consolidada['TOTAL_VR'] * 0.8
base_consolidada['DESCONTO_COLABORADOR'] = base_consolidada['TOTAL_VR'] * 0.2

# STEP 7: Gerar planilha final
colaboradores_vr = base_consolidada[base_consolidada['TOTAL_VR'] > 0].copy()

planilha_final = pd.DataFrame()
planilha_final['Matricula'] = colaboradores_vr['MATRICULA']
planilha_final['Admissão'] = ''
planilha_final['Sindicato do Colaborador'] = colaboradores_vr['Sindicato']
planilha_final['Competência'] = '2025-05-01'
planilha_final['Dias'] = colaboradores_vr['DIAS_VR']
planilha_final['VALOR DIÁRIO VR'] = colaboradores_vr['VALOR_DIARIO_VR']
planilha_final['TOTAL'] = colaboradores_vr['TOTAL_VR']
planilha_final['Custo empresa'] = colaboradores_vr['CUSTO_EMPRESA']
planilha_final['Desconto profissional'] = colaboradores_vr['DESCONTO_COLABORADOR']
planilha_final['OBS GERAL'] = ''

# Adicionar observações
for idx, row in planilha_final.iterrows():
    matricula = row['Matricula']
    obs = []

    if matricula in ferias_dict:
        dias_ferias = ferias_dict[matricula]
        obs.append(f"FÉRIAS: {dias_ferias} dias")

    if matricula in desligamento_dict:
        dia_desl = desligamento_dict[matricula]
```

```
            obs.append(f"DESLIGAMENTO: dia {dia_desl}")

        planilha_final.at[idx, 'OBS GERAL'] = '; '.join(obs)

    planilha_final = planilha_final.sort_values('Matricula').reset_index(drop=True)

    # Salvar arquivos
    data_atual = datetime.now().strftime('%Y%m%d')
    nome_arquivo = f'VR_AUTOMATIZADO_{data_atual}.xlsx'
    planilha_final.to_excel(nome_arquivo, index=False)

    print(f"✓ Processamento concluído!")
    print(f"✓ Arquivo gerado: {nome_arquivo}")
    print(f"✓ Total de colaboradores: {len(planilha_final)}")
    print(f"✓ Valor total VR: R$ {planilha_final['TOTAL'].sum():,.2f}")

    return planilha_final

# Para executar a automação, descomente a linha abaixo:
# resultado = automatizar_vr_mensal()

print("Função de automação criada com sucesso!")
print("Para usar mensalmente, execute: resultado = automatizar_vr_mensal()")
```

```
Função de automação criada com sucesso!
Para usar mensalmente, execute: resultado = automatizar_vr_mensal()
```

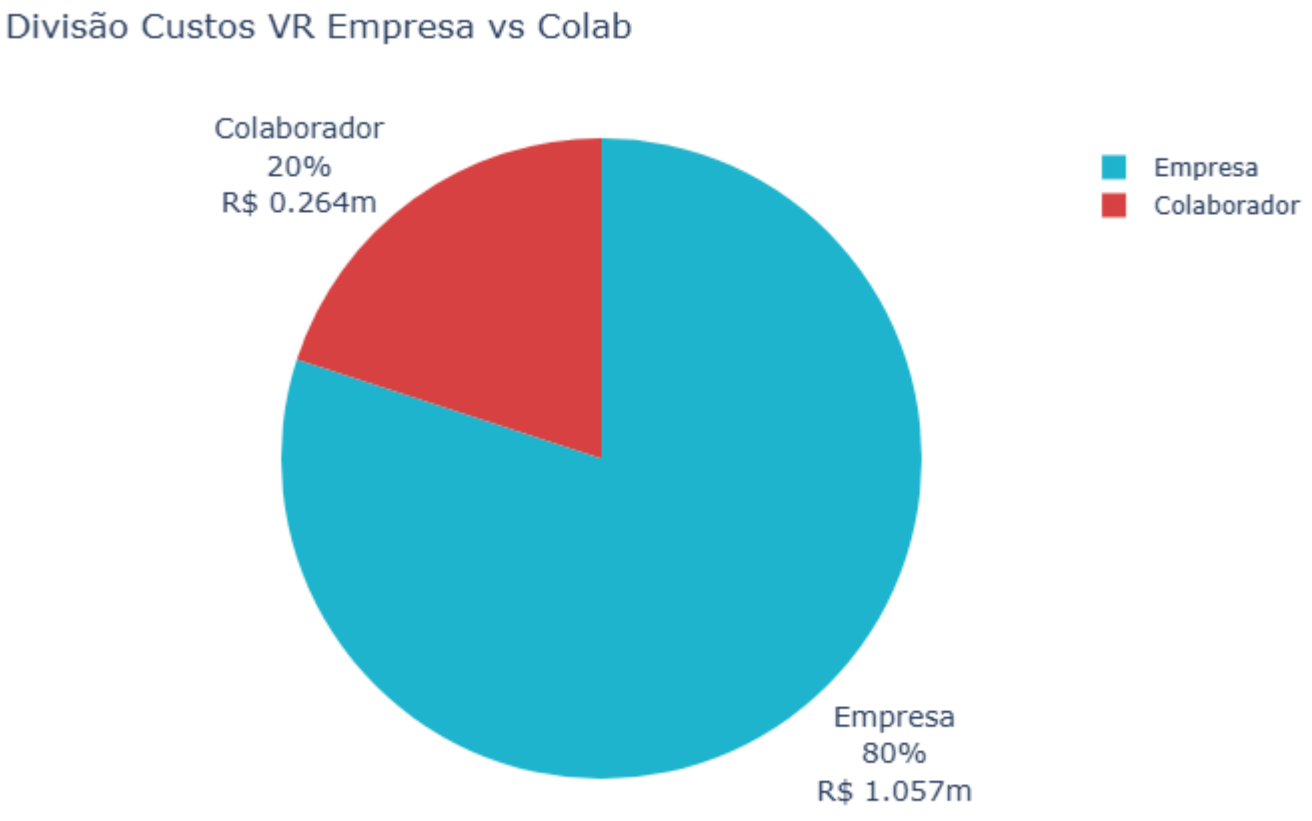
In [35]:

```
import plotly.graph_objects as go

# Data with abbreviated state names to meet 15-character limit
estados = ["R. Grande Sul", "São Paulo", "Paraná", "Rio de Janeiro"]
colaboradores = [1124, 414, 132, 100]
valores = [813995, 334425, 100765, 71925]

# Convert valores to thousands for better scale comparison
valores_k = [v/1000 for v in valores]

fig = go.Figure()

# Add colaboradores bars
fig.add_trace(go.Bar(
    x=estados,
    y=colaboradores,
    name='Colaboradores',
    marker_color='#1FB8CD',
    text=colaboradores,
    textposition='outside',
    cliponaxis=False
))

# Add valores bars
fig.add_trace(go.Bar(
    x=estados,
    y=valores_k,
    name='Valor (k R$)',
    marker_color='#DB4545',
    text=[f'{v:.0f}k' for v in valores_k],
    textposition='outside',
    cliponaxis=False
))

fig.update_layout(
    title='Colaboradores e VR por Estado',
    barmode='group',
    legend=dict(orientation='h', yanchor='bottom', y=1.05, xanchor='center', x=0.5)
)

fig.update_yaxes(title='Quantidade')
fig.update_xaxes(title='Estado')

fig.show()
fig.write_image('colaboradores_vr_estados.png')
```



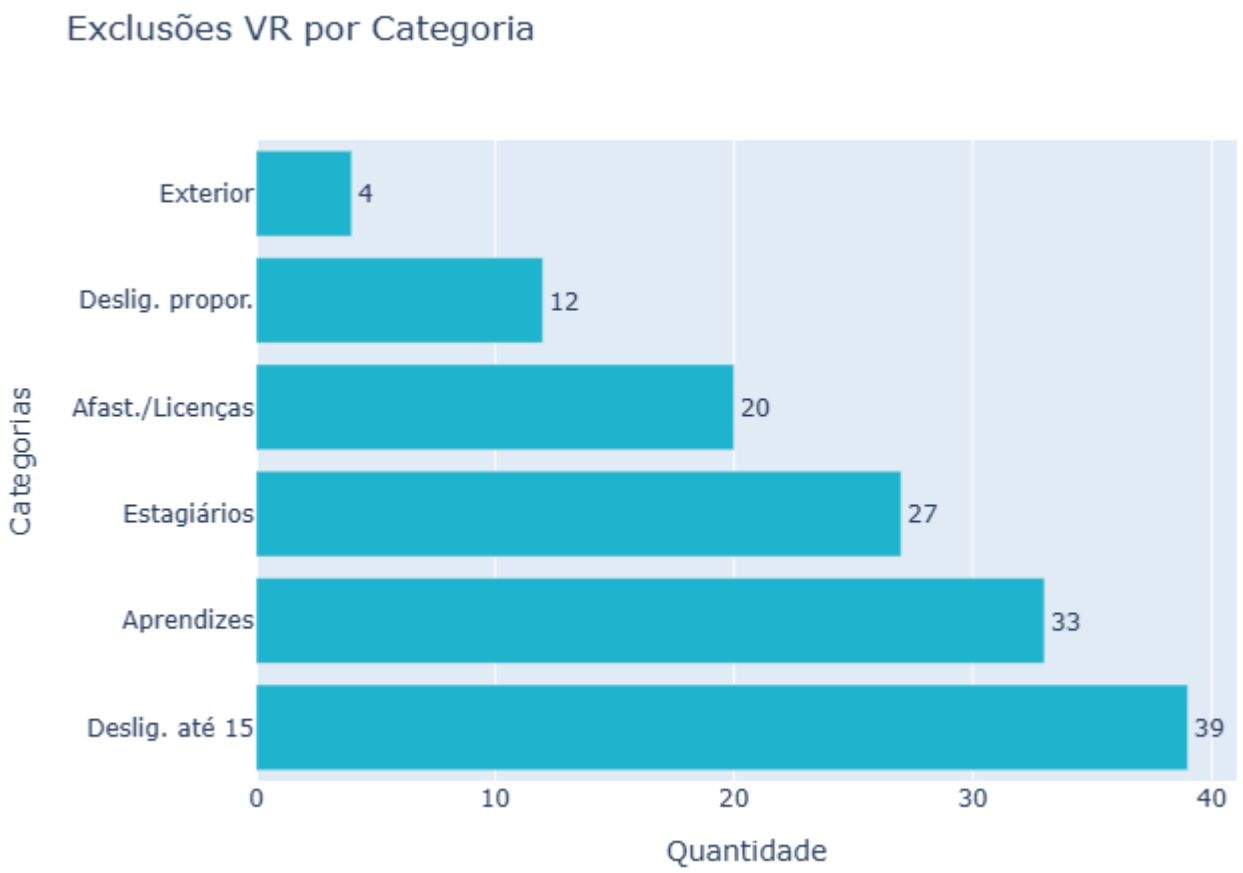
Colaboradores e VR por Estado

In [36]:
```python
import plotly.express as px
import plotly.graph_objects as go

# Data for the pie chart
categories = ["Empresa", "Colaborador"]
valores = [1056888, 264222]
percentuais = [80, 20]

# Format values more precisely
valor_empresa = "1.057m"
valor_colab = "0.264m"

# Create custom hover text with full information
hover_text = [
    f"Empresa<br>80%<br>R$ {valor_empresa}",
    f"Colaborador<br>20%<br>R$ {valor_colab}"
]

# Create pie chart with labels outside to avoid cramping
fig = go.Figure(data=[go.Pie(
    labels=categories,
    values=valores,
    text=[f"{perc}%<br>R$ {val}" for perc, val in zip(percentuais, [valor_empresa, valor_colab])],
    textinfo='label+text',
    textposition='outside',
    hovertext=hover_text,
    hoverinfo='text',
    marker=dict(colors=['#1FB8CD', '#DB4545'])
)])

# Update layout
fig.update_layout(
    title="Divisão Custos VR Empresa vs Colab",
    uniformtext_minsize=14,
    uniformtext_mode='hide'
)


fig.show()
# Save as PNG

fig.write_image("vr_costs_division.png")
```



Divisão Custos VR Empresa vs Colab

In [37]:
```python
import plotly.graph_objects as go
import plotly.io as pio

# Data from the provided JSON
categorias = ["Desligados até dia 15", "Aprendizes", "Estagiários", "Afastados/Licenças", "Desligados proporcional", "Exterior
quantidades = [39, 33, 27, 20, 12, 4]

# Abbreviate category names to fit 15-character limit
categorias_abrev = ["Deslig. até 15", "Aprendizes", "Estagiários", "Afast./Licenças", "Deslig. propor.", "Exterior"]

# Create horizontal bar chart
fig = go.Figure(go.Bar(
    x=quantidades,
    y=categorias_abrev,
    orientation='h',
    marker_color='#1FB8CD',  # Using the first brand color
    text=quantidades,
    textposition='outside',
    cliponaxis=False
))

# Update layout
fig.update_layout(
    title="Exclusões VR por Categoria",
    xaxis_title="Quantidade",
    yaxis_title="Categorias"
)

# Update axes
fig.update_xaxes(showgrid=True)
fig.update_yaxes(showgrid=False)

fig.show()
# Save as PNG
fig.write_image("exclusoes_vr_categorias.png")
```



Exclusões VR por Categoria

Conclusão

Este notebook automatiza completamente o processo de cálculo e compra de Vale Refeição, incluindo:

✅ **Consolidação de bases de dados múltiplas**
✅ **Aplicação de regras de negócio e exclusões**
✅ **Cálculo automático de dias e valores**
✅ **Geração de planilha para fornecedor**
✅ **Relatório completo de validações**
✅ **Função reutilizável para uso mensal**

Benefícios Alcançados:

- **Eliminação do processo manual** (economia de horas de trabalho)
- **Redução significativa de erros** através de cálculos automatizados
- **Padronização** do processo com regras consistentes
- **Rastreabilidade completa** de todos os cálculos
- **Conformidade** com acordos coletivos e regulamentações

Para Uso Mensal:

1. Atualize os arquivos Excel com dados do mês corrente
2. Execute a função `automatizar_vr_mensal()`
3. Revise os resultados e validações
4. Envie a planilha final para o fornecedor

Data de Criação: 18/08/2025
Versão: 1.0
Competência Processada: Maio/2025