In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sea
import matplotlib.pyplot as plt
from datetime import timedelta
```

In [2]:
```
path  = "C:\\Users\\mateu\\Documents\\MEGA\\Matérias UFC CD\\Modelagem Estatistica\\trabalhoPratico\\trabalho2\\dataset\\insta
insta = pd.read_csv(path)
insta.head(3)
```

Out[2]:

	Segunda	Terca	Quarta	Quinta	Sexta	Sabado	Domingo	idade	genero	trabalha	trabalhaInstagram	videosLongos	moraS
0	56 minutos	1 hora e 16 minutos	47 minutos	1 hora e 22 minutos	22 minutos	55 minutos	38 minutos	24	Masculino	Sim	Não	Sim	Nã
1	1 h 22 min	1 h 27 min	53 mim	1 h 13 min	38 min	1 h 50 min	32 min	23	Hétero	Sim	Não	Sim	Sir
2	56min	1h 35min	1h 39min	43min	58min	2h 58min	1h 14min	22	Masculino	Sim	Não	Sim	Nã

In [3]:
```
insta.shape
```

Out[3]:
```
(15, 18)
```

Limpeza dos dados

In [4]:
```
semanas = ["Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado","Domingo", "translado"]


def to_minutos(x):
    if "h" in x and "m" in x:
        # Modificando valores incosistentes
        x = x.lower()
        x = x.replace("e","")
        x = x.replace("horas", "h").replace("horas", "h").replace("hora", "h")
        x = x.split("h")

        # Convertendo para numericos
        horas   = int(x[0])
        minutos = ''
        for i in x[1]:
            if str.isdigit(i):  minutos += i
        minutos = float(minutos)

        # Retorna as horas min em segundos
        return timedelta(hours=horas, minutes=minutos).seconds
    if "m" in x and '~' not in x:
        minutos = ''
        for i in x:
            if str.isdigit(i):  minutos += i
        minutos = float(minutos)
        return timedelta(minutes=minutos).seconds

    if ":" in x:
        x = x.split(':')
        horas   = int(x[0])
        minutos = ''
        for i in x[1]:
            if str.isdigit(i):  minutos += i
        minutos = float(minutos)

        return timedelta(hours=horas, minutes=minutos).seconds
    if "h"  in x:
        horas   = int(x[0])
        try:
            minutos = ''
```

```
            for i in x[1]:
                if str.isdigit(i):  minutos += i
            minutos = float(minutos)
            return timedelta(hours=horas, minutes=minutos)
        except:
            return timedelta(hours=horas).seconds

    if "hr" in x or "horas" in x or "hora" in x:
        horas = ''
        for i in x:
            if str.isdigit(i):  horas += i
        horas = float(horas)
        return timedelta(hours=horas).seconds

    if "~" in x: return timedelta(minutes=40).seconds
    return x

for dia in semanas:

    insta[dia] = insta[dia].apply(lambda x:to_minutos(x))
```

Modificando incosistencias do atributo `idade`

In [5]:
```
insta["idade"] = insta["idade"]\
                    .str.replace("ano",'')\
                    .str.replace('s','').astype(int)
```

Como atributo `genero` não tem valores diferente será removido do conj. dados

In [6]:
```
insta.drop(columns=['genero'], inplace=True)
```

Como o atributo `Semestre` é repetido será removido do conj. de dados

In [7]:
```
insta.drop(columns=['Semestre'], inplace=True)
```

O atributo `trabalhaInstagram` também tem elementos repetidos

In [8]:
```
insta.drop(columns=['trabalhaInstagram'], inplace=True)
```

Modificando incosistencias do atributo `acesso`

In [9]:
```
insta['acesso'] = insta['acesso'].str.lower().str.replace('.','')
```

In [10]:
```
import re

texts = insta['acesso'].tolist()

# Expressão regular para extrair "4g" e "wi-fi"
keywords = [' '.join(re.findall(r'\b(?:4g|wi-fi)\b', text, re.IGNORECASE)) for text in texts]

insta['acesso'] = keywords
```

Transformando o valor `onibus` em np.NA

In [11]:
```
def to_na(x):
    if x =='onibus':
        return pd.NA
    return x

insta['translado'] = insta['translado'].apply(lambda x: to_na(x))
```

Paras os valores da semana abaixo de 1 serão tratados como minutos

In [12]:
```
def to_minutos(x):

    if pd.isna(x): return x
    x = float(x)
    if x > -1 and x < 1:
        return timedelta(minutes=x).seconds
    if x >= 1 and x<=2:
        return timedelta(hours=x).seconds
    if x > 10 and x < 60:
        return timedelta(minutes=x).seconds
    return x
```

In [13]:
```
insta
```

Out[13]:

	Segunda	Terca	Quarta	Quinta	Sexta	Sabado	Domingo	idade	trabalha	videosLongos	moraSo	streamingDiario	acesso	transla
0	3360	4560	2820	4920	1320	3300	2280	24	Sim	Sim	Não	Sim	4g wi-fi	24
1	4920	5220	3180	4380	2280	6600	1920	23	Sim	Sim	Sim	Não	4g wi-fi	24
2	3360	5700	5940	2580	3480	10680	4440	22	Sim	Sim	Não	Não	4g wi-fi	60
3	0	0	0.85	0	0	0	0	21	Sim	Sim	Não	Não	wi-fi	72
4	0.5	1	0.35	1	0.75	2	1.5	22	Não	Sim	Não	Sim	4g wi-fi	36
5	13020	10380	5400	12660	11460	16080	17880	23	Não	Sim	Não	Sim	wi-fi	<N
6	1500	1860	3120	4980	4440	3420	4800	24	Sim	Não	Não	Não	4g wi-fi	54
7	9600	11640	13680	8160	11580	17880	16260	25	Não	Sim	Não	Não	wi-fi	108
8	10800	3600	10800	5400	1200	1740	3600	22	Sim	Não	Não	Sim	wi-fi	72
9	6480	8040	7200	7200	10080	7440	6900	21	Sim	Sim	Não	Sim	4g wi-fi	72
10	720	1500	420	780	1260	2160	2340	21	Não	Sim	Não	Não	wi-fi	18
11	1080	600	540	0	480	300	300	22	Sim	Não	Não	Sim	wi-fi	108
12	7560	9300	9000	7740	6780	5760	5580	23	Sim	Sim	Não	Sim	4g wi-fi	252
13	39	43	52	38	35	21	61	23	Sim	Sim	Sim	Não	4g wi-fi	

	Segunda	Terca	Quarta	Quinta	Sexta	Sabado	Domingo	idade	trabalha	videosLongos	moraSo	streamingDiario	acesso	transla
14	3240	1380	7200	2220	2520	4740	4860	22	Sim	Sim	Sim	Não	wi-fi	2

In [14]:
```
for dias in semanas:
    insta[dias] = insta[dias].apply(lambda x:to_minutos(x))
    insta[dias] = insta[dias].apply(lambda x:to_minutos(x))
```

In [15]:
```
insta
```

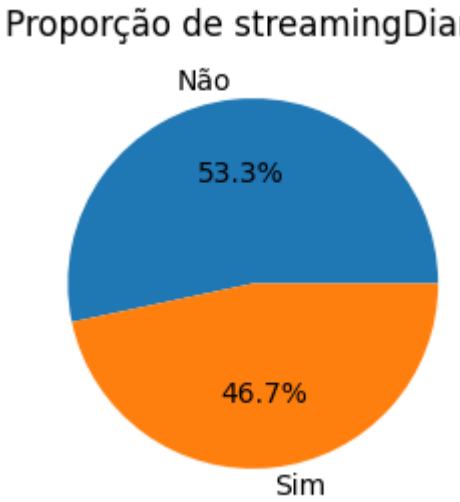
Out[15]:

	Segunda	Terca	Quarta	Quinta	Sexta	Sabado	Domingo	idade	trabalha	videosLongos	moraSo	streamingDiario	acesso	tra
0	3360.0	4560.0	2820.0	4920.0	1320.0	3300.0	2280.0	24	Sim	Sim	Não	Sim	4g wi-fi	
1	4920.0	5220.0	3180.0	4380.0	2280.0	6600.0	1920.0	23	Sim	Sim	Sim	Não	4g wi-fi	
2	3360.0	5700.0	5940.0	2580.0	3480.0	10680.0	4440.0	22	Sim	Sim	Não	Não	4g wi-fi	
3	0.0	0.0	3060.0	0.0	0.0	0.0	0.0	21	Sim	Sim	Não	Não	wi-fi	
4	1800.0	3600.0	1260.0	3600.0	2700.0	7200.0	5400.0	22	Não	Sim	Não	Sim	4g wi-fi	
5	13020.0	10380.0	5400.0	12660.0	11460.0	16080.0	17880.0	23	Não	Sim	Não	Sim	wi-fi	
6	1500.0	1860.0	3120.0	4980.0	4440.0	3420.0	4800.0	24	Sim	Não	Não	Não	4g wi-fi	
7	9600.0	11640.0	13680.0	8160.0	11580.0	17880.0	16260.0	25	Não	Sim	Não	Não	wi-fi	
8	10800.0	3600.0	10800.0	5400.0	1200.0	1740.0	3600.0	22	Sim	Não	Não	Sim	wi-fi	
9	6480.0	8040.0	7200.0	7200.0	10080.0	7440.0	6900.0	21	Sim	Sim	Não	Sim	4g wi-fi	
10	720.0	1500.0	420.0	780.0	1260.0	2160.0	2340.0	21	Não	Sim	Não	Não	wi-fi	
11	1080.0	600.0	540.0	0.0	480.0	300.0	300.0	22	Sim	Não	Não	Sim	wi-fi	
12	7560.0	9300.0	9000.0	7740.0	6780.0	5760.0	5580.0	23	Sim	Sim	Não	Sim	4g wi-fi	
13	2340.0	2580.0	3120.0	2280.0	2100.0	1260.0	61.0	23	Sim	Sim	Sim	Não	4g wi-fi	

	Segunda	Terca	Quarta	Quinta	Sexta	Sabado	Domingo	idade	trabalha	videosLongos	moraSo	streamingDiario	acesso	tra
14	3240.0	1380.0	7200.0	2220.0	2520.0	4740.0	4860.0	22	Sim	Sim	Sim	Não	wi-fi	

Analise de dados

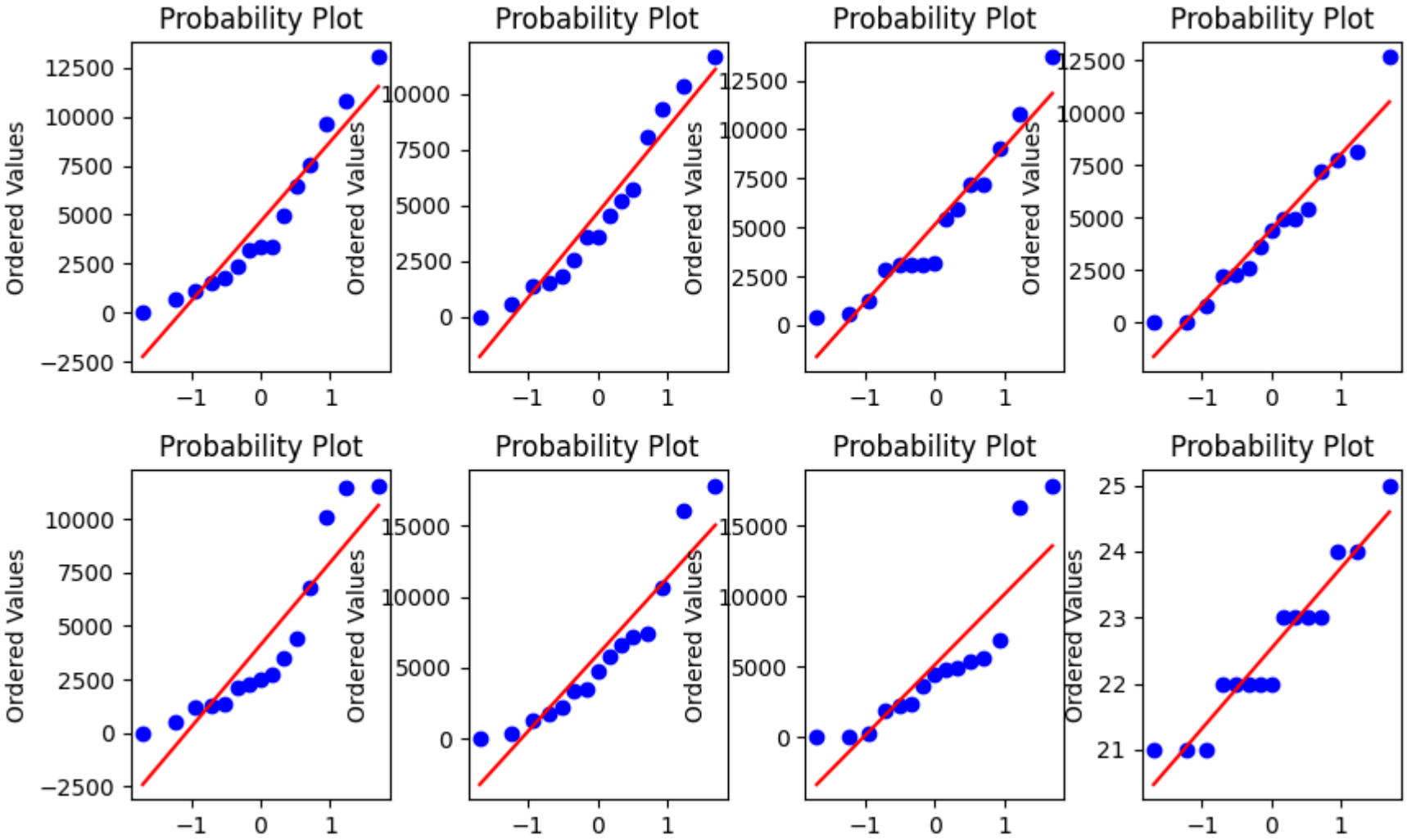
In [16]:
```
subset = insta['streamingDiario'].value_counts()

plt.figure(figsize=(3, 3))
plt.title("Proporção de streamingDiario")
plt.pie(subset, labels=subset.index, autopct='%1.1f%%')
plt.show()
```



Verificandos se os atributos numericos seguem uma normal

In [17]:
```
from scipy.stats import probplot
col = ["Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado","Domingo", "idade"]
```

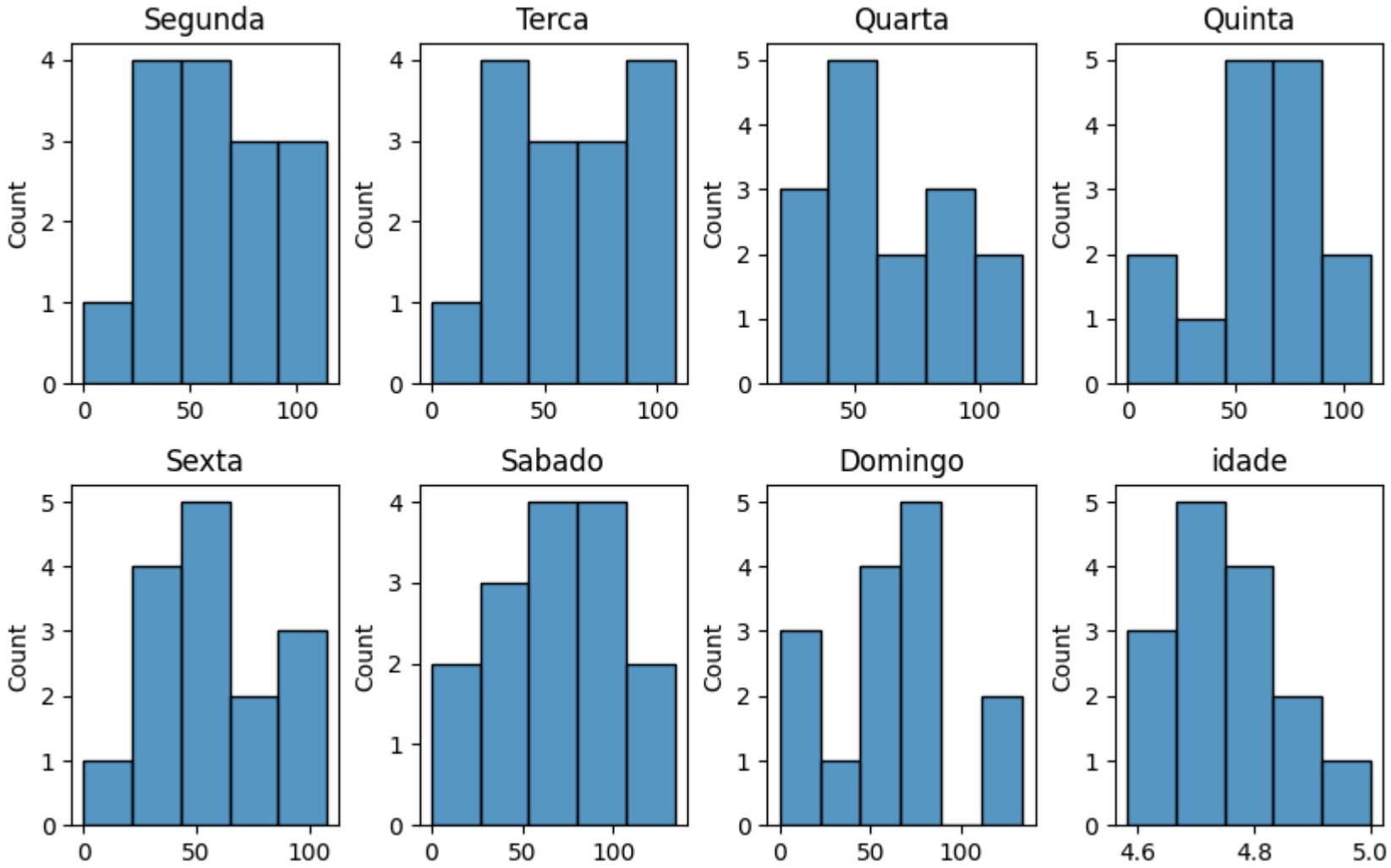
```
fig, axis = plt.subplots(2, 4, figsize=(10, 6))
fig.subplots_adjust(hspace = .3, wspace=.3)
for col_, ax in zip(col, axis.flatten()):
    ax.set_title(col_)
    probplot(x=insta[col_], dist='norm', plot=ax)
    ax.set_xlabel('')
plt.show()
```



In [18]:
```
from scipy.stats import shapiro
for col_ in col:
    stat, p = shapiro(insta[col_])
    print(f" {col_}\nstat = {stat:.2f}, p = {p:.2f}", end=' ')
    if p > 0.05:
        print("Seguem uma normal")
    else:
        print("Não seguem uma normal")
```

```
 Segunda
stat = 0.90, p = 0.10 Seguem uma normal
 Terca
stat = 0.92, p = 0.23 Seguem uma normal
 Quarta
stat = 0.92, p = 0.20 Seguem uma normal
 Quinta
stat = 0.94, p = 0.41 Seguem uma normal
 Sexta
stat = 0.82, p = 0.01 Não seguem uma normal
 Sabado
stat = 0.88, p = 0.04 Não seguem uma normal
 Domingo
stat = 0.79, p = 0.00 Não seguem uma normal
 idade
stat = 0.92, p = 0.17 Seguem uma normal
```

In [19]:
```
fig, axis = plt.subplots(2, 4, figsize=(10, 6))
fig.subplots_adjust(hspace = .3, wspace=.3)

for col_, ax in zip(col, axis.flatten()):
    ax.set_title(col_)
    sea.histplot(x=np.sqrt(insta[col_]), ax=ax)
    ax.set_xlabel('')
plt.show()
```



Verificando a média de horas consumidas na semana

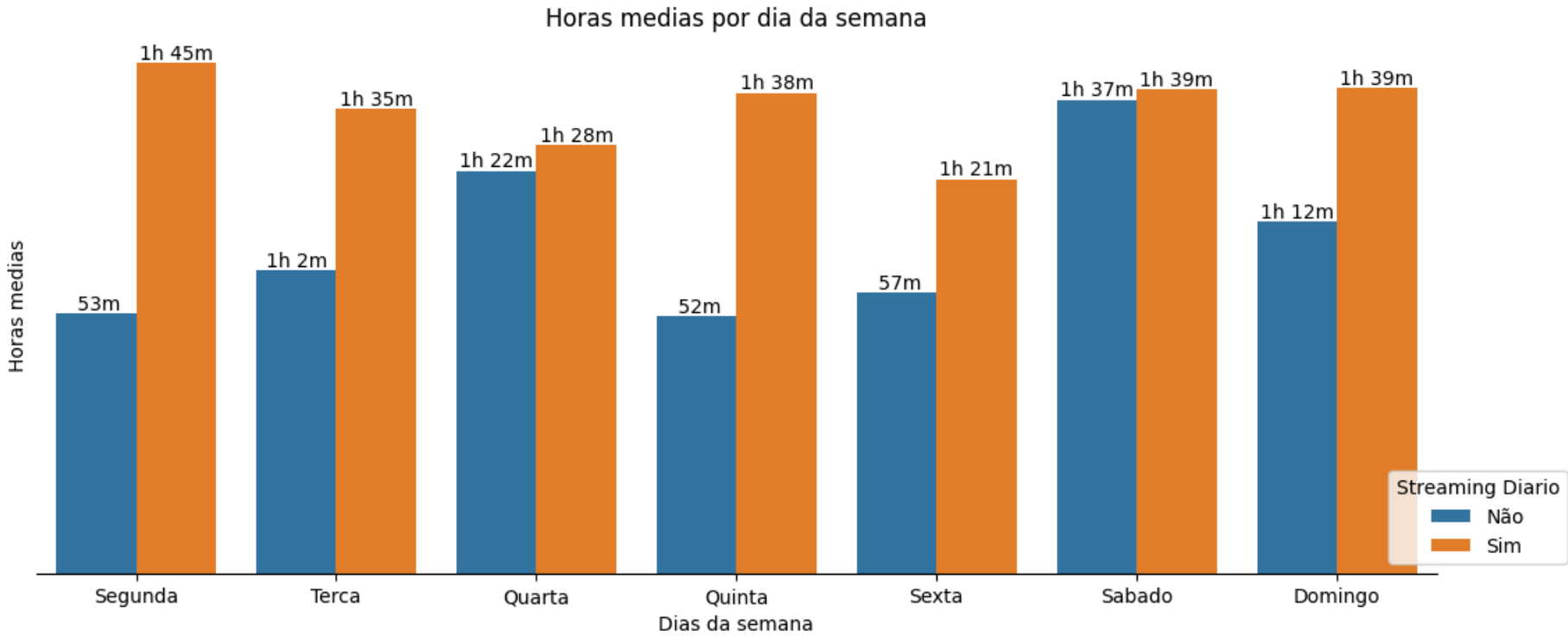
In [20]:
```
subset = insta.groupby(by=['streamingDiario'])\
    [['Segunda', 'Terca', 'Quarta', 'Quinta', 'Sexta', 'Sabado', 'Domingo']]\
        .mean().round(2).reset_index()
```

```
# Reshape
subset = pd.melt(subset, id_vars=['streamingDiario'], value_vars=['Segunda', 'Terca', 'Quarta', 'Quinta', 'Sexta', 'Sabado', '
```

Convertendo para horas (ex : 1h30m)

In [21]:
```
horas = subset['value'].apply(lambda x:timedelta(hours=(x/3600)))
horas = pd.to_timedelta(horas, unit='h').dt.components[['hours',"minutes"]]
subset['horas_medias_label'] = horas\
    .apply(lambda x: (x['hours'].astype(str)+'h' if x['hours']!=0 else '')+' '+ x['minutes'].astype(str)+'m', axis=1)
```

In [22]:
```
plt.figure(figsize=(13, 5))
# Plot
bar = sea.barplot(x='variable', y='value', hue='streamingDiario', data=subset)

# Label
for rect in bar.patches:
    if rect.get_height() == 0: continue
    height = rect.get_height()
    label  = subset.loc[subset['value']==height, 'horas_medias_label'].values[0]
    bar.text(rect.get_x() + rect.get_width() / 2, height, f'{label}', ha='center', va='bottom')

plt.title("Horas medias por dia da semana")

# label
plt.ylabel("Horas medias")
plt.xlabel("Dias da semana")
# Eixo X
plt.yticks([])

# Removendo spines
plt.gca().spines[['top', 'left', 'right']].set_visible(False)

# Colocando legenda
plt.legend(bbox_to_anchor=(0, 0, 1.1, 1), title='Streaming Diario')
plt.show()
```



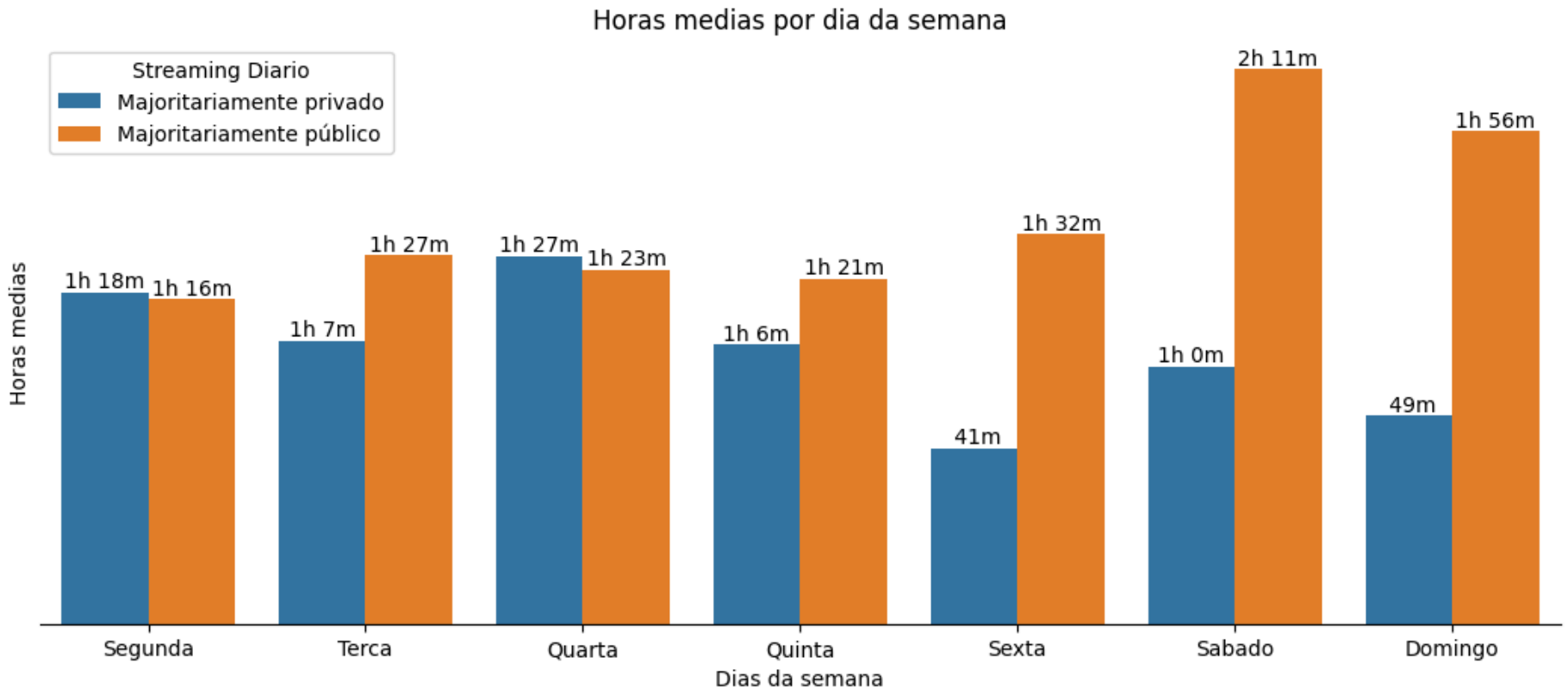
> Não há uma diferença discrepante para os que usando streaming diario ou não

Verificando o média de horas e transporte

In [23]:
```
subset = insta.groupby(['transporte'])[["Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado", "Domingo"]].mean().reset_in
subset = pd.melt(subset, id_vars=['transporte'], value_vars=['Segunda', 'Terca', 'Quarta', 'Quinta', 'Sexta', 'Sabado', 'Domin
```

In [24]:
```
horas = subset['value'].apply(lambda x:timedelta(hours=(x/3600)))
horas = pd.to_timedelta(horas, unit='h').dt.components[['hours',"minutes"]]

subset['horas_medias_label'] = horas\
       .apply(lambda x: (x['hours'].astype(str)+'h' if x['hours']!=0 else '')+' '+ x['minutes'].astype(str)+'m', axis=1)
```

In [25]:
```
plt.figure(figsize=(13, 5))
# Plot
bar = sea.barplot(x='variable', y='value', hue='transporte', data=subset)

# Label
for rect in bar.patches:
    if rect.get_height() == 0: continue
    height = rect.get_height()
    label  = subset.loc[subset['value']==height, 'horas_medias_label'].values[0]
    bar.text(rect.get_x() + rect.get_width() / 2, height, f'{label}', ha='center', va='bottom')

plt.title("Horas medias por dia da semana")

# label
plt.ylabel("Horas medias")
plt.xlabel("Dias da semana")
# Eixo X
plt.yticks([])

# Removendo spines
plt.gca().spines[['top', 'left', 'right']].set_visible(False)

# Colocando legenda
plt.legend(bbox_to_anchor=(0, 0, 1.1, 1), title='Streaming Diario')
plt.show()
```

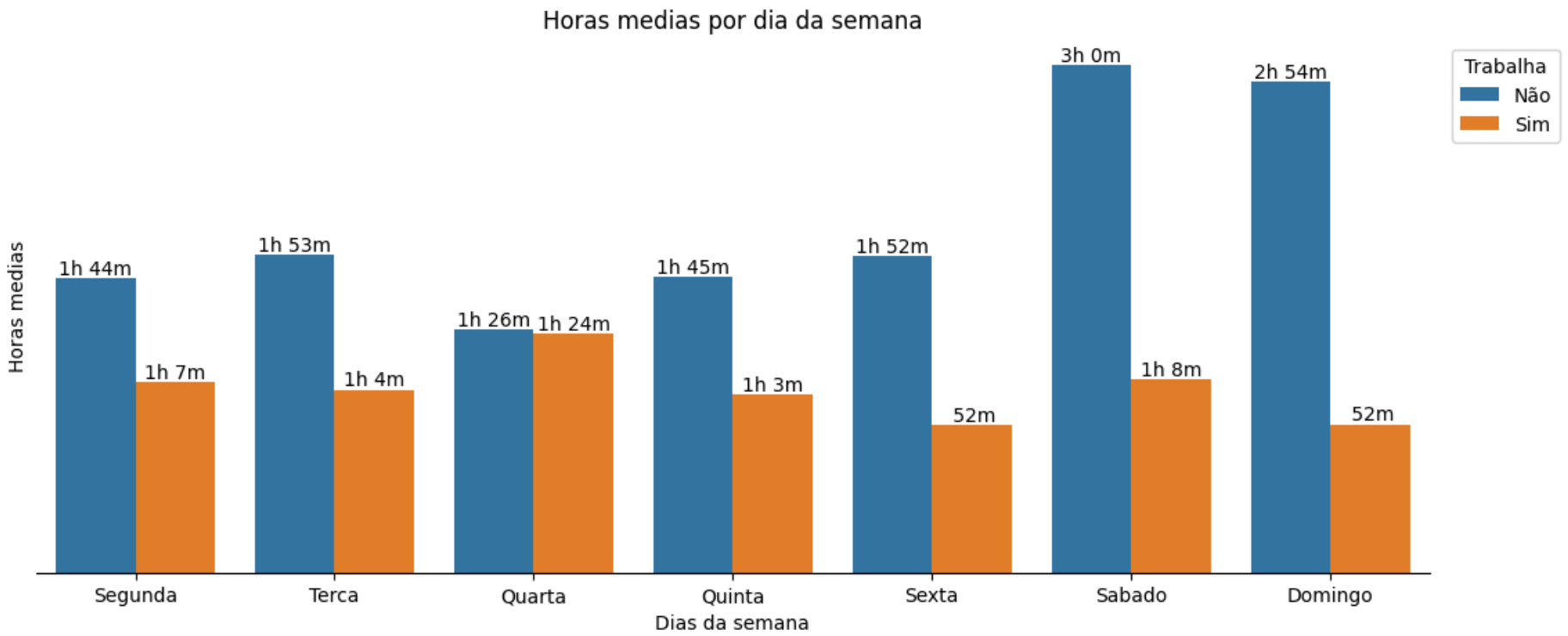


In [26]:
```
subset = insta.groupby(['trabalha'])[["Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado", "Domingo"]].mean().round(2).r
subset = pd.melt(subset, id_vars=['trabalha'], value_vars=['Segunda', 'Terca', 'Quarta', 'Quinta', 'Sexta', 'Sabado', 'Domingo
```

In [27]:
```
horas = subset['value'].apply(lambda x:timedelta(hours=(x/3600)))
horas = pd.to_timedelta(horas, unit='h').dt.components[['hours',"minutes"]]
subset['horas_medias_label'] = horas\
    .apply(lambda x: (x['hours'].astype(str)+'h' if x['hours']!=0 else '')+' '+ x['minutes'].astype(str)+'m', axis=1)
```

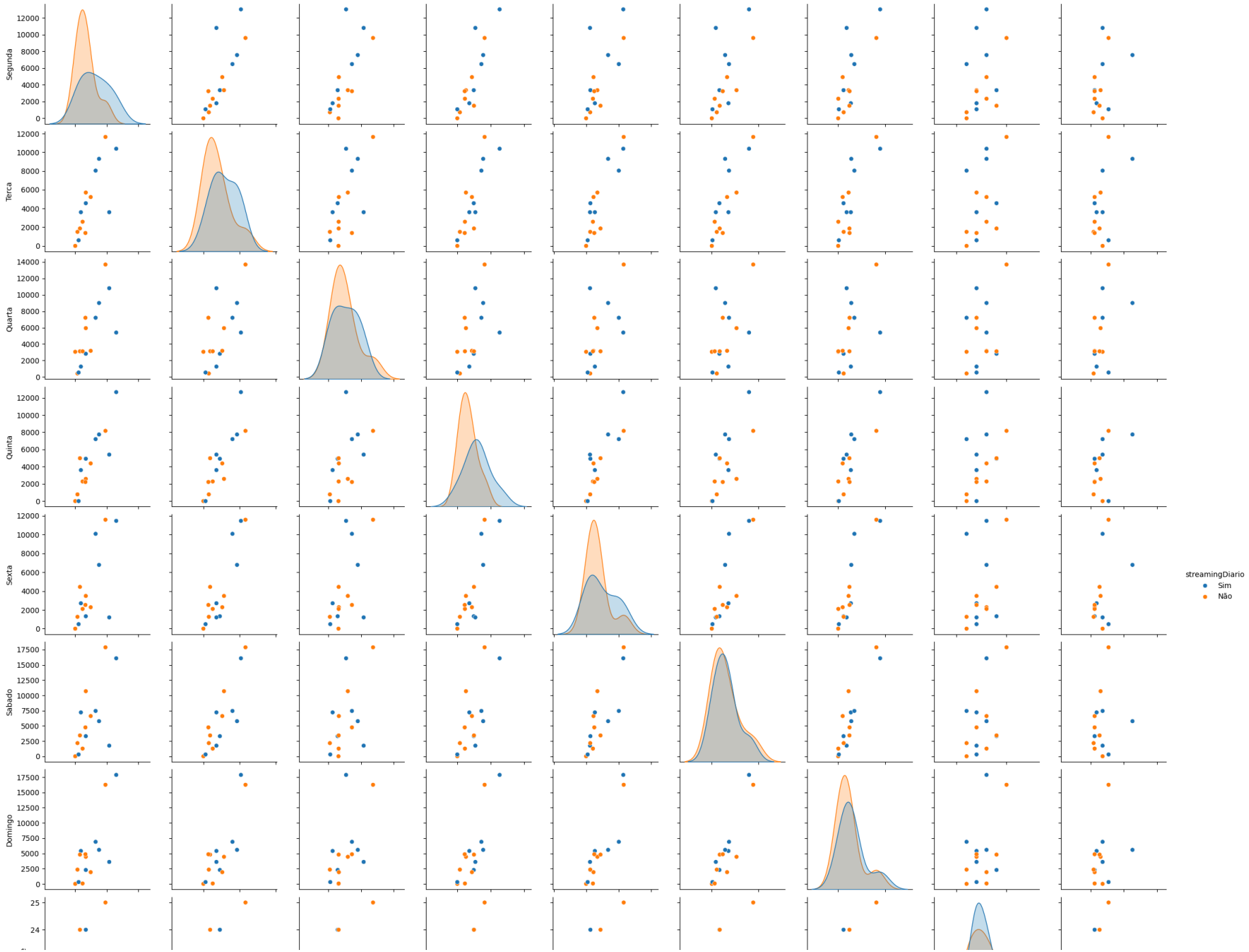
In [28]:
```
plt.figure(figsize=(13, 5))
# Plot
bar = sea.barplot(x='variable', y='value', hue='trabalha', data=subset)

# Label
for rect in bar.patches:
    if rect.get_height() == 0: continue
```

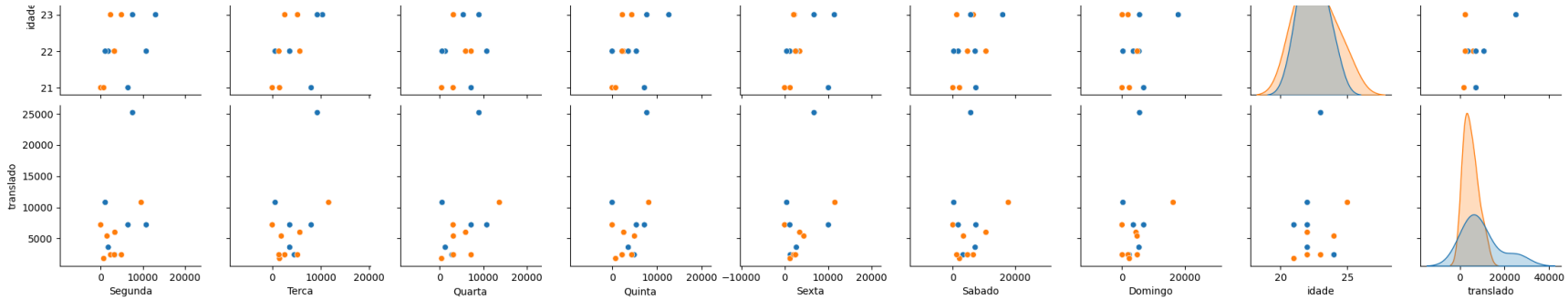
```
    height = rect.get_height()
    label  = subset.loc[subset['value']==height, 'horas_medias_label'].values[0]
    bar.text(rect.get_x() + rect.get_width() / 2, height, f'{label}', ha='center', va='bottom')

plt.title("Horas medias por dia da semana")

# Label
plt.ylabel("Horas medias")
plt.xlabel("Dias da semana")
# Eixo X
plt.yticks([])

# Removendo spines
plt.gca().spines[['top', 'left', 'right']].set_visible(False)

# Colocando legenda
plt.legend(bbox_to_anchor=(0, 0, 1.1, 1), title="Trabalha")
plt.show()
```



In [29]:
```
sea.pairplot(insta, hue='streamingDiario')
```

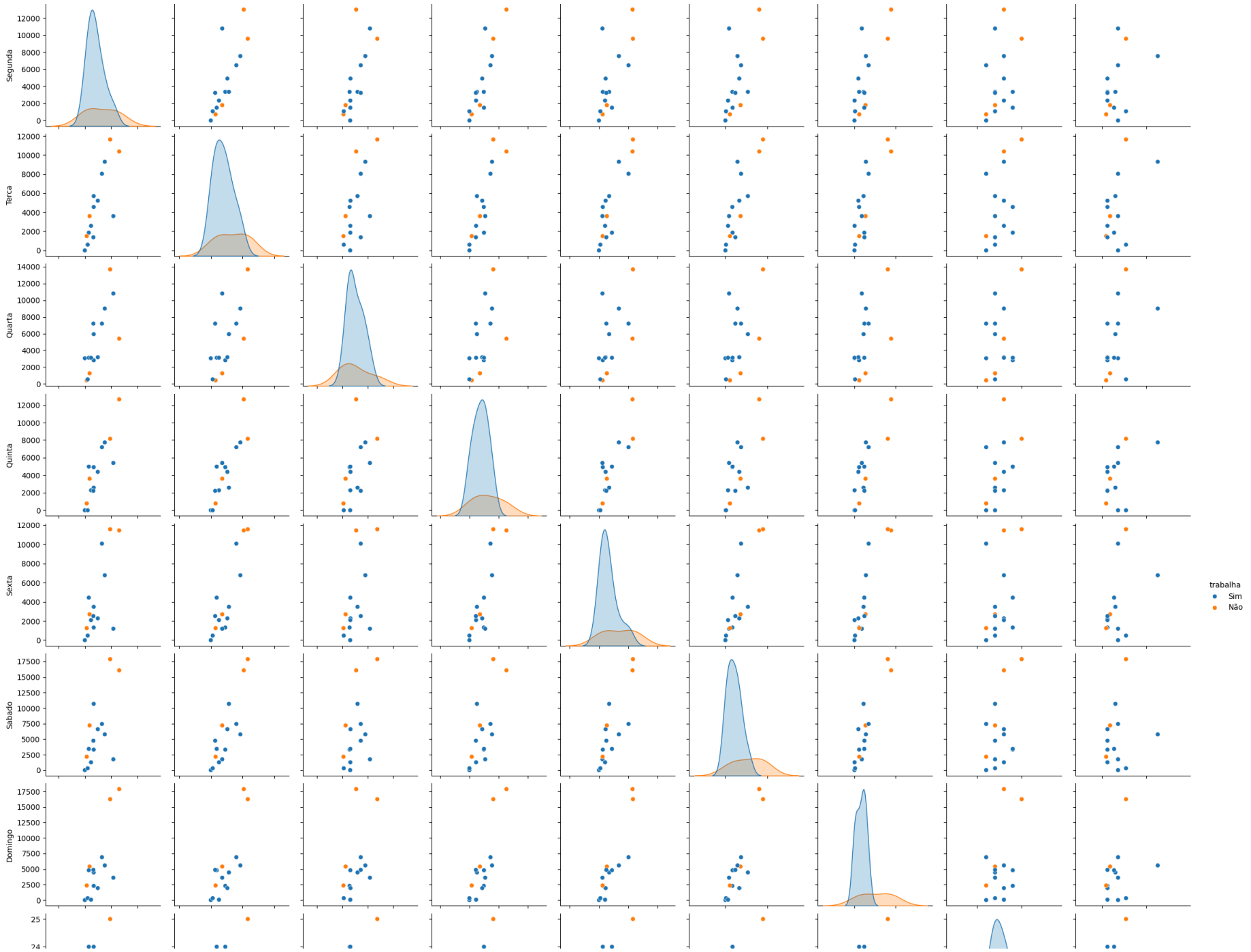
Out[29]:
```
<seaborn.axisgrid.PairGrid at 0x26fc8dbc530>
```

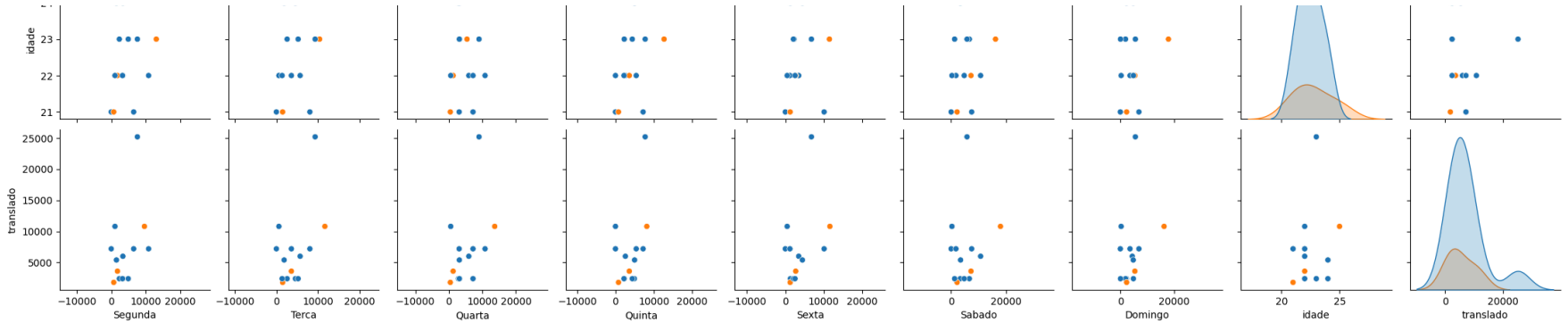

Segunda
Terca
Quarta
Quinta
Sexta
Sabado
Domingo
streamingDiario
Sim
Não



In [30]:
```
sea.pairplot(insta, hue='trabalha')
```

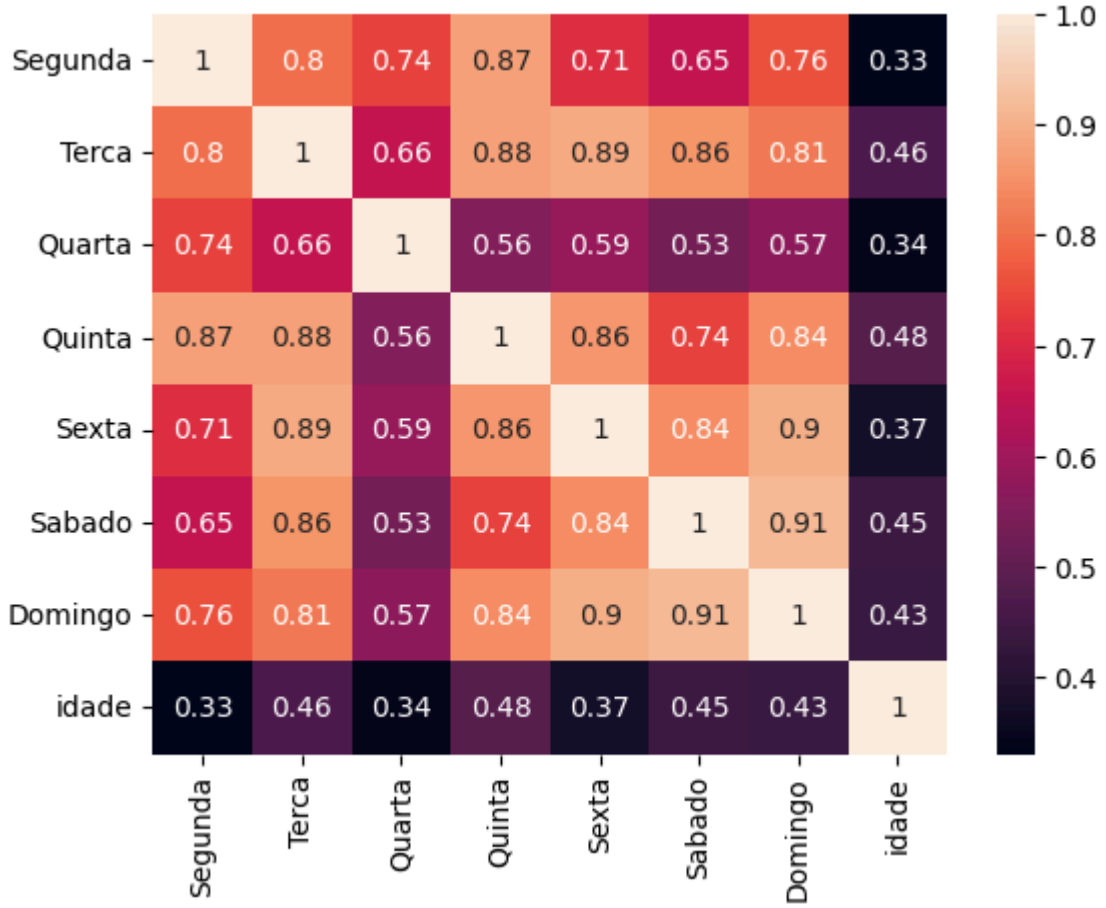
Out[30]: <seaborn.axisgrid.PairGrid at 0x26fc6badfd0>


Segunda
Terca
Quarta
Quinta
Sexta
Sabado
Domingo
trabalha
Sim
Não



In [31]:
```
head_ = insta.select_dtypes(include=['float','int']).corr()

sea.heatmap(head_, annot=True)
```

Out[31]: <Axes: >



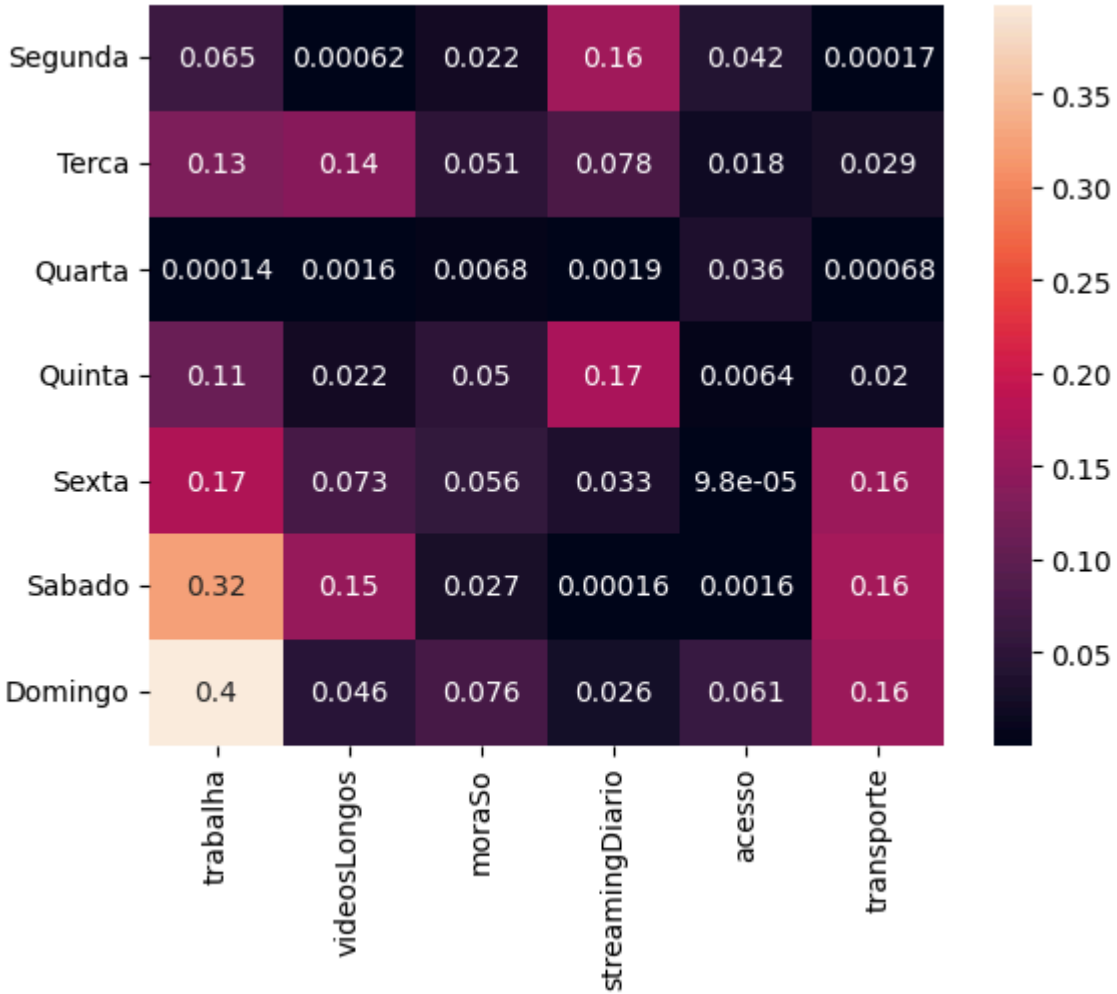
In [32]:
```
from statsmodels.formula.api import ols
from sklearn.preprocessing import LabelEncoder
import statsmodels.api as sm

data_cor = insta.copy()
data_cor['translado'] = data_cor['translado'].fillna(np.mean(data_cor['translado']))

# Peando valores categoricos e numericos
cat = data_cor.select_dtypes(include=['object']).columns
num = data_cor.select_dtypes(include=['float','int']).columns[0:7]
# Dicionario para adcionar a correlação
dicionario = dict()
```

```
# Calculando
for i in cat:
    dicionario[i] = []
    data_cor[i] = LabelEncoder().fit_transform(data_cor[i])


    for j in num:
        formula = f"{j} ~ C({i})"
        model = ols(formula, data=data_cor).fit()

        # Calcula a ANOVA
        anova = sm.stats.anova_lm(model)

        # Calcula o Eta²
        eta_squared = anova['sum_sq'][f'C({i})'] / anova['sum_sq'].sum()

        # Salva o resultado no dicionário
        dicionario[i].append(eta_squared)
```

```
C:\Users\mateu\AppData\Local\Temp\ipykernel_12684\2599589047.py:6: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will change in a future version. Call result.infer_objects(copy=False) instead. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  data_cor['translado'] = data_cor['translado'].fillna(np.mean(data_cor['translado']))
```

In [33]:
```
sea.heatmap(pd.DataFrame(dicionario, index=num), annot=True)
```

Out[33]: <Axes: >



> Mora só não tem correlação alta com as target

Tratamento

In [34]:
```
insta['translado'] = insta['translado'].fillna(np.mean(insta['translado']))
```

```
C:\Users\mateu\AppData\Local\Temp\ipykernel_12684\1095521838.py:1: FutureWarning: Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will change in a future version. Call result.infer_objects(copy=False) instead. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  insta['translado'] = insta['translado'].fillna(np.mean(insta['translado']))
```

In [35]:
```
insta.drop(columns=['acesso', 'moraSo'], inplace=True)
```

Criando modelo

In [36]:
```
dataset = insta.copy()
```

In [37]:
```
import keras
import tensorflow as tf
```

In [38]:
```
from sklearn.preprocessing import LabelEncoder

cat = insta.select_dtypes(include=['object']).columns
for col in cat:
    insta[col] = LabelEncoder().fit_transform(insta[col])

insta_  = pd.get_dummies(insta,columns=cat, dtype=int)
```

In [39]:
```
col = ["Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado", "Domingo"]
X = insta_.drop(columns=col).values.astype(np.float32)
Y = np.sqrt(insta_[col].values).astype(np.float32)
X.shape, Y.shape
```

Out[39]:
```
((15, 10), (15, 7))
```

In [40]:
```
X_ = (X - X.mean(axis=0))/X.std(axis=0)
```

In [41]:
```
data = tf.data.Dataset.from_tensor_slices((X_, Y))
data = data.shuffle(buffer_size=3).batch(batch_size=32).repeat(count=200)

# definindo optimizer loss e metrica
optimizer = keras.optimizers.Adam()
loss      = keras.losses.MeanSquaredError()
```

```
model     = keras.models.Sequential(
                    layers=[
                        keras.layers.Dense(14, activation=keras.activations.leaky_relu, kernel_initializer=keras.initializers.
                        keras.layers.Dense(7)
            ])
# treinado
for enum, (x, y) in enumerate(data):
    with tf.GradientTape() as tape:
        pred = model(x,  training=True)
        loss_ = loss(y, pred)
        loss_ = tf.add_n([loss_], model.losses)
    grad = tape.gradient(loss_, model.variables)
    optimizer.apply_gradients(zip(grad, model.variables))

    if enum % 100 == 0:
        print(loss_)
```

```
tf.Tensor(4824.7275, shape=(), dtype=float32)
tf.Tensor(4653.02, shape=(), dtype=float32)
```

In [42]:
```
pred = model(X)
pred = (y - pred).numpy()
```

In [43]:
```
from sklearn.metrics import mean_squared_error, r2_score
for i in range(7):
    print(col[i], mean_squared_error(pred[:, i], Y[:, i]), r2_score(pred[:, i], Y[:, i]))
```

```
Segunda 362797020.0 -1.4264318943023682
Terca 151056800.0 -1.4277372360229492
Quarta 54216948.0 -1.4308154582977295
Quinta 95307256.0 -1.4218645095825195
Sexta 441224100.0 -1.4247488975524902
Sabado 574507800.0 -1.4173550605773926
Domingo 164934740.0 -1.4230809211730957
```

In [44]:
```
from sklearn.ensemble import RandomForestRegressor

for i in range(7):
    model = RandomForestRegressor(random_state=32)
    model.fit(X, Y[:, i])
```

```
    pred = model.predict(X)
    print(col[i], "\nMSE :",round(mean_squared_error(pred, Y[:, i]),2), "R2",round(r2_score(pred, Y[:, i]), 2))
```

```
Segunda
MSE : 204.92 R2 0.39
Terca
MSE : 220.68 R2 0.16
Quarta
MSE : 166.44 R2 0.31
Quinta
MSE : 196.58 R2 0.45
Sexta
MSE : 210.83 R2 0.34
Sabado
MSE : 321.99 R2 0.3
Domingo
MSE : 275.56 R2 0.5
```

> Resultados significantes para a arvore aletaria , mas será os parametros são estatisticamente significantes?

In [45]:
```
from statsmodels.regression.linear_model import OLS

for i in range(7):
    model = OLS(Y[:, i], X)
    model = model.fit()
    if  np.all(model.pvalues>0.05):
        print("todos parametros não estatisticamente significante")
    else:
        print(col[i], "\nMSE :",round(mean_squared_error(pred, Y[:, i]),2), "R2",round(r2_score(pred, Y[:, i]), 2))
```

```
todos parametros não estatisticamente significante
todos parametros não estatisticamente significante
todos parametros não estatisticamente significante
todos parametros não estatisticamente significante
todos parametros não estatisticamente significante
todos parametros não estatisticamente significante
todos parametros não estatisticamente significante
```

> Todos os parametros não são estatisticamente significante

Conclusão

- baixa qualidade dos dados impossibilita predição e bons resultados

Quais hipóteses iniciais sobre o uso do aplicativo você formulou e quais foram verificadas?

> Nenhuma, os dados são de baixa qualidade assim impossibilitando uma analise minuciosa e criação de uma modelo bom

Usando `streamingDierio` como target

In [46]:
```
from sklearn.preprocessing import LabelEncoder

cat = dataset.drop(columns=['streamingDiario'])\
    .select_dtypes(include=['object']).columns
for col in cat:
    insta[col] = LabelEncoder().fit_transform(insta[col])

insta_  = pd.get_dummies(insta,columns=cat, dtype=int)
```

In [47]:
```
# import GLM familia poison no statsmodels
from statsmodels.api import add_constant, GLM, families

y = insta['streamingDiario']
x = insta_.drop(columns=['streamingDiario'])
```

In [48]:
```
X = add_constant(x)
```

In [49]:
```
# Define o modelo GLM com distribuição binomial (logística)
model = GLM(y, X, family=families.NegativeBinomial(alpha=10))

# Ajusta o modelo
```

```
result = model.fit()

# Exibe o resumo dos resultados
print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:        streamingDiario   No. Observations:                   15
Model:                            GLM   Df Residuals:                        2
Model Family:        NegativeBinomial   Df Model:                           12
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -18.464
Date:                Sat, 22 Feb 2025   Deviance:                   3.1618e-09
Time:                        13:09:02   Pearson chi2:                 1.58e-09
No. Iterations:                     22   Pseudo R-squ. (CS):             0.1938
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             36.9023   2.98e+05      0.000      1.000   -5.84e+05    5.84e+05
Segunda           -0.0018     23.157  -7.98e-05      1.000     -45.388      45.385
Terca              0.0195     28.087      0.001      0.999     -55.031      55.070
Quarta            -0.0015      4.057     -0.000      1.000      -7.953       7.950
Quinta             0.0016     11.382      0.000      1.000     -22.307      22.311
Sexta             -0.0089     22.923     -0.000      1.000     -44.936      44.918
Sabado            -0.0079     10.871     -0.001      0.999     -21.314      21.298
Domingo            0.0030     25.799      0.000      1.000     -50.562      50.568
idade             -4.6960   3.49e+04     -0.000      1.000   -6.83e+04    6.83e+04
translado         -0.0011      1.822     -0.001      1.000      -3.573       3.570
trabalha_0        24.3244   1.16e+05      0.000      1.000   -2.27e+05    2.27e+05
trabalha_1        12.5780   1.83e+05   6.88e-05      1.000   -3.59e+05    3.59e+05
videosLongos_0    29.7556   1.82e+05      0.000      1.000   -3.56e+05    3.56e+05
videosLongos_1     7.1467   1.18e+05   6.07e-05      1.000   -2.31e+05    2.31e+05
transporte_0       3.6314   1.67e+05   2.17e-05      1.000   -3.28e+05    3.28e+05
transporte_1      33.2709   1.32e+05      0.000      1.000    -2.6e+05     2.6e+05
==============================================================================
```

```
c:\Users\mateu\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels\genmod\generalized_linear_model.py:1342: P
erfectSeparationWarning: Perfect separation or prediction detected, parameter may not be identified
  warnings.warn(msg, category=PerfectSeparationWarning)
c:\Users\mateu\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels\genmod\generalized_linear_model.py:1342: P
erfectSeparationWarning: Perfect separation or prediction detected, parameter may not be identified
  warnings.warn(msg, category=PerfectSeparationWarning)
c:\Users\mateu\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels\genmod\generalized_linear_model.py:1342: P
erfectSeparationWarning: Perfect separation or prediction detected, parameter may not be identified
  warnings.warn(msg, category=PerfectSeparationWarning)
c:\Users\mateu\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels\genmod\generalized_linear_model.py:1342: P
erfectSeparationWarning: Perfect separation or prediction detected, parameter may not be identified
  warnings.warn(msg, category=PerfectSeparationWarning)
```

> Pessimos resultados