
Conix
BARBEARIA

Conix
BARBEARIA

QUEM SOMOS

Somos uma barbearia situada no Recreio dos Bandeirantes desde setembro de 2024.

Funcionamos de Segunda á Sábado, das 10h ás 20h.

ONDE ESTAMOS LOCALIZADOS

Barra World, Sala 107C - Setor Japão

QUAIS DADOS USAMOS NESTE PROJETO?

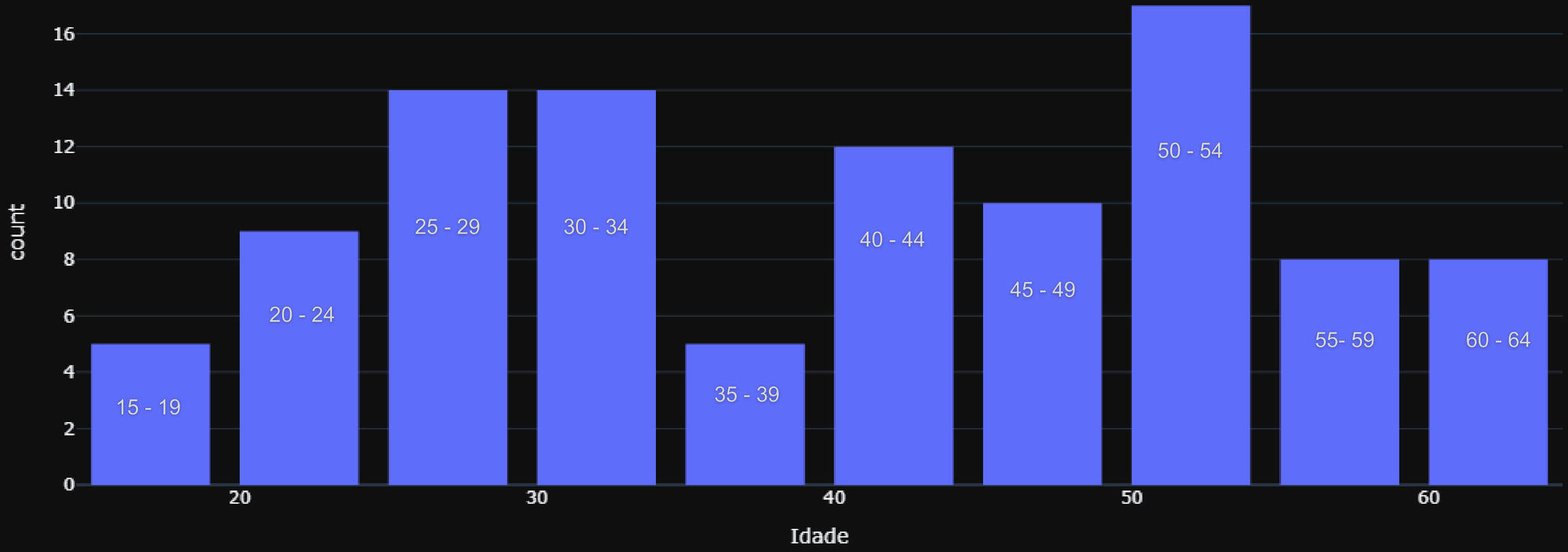
Neste projeto, utilizamos como base de dados um período de três meses e meio, compreendendo os meses de **janeiro, fevereiro, março e parte de abril**.

Este projeto está dividido em cinco grupos de análise:
Salão, Pagamento, Faturamento, Agendamento e Profissional

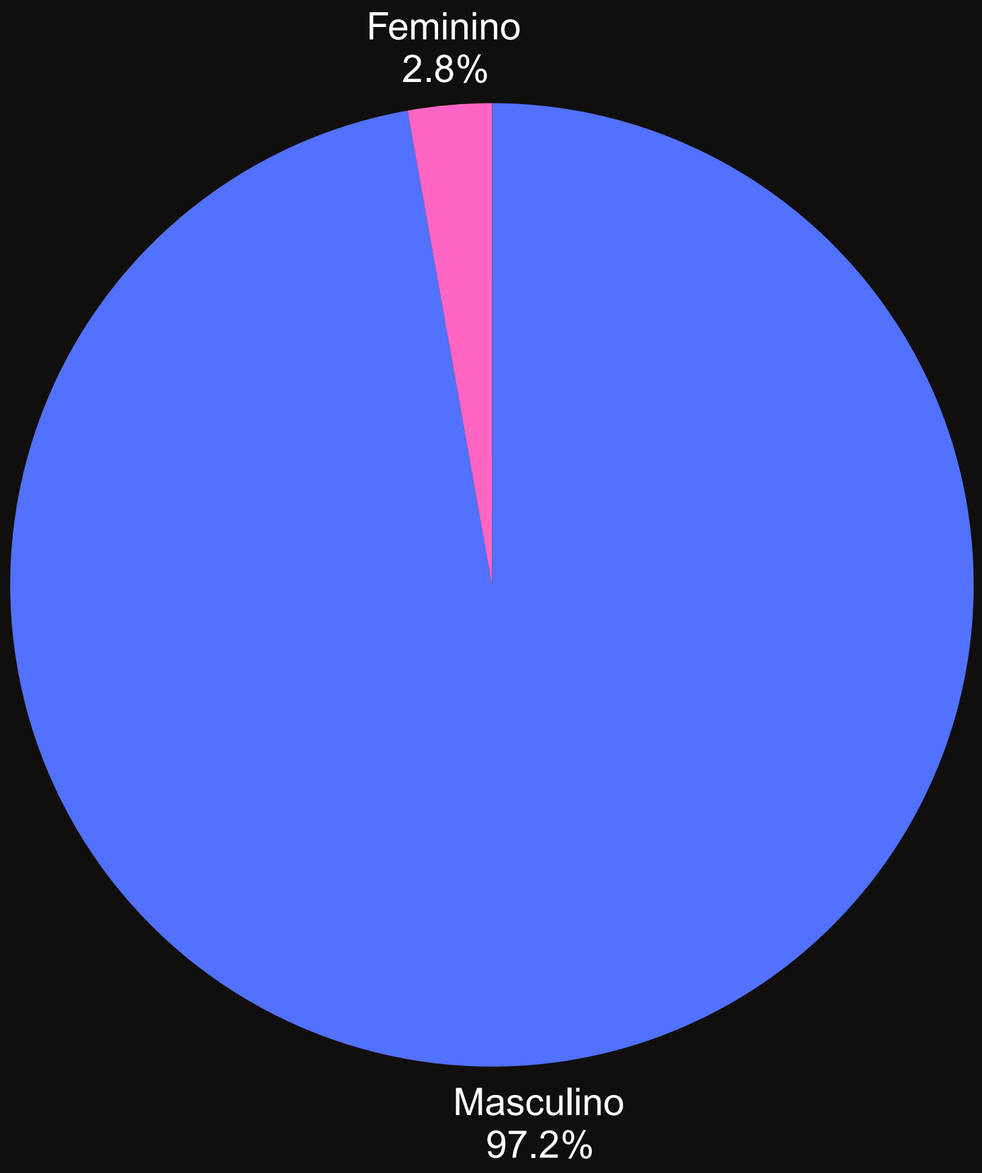
Conix
BARBEARIA

SALÃO

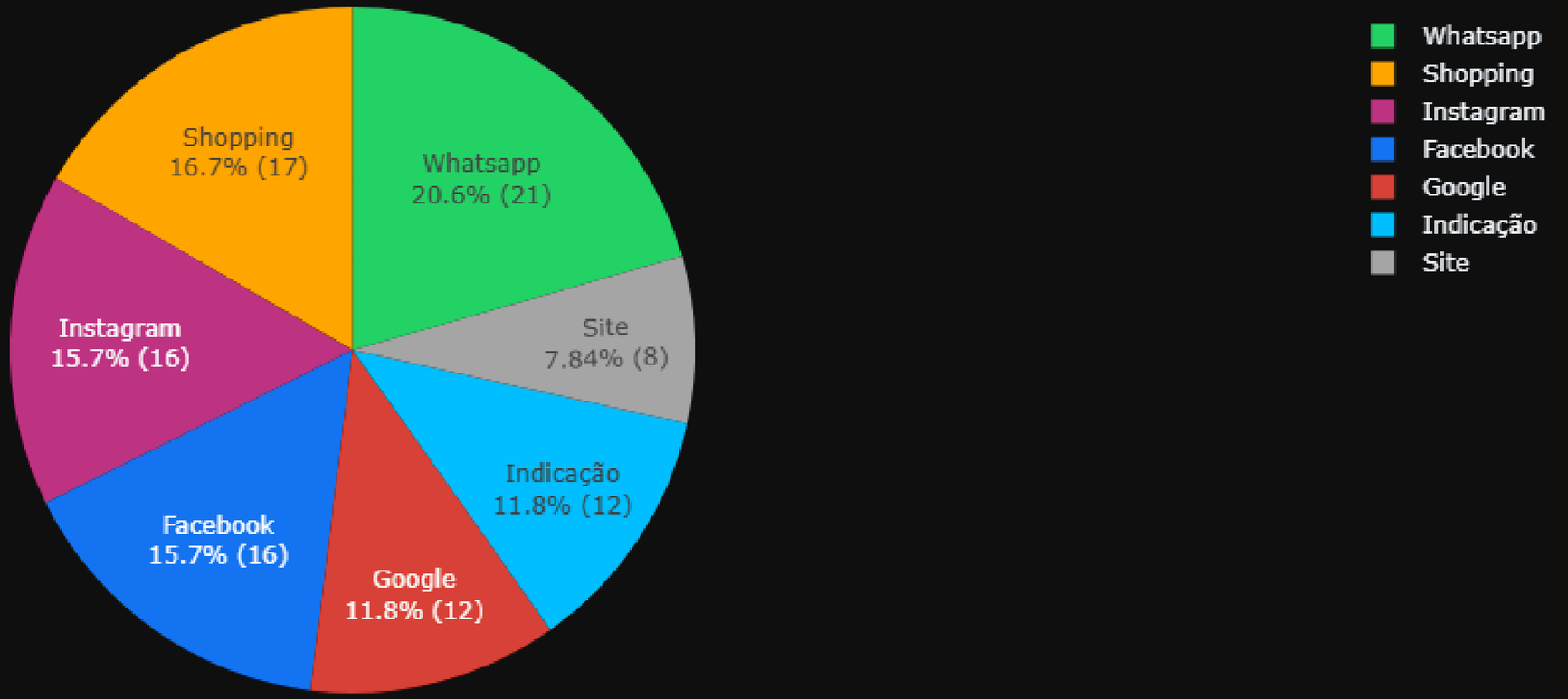
Distribuição de Idade



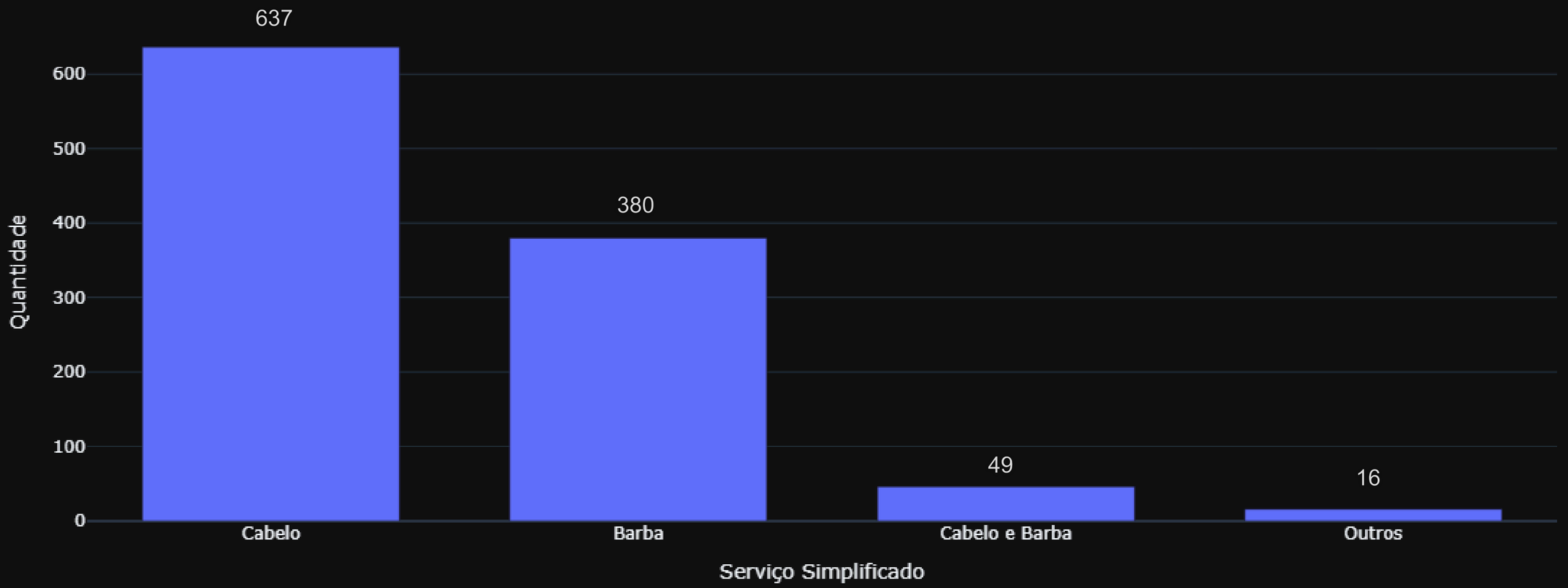
Distribuição de Atendimento por Gênero



Origem do Primeiro Contato

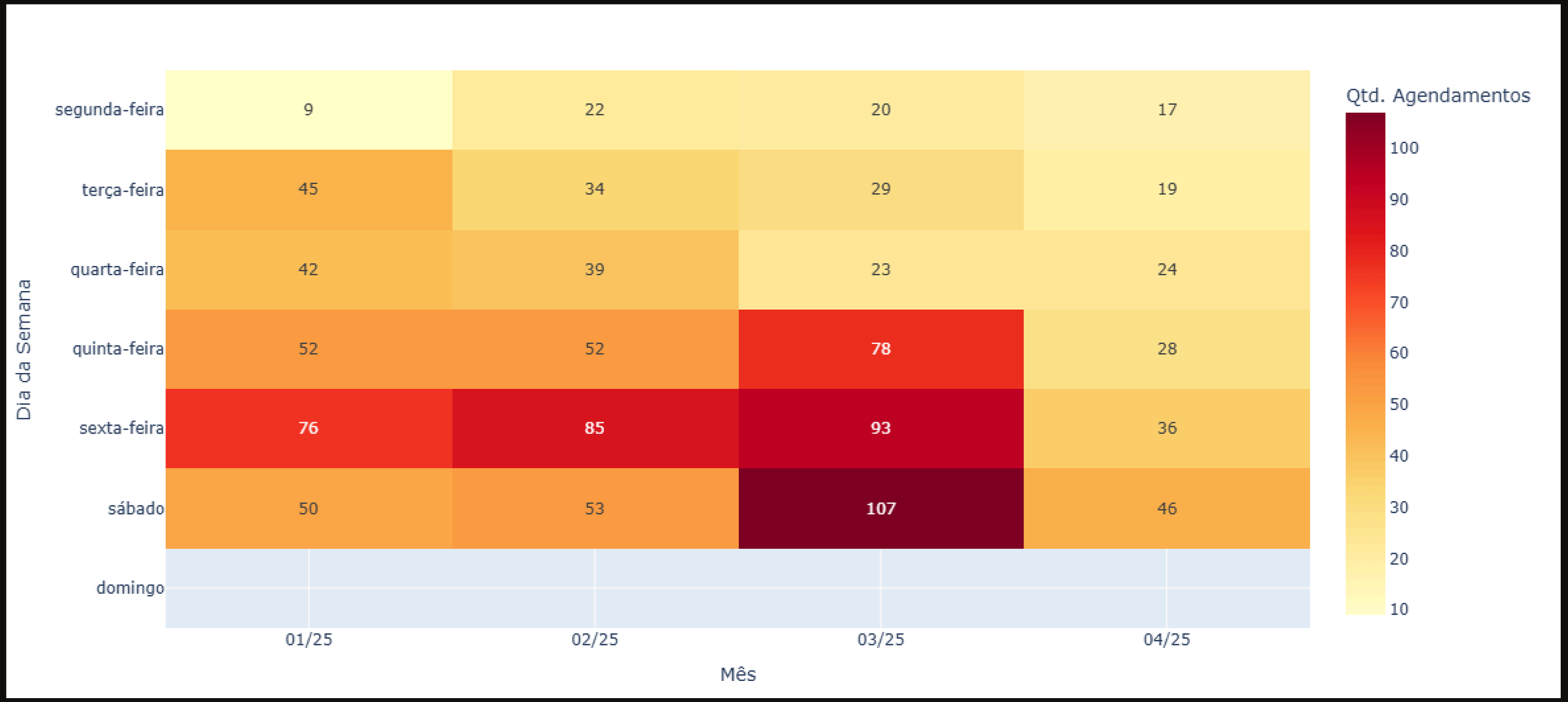


Serviços Mais Realizados

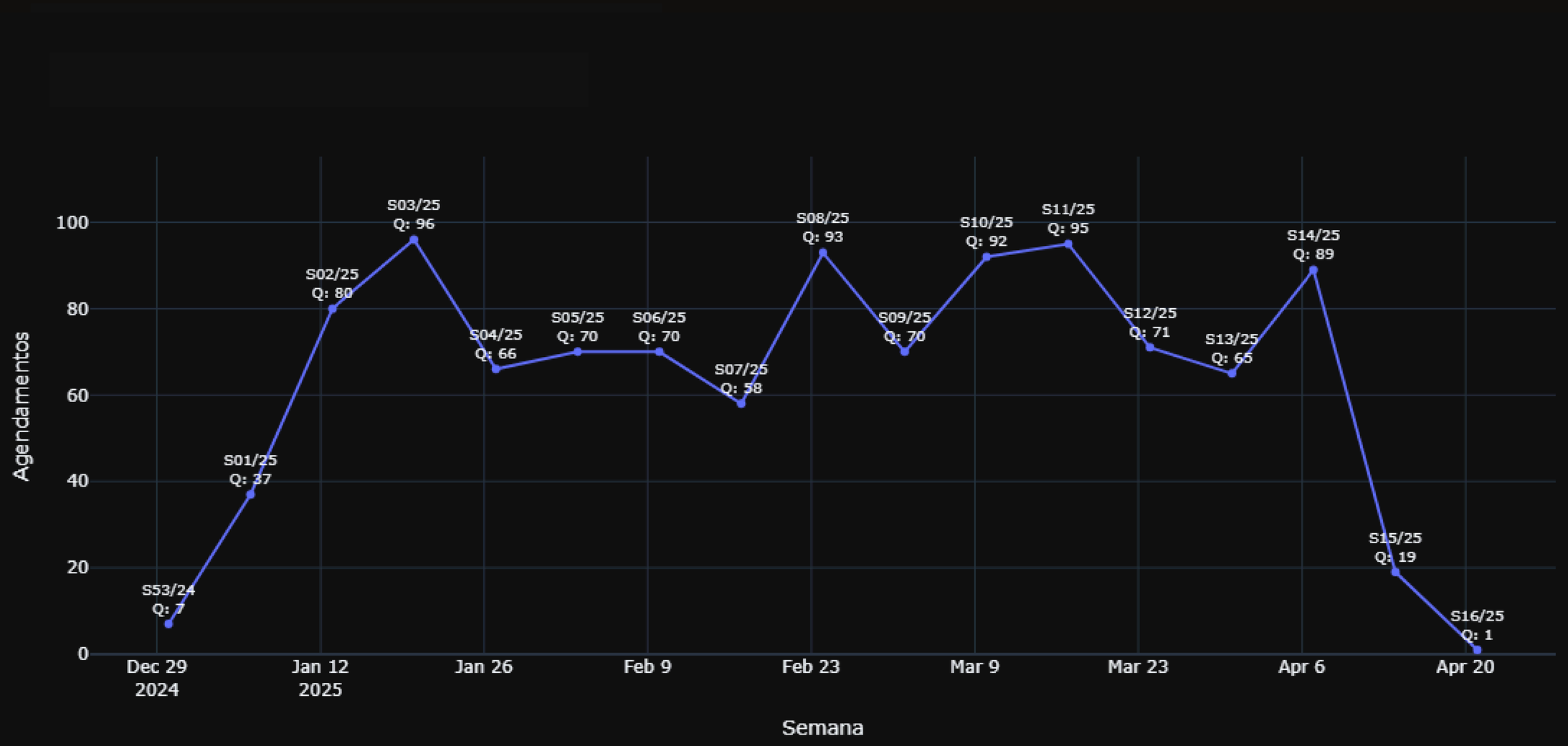


AGENDAMENTOS

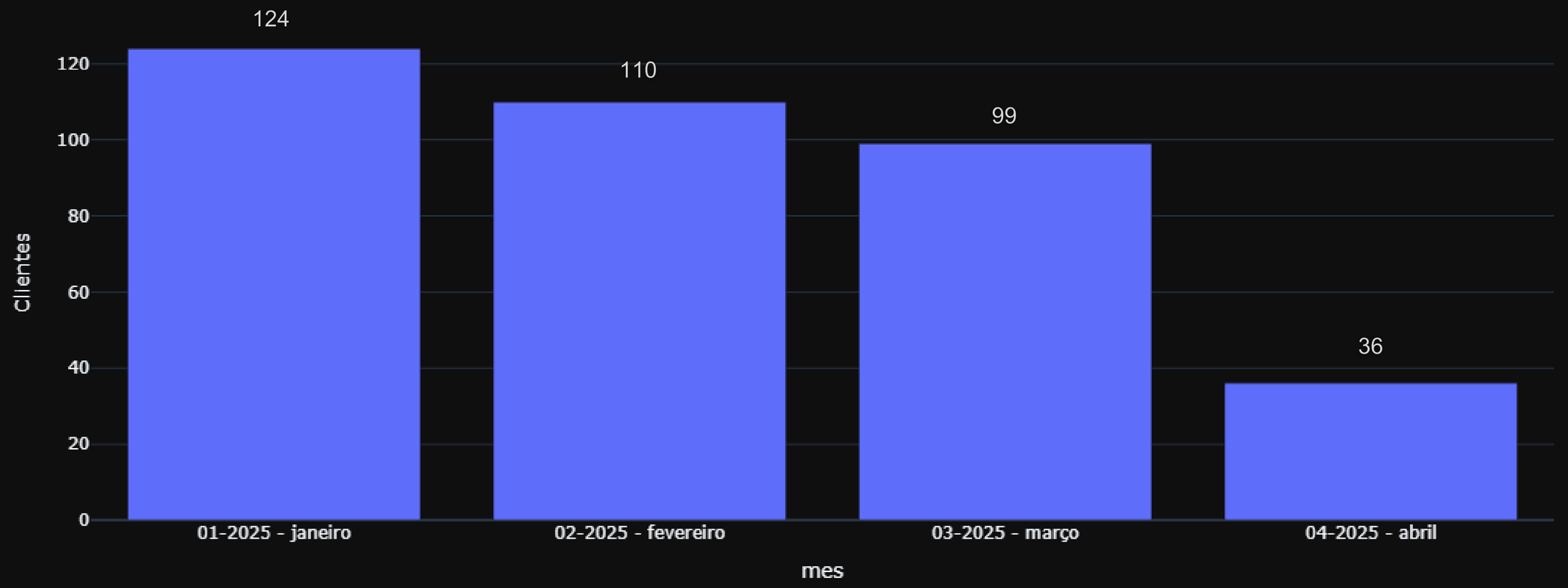
Mapa de Calor: Agendamentos por Dia da Semana e Mês



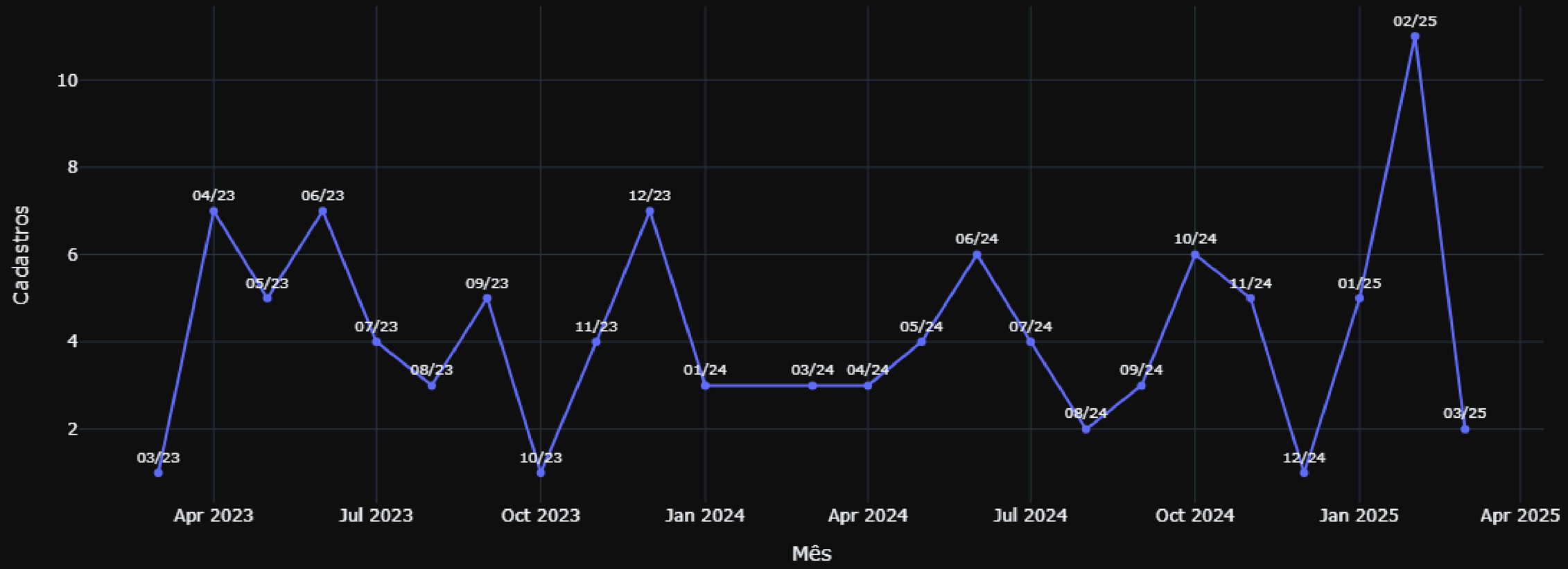
Evolução de Agendamentos por Semana



Clientes Únicos por Mês

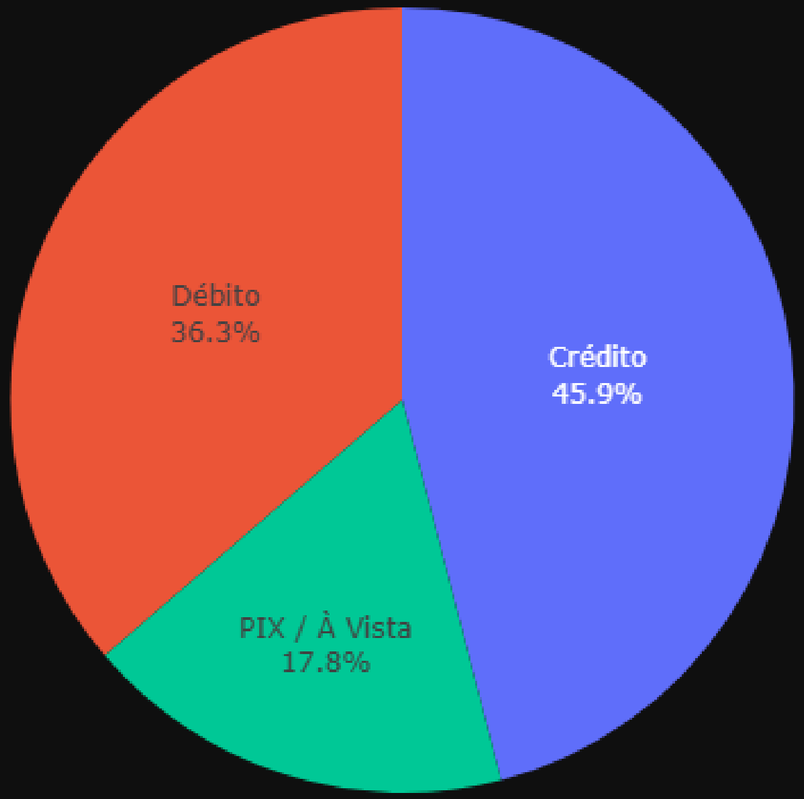


Evolução de Cadastros por Mês

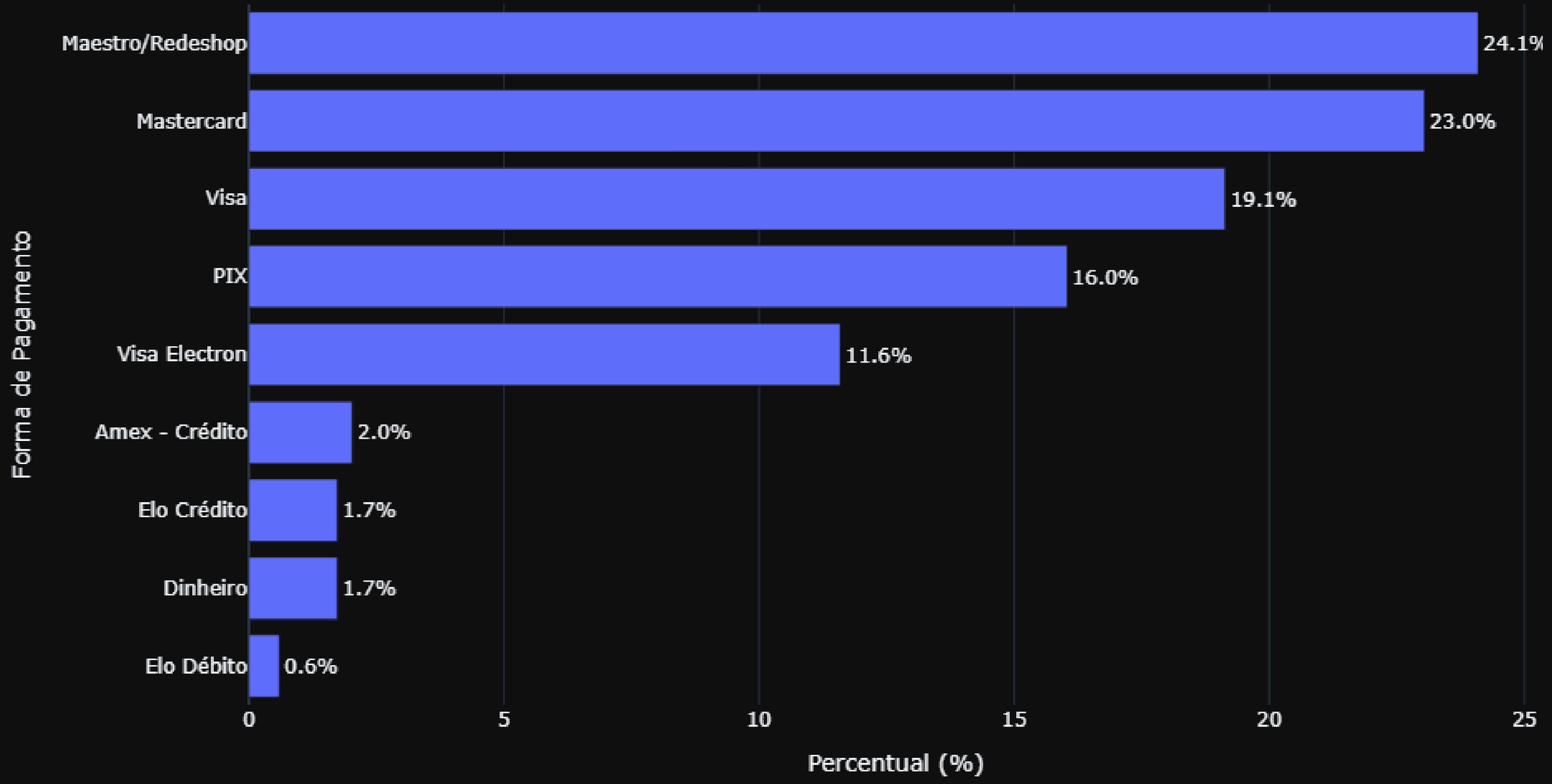


PAGAMENTO

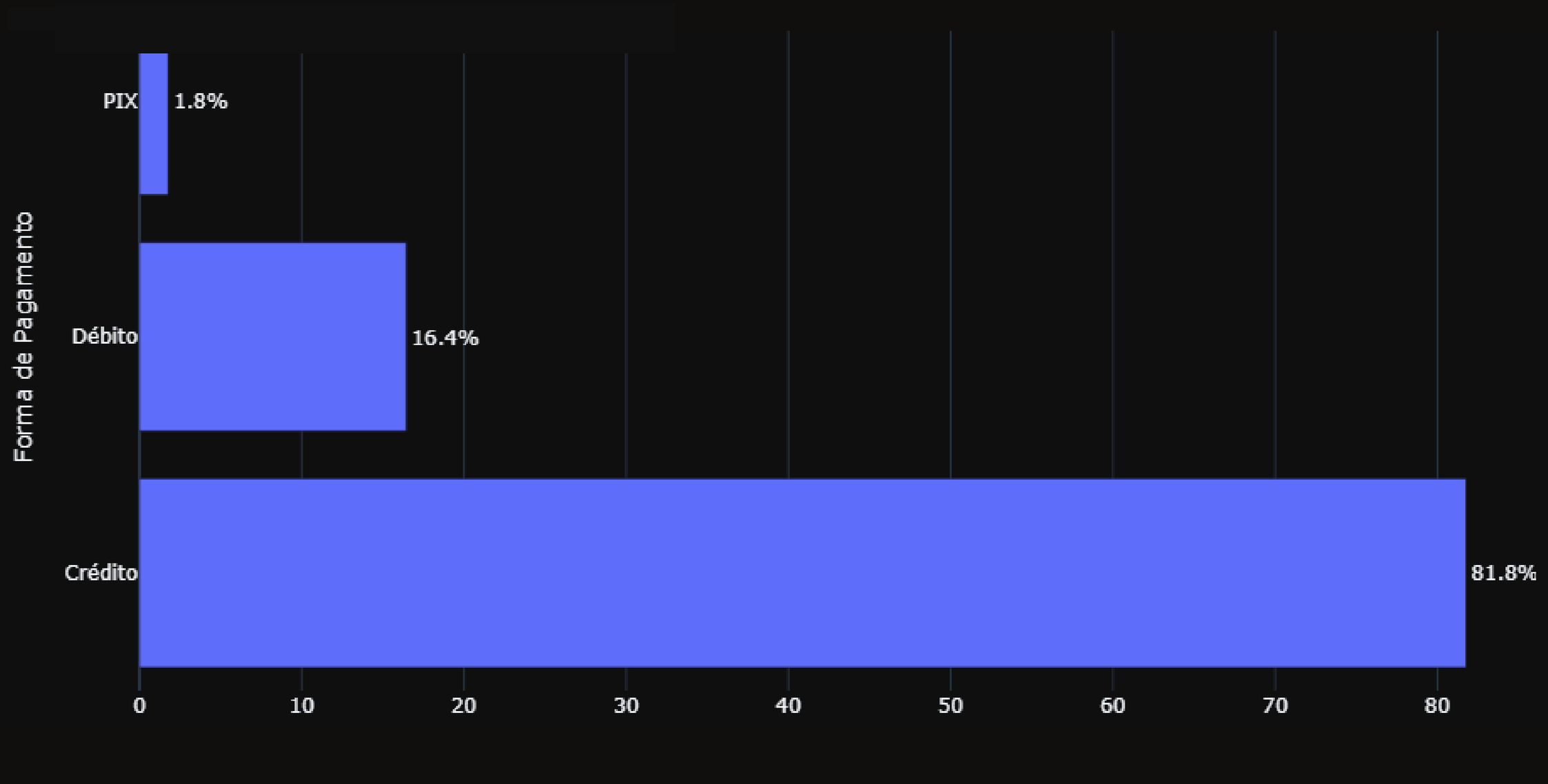
Distribuição por Tipo de Pagamento



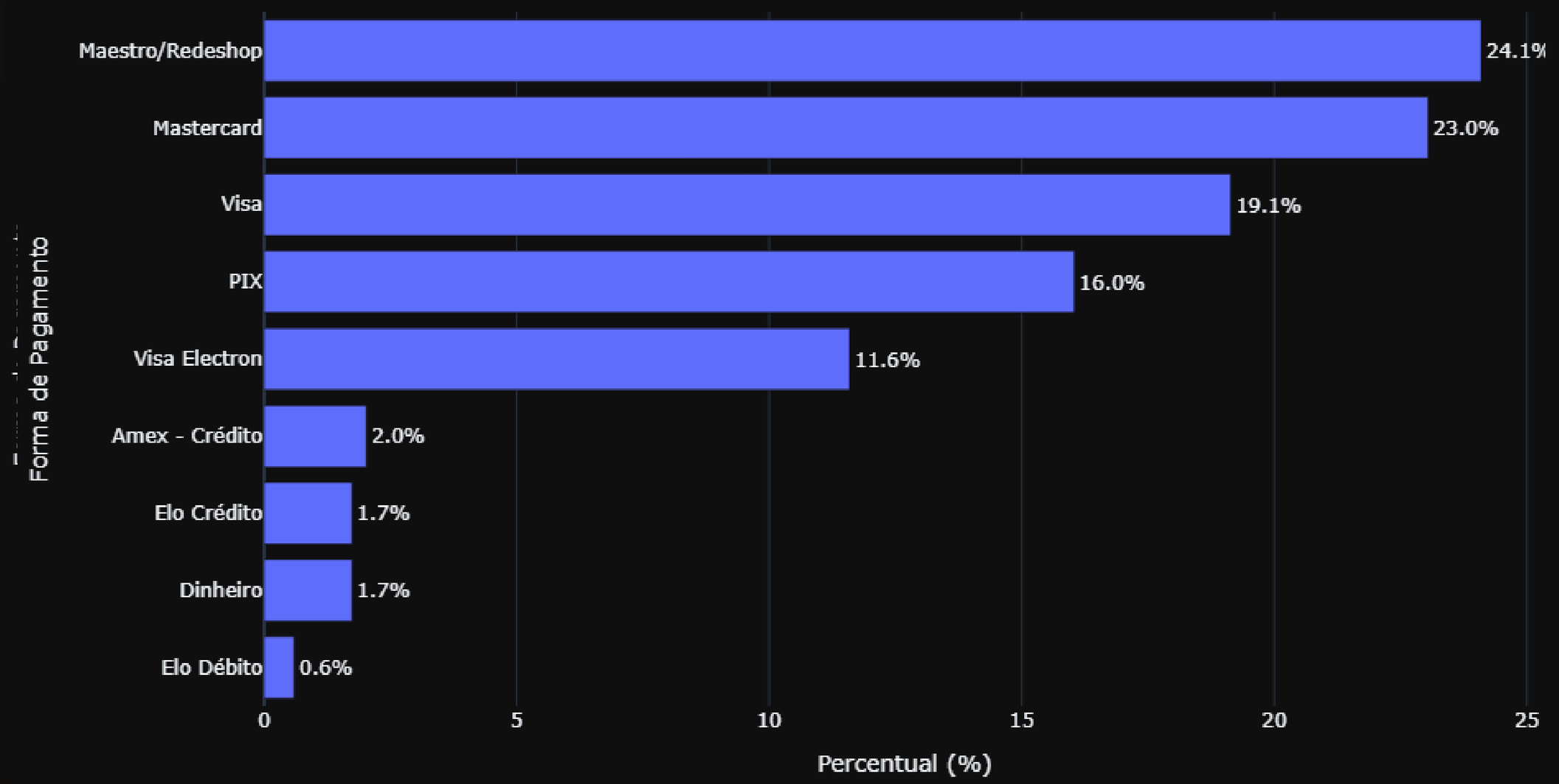
Total de Cada Forma de Pagamento



Total de Taxas Pagas por Forma de Pagamento

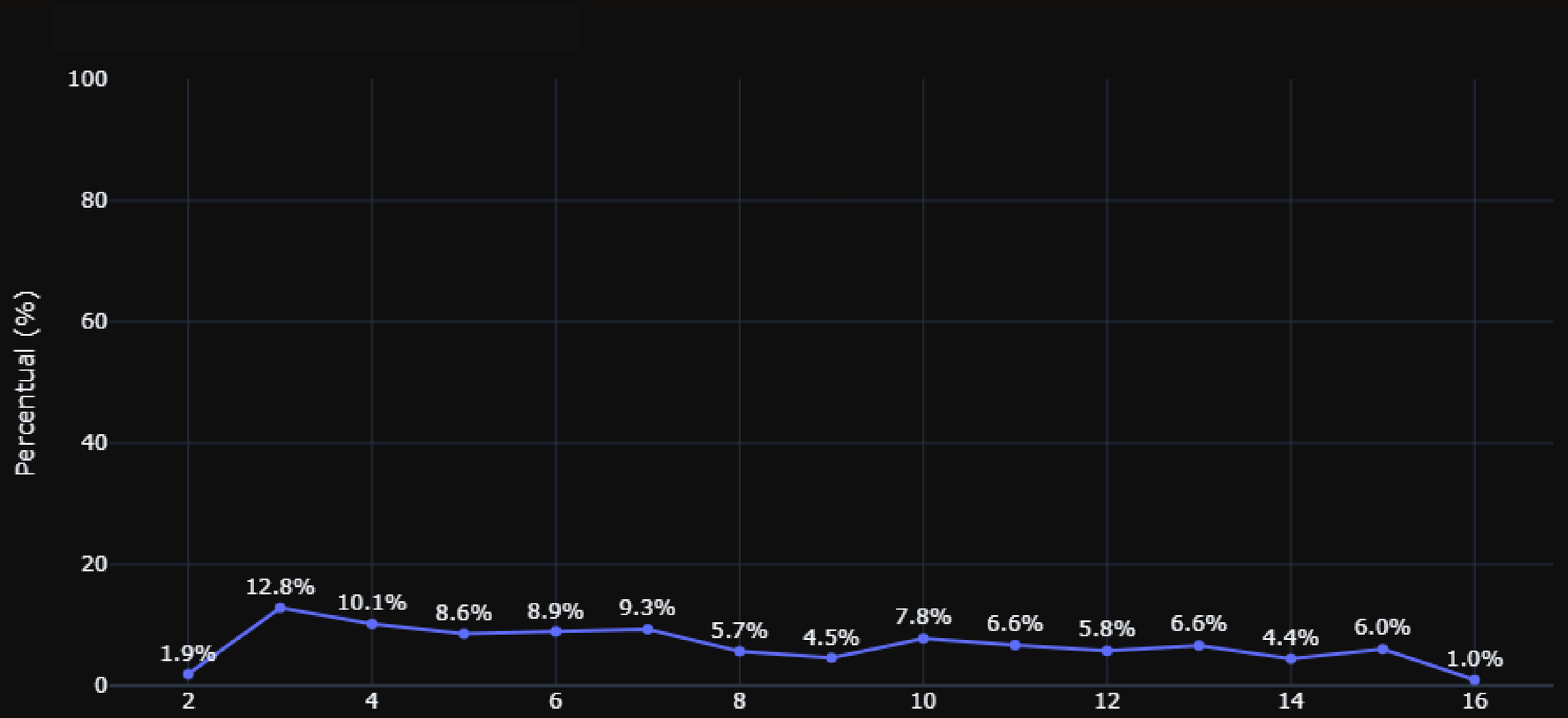


Total de Taxas Pagas por Bandeira

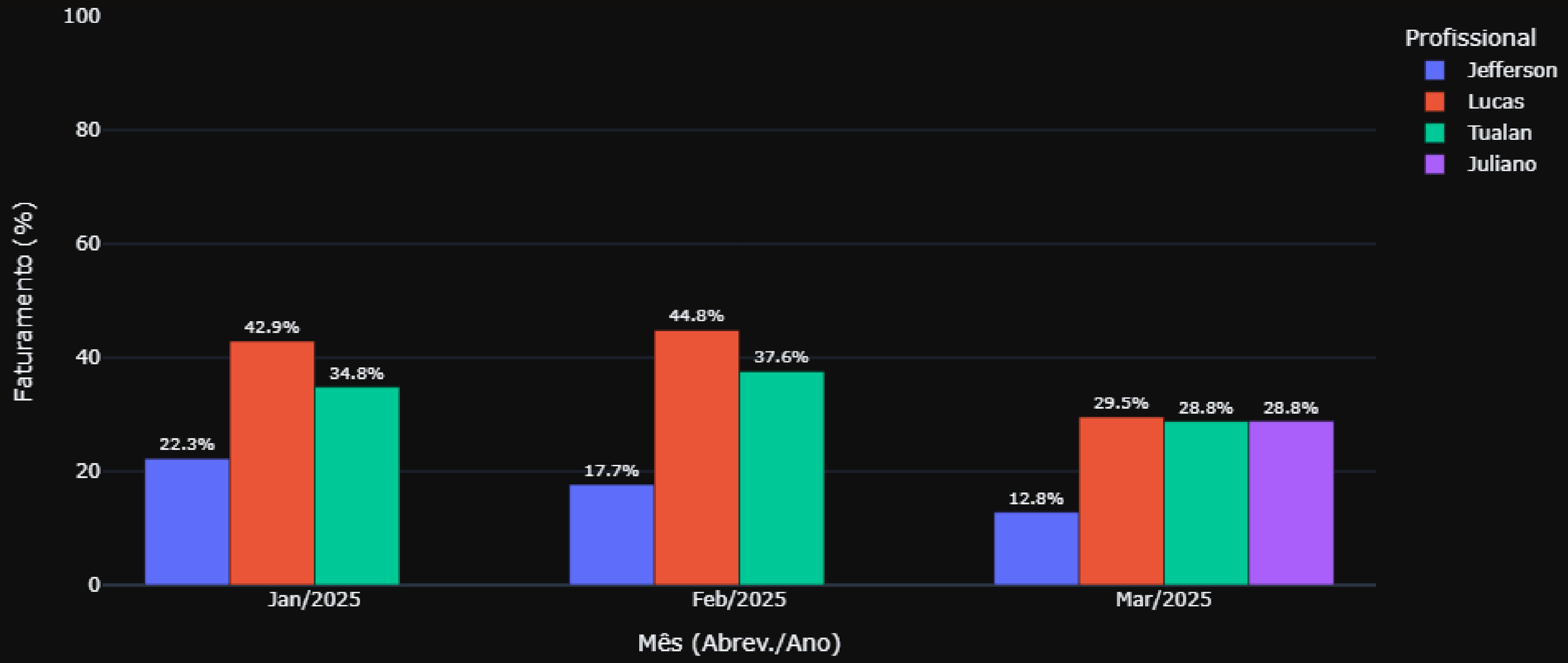


FATURAMENTO

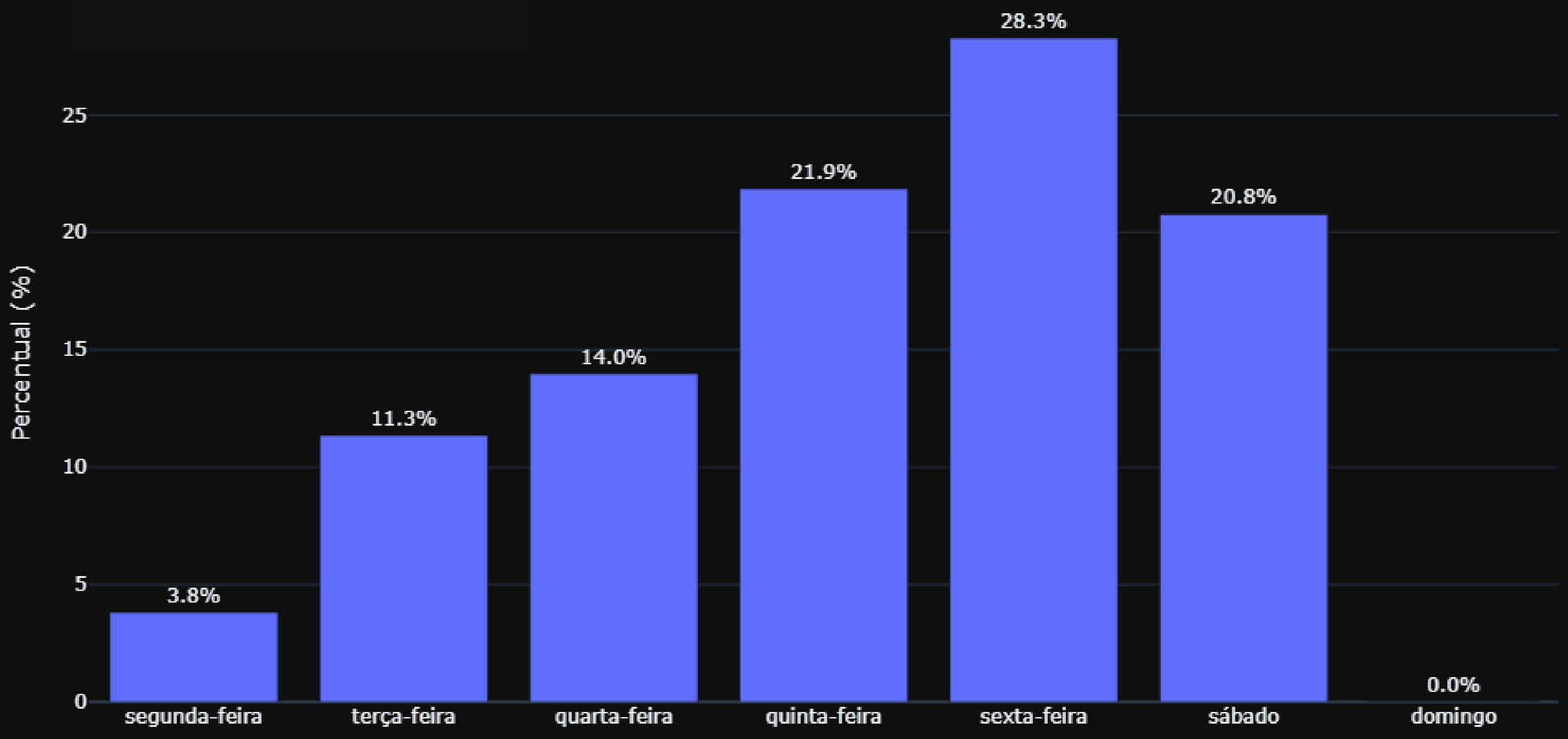
Faturamento por Semana



Faturamento por Profissional por Mês (Exceto Abril)

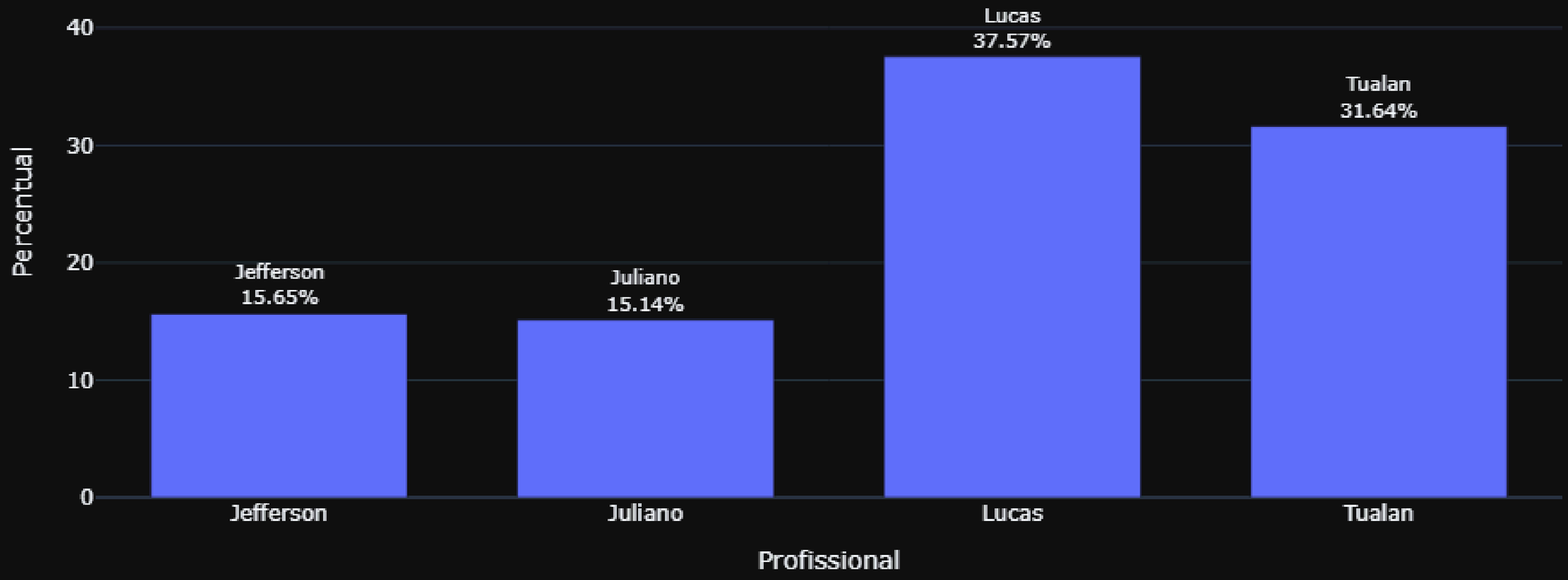


Faturamento por Dia da Semana

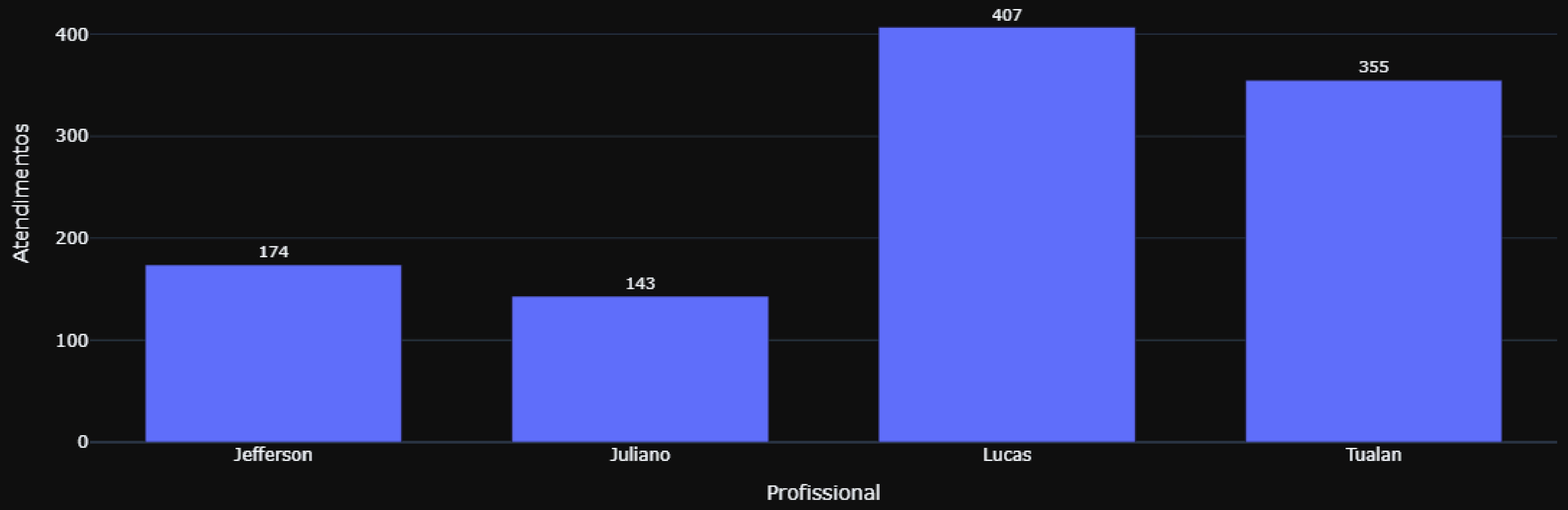


PROFISSIONAL

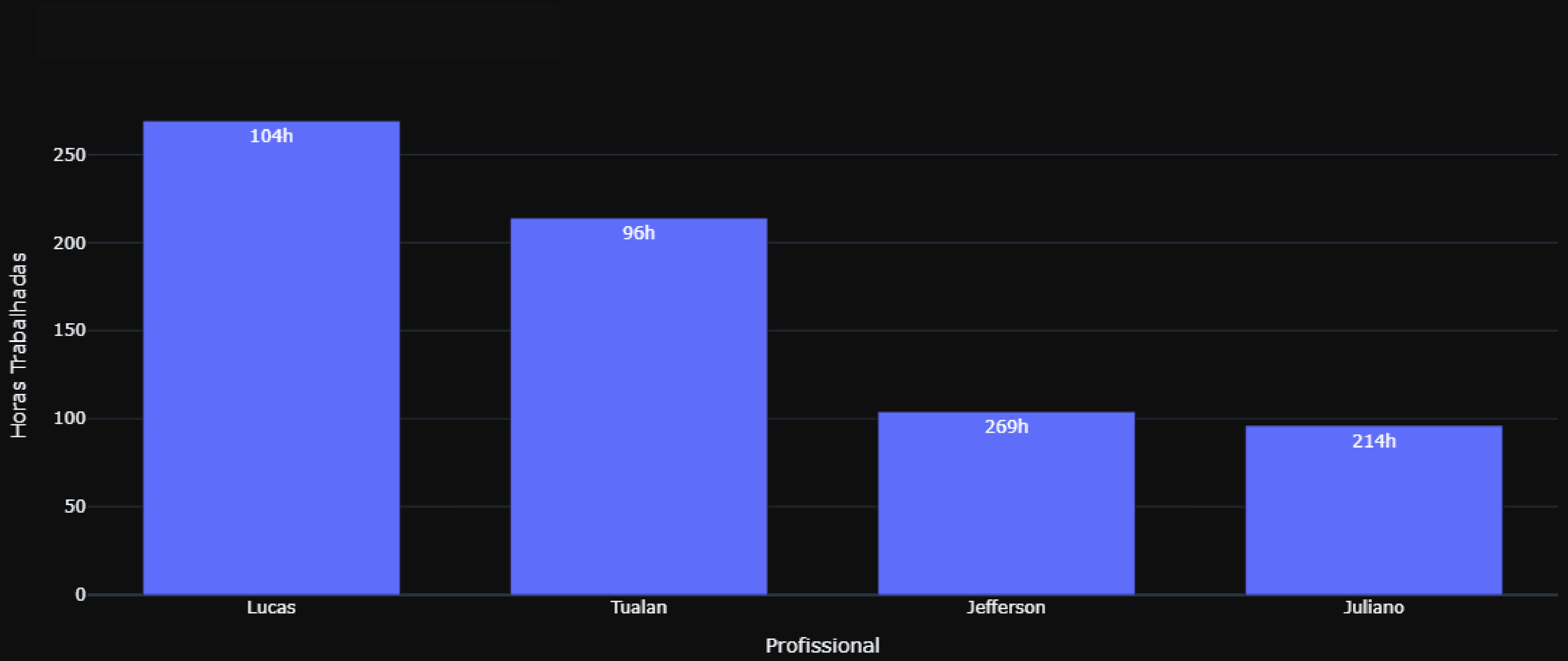
Faturamento por Profissional



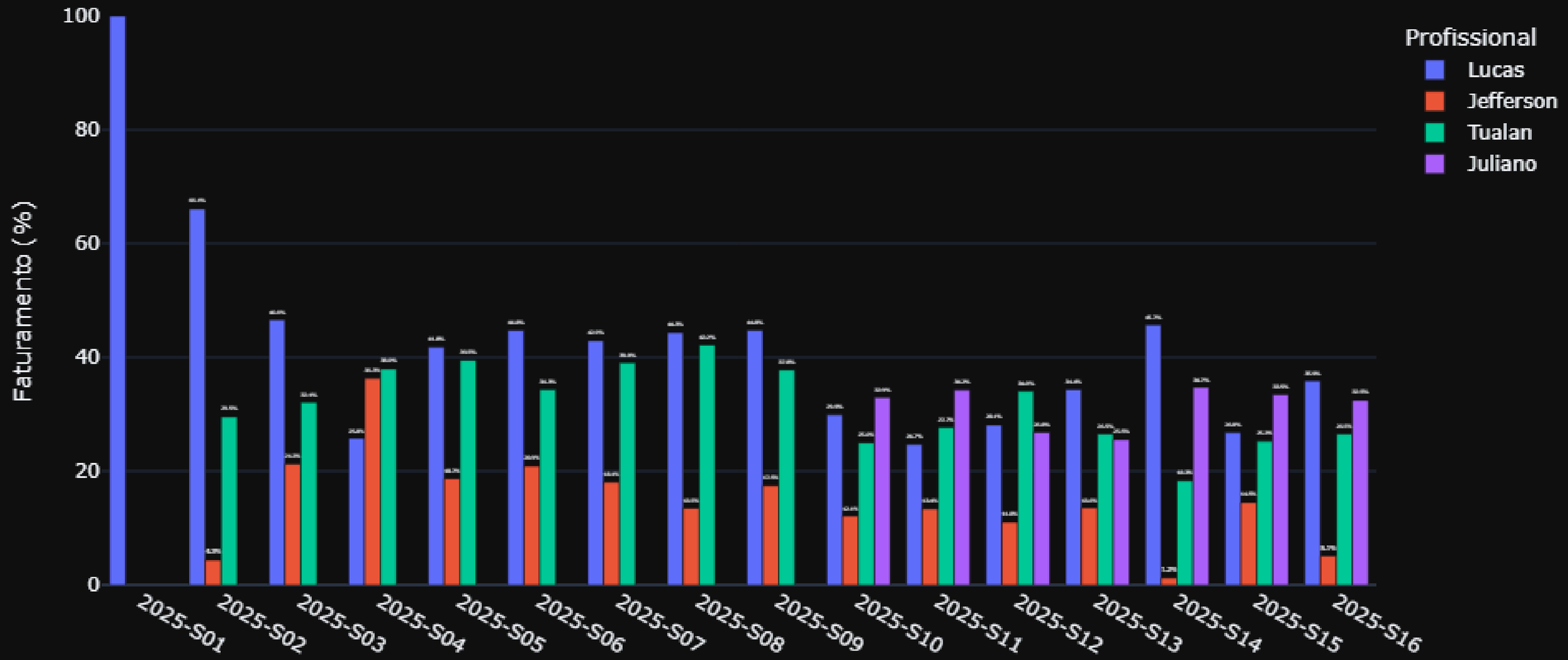
Atendimentos por Profissional



Horas Trabalhadas por Profissional



Faturamento por Profissional por Semana (Exceto Abril)



TÉCNICAS UTILIZADAS

Distribuição de Idade

```
# @title Gráfico - Idade {"vertical-output":true,"display-mode":"form"}
# Convertendo a coluna de cadastro para datetime
clientes_df["CADASTRO"] = pd.to_datetime(clientes_df["CADASTRO"], dayfirst=T

# Gráfico Idade dos Clientes
fig2 = px.histogram(
    clientes_df, x="IDADE",
    nbins=10,
    labels={"IDADE": "Idade"},
    title="Distribuição de Idade",
    template="plotly_dark"
)
fig2.update_layout(bargap=0.2)
fig2.show()
```

Distribuição de Atendimento por Gênero

```
# @title Gráfico - Sexo {"vertical-output":true,"display-mode":"form"}
# Gráfico 1: Distribuição por Sexo
sexo_df = clientes_df["SEXO"].value_counts().reset_index()
sexo_df.columns = ["SEXO", "count"]

color_map = {
    "M": "#1877F2",
    "F": "#C13584"
}

fig1 = px.pie(
    sexo_df,
    values="count", names="SEXO",
    # x="SEXO", y="count",
    # labels={"SEXO": "Sexo", "count": "Número de Clientes"},
    title="Distribuição por Sexo",
    template="plotly_dark",
    color="SEXO",
    color_discrete_map=color_map
)
fig1.show()
```

Origem do Primeiro Contato

```
# @title Gráfico - Contato {"vertical-output":true,"display-mode":"form"}
# Gráfico 3: Primeiro Contato
contato_df = clientes_df["PRIMEIRO CONTATO"].value_counts().reset_index()
contato_df.columns = ["Canal", "count"]

color_map = {
    "Whatsapp": "#25D366",
    "Instagram": "#C13584",
    "Facebook": "#1877F2",
    "Google": "#DB4437",
    "Indicação": "#00BFFF",
    "Shopping": "#FFA500",
    "Site": "#A9A9A9"
}

fig3 = px.pie(
    contato_df,
    values="count", names="Canal",
    title="Origem do Primeiro Contato",
    template="plotly_dark",
    color="Canal",
    color_discrete_map=color_map
)
# fig3.update_traces(textinfo="label+percent")
fig3.update_traces(
    texttemplate="%{label}<br>%{percent} (%{value})",
    textposition="inside"
)
fig3.show()
```

Serviços Mais Realizados

```
# Juntar Corte Jeff para apenas Corte
# @title Gráfico - Serviços Mais Realizados {"vertical-output":true,"display-mode":"form"}

# Unificar categorias de serviços
def categorizar_servico(servico):
    servico = servico.lower()
    if "barba" in servico and "corte" in servico:
        return "Cabelo e Barba"
    elif "barba" in servico:
        return "Barba"
    elif "corte" in servico or "juliano" in servico or "promocional" in servico:
        return "Cabelo"
    else:
        return "Outros"

df["Serviço Simplificado"] = df["Serviço"].apply(categorizar_servico)

df_serv = df.groupby("Serviço Simplificado").size().reset_index(name="Quantidade")

fig = px.bar(df_serv.sort_values("Quantidade", ascending=False),
             x="Serviço Simplificado", y="Quantidade",
             title="Serviços Mais Realizados (Categorias Unificadas)",
             template="plotly_dark")
fig.update_layout(bargap=0.3)
fig.show()
```

Mapa de Calor: Agendamentos por Dia da Semana e Mês

```
# @title Gráfico - Agendamentos por Dia da Semana e Mês {"vertical-output":true,"display-mode":"form"}


# Criar colunas auxiliares
df["mes"] = df["Data"].dt.strftime("%m/%y")  # Formato MM/AA
df["dia_da_semana_en"] = df["Data"].dt.day_name()

# Traduzir dias da semana manualmente
traducao_dias = {
    'Monday': 'segunda-feira',
    'Tuesday': 'terça-feira',
    'Wednesday': 'quarta-feira',
    'Thursday': 'quinta-feira',
    'Friday': 'sexta-feira',
    'Saturday': 'sábado',
    'Sunday': 'domingo'
}
df["dia_da_semana"] = df["dia_da_semana_en"].map(traducao_dias)

# Contar agendamentos
heatmap_data = df.groupby(['mes', 'dia_da_semana']).size().reset_index(name='agendamentos')
heatmap_pivot = heatmap_data.pivot(index='dia_da_semana', columns='mes', values='agendamentos').fillna(0)

# Ordenar os dias da semana
dias_ordem = ['segunda-feira', 'terça-feira', 'quarta-feira', 'quinta-feira', 'sexta-feira', 'sábado', 'domingo']
heatmap_pivot = heatmap_pivot.reindex(dias_ordem)

# Criar gráfico
fig_heatmap = px.imshow(
    heatmap_pivot,
    labels=dict(x='Mês', y='Dia da Semana', color='Qtd. Agendamentos'),
    x=heatmap_pivot.columns,
    y=heatmap_pivot.index,
    color_continuous_scale='YlOrRd',
    text_auto='.0f',
    aspect='auto',
    title='Agendamentos por Dia da Semana e Mês'
)

fig_heatmap.update_layout(margin=dict(l=40, r=40, t=50, b=40))
fig_heatmap.show()
```

Evolução de Agendamentos por Semana

```
# @title Gráfico - Evolução de Agendamentos por Semana {"vertical-output":true,"display-mode":"form"}
# Agrupar por semana e extrair o início da semana como data
agendamentos_semanais = df.groupby(df["Data"].dt.to_period("W")).size().reset_index(name="Qtd")
agendamentos_semanais["Data"] = agendamentos_semanais["Data"].apply(lambda x: x.start_time)
agendamentos_semanais["Semana"] = agendamentos_semanais["Data"].dt.strftime("S%W/%y")

fig = px.line(
    agendamentos_semanais,
    x="Data", y="Qtd",
    title="Evolução de Agendamentos por Semana",
    labels={"Data": "Semana", "Qtd": "Agendamentos"},
    template="plotly_dark"
)

fig.update_traces(
    mode="lines+markers+text",
    texttemplate="%{customdata[0]}<br>Q: %{y}",
    textposition="top center",
    textfont_size=10,
    customdata=agendamentos_semanais[["Semana"]]
)

fig.update_yaxes(range=[0, agendamentos_semanais["Qtd"].max() * 1.2])
fig.show()
```

Clientes Únicos por Mês

```
# @title Gráfico - Clientes Únicos por Mês {"vertical-output":true,"display-mode":"form"}


# Valido alterar - Usando rankingDeClientes.csv , distribuir clientes novos e antigos por semana
clientes_mes = df.groupby('mes')['cliente'].nunique().reset_index(name='quantidade')
fig = px.bar(clientes_mes, x='mes', y='quantidade', title='Clientes Únicos por Mês', labels={'quantidade': 'Clientes'})
fig.show()
```

Evolução de Cadastros por Mês

```
# @title Gráfico - Cadastros {"vertical-output":true,"display-mode":"form"}
# Agrupa por mês e formata para MM/YY
cadastros_mensais = clientes_df.groupby(clientes_df["CADASTRO"].dt.to_period("M")).size().reset_index(name="Qtd")
cadastros_mensais["CADASTRO"] = pd.to_datetime(cadastros_mensais["CADASTRO"].astype(str))
cadastros_mensais["label"] = cadastros_mensais["CADASTRO"].dt.strftime("%m/%y")

fig = px.line(
    cadastros_mensais,
    x="CADASTRO", y="Qtd",
    title="Evolução de Cadastros por Mês",
    labels={"CADASTRO": "Mês", "Qtd": "Cadastros"},
    markers=True,
    text="label",  # 👈 MM/YY acima dos pontos
    template="plotly_dark"
)

fig.update_traces(
    textposition="top center"
)
fig.update_traces(textfont_size=10)
fig.show()
```

Distribuição por Tipo de Pagamento

```
# @title Gráfico - Distribuição por Tipo de Pagamento (Agrupado) {"vertical-output":true,"display-


import plotly.express as px

# Criar uma cópia da coluna com categorias agrupadas
df_grafico = df.copy()
df_grafico['categoria'] = df_grafico['tipo_de_forma_de_pagamento'].replace({
    'PIX': 'PIX / À Vista',
    'À Vista': 'PIX / À Vista'
})

# Agrupar com base na nova categoria
agrupado_categoria = df_grafico.groupby('categoria')['valor_pago'].sum()

# Criar o gráfico de pizza
fig_pizza = px.pie(
    agrupado_categoria.reset_index(),
    names='categoria',
    values='valor_pago',
    title='Distribuição por Tipo de Pagamento (Agrupado)',
    hole=0
)

fig_pizza.update_traces(
    textinfo='label+percent',
    textposition='inside',
    showlegend=False
)

fig_pizza.show()
```

Total de Cada Forma de Pagamento

```
# @title Gráfico - % (novo) de Cada Forma de Pagamento {"vertical-output":true,"display-mode":"form"}
import plotly.express as px


porcent_forma = (
    df.groupby('forma_de_pagamento')['valor_pago']
    .sum()
    .reset_index()
    .sort_values(by='valor_pago', ascending=True)
)

# Calcular o total geral
total_pago = porcent_forma['valor_pago'].sum()


porcent_forma['percentual'] = (porcent_forma['valor_pago'] / total_pago) * 100


fig_forma_barra = px.bar(
    porcent_forma,
    x='percentual',
    y='forma_de_pagamento',
    orientation='h',
    title='% de Cada Forma de Pagamento',
    template='plotly_dark',
    labels={'percentual': 'Percentual (%)', 'forma_de_pagamento': 'Forma de Pagamento'}
)


fig_forma_barra.update_traces(
    text=porcent_forma['percentual'].apply(lambda x: f'{x:.1f}%'),
    textposition='outside'
)

fig_forma_barra.update_layout(
    yaxis_title='Forma de Pagamento',
    xaxis_title='Percentual (%)',
    margin=dict(l=100, r=50, t=50, b=50)
)

fig_forma_barra.show()
```

Total de Taxas Pagas por Forma de Pagamento

```python
# @title Gráfico - Total de Taxas Pagas por Forma de Pagamento %{"vertical-output":true,"display-mode":"f

import plotly.express as px
import pandas as pd

# Processar coluna de taxa
coluna_taxa = (
    df['valor_de_desconto_da_operadora_(r$)']
    .astype(str)
    .str.replace('R$', '', regex=False)
    .str.replace('.', '', regex=False)
    .str.replace(',', '.', regex=False)
    .astype(float)
    .abs()
)

formas_pagamento = df['tipo_de_forma_de_pagamento']

df_temp = pd.DataFrame({
    'Forma de Pagamento': formas_pagamento,
    'Taxa (R$)': coluna_taxa
})

agrupado_taxa = (
    df_temp.groupby('Forma de Pagamento')['Taxa (R$)']
    .sum()
    .loc[lambda x: x > 0]
    .sort_values(ascending=False)
)

total_taxa = agrupado_taxa.sum()
df_percentual = agrupado_taxa.reset_index()
```

```python
total_taxa = agrupado_taxa.sum()
df_percentual = agrupado_taxa.reset_index()
df_percentual['Percentual'] = (df_percentual['Taxa (R$)'] / total_taxa) * 100

fig_taxa = px.bar(
    df_percentual,
    x='Percentual',
    y='Forma de Pagamento',
    orientation='h',
    title='Percentual de Taxas Pagas por Forma de Pagamento',
    labels={
        'Forma de Pagamento': 'Forma de Pagamento',
        'Percentual': 'Percentual (%)'
    },
    template='plotly_dark'
)

# Adicionar rótulos em porcentagem
fig_taxa.update_traces(
    text=df_percentual['Percentual'].apply(lambda x: f'{x:.1f}%'),
    textposition='outside'
)

fig_taxa.update_layout(
    xaxis_title='Percentual (%)',
    margin=dict(l=100, r=50, t=50, b=50)
)

fig_taxa.show()
```

Total de Taxas Pagas por Bandeira

```
# @title Gráfico - Total de Taxas Pagas por Bandeira {"vertical-output":true,"display-mode":"form"}


import plotly.express as px
import pandas as pd

# Extrair a coluna de taxa como float absoluto
coluna_taxa = (
    df['valor_de_desconto_da_operadora_(r$)']
    .astype(str)
    .str.replace('R$', '', regex=False)
    .str.replace('.', '', regex=False)
    .str.replace(',', '.', regex=False)
    .astype(float)
    .abs()
)

# Bandeira (forma de pagamento detalhada)
bandeiras = df['forma_de_pagamento']

# DataFrame temporário
df_temp = pd.DataFrame({
    'Bandeira': bandeiras,
    'Taxa (R$)': coluna_taxa
})

# Agrupar e remover valores zerados
agrupado_taxa_bandeira = (
    df_temp.groupby('Bandeira')['Taxa (R$)']
    .sum()
    .loc[lambda x: x > 0]
    .sort_values(ascending=False)
)

# Gráfico de barras horizontal
fig_bandeira = px.bar(
    agrupado_taxa_bandeira.reset_index(),
    x='Taxa (R$)',
    y='Bandeira',
    orientation='h',
    title='Total de Taxas Pagas por Bandeira',
    labels={
        'Bandeira': 'Bandeira',
        'Taxa (R$)': 'Total de Taxas (R$)'
    }
)

# Adicionar texto formatado em reais
fig_bandeira.update_traces(
text=agrupado_taxa_bandeira.apply(lambda x: f"R$ {x:,.2f}".replace('.', ',').replace(',', '.', 1)),
textposition='outside'

)


fig_bandeira.show()
```

Faturamento por Semana

```
# @title Gráfico - Faturamento por Semana %{"vertical-output":true,"display-mode":"form"}

df_semana = df.groupby('semana', as_index=False)['valor_pago'].sum()

# Calcular o total geral
total_geral = df_semana['valor_pago'].sum()


df_semana['percentual'] = (df_semana['valor_pago'] / total_geral) * 100


df_semana['texto'] = df_semana['percentual'].round(1).astype(str) + '%'


fig3 = px.line(
    df_semana,
    x='semana', y='percentual',
    title='Faturamento por Semana (%)',
    labels={'semana': 'Semana', 'percentual': 'Percentual (%)'}
)


fig3.update_traces(mode='lines+markers+text', text=df_semana['texto'], textposition='top center')
fig3.update_layout(yaxis_range=[0, 100])  # Ajuste do eixo Y para 0-100%

fig3.show()
```

Faturamento por Profissional por Mês (Exceto Abril)

```python
# @title Gráfico - Faturamento por Profissional por Mês % (sem o último Mês) {"vertical-output":true,"display-mode":"form

# Garantir o tipo datetime correto
df["Data"] = pd.to_datetime(df["Data"], format="%d/%m/%Y")

# Extrair ano e mês
df["Ano"] = df["Data"].dt.year
df["Mes"] = df["Data"].dt.month
df["Ano_Mes_dt"] = df["Data"].dt.to_period("M").dt.to_timestamp()
df["Ano_Mes"] = df["Ano_Mes_dt"].dt.strftime("%b/%Y")  # Ex: Jan/2025

# Remover o último mês detectado
ultimo_ano = df["Ano"].max()
ultimo_mes = df[df["Ano"] == ultimo_ano]["Mes"].max()
df_filtrado = df[~((df["Ano"] == ultimo_ano) & (df["Mes"] == ultimo_mes))]

# Agrupar faturamento por mês e profissional
faturamento_mes = df_filtrado.groupby(["Ano_Mes", "Profissional"])["Valor"].sum().reset_index()

total_por_mes = faturamento_mes.groupby("Ano_Mes")["Valor"].transform("sum")
faturamento_mes["Percentual"] = (faturamento_mes["Valor"] / total_por_mes) * 100
faturamento_mes["Texto"] = faturamento_mes["Percentual"].round(1).astype(str) + "%"

fig = px.bar(
    faturamento_mes,
    x="Ano_Mes",
    y="Percentual",
    color="Profissional",
    barmode="group",
    text="Texto",
    title="Faturamento por Profissional por Mês (%)",
    labels={"Percentual": "Faturamento (%)", "Ano_Mes": "Mês"},
    template="plotly_dark",
    category_orders={"Ano_Mes": sorted(faturamento_mes["Ano_Mes"].unique(), key=lambda x: pd.to_datetime(x, format="%b/%
)
```

```python
fig.update_traces(
    textposition="outside",
    textfont_size=10,
    texttemplate="%{text}"
)

fig.update_layout(
    bargap=0.2,
    xaxis_title="Mês (Abrev./Ano)",
    yaxis_title="Faturamento (%)"
)
fig.update_yaxes(range=[0, 100])

fig.show()
```

Faturamento por Dia da Semana

```
# @title Gráfico - Faturamento por Dia da Semana {"vertical-output":true,"display-mode":"form"}


# Valido
dias = df.groupby('dia_da_semana')['valor_pago'].sum().reset_index()
dias['ordem'] = dias['dia_da_semana'].map({
    'segunda-feira': 0,
    'terça-feira': 1,
    'quarta-feira': 2,
    'quinta-feira': 3,
    'sexta-feira': 4,
    'sábado': 5,
    'domingo': 6
})
dias = dias.sort_values('ordem')

fig5 = px.bar(
    dias,
    x='dia_da_semana', y='valor_pago',
    title='Faturamento por Dia da Semana',
    labels={'dia_da_semana': 'Dia', 'valor_pago': 'Valor Pago (R$)'}
)
fig5.update_layout(margin=dict(l=40, r=40, t=50, b=40))
fig5.update_traces(text=dias['valor_pago'].apply(format_brl), textposition='outside')
fig5.show()
```

Faturamento por Profissional

```
# Valido - Mostrar total de cada servico
# @title Gráfico - Faturamento por Profissional (Percentual) {"vertical-output":true,"display-mode":"form"}

df_fat = df.groupby("Profissional")["Valor"].sum().reset_index()
df_fat["Valor_Label"] = "R$: " + df_fat["Valor"].round(2).astype(str)


df_fat["Percentual"] = (df_fat["Valor"] / df_fat["Valor"].sum()) * 100
df_fat["Percentual_Label"] = df_fat["Percentual"].round(2).astype(str) + "%"

fig = px.bar(
    df_fat,
    x="Profissional",
    y="Percentual",
    title="Faturamento por Profissional (%)",
    template="plotly_dark",
    text="Percentual_Label"
)

fig.update_traces(
    textposition="outside",
    textfont_size=11,
    texttemplate="%{x}<br>%{text}"
)

fig.update_yaxes(range=[0, df_fat["Percentual"].max() * 1.3])
fig.update_layout(bargap=0.3, height=500)
```

Atendimentos por Profissional

```
# @title Gráfico - Atendimentos por Profissional {"vertical-output":true,"display-mode":"form"}

df_qtd = df.groupby("Profissional").size().reset_index(name="Atendimentos")
df_qtd["Texto"] = df_qtd["Atendimentos"].astype(str)

fig = px.bar(
    df_qtd,
    x="Profissional",
    y="Atendimentos",
    title="Atendimentos por Profissional",
    template="plotly_dark",
    text="Texto"
)

fig.update_traces(
    textposition="outside",
    textfont_size=11,
    texttemplate="%{text}"
)

# Garantir espaço suficiente para os rótulos
fig.update_yaxes(range=[0, df_qtd["Atendimentos"].max() * 1.2])
fig.update_layout(bargap=0.3)

fig.show()
```

Horas Trabalhadas por Profissional

```
# @title Gráfico - Horas Trabalhadas por Profissional {"vertical-output":true,"display-mode":"form"}

# Calcular horas a partir de minutos
horas_trabalhadas = df.groupby("Profissional")["Duracao_min"].sum().reset_index()
horas_trabalhadas["Horas"] = (horas_trabalhadas["Duracao_min"] / 60).round(0)

# Adicionar coluna de texto com "h"
horas_trabalhadas["Horas_str"] = horas_trabalhadas["Horas"].astype(int).astype(str) + "h"



# Criar gráfico
fig = px.bar(
    horas_trabalhadas.sort_values("Horas", ascending=False),
    x="Profissional", y="Horas",
    title="Horas Trabalhadas por Profissional",
    labels={"Horas": "Horas Trabalhadas"},
    text=horas_trabalhadas["Horas_str"],  # Usar string personalizada
    template="plotly_dark"
)

fig.update_layout(bargap=0.3)
fig.show()
```

Faturamento por Profissional por Semana (Exceto Abril)

```python
# Extrair ano e semana ISO
df["Ano"] = df["Data"].dt.isocalendar().year
df["Semana"] = df["Data"].dt.isocalendar().week
df["Ano_Semana"] = df["Ano"].astype(str) + "-S" + df["Semana"].astype(str).str.zfill(2)

# Remover a última semana do ano mais recente
ultima_semana = df["Semana"].max()
ultimo_ano = df["Ano"].max()
df_filtrado = df[~((df["Ano"] == ultimo_ano) & (df["Semana"] == ultima_semana))]

# Agrupar por semana e profissional
faturamento_semana = df_filtrado.groupby(["Ano_Semana", "Profissional"])["Valor"].sum().reset_index()


total_por_semana = faturamento_semana.groupby("Ano_Semana")["Valor"].transform("sum")
faturamento_semana["Percentual"] = (faturamento_semana["Valor"] / total_por_semana) * 100
faturamento_semana["Texto"] = faturamento_semana["Percentual"].round(1).astype(str) + "%"

# Criar gráfico
fig = px.bar(
    faturamento_semana,
    x="Ano_Semana",
    y="Percentual",
    color="Profissional",
    barmode="group",
    text="Texto",
    title="Faturamento por Profissional por Semana (%)",
    labels={"Percentual": "Faturamento (%)", "Ano_Semana": "Semana"},
    template="plotly_dark"
)

# Ajustes visuais
fig.update_traces(
    textposition="outside",
    textfont_size=10,
    texttemplate="%{text}"
)

fig.update_layout(
    bargap=0.2,
    xaxis_title="Semana (Ano-Semana)",
    yaxis_title="Faturamento (%)"
)
```

```python
# Ajustes visuais
fig.update_traces(
    textposition="outside",
    textfont_size=10,
    texttemplate="%{text}"
)

fig.update_layout(
    bargap=0.2,
    xaxis_title="Semana (Ano-Semana)",
    yaxis_title="Faturamento (%)"
)
fig.update_yaxes(range=[0, 100])  # Percentuais de 0% a 100%

fig.show()
```