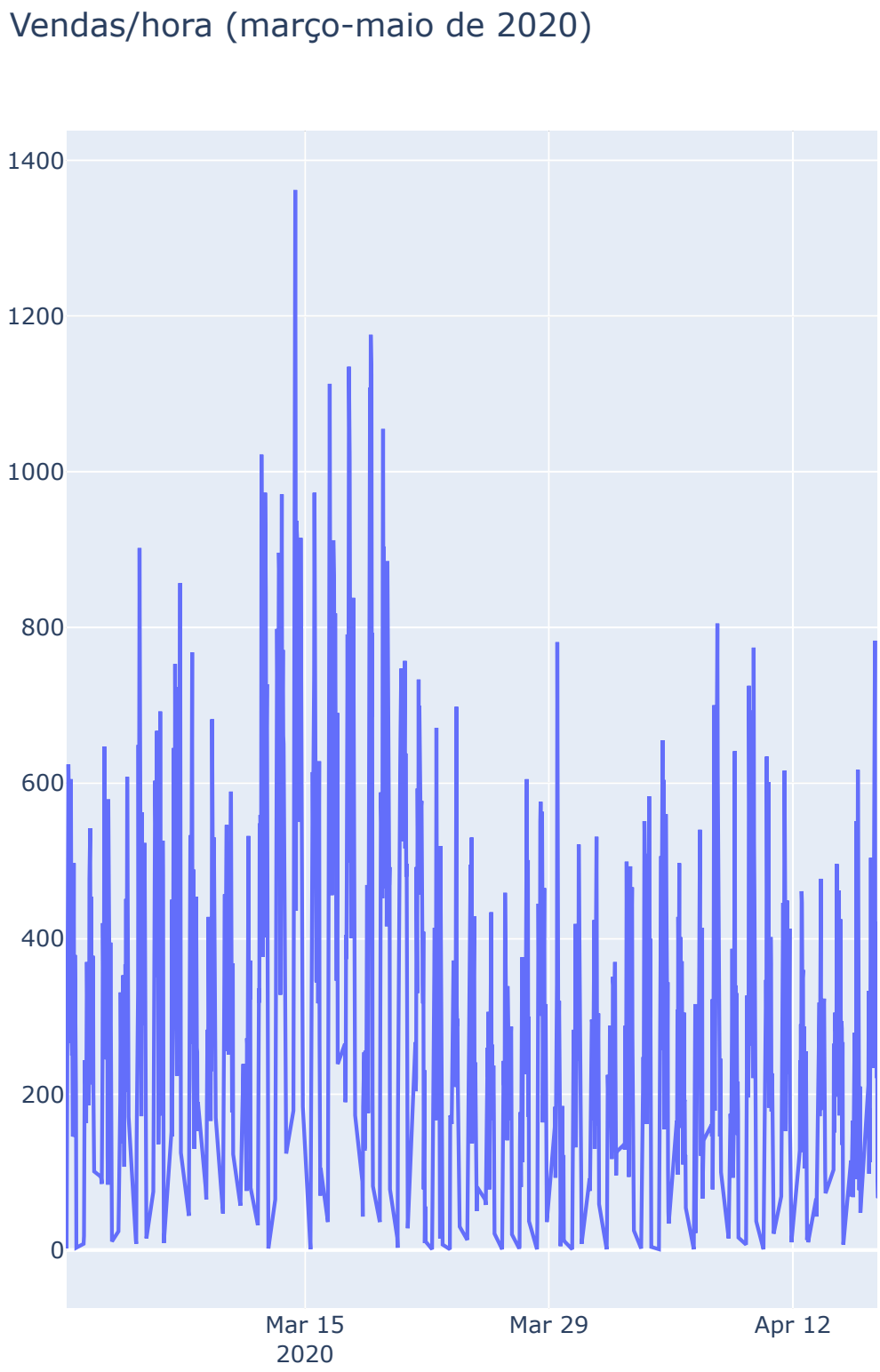
In [1]:
```
import datacleaner
import config
import os
import sys
import pandas as pd
import numpy as np
from fbprophet import Prophet
from fbprophet.plot import add_changepoints_to_plot
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
# Offline mode
from plotly.offline import init_notebook_mode, iplot
from scipy.stats import boxcox
from scipy.special import inv_boxcox
from statsmodels.tsa.seasonal import seasonal_decompose
from datetime import datetime

# "high resolution"
%config InlineBackend.figure_format = 'retina'
init_notebook_mode(connected=True)
```

In [2]:
```
## Cria 3 dataframes:
## df_274_time_sale = quantidade de produtos vendidos na loj
## df_274_time_cli = quantidade de clientes na loja por hora

#df_complete = datacleaner.get_Dataframes(None,None,full_dat
#df_complete.reset_index(inplace=True)

df_274_time_sale = datacleaner.get_Dataframes_time(274,'s')
df_274_time_cli = datacleaner.get_Dataframes_time(274,'c')
df_274_time_sale.reset_index(inplace=True)
df_274_time_cli.reset_index(inplace=True)

## Cria dois dataframes:
## df_274_sale = quantidade de produtos vendidos na loja por
## df_274_cli  = quantidade de clientes na loja por data

df_274_sale = datacleaner.get_Dataframes(274,'s')
df_274_sale.reset_index(inplace=True)

sns.set(style="darkgrid")
sns.set_context("notebook", font_scale=1.5, rc={"lines.linew
```

Quantidade de produtos vendidos/hora (l
274)

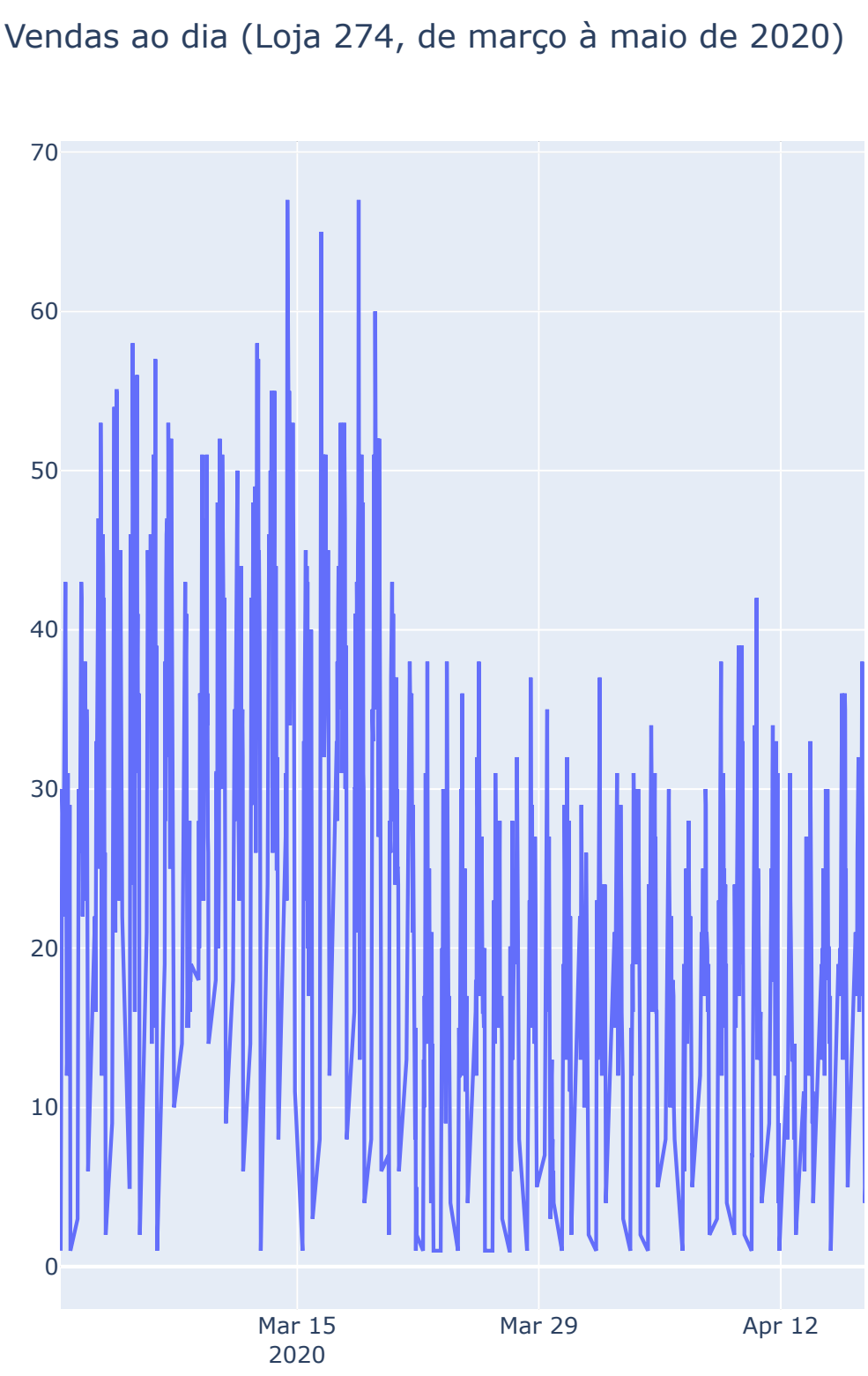
In [3]:
```
sales_data_l274 = go.Scatter(x=df_274_time_sale.ds,y=df_274_
layout = go.Layout(height=800,
                   width=1000,
                   title='Vendas/hora (março-maio de 2020)')
fig_sales_hour = go.Figure(data=[sales_data_l274],layout=lay
iplot(fig_sales_hour)
```



Analisando o gráfico acima, observamos visualmente uma sazonalidade pelos picos de alto e baixo no gráfico.

Quantidade de clientes/hora (loja 274)

In [4]:
```
cli_data_l274 = go.Scatter(x=df_274_time_cli.ds,y=df_274_tim
layout = go.Layout(height=800,
                   width=1000,
                   title='Vendas ao dia (Loja 274, de março
fig_cli = go.Figure(data=[cli_data_l274],layout=layout)
iplot(fig_cli)
```

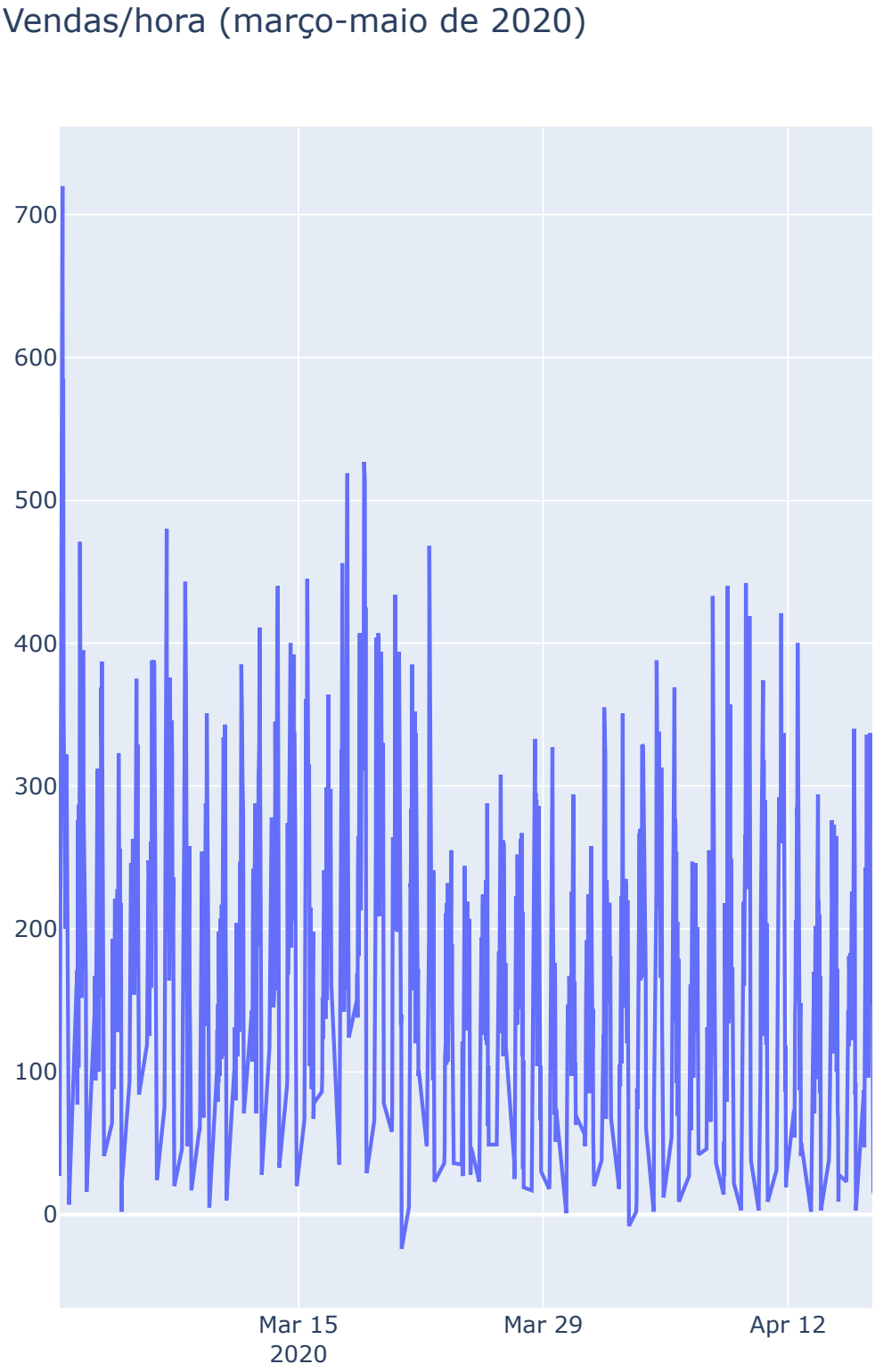


O gráfico de quantidade de clientes por hora na loja 274 apresenta semelhanças com o gráfico de quantidade de produto hora na mesma loja logo, inferimos que o comportamento da sé sazonal como a série anterior.

Loja 432

In [5]:
```
df_432_time_sale = datacleaner.get_Dataframes_time(432,'s')
df_432_time_cli = datacleaner.get_Dataframes_time(432,'c')
df_432_time_sale.reset_index(inplace=True)
df_432_time_cli.reset_index(inplace=True)
```

In [6]:
```
sales_data_l432 = go.Scatter(x=df_432_time_sale.ds,y=df_432_
layout = go.Layout(height=800,
                   width=1000,
                   title='Vendas/hora (março-maio de 2020)')
fig_sales_hour_432 = go.Figure(data=[sales_data_l432],layout=
iplot(fig_sales_hour_432)
```



Analisando o gráfico acima, também observamos visualmente o tipo de sazonalidade que os gráficos anteriores. Além da sazonalidade também observamos alguns dados negativos.

In [7]:
```
df_432_time_sale.loc[df_432_time_sale['y'] < 0]
```

Out[7]:

	ds	y
548	2020-03-20 21:30:00	-24
890	2020-04-02 21:30:00	-8
1444	2020-04-23 21:00:00	-2
2141	2020-05-20 22:00:00	-3
2170	2020-05-21 22:00:00	-6
2292	2020-05-26 21:00:00	-16

1. Em 20-03 houve -24 produtos vendidos e -1 cliente na loja
2. Em 02-04 houve -8 produtos vendidos e -1 cliente na loja
3. Em 23-04 houve -2 produtos vendidos e 0 cliente na loja
4. Em 20-05 houve -3 produtos vendidos e -1 cliente na loja
5. Em 21-05 houve -6 produtos vendidos e -1 cliente na loja
6. Em 26-05 houve -16 produtos vendidos e 1 cliente na loja

Nesse caso notamos que esse comportamento ocorre sempre após 21 horas, aparentemente próximo do horário de encerramento d atendimento. Resta a dúvida sobre qual motivo leva os dados terem esses valores Verificamos que esse comportamento não s repete na loja 274.

In [8]:
```
df_274_time_sale.loc[df_274_time_sale['y'] < 0]
```

Out[8]:

	ds	y

Removendo os valores negativos do dataframe de vendas da loj

In [9]:
```
df_432_time_sale_val_positivo = df_432_time_sale.copy()
df_432_time_sale_val_positivo = df_432_time_sale_val_positiv
df_432_time_sale_val_positivo
```

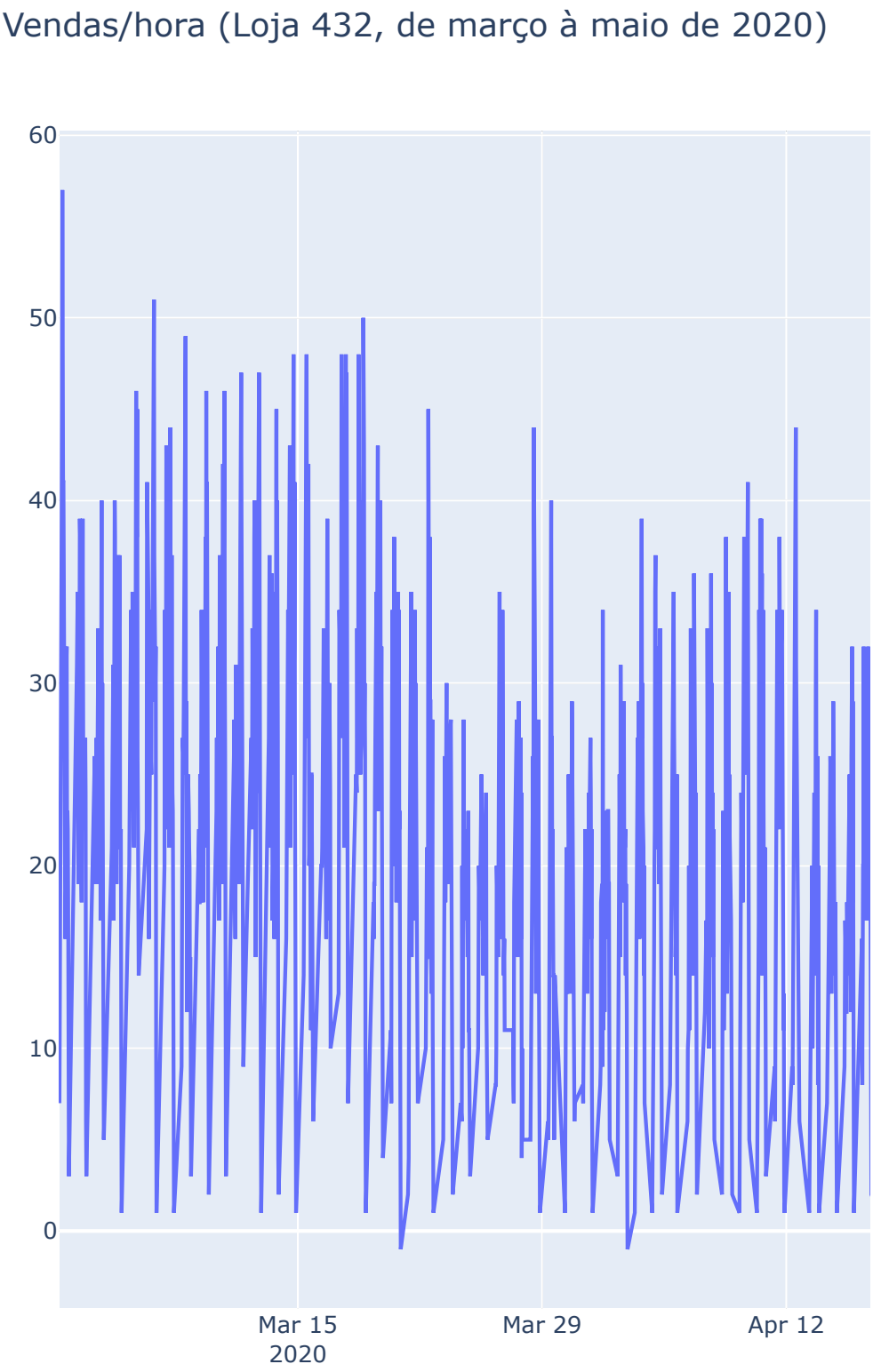
Out[9]:

	ds	y
0	2020-03-01 08:00:00	27
1	2020-03-01 08:30:00	170
2	2020-03-01 09:00:00	285
3	2020-03-01 09:30:00	148
4	2020-03-01 10:00:00	333
...	...	...
2423	2020-05-31 16:30:00	81
2424	2020-05-31 17:00:00	184
2425	2020-05-31 17:30:00	144
2426	2020-05-31 18:00:00	29
2427	2020-05-31 18:30:00	56

2421 rows × 2 columns

Quantidade de clientes/dia (loja 432)

In [10]:
```
cli_data_l432 = go.Scatter(x=df_432_time_cli.ds,y=df_432_tim
layout = go.Layout(height=800,
                   width=1000,
                   title='Vendas/hora (Loja 432, de março à
fig_cli_432 = go.Figure(data=[cli_data_l432],layout=layout)
iplot(fig_cli_432)
```



O gráfico de quantidade de clientes por hora na loja 432 apresenta semelhanças com o gráfico de quantidade de produto hora na mesma loja. Também notamos nessa loja momentos em qu número de clientes foi negativo:

In [11]:
```
df_432_time_cli.loc[df_432_time_cli['y'] < 0]
```

Out[11]:

	ds	y
548	2020-03-20 21:30:00	-1
890	2020-04-02 21:30:00	-1
1320	2020-04-19 09:00:00	-1
2141	2020-05-20 22:00:00	-1
2170	2020-05-21 22:00:00	-1

In [12]:
```
df_432_time_cli_val_positivo = df_432_time_cli.copy()
df_432_time_cli_val_positivo=df_432_time_cli_val_positivo.lo
```

Na linha acima removemos os valores negativos do dataframe.
Dúvida: Os valores negativos verificados provocam as seguintes dúvidas:

- Há necessidade de normalizar os dados por esse motivo?
- Não deveríamos questionar à Sisqual se sabem sobre essa situação e porqu ocorre dessa forma?

Seasonality

O Prophet, para realizar a previsão, utiliza dois modelos Aditivo e Multiplicativo e
ser determinados pelos parâmetros. O modelo aditivo sugere que os componente
adicionados conforme a equação abaixo:

$$y(t) = Level + Trend + Seasonality + Noise$$

O modelo multiplicativo sugere que os componentes são multiplicados:

$$y(t) = Level + Trend + Seasonality + Noise$$

Onde:

- **Level** = Valores médios da série.
- **Trend** = Aumento ou a redução dos valores da série.
- **Seasonality** = Ciclo de repetições da série.
- **Noise** = Variações randômicas da série.

O Prophet decompõe o seu modelo em trend, seasonality (semanal, mensal, diári
holidays (feriados).

Para isso é preciso primeiro criar o modelo no Prophet. Faze isso preenchendo os seguintes parâmetros:

- Parâmetro growth (saturação)

> Quando a previsão cresce, alguns pontos atingem o valor máximo possível, como um teto, isso é chamado de carrying capacity. Por essa razão deve-se saturar o "growth" da previsão para que o modelo comporte essas variações. É possível definir o carrying capacity (cap) adicionando esse valor como uma coluna do dataframe.

Por padrão o Prophet usa um modelo de saturação linear. Utilizaremos nesse caso o modelo logistico.

- Parâmetro interval_width

> O interval_width de confiança = 95%; isso define o intervalo de incerteza para produzir um intervalo de confiança em torno do valor previsto.

- Parâmetros Trend Changepoints

> Para os dados que estamos a analisar, os *changepoints* representam a data, momento ou índice de tempo que define um ponto quando uma data começa a mudar sua direção, quer seja crescente ou decrescente, seria uma espécie de ponto de inflexão. Prophet usa um entre dois métodos para definir a tendência de *changepoints* Especificar a flexibilidade das tendências dos *changepoints* Especificar o local dos *changepoints*, significa definir os *changepoints* no próprio dataframe como uma série (quando a tendência começa a mudar). No nosso caso vamos utilizar a flexibilidade de tendência, ou seja o modelo será flexível a tendência.

- Parâmetro changepoint_prior_scale

> Representa o quão flexível o modelo irá se comportar contra os *trend changepoints*

- Parâmetros de sazonalidade

> Definem o período de sazonalidade a ser obtido pelo prophet. Em nosso caso desejamos que o algoritmo considere sazonalidades (a frequência de repetição de um evento) anual e mensal.

- Parâmetro holiday

> Os feriados e eventos influenciam no comportamento de compra e deslocamento das pessoas. Para esse estudo foram adicionados os feriados nacionais conhecidos, entretanto, cabe destacar que podem existir eventos ou feriados municipais que impactam o comércio local onde as lojas se encontram.

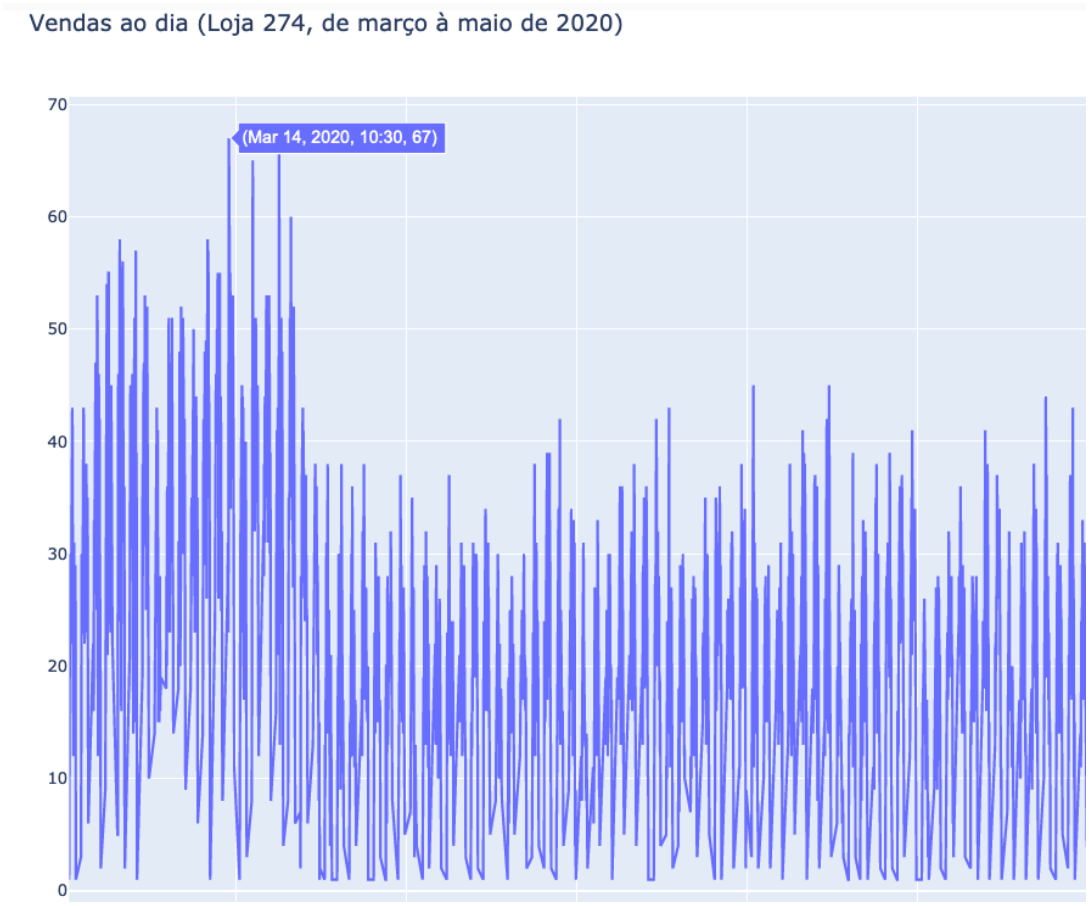
In [13]:
```
# definindo o cap (carrying capacity)
df_274_time_sale['cap'] = 1362
df_274_time_sale.reset_index(drop=True)
```

Out[13]:

	ds	y	cap
0	2020-03-01 07:30:00	2	1362
1	2020-03-01 08:00:00	144	1362
2	2020-03-01 08:30:00	117	1362
3	2020-03-01 09:00:00	65	1362
4	2020-03-01 09:30:00	271	1362
...	...	...	...
2550	2020-05-31 18:30:00	268	1362
2551	2020-05-31 19:00:00	173	1362
2552	2020-05-31 19:30:00	136	1362
2553	2020-05-31 20:00:00	23	1362
2554	2020-05-31 20:30:00	80	1362

2555 rows × 3 columns

O valor cap gerado teve por base o máximo valor atingido no período da série



Criação do dataframe para futuro

- Identificando os valores criados fora do range de atendimento da loja
- Frequência horária
- Cada 30 minutos de cada hora
- Carry Capacity (teto) = 1362 (valor carece de validação)
  - Carry Capacity (base) = Não utilizado

In [14]:
```
m_274_time_sale = Prophet(growth='logistic',
                          interval_width=0.95,
                          changepoint_prior_scale=0.095,
                          yearly_seasonality=True,
                          weekly_seasonality=True,
                          holidays=datacleaner.get_Holiday()
m_274_time_sale.add_country_holidays(country_name='BR')
m_274_time_sale.fit(df_274_time_sale)
```

Out[14]:
```
<fbprophet.forecaster.Prophet at 0x125e7b940>
```

In [15]:
```
m_274_time_sale_linear = Prophet(growth='linear',
                                 interval_width=0.95,
                                 changepoint_prior_scale=0.095,
                                 yearly_seasonality=True,
                                 weekly_seasonality=True,
                                 holidays=datacleaner.get_Holiday()
m_274_time_sale_linear.add_country_holidays(country_name='BR
m_274_time_sale_linear.fit(df_274_time_sale)
```

Out[15]:
```
<fbprophet.forecaster.Prophet at 0x1270484f0>
```

Criar um modelo com saturação aditiva para posterior compara

In [16]:
```
m_274_time_sale.train_holiday_names
```

Out[16]:
```
0                          parana
1                New Year's Day
2                    Tiradentes
3                  Worker's Day
4              Independence Day
5    Our Lady of the Apparition
6                All Souls' Day
7     Republic Proclamation Day
8                      Christmas
dtype: object
```

In [17]:
```
future_274_time_sale = m_274_time_sale.make_future_dataframe
future_274_time_sale['cap'] = 1362
teste_future_274_time_sale = future_274_time_sale.loc[future_
teste_future_274_time_sale
```

Out[17]:

	ds	cap
2555	2020-06-01 20:30:00	1362
2556	2020-06-02 20:30:00	1362
2557	2020-06-03 20:30:00	1362
2558	2020-06-04 20:30:00	1362
2559	2020-06-05 20:30:00	1362
...	...	...
4350	2025-05-01 20:30:00	1362
4351	2025-05-02 20:30:00	1362
4352	2025-05-03 20:30:00	1362
4353	2025-05-04 20:30:00	1362
4354	2025-05-05 20:30:00	1362

Criando o dataframe de previsão futura para uma frequência d
e um período de 5 anos

In [18]:
```
#future_274_time_sale['ds'] = pd.to_datetime(future_274_time
#future_274_time_sale=future_274_time_sale.set_index(pd.Date
#future_274_time_sale.loc[future_274_time_sale['ds'].between
```

O prophet está projetando dados para horários em que a loja possui dados, isto é, após 20:30 até 07:00, como não há dado Prophet cria dados negativos que podem impactar nas análises Dessa forma para evitar essa ocorrência foi necessário remov dataframe, projetado no futuro, esses valores. Para fazer is um tratamento nos dados importante, pois constatamos que provavelmente, não existe um horário fixo para a abertura e encerramento da loja Existem dias em que a loja teve cliente produtos vendidos a partir das 06:00 da manhã e dias em que encerramento ocorreu as 20:30, 21:00, 21:30 e 22:00.

In [19]:
```
df_sale_test = df_274_time_sale.copy()
df_sale_test['ds'] = pd.to_datetime(df_sale_test['ds'])
df_sale_test = df_sale_test.set_index(pd.DatetimeIndex(df_sa
df_sale_test.loc[df_sale_test['ds'].between_time('06:00','07
```

Out[19]:

	ds	y	cap
ds			
2020-03-02 07:00:00	2020-03-02 07:00:00	8	1362
2020-03-03 07:00:00	2020-03-03 07:00:00	94	1362
2020-03-04 07:00:00	2020-03-04 07:00:00	24	1362
2020-03-05 07:00:00	2020-03-05 07:00:00	12	1362
2020-03-06 07:00:00	2020-03-06 07:00:00	75	1362
2020-03-09 07:00:00	2020-03-09 07:00:00	81	1362
2020-03-10 07:00:00	2020-03-10 07:00:00	47	1362
2020-03-11 07:00:00	2020-03-11 07:00:00	57	1362
2020-03-12 07:00:00	2020-03-12 07:00:00	32	1362
2020-03-13 07:00:00	2020-03-13 07:00:00	65	1362
2020-03-17 07:00:00	2020-03-17 07:00:00	265	1362
2020-03-18 07:00:00	2020-03-18 07:00:00	88	1362
2020-03-19 07:00:00	2020-03-19 07:00:00	36	1362
2020-03-20 07:00:00	2020-03-20 07:00:00	16	1362
2020-03-22 07:00:00	2020-03-22 07:00:00	1	1362
2020-03-27 07:00:00	2020-03-27 07:00:00	2	1362
2020-04-03 07:00:00	2020-04-03 07:00:00	2	1362
2020-05-04 06:30:00	2020-05-04 06:30:00	2	1362
2020-05-05 06:00:00	2020-05-05 06:00:00	1	1362
2020-05-08 07:00:00	2020-05-08 07:00:00	1	1362
2020-05-12 06:30:00	2020-05-12 06:30:00	4	1362
2020-05-29 06:30:00	2020-05-29 06:30:00	1	1362
2020-05-30 06:00:00	2020-05-30 06:00:00	10	1362

In [20]:
```
df_sale_test.between_time('00:00','05:00')
```

Out[20]:

	ds	y	cap
ds			

In [21]:
```
future_274_time_sale_adjusted = future_274_time_sale.copy()
future_274_time_sale_adjusted['ds'] = pd.to_datetime(future_
future_274_time_sale_adjusted = future_274_time_sale_adjuste
#future_274_time_sale_adjusted = future_274_time_sale_adjust
#future_274_time_sale_adjusted.reset_index(drop=True)
future_274_time_sale_adjusted.between_time('00:00','05:00')
```

Out[21]:

	ds	cap
ds		

In [24]:
```
future_274_time_sale_adjusted = future_274_time_sale.copy()
future_274_time_sale_adjusted['ds'] = pd.to_datetime(future_
future_274_time_sale_adjusted = future_274_time_sale_adjuste
future_274_time_sale_adjusted = future_274_time_sale_adjuste
#future_274_time_sale_adjusted.reset_index(drop=True)
future_274_time_sale_adjusted
```

Out[24]:

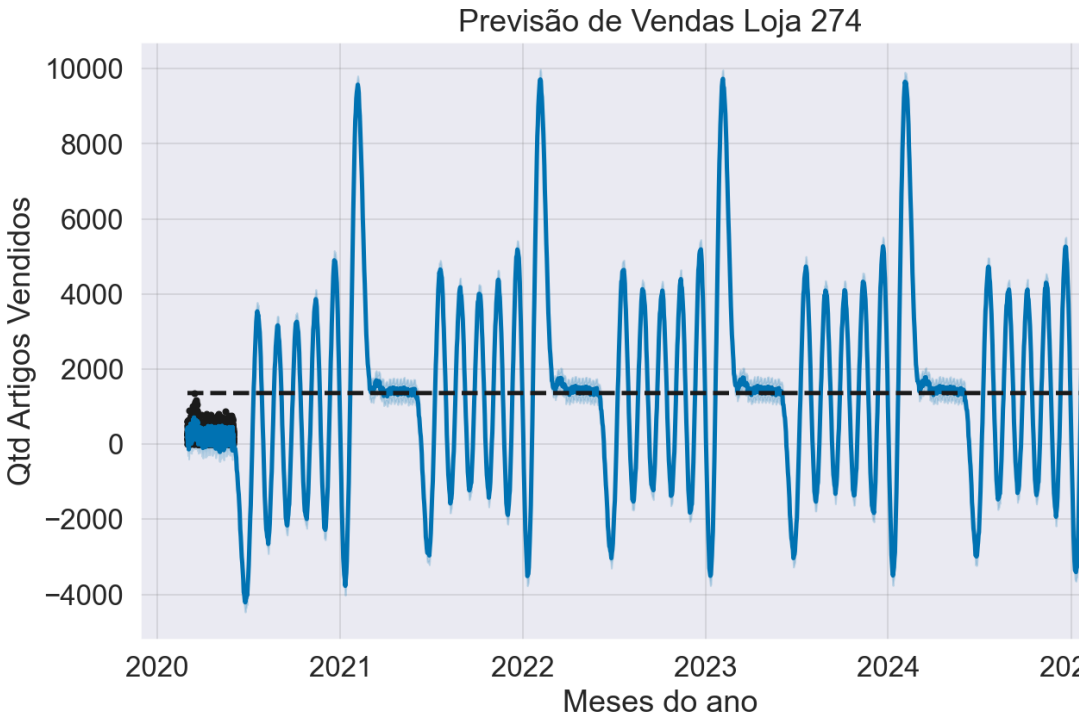
	ds	cap
ds		
2020-03-01 07:30:00	2020-03-01 07:30:00	1362
2020-03-01 08:00:00	2020-03-01 08:00:00	1362
2020-03-01 08:30:00	2020-03-01 08:30:00	1362
2020-03-01 09:00:00	2020-03-01 09:00:00	1362
2020-03-01 09:30:00	2020-03-01 09:30:00	1362
...	...	...
2025-05-01 20:30:00	2025-05-01 20:30:00	1362
2025-05-02 20:30:00	2025-05-02 20:30:00	1362
2025-05-03 20:30:00	2025-05-03 20:30:00	1362
2025-05-04 20:30:00	2025-05-04 20:30:00	1362
2025-05-05 20:30:00	2025-05-05 20:30:00	1362

4355 rows × 2 columns

O dataframe projetado para 5 anos no futuro. O parâmetro uncertainty=True assume uma média da frequência e magnitude da tendência verificada no pa que seja a mesma para o futuro.

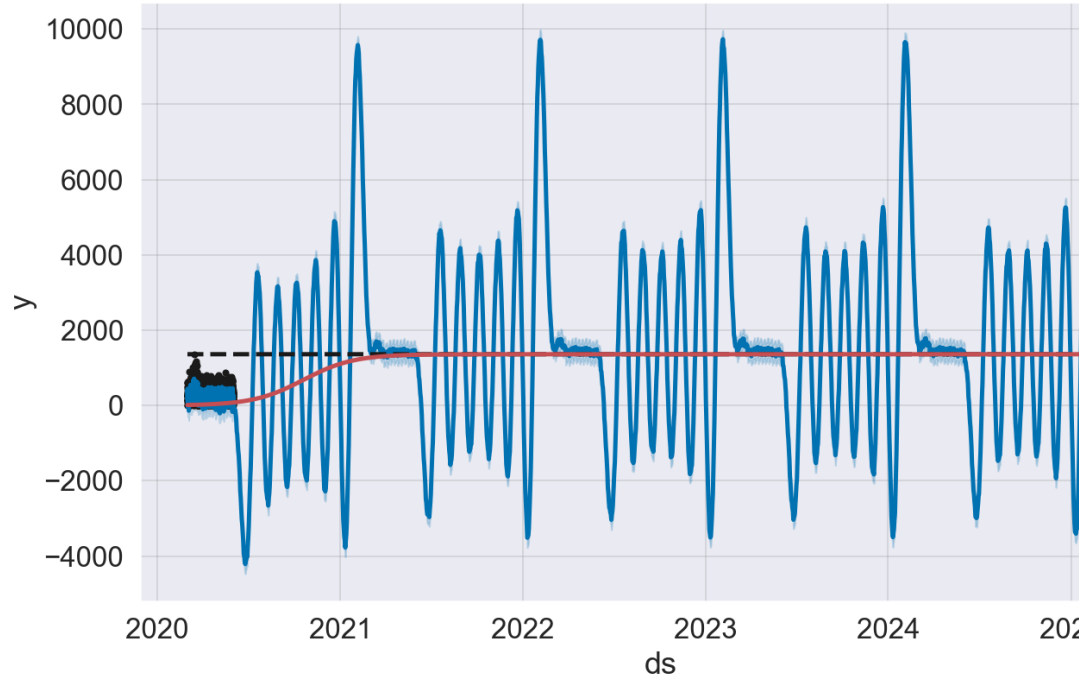
In [25]:
```
forecast_time_sale = m_274_time_sale.predict(future_274_time_
fig_time_sale = m_274_time_sale.plot(forecast_time_sale, unc
plt.title('Previsão de Vendas Loja 274')
```

Out[25]:
```
Text(0.5, 1.0, 'Previsão de Vendas Loja 274')
```



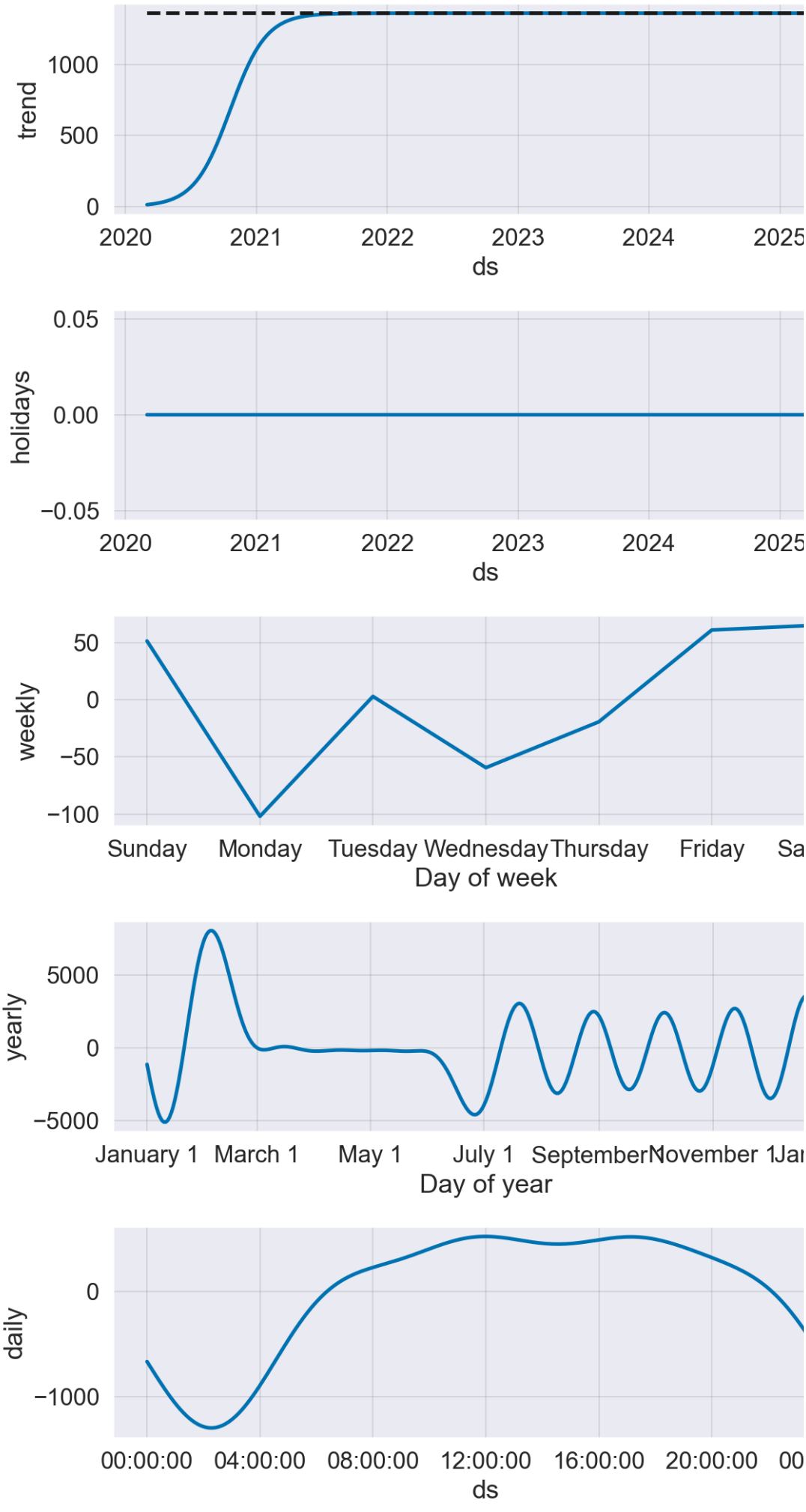
Visualizando os trends changepoints anual de venda

In [26]:
```
fig = m_274_time_sale.plot(forecast_time_sale)
trend_change_points_anual = add_changepoints_to_plot(fig.gca
```



Visualizando os componentes da série

In [27]:
```
fig_comp_time_sale = m_274_time_sale.plot_components(forecas
```



Visualizando os dados previstos para o futuro

In [28]:
```
future_no_art_sales_pred = forecast_time_sale.loc[forecast_t
future_no_art_sales_pred = future_no_art_sales_pred[['ds', '
future_no_art_sales_pred.sample(5)
```

Out[28]:

	ds	yhat	yhat_lower	yhat_upper
3708	2023-07-29 20:30:00	2063.485540	1808.223419	2309.853216
3566	2023-03-09 20:30:00	1654.711607	1403.095842	1913.537016
2656	2020-09-10 20:30:00	-953.547244	-1209.405730	-716.135893
3215	2022-03-23 20:30:00	1517.372910	1260.569314	1757.147226
3175	2022-02-11 20:30:00	7873.319548	7627.521731	8126.978455

Previsão de artigos vendidos por hora com inter
de 30 minutos para cada hora

In [29]:
```
future_274_time_sale = m_274_time_sale.make_future_dataframe
future_274_time_sale['cap'] = 1362
teste_future_274_time_sale = future_274_time_sale.loc[future_
teste_future_274_time_sale
```

Out[29]:

	ds	cap
2558	2020-06-01 00:30:00	1362
2559	2020-06-01 01:30:00	1362
2560	2020-06-01 02:30:00	1362
2561	2020-06-01 03:30:00	1362
2562	2020-06-01 04:30:00	1362
...	...	...
4350	2020-08-14 16:30:00	1362
4351	2020-08-14 17:30:00	1362
4352	2020-08-14 18:30:00	1362
4353	2020-08-14 19:30:00	1362
4354	2020-08-14 20:30:00	1362

1797 rows × 2 columns

In [30]:
```
df_sale_test = df_274_time_sale.copy()
df_sale_test['ds'] = pd.to_datetime(df_sale_test['ds'])
df_sale_test = df_sale_test.set_index(pd.DatetimeIndex(df_sa
df_sale_test.loc[df_sale_test['ds'].between_time('06:00','07
```

Out[30]:

	ds	y	cap
ds			
2020-03-02 07:00:00	2020-03-02 07:00:00	8	1362
2020-03-03 07:00:00	2020-03-03 07:00:00	94	1362
2020-03-04 07:00:00	2020-03-04 07:00:00	24	1362
2020-03-05 07:00:00	2020-03-05 07:00:00	12	1362
2020-03-06 07:00:00	2020-03-06 07:00:00	75	1362
2020-03-09 07:00:00	2020-03-09 07:00:00	81	1362
2020-03-10 07:00:00	2020-03-10 07:00:00	47	1362
2020-03-11 07:00:00	2020-03-11 07:00:00	57	1362
2020-03-12 07:00:00	2020-03-12 07:00:00	32	1362
2020-03-13 07:00:00	2020-03-13 07:00:00	65	1362
2020-03-17 07:00:00	2020-03-17 07:00:00	265	1362
2020-03-18 07:00:00	2020-03-18 07:00:00	88	1362
2020-03-19 07:00:00	2020-03-19 07:00:00	36	1362
2020-03-20 07:00:00	2020-03-20 07:00:00	16	1362
2020-03-22 07:00:00	2020-03-22 07:00:00	1	1362
2020-03-27 07:00:00	2020-03-27 07:00:00	2	1362
2020-04-03 07:00:00	2020-04-03 07:00:00	2	1362
2020-05-04 06:30:00	2020-05-04 06:30:00	2	1362
2020-05-05 06:00:00	2020-05-05 06:00:00	1	1362
2020-05-08 07:00:00	2020-05-08 07:00:00	1	1362
2020-05-12 06:30:00	2020-05-12 06:30:00	4	1362
2020-05-29 06:30:00	2020-05-29 06:30:00	1	1362
2020-05-30 06:00:00	2020-05-30 06:00:00	10	1362

In [31]:
```
df_sale_test.between_time('00:00','05:00')
```

Out[31]:

	ds	y	cap
ds			

In [32]:
```
future_274_time_sale_adjusted = future_274_time_sale.copy()
future_274_time_sale_adjusted['ds'] = pd.to_datetime(future_
future_274_time_sale_adjusted = future_274_time_sale_adjuste
#future_274_time_sale_adjusted = future_274_time_sale_adjust
#future_274_time_sale_adjusted.reset_index(drop=True)
future_274_time_sale_adjusted.between_time('00:00','05:00')
```

Out[32]:

	ds	cap
ds		
2020-06-01 00:30:00	2020-06-01 00:30:00	1362
2020-06-01 01:30:00	2020-06-01 01:30:00	1362
2020-06-01 02:30:00	2020-06-01 02:30:00	1362
2020-06-01 03:30:00	2020-06-01 03:30:00	1362
2020-06-01 04:30:00	2020-06-01 04:30:00	1362
...	...	...
2020-08-14 00:30:00	2020-08-14 00:30:00	1362
2020-08-14 01:30:00	2020-08-14 01:30:00	1362
2020-08-14 02:30:00	2020-08-14 02:30:00	1362
2020-08-14 03:30:00	2020-08-14 03:30:00	1362
2020-08-14 04:30:00	2020-08-14 04:30:00	1362

375 rows × 2 columns

In [34]:
```
future_274_time_sale_adjusted = future_274_time_sale.copy()
future_274_time_sale_adjusted['ds'] = pd.to_datetime(future_
future_274_time_sale_adjusted = future_274_time_sale_adjuste
future_274_time_sale_adjusted = future_274_time_sale_adjuste
#future_274_time_sale_adjusted.reset_index(drop=True)
future_274_time_sale_adjusted
```

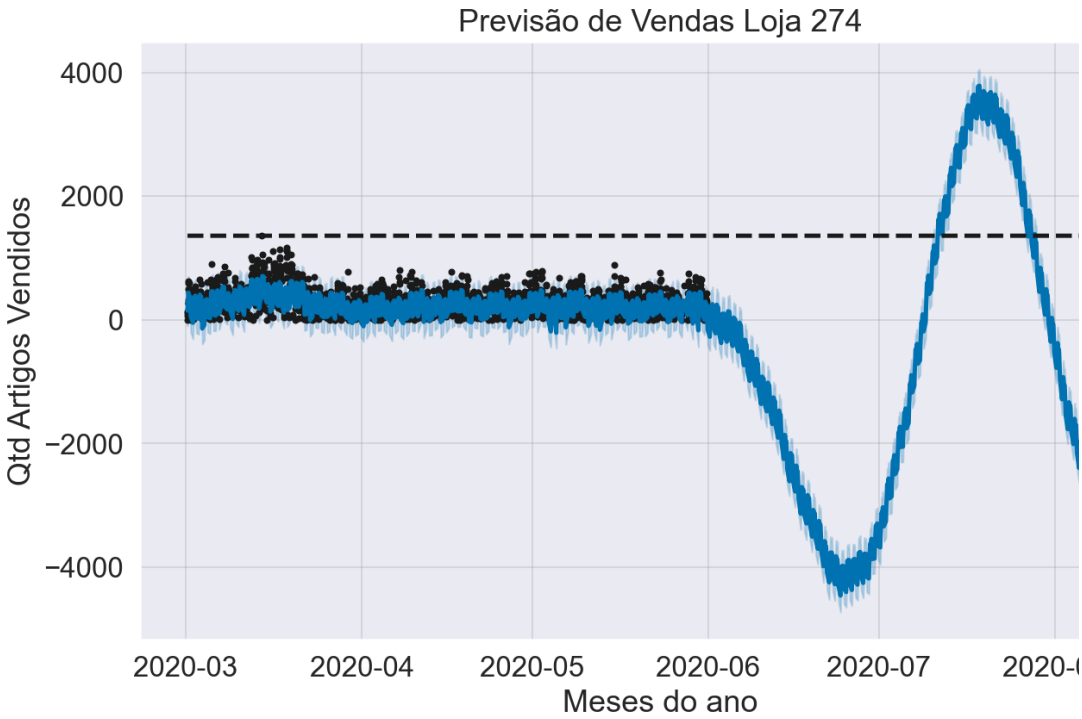
Out[34]:

	ds	cap
ds		
2020-03-01 07:30:00	2020-03-01 07:30:00	1362
2020-03-01 08:00:00	2020-03-01 08:00:00	1362
2020-03-01 08:30:00	2020-03-01 08:30:00	1362
2020-03-01 09:00:00	2020-03-01 09:00:00	1362
2020-03-01 09:30:00	2020-03-01 09:30:00	1362
...	...	...
2020-08-14 16:30:00	2020-08-14 16:30:00	1362
2020-08-14 17:30:00	2020-08-14 17:30:00	1362
2020-08-14 18:30:00	2020-08-14 18:30:00	1362
2020-08-14 19:30:00	2020-08-14 19:30:00	1362
2020-08-14 20:30:00	2020-08-14 20:30:00	1362

3755 rows × 2 columns

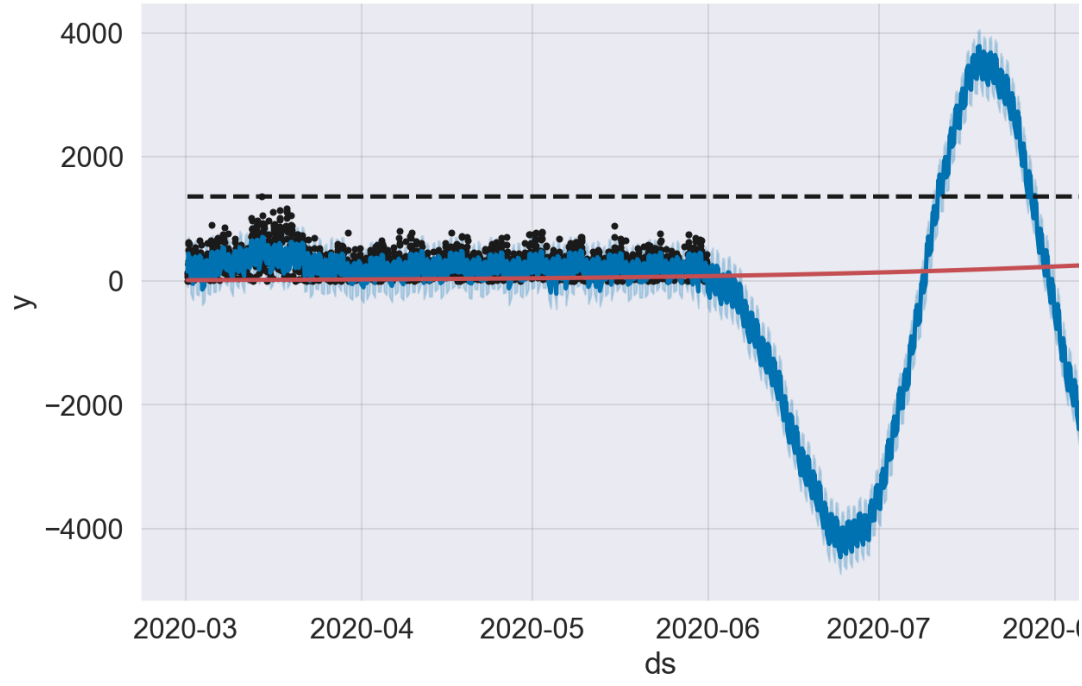
In [35]:
```
forecast_time_sale = m_274_time_sale.predict(future_274_time_
fig_time_sale = m_274_time_sale.plot(forecast_time_sale, unc
plt.title('Previsão de Vendas Loja 274')
```

Out[35]:
```
Text(0.5, 1.0, 'Previsão de Vendas Loja 274')
```



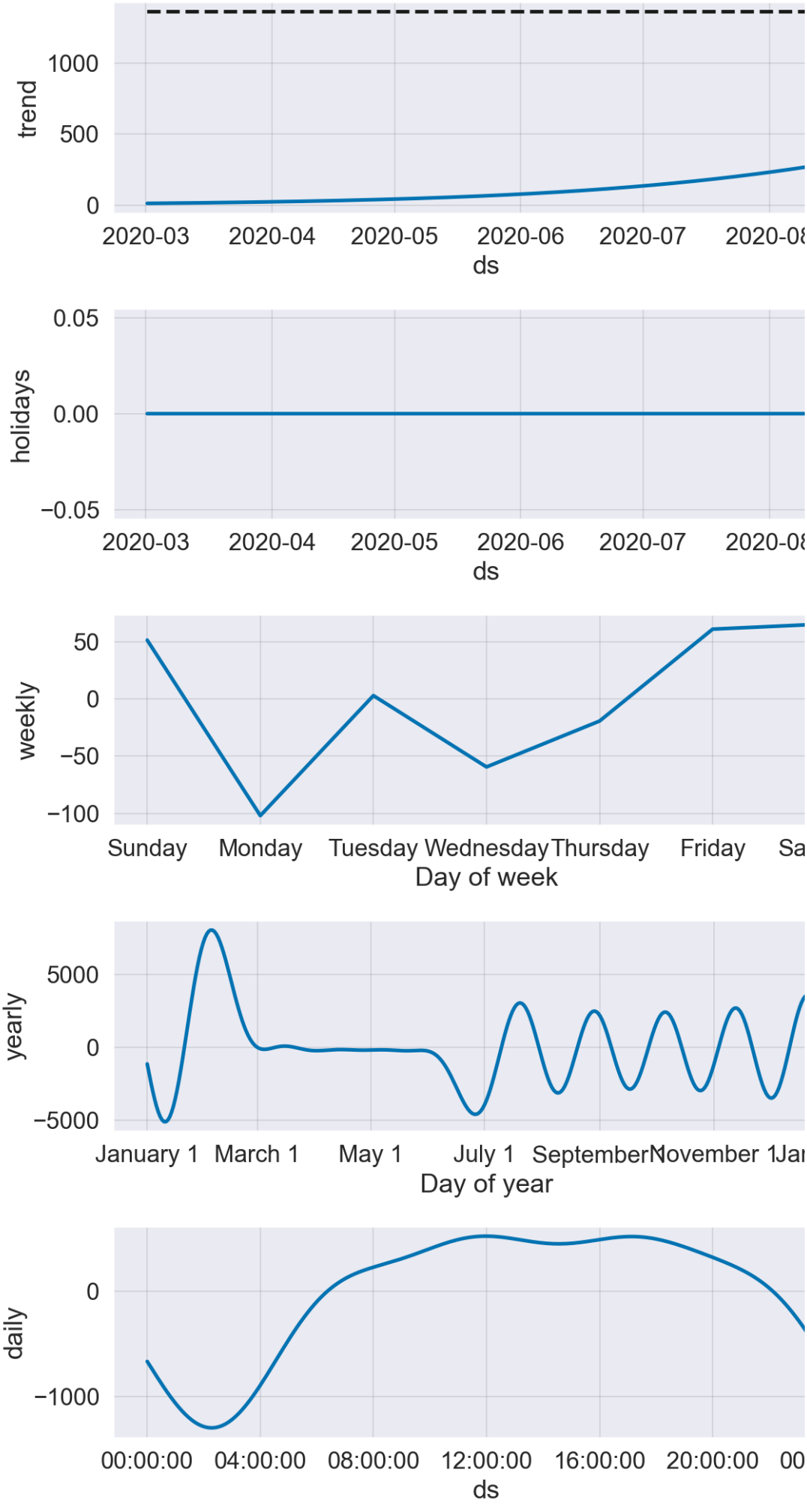
Visualizando os trends changepoints de venda

In [36]:
```
figtrend_changepoints = m_274_time_sale.plot(forecast_time_sa
trend_changepoints_diario = add_changepoints_to_plot(figtren
```



Visualizando os componentes da série

In [37]:
```
fig_comp_time_sale = m_274_time_sale.plot_components(forecas
```



Visualizando os dados previstos para o futuro

In [38]:
```
future_no_art_sales_pred = forecast_time_sale.loc[forecast_t
future_no_art_sales_pred = future_no_art_sales_pred[['ds', '
future_no_art_sales_pred.sample(5)
```

Out[38]:

	ds	yhat	yhat_lower	yhat_upper
3637	2020-08-07 15:30:00	-2262.320423	-2505.568246	-2005.318634
3573	2020-08-03 15:30:00	-1411.388833	-1673.231975	-1165.160793
3491	2020-07-29 13:30:00	768.646550	513.588271	1028.420518
3347	2020-07-20 13:30:00	3630.943841	3358.155948	3893.613491
3117	2020-07-06 07:30:00	-1488.305207	-1736.398621	-1253.883194

Previsão de clientes/hora (loja 274)

In [39]:
```
df_274_time_cli['cap'] = 67
df_274_time_cli.reset_index(drop=True)
m_274_time_cli = Prophet(growth='logistic',
                         interval_width=0.95,
                         changepoint_prior_scale=0.095,
                         yearly_seasonality=True,
                         weekly_seasonality=True,
                         holidays=datacleaner.get_Holiday())
m_274_time_cli.add_country_holidays(country_name='BR')
m_274_time_cli.fit(df_274_time_cli)
```

Out[39]:
```
<fbprophet.forecaster.Prophet at 0x1277d0070>
```

Criando previsão diária para um período de 5 anos

In [40]:
```
future_274_time_cli = m_274_time_cli.make_future_dataframe(p
future_274_time_cli['cap'] = 67
teste_future_274_time_cli = future_274_time_cli.loc[future_2
teste_future_274_time_cli
```

Out[40]:

	ds	cap
2555	2020-06-01 20:30:00	67
2556	2020-06-02 20:30:00	67
2557	2020-06-03 20:30:00	67
2558	2020-06-04 20:30:00	67
2559	2020-06-05 20:30:00	67
...	...	...
4350	2025-05-01 20:30:00	67
4351	2025-05-02 20:30:00	67
4352	2025-05-03 20:30:00	67
4353	2025-05-04 20:30:00	67
4354	2025-05-05 20:30:00	67

1800 rows × 2 columns

In [41]:
```
df_cli_test = df_274_time_cli.copy()
df_cli_test['ds'] = pd.to_datetime(df_cli_test['ds'])
df_cli_test = df_cli_test.set_index(pd.DatetimeIndex(df_cli_
df_cli_test.loc[df_cli_test['ds'].between_time('06:00','07:0
```

Out[41]:

	ds	y	cap
ds			
2020-03-02 07:00:00	2020-03-02 07:00:00	3	67
2020-03-03 07:00:00	2020-03-03 07:00:00	22	67
2020-03-04 07:00:00	2020-03-04 07:00:00	9	67
2020-03-05 07:00:00	2020-03-05 07:00:00	5	67
2020-03-06 07:00:00	2020-03-06 07:00:00	21	67
2020-03-09 07:00:00	2020-03-09 07:00:00	18	67
2020-03-10 07:00:00	2020-03-10 07:00:00	18	67
2020-03-11 07:00:00	2020-03-11 07:00:00	18	67
2020-03-12 07:00:00	2020-03-12 07:00:00	14	67
2020-03-13 07:00:00	2020-03-13 07:00:00	24	67
2020-03-17 07:00:00	2020-03-17 07:00:00	33	67
2020-03-18 07:00:00	2020-03-18 07:00:00	15	67
2020-03-19 07:00:00	2020-03-19 07:00:00	8	67
2020-03-20 07:00:00	2020-03-20 07:00:00	7	67
2020-03-22 07:00:00	2020-03-22 07:00:00	1	67
2020-03-27 07:00:00	2020-03-27 07:00:00	1	67
2020-04-03 07:00:00	2020-04-03 07:00:00	1	67
2020-05-04 06:30:00	2020-05-04 06:30:00	1	67
2020-05-05 06:00:00	2020-05-05 06:00:00	1	67
2020-05-08 07:00:00	2020-05-08 07:00:00	1	67
2020-05-12 06:30:00	2020-05-12 06:30:00	1	67
2020-05-29 06:30:00	2020-05-29 06:30:00	1	67
2020-05-30 06:00:00	2020-05-30 06:00:00	2	67

In [42]:
```
future_274_time_cli_adjusted = future_274_time_cli.copy()
future_274_time_cli_adjusted['ds'] = pd.to_datetime(future_2
future_274_time_cli_adjusted = future_274_time_cli_adjusted.
future_274_time_cli_adjusted = future_274_time_cli_adjusted.
future_274_time_cli_adjusted
```

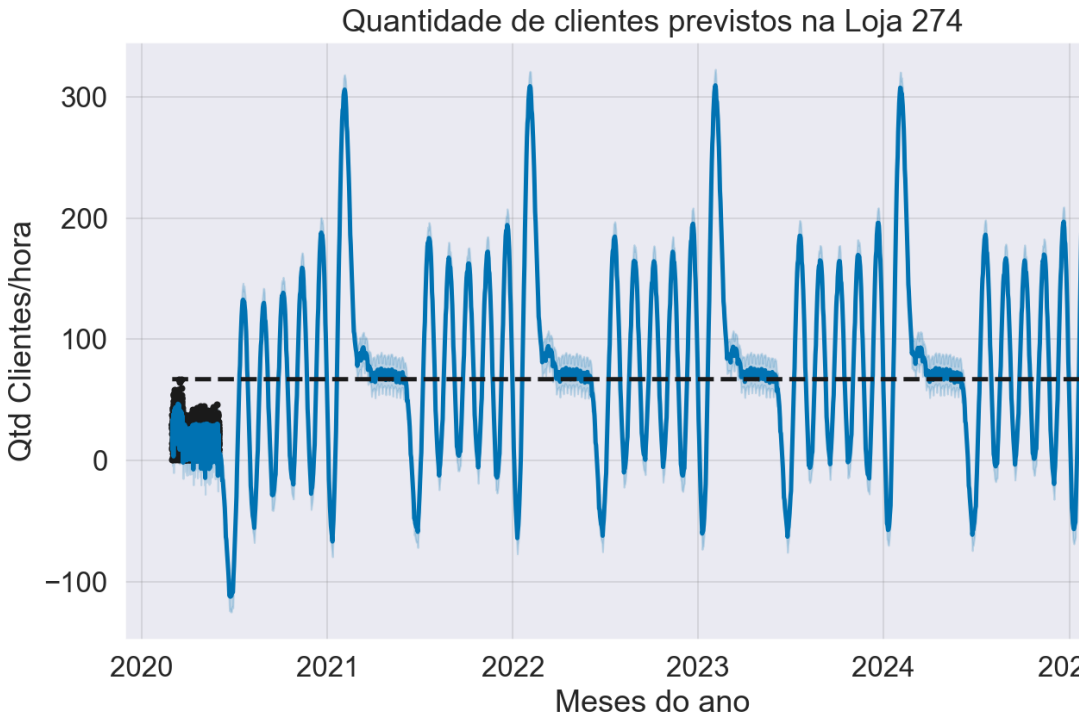
Out[42]:

	ds	cap
ds		
2020-03-01 07:30:00	2020-03-01 07:30:00	67
2020-03-01 08:00:00	2020-03-01 08:00:00	67
2020-03-01 08:30:00	2020-03-01 08:30:00	67
2020-03-01 09:00:00	2020-03-01 09:00:00	67
2020-03-01 09:30:00	2020-03-01 09:30:00	67
...	...	...
2025-05-01 20:30:00	2025-05-01 20:30:00	67
2025-05-02 20:30:00	2025-05-02 20:30:00	67
2025-05-03 20:30:00	2025-05-03 20:30:00	67
2025-05-04 20:30:00	2025-05-04 20:30:00	67
2025-05-05 20:30:00	2025-05-05 20:30:00	67

4355 rows × 2 columns

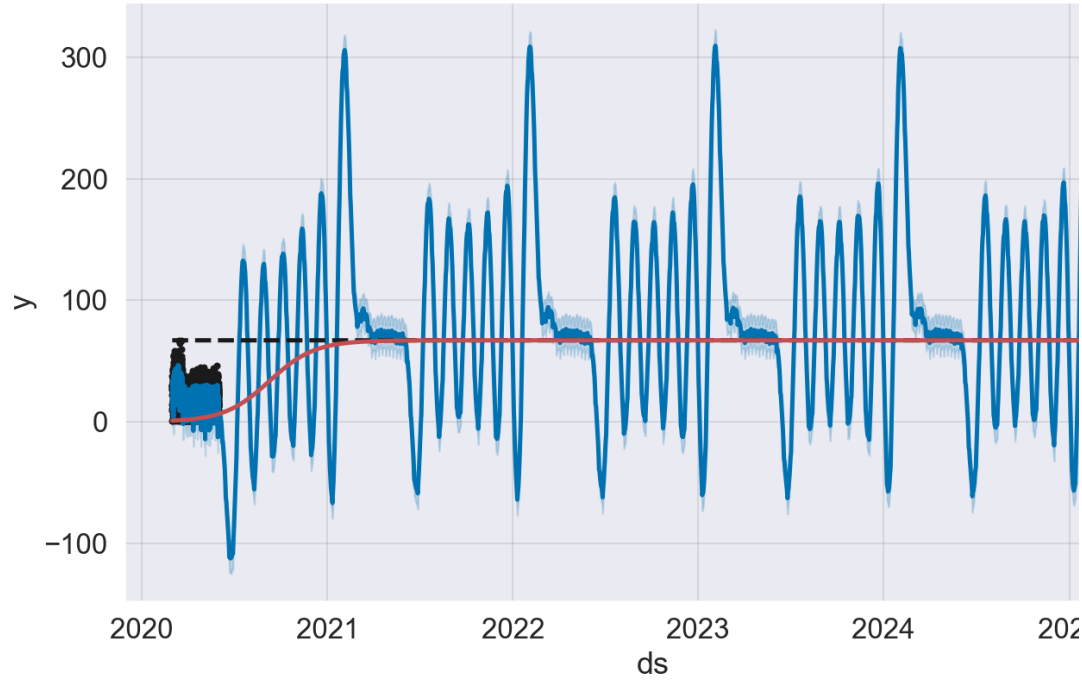
In [43]:
```
forecast_time_cli = m_274_time_cli.predict(future_274_time_c
fig_time_cli = m_274_time_cli.plot(forecast_time_cli, uncert
plt.title('Quantidade de clientes previstos na Loja 274')
```

Out[43]:
```
Text(0.5, 1.0, 'Quantidade de clientes previstos na Loja 274
```



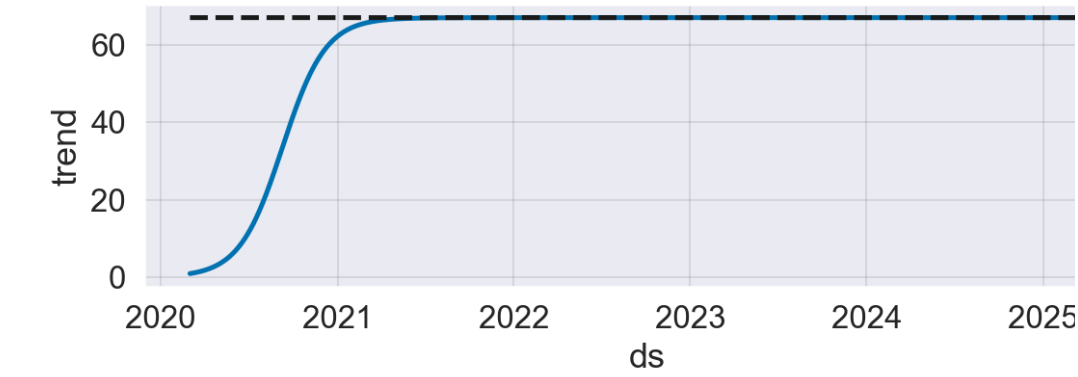
Visualizando os trends changepoints de venda

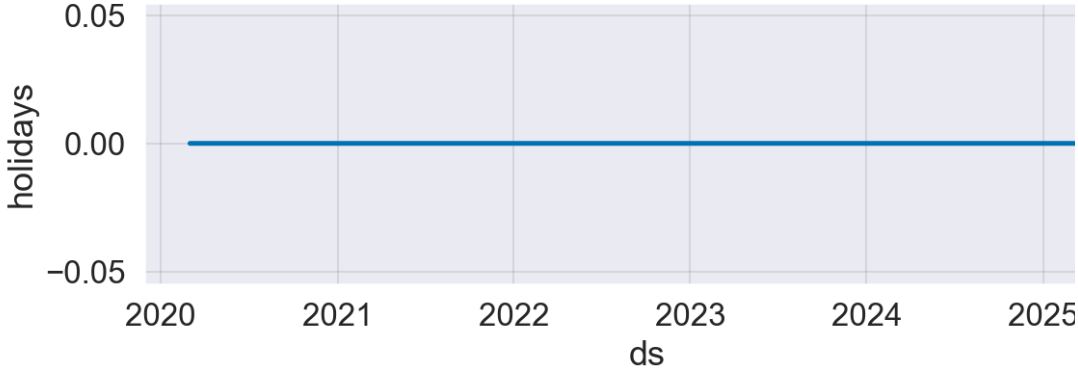
In [44]:
```
figtrend_changepoints_anual_cli = m_274_time_cli.plot(foreca
trend_changepoints_anual_cli = add_changepoints_to_plot(figt
```

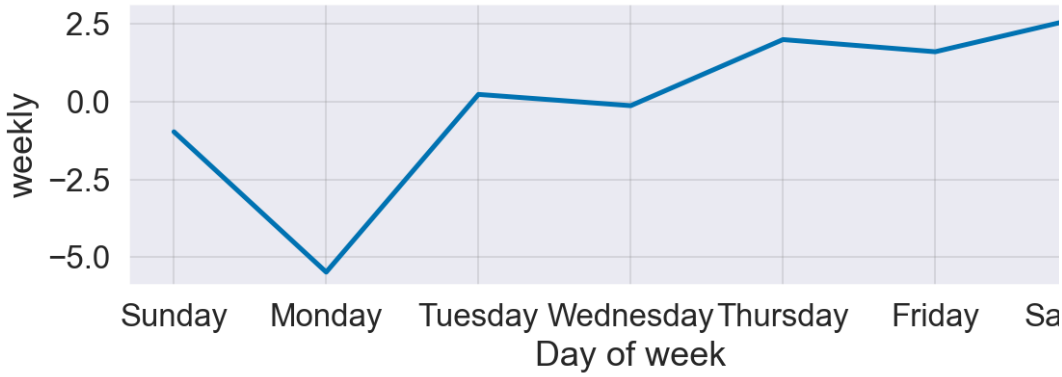


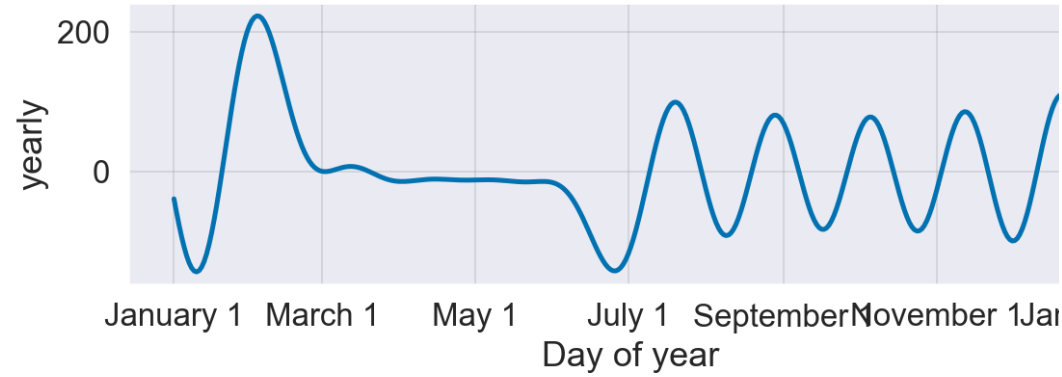
Visualizando os componentes da série

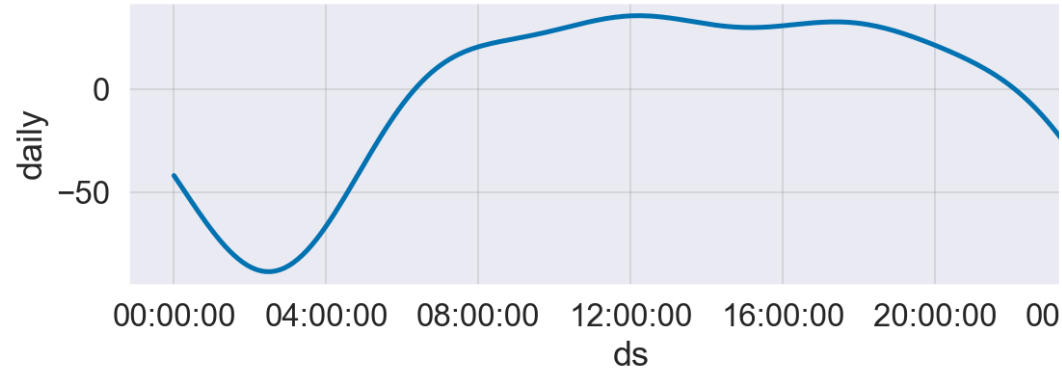
In [45]:
```
fig_comp_time_cli = m_274_time_cli.plot_components(forecast_
```











Visualizando os dados previstos para o futuro

In [46]:
```
future_no_cli_pred = forecast_time_cli.loc[forecast_time_cli
future_no_cli_pred = future_no_cli_pred[['ds', 'yhat', 'yhat_
future_no_cli_pred.sample(5)
```

Out[46]:

	ds	yhat	yhat_lower	yhat_upper
3557	2023-02-28 20:30:00	85.617755	72.289877	97.919426
2870	2021-04-12 20:30:00	72.843650	60.945757	86.357255
3720	2023-08-10 20:30:00	-3.165808	-15.466400	10.692331
2750	2020-12-13 20:30:00	125.209653	112.700816	137.682976
2975	2021-07-26 20:30:00	131.972046	118.987245	144.351247

Criando previsão com frequência horária e com intervalos de 30 minutos

In [47]:
```
future_274_time_cli = m_274_time_cli.make_future_dataframe(p
future_274_time_cli['cap'] = 67
teste_future_274_time_cli = future_274_time_cli.loc[future_2
teste_future_274_time_cli
```

Out[47]:

	ds	cap
2558	2020-06-01 00:30:00	67
2559	2020-06-01 01:30:00	67
2560	2020-06-01 02:30:00	67
2561	2020-06-01 03:30:00	67
2562	2020-06-01 04:30:00	67
...	...	...
4350	2020-08-14 16:30:00	67
4351	2020-08-14 17:30:00	67
4352	2020-08-14 18:30:00	67
4353	2020-08-14 19:30:00	67
4354	2020-08-14 20:30:00	67

1797 rows × 2 columns

In [48]:
```
df_cli_test = df_274_time_cli.copy()
df_cli_test['ds'] = pd.to_datetime(df_cli_test['ds'])
df_cli_test = df_cli_test.set_index(pd.DatetimeIndex(df_cli_
df_cli_test.loc[df_cli_test['ds'].between_time('06:00','07:0
```

Out[48]:

	ds	y	cap
ds			
2020-03-02 07:00:00	2020-03-02 07:00:00	3	67
2020-03-03 07:00:00	2020-03-03 07:00:00	22	67
2020-03-04 07:00:00	2020-03-04 07:00:00	9	67
2020-03-05 07:00:00	2020-03-05 07:00:00	5	67
2020-03-06 07:00:00	2020-03-06 07:00:00	21	67
2020-03-09 07:00:00	2020-03-09 07:00:00	18	67
2020-03-10 07:00:00	2020-03-10 07:00:00	18	67
2020-03-11 07:00:00	2020-03-11 07:00:00	18	67
2020-03-12 07:00:00	2020-03-12 07:00:00	14	67
2020-03-13 07:00:00	2020-03-13 07:00:00	24	67
2020-03-17 07:00:00	2020-03-17 07:00:00	33	67
2020-03-18 07:00:00	2020-03-18 07:00:00	15	67
2020-03-19 07:00:00	2020-03-19 07:00:00	8	67
2020-03-20 07:00:00	2020-03-20 07:00:00	7	67
2020-03-22 07:00:00	2020-03-22 07:00:00	1	67
2020-03-27 07:00:00	2020-03-27 07:00:00	1	67
2020-04-03 07:00:00	2020-04-03 07:00:00	1	67
2020-05-04 06:30:00	2020-05-04 06:30:00	1	67
2020-05-05 06:00:00	2020-05-05 06:00:00	1	67
2020-05-08 07:00:00	2020-05-08 07:00:00	1	67
2020-05-12 06:30:00	2020-05-12 06:30:00	1	67
2020-05-29 06:30:00	2020-05-29 06:30:00	1	67
2020-05-30 06:00:00	2020-05-30 06:00:00	2	67

In [49]:
```
future_274_time_cli_adjusted = future_274_time_cli.copy()
future_274_time_cli_adjusted['ds'] = pd.to_datetime(future_2
future_274_time_cli_adjusted = future_274_time_cli_adjusted.
future_274_time_cli_adjusted = future_274_time_cli_adjusted.
#future_274_time_cli_adjusted.reset_index(drop=True)
future_274_time_cli_adjusted
```

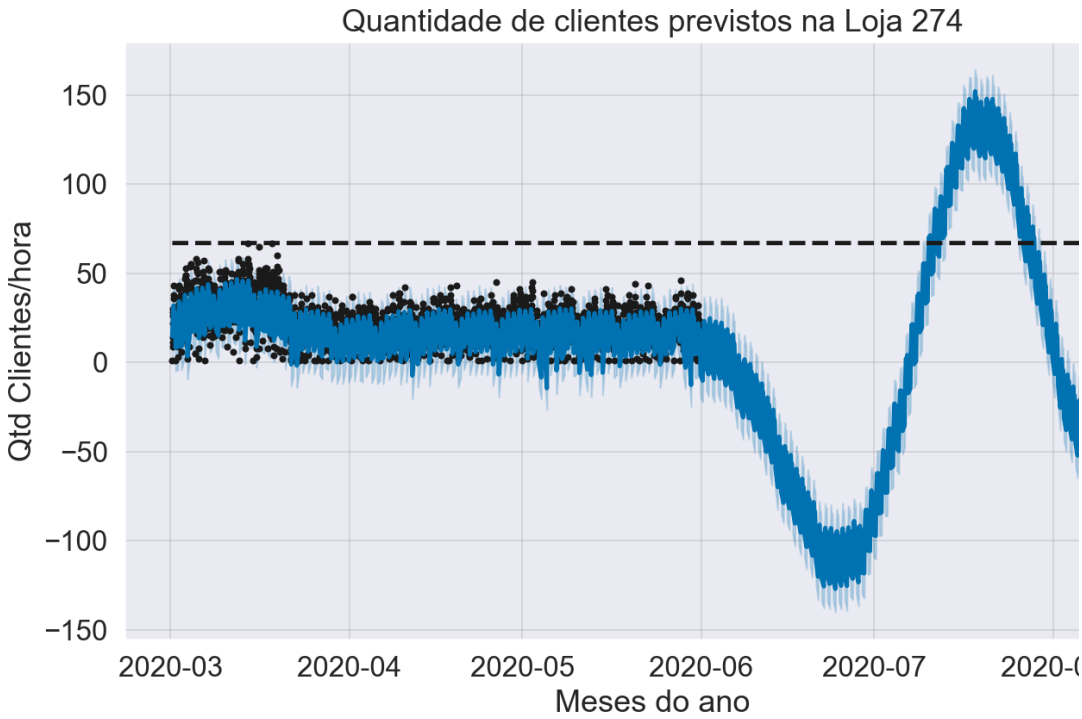
Out[49]:

	ds	cap
ds		
2020-03-01 07:30:00	2020-03-01 07:30:00	67
2020-03-01 08:00:00	2020-03-01 08:00:00	67
2020-03-01 08:30:00	2020-03-01 08:30:00	67
2020-03-01 09:00:00	2020-03-01 09:00:00	67
2020-03-01 09:30:00	2020-03-01 09:30:00	67
...	...	...
2020-08-14 16:30:00	2020-08-14 16:30:00	67
2020-08-14 17:30:00	2020-08-14 17:30:00	67
2020-08-14 18:30:00	2020-08-14 18:30:00	67
2020-08-14 19:30:00	2020-08-14 19:30:00	67
2020-08-14 20:30:00	2020-08-14 20:30:00	67

3755 rows × 2 columns

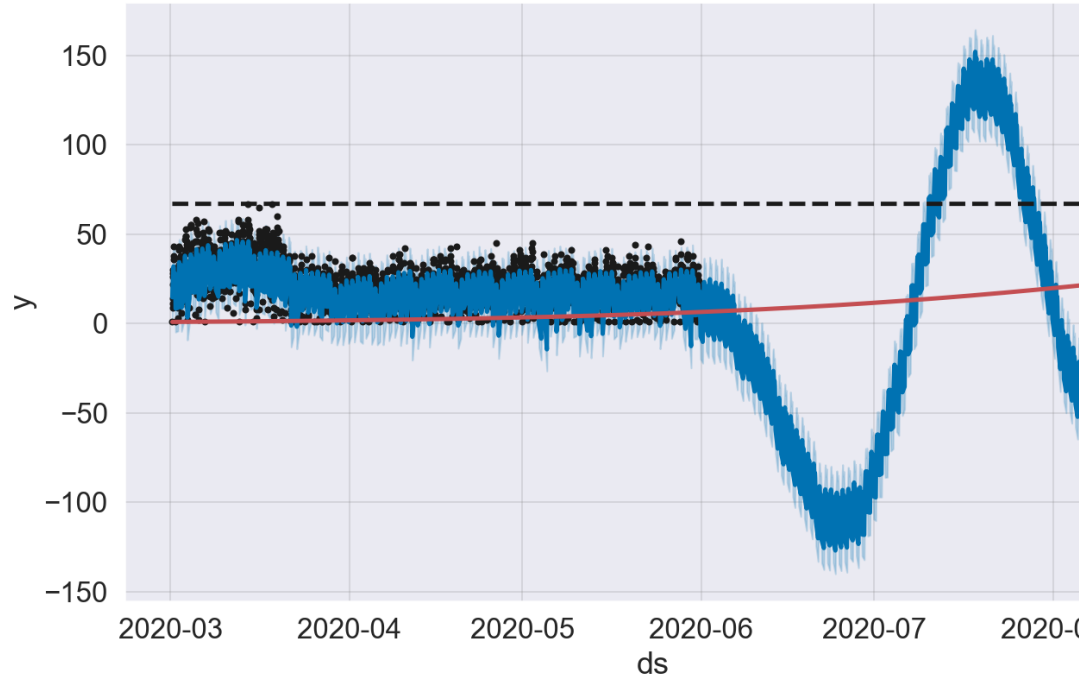
In [50]:
```
forecast_time_cli = m_274_time_cli.predict(future_274_time_c
fig_time_cli = m_274_time_cli.plot(forecast_time_cli, uncert
plt.title('Quantidade de clientes previstos na Loja 274')
```

Out[50]:
```
Text(0.5, 1.0, 'Quantidade de clientes previstos na Loja 274
```



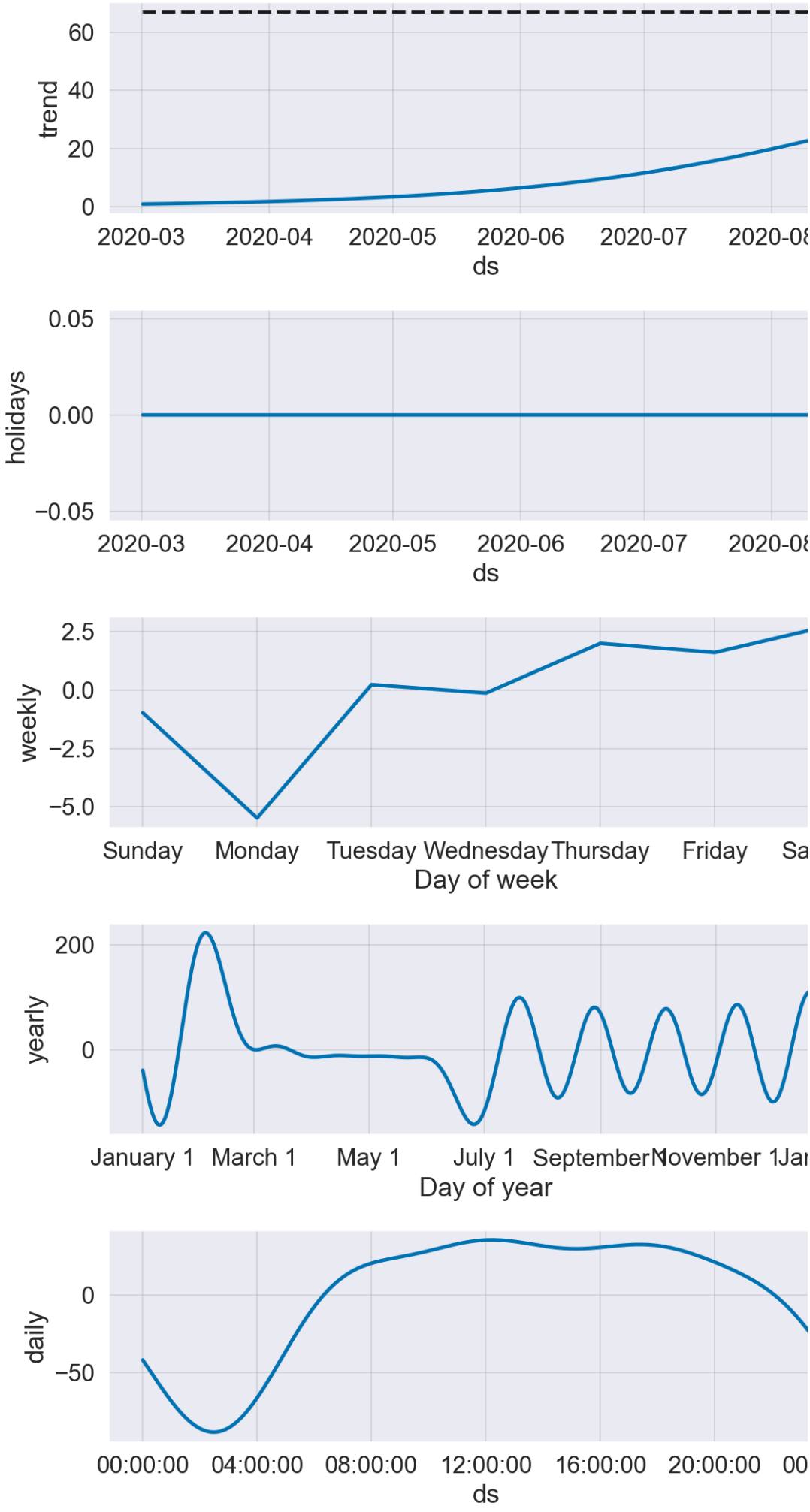
Visualizando os trends changepoints de venda

In [51]:
```
figtrend_changepoints_diario_cli = m_274_time_cli.plot(foreca
trend_changepoints_diario_cli = add_changepoints_to_plot(fig
```



Visualizando os componentes da série

In [52]:
```
fig_comp_time_cli = m_274_time_cli.plot_components(forecast_
```



Visualizando os dados previstos para o futuro

In [53]:
```
future_no_cli_pred = forecast_time_cli.loc[forecast_time_cli
future_no_cli_pred = future_no_cli_pred[['ds', 'yhat', 'yhat_
future_no_cli_pred.sample(5)
```

Out[53]:

	ds	yhat	yhat_lower	yhat_upper
3469	2020-07-28 07:30:00	60.909962	48.458081	72.766964
2571	2020-06-01 21:30:00	-2.773774	-15.351947	9.789926
3447	2020-07-26 17:30:00	91.559821	79.964253	104.475494
3053	2020-07-02 07:30:00	-70.061231	-83.882912	-56.371600
3274	2020-07-15 20:30:00	118.705140	105.489876	130.468003

Saturação linear para quantidade de produtos vendidos

In [54]:
```
future_274_time_sale_l = m_274_time_sale_linear.make_future_
future_274_time_sale_l['cap'] = 1362
teste_future_274_time_sale_l = future_274_time_sale_l.loc[fu
teste_future_274_time_sale_l
```

Out[54]:

	ds	cap
2558	2020-06-01 00:30:00	1362
2559	2020-06-01 01:30:00	1362
2560	2020-06-01 02:30:00	1362
2561	2020-06-01 03:30:00	1362
2562	2020-06-01 04:30:00	1362
...	...	...
4350	2020-08-14 16:30:00	1362
4351	2020-08-14 17:30:00	1362
4352	2020-08-14 18:30:00	1362
4353	2020-08-14 19:30:00	1362
4354	2020-08-14 20:30:00	1362

1797 rows × 2 columns

In [55]:
```
df_sale_test_a = df_274_time_sale.copy()
df_sale_test_a['ds'] = pd.to_datetime(df_sale_test_a['ds'])
df_sale_test_a = df_sale_test_a.set_index(pd.DatetimeIndex(d
df_sale_test_a.loc[df_sale_test_a['ds'].between_time('06:00'
```

Out[55]:

	ds	y	cap
ds			
2020-03-02 07:00:00	2020-03-02 07:00:00	8	1362
2020-03-03 07:00:00	2020-03-03 07:00:00	94	1362
2020-03-04 07:00:00	2020-03-04 07:00:00	24	1362
2020-03-05 07:00:00	2020-03-05 07:00:00	12	1362
2020-03-06 07:00:00	2020-03-06 07:00:00	75	1362
2020-03-09 07:00:00	2020-03-09 07:00:00	81	1362
2020-03-10 07:00:00	2020-03-10 07:00:00	47	1362
2020-03-11 07:00:00	2020-03-11 07:00:00	57	1362
2020-03-12 07:00:00	2020-03-12 07:00:00	32	1362
2020-03-13 07:00:00	2020-03-13 07:00:00	65	1362
2020-03-17 07:00:00	2020-03-17 07:00:00	265	1362
2020-03-18 07:00:00	2020-03-18 07:00:00	88	1362
2020-03-19 07:00:00	2020-03-19 07:00:00	36	1362
2020-03-20 07:00:00	2020-03-20 07:00:00	16	1362
2020-03-22 07:00:00	2020-03-22 07:00:00	1	1362
2020-03-27 07:00:00	2020-03-27 07:00:00	2	1362
2020-04-03 07:00:00	2020-04-03 07:00:00	2	1362
2020-05-04 06:30:00	2020-05-04 06:30:00	2	1362
2020-05-05 06:00:00	2020-05-05 06:00:00	1	1362
2020-05-08 07:00:00	2020-05-08 07:00:00	1	1362
2020-05-12 06:30:00	2020-05-12 06:30:00	4	1362
2020-05-29 06:30:00	2020-05-29 06:30:00	1	1362
2020-05-30 06:00:00	2020-05-30 06:00:00	10	1362

In [56]:
```
future_274_time_sale_adjusted_l = future_274_time_sale_l.cop
future_274_time_sale_adjusted_l['ds'] = pd.to_datetime(futur
future_274_time_sale_adjusted_l = future_274_time_sale_adjus
future_274_time_sale_adjusted_l = future_274_time_sale_adjus
#future_274_time_sale_adjusted.reset_index(drop=True)
future_274_time_sale_adjusted_l
```

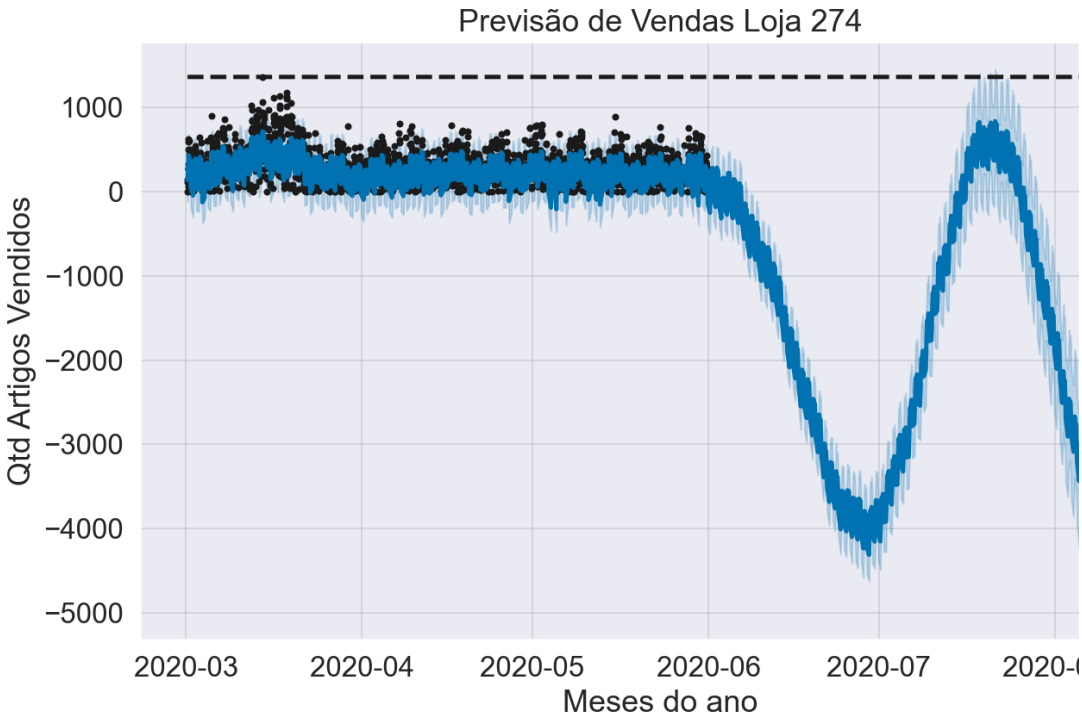
Out[56]:

	ds	cap
ds		
2020-03-01 07:30:00	2020-03-01 07:30:00	1362
2020-03-01 08:00:00	2020-03-01 08:00:00	1362
2020-03-01 08:30:00	2020-03-01 08:30:00	1362
2020-03-01 09:00:00	2020-03-01 09:00:00	1362
2020-03-01 09:30:00	2020-03-01 09:30:00	1362
...	...	...
2020-08-14 16:30:00	2020-08-14 16:30:00	1362
2020-08-14 17:30:00	2020-08-14 17:30:00	1362
2020-08-14 18:30:00	2020-08-14 18:30:00	1362
2020-08-14 19:30:00	2020-08-14 19:30:00	1362
2020-08-14 20:30:00	2020-08-14 20:30:00	1362

3755 rows × 2 columns

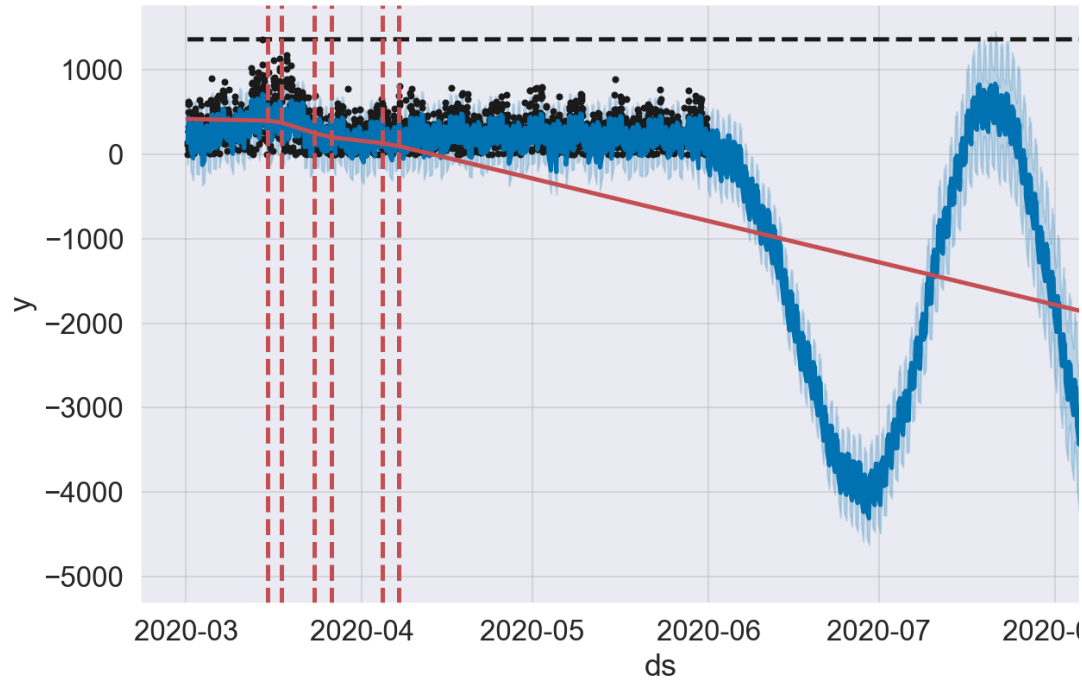
In [57]:
```
forecast_time_sale_l = m_274_time_sale_linear.predict(future_
fig_time_sale_a = m_274_time_sale_linear.plot(forecast_time_
plt.title('Previsão de Vendas Loja 274')
```

Out[57]:
```
Text(0.5, 1.0, 'Previsão de Vendas Loja 274')
```



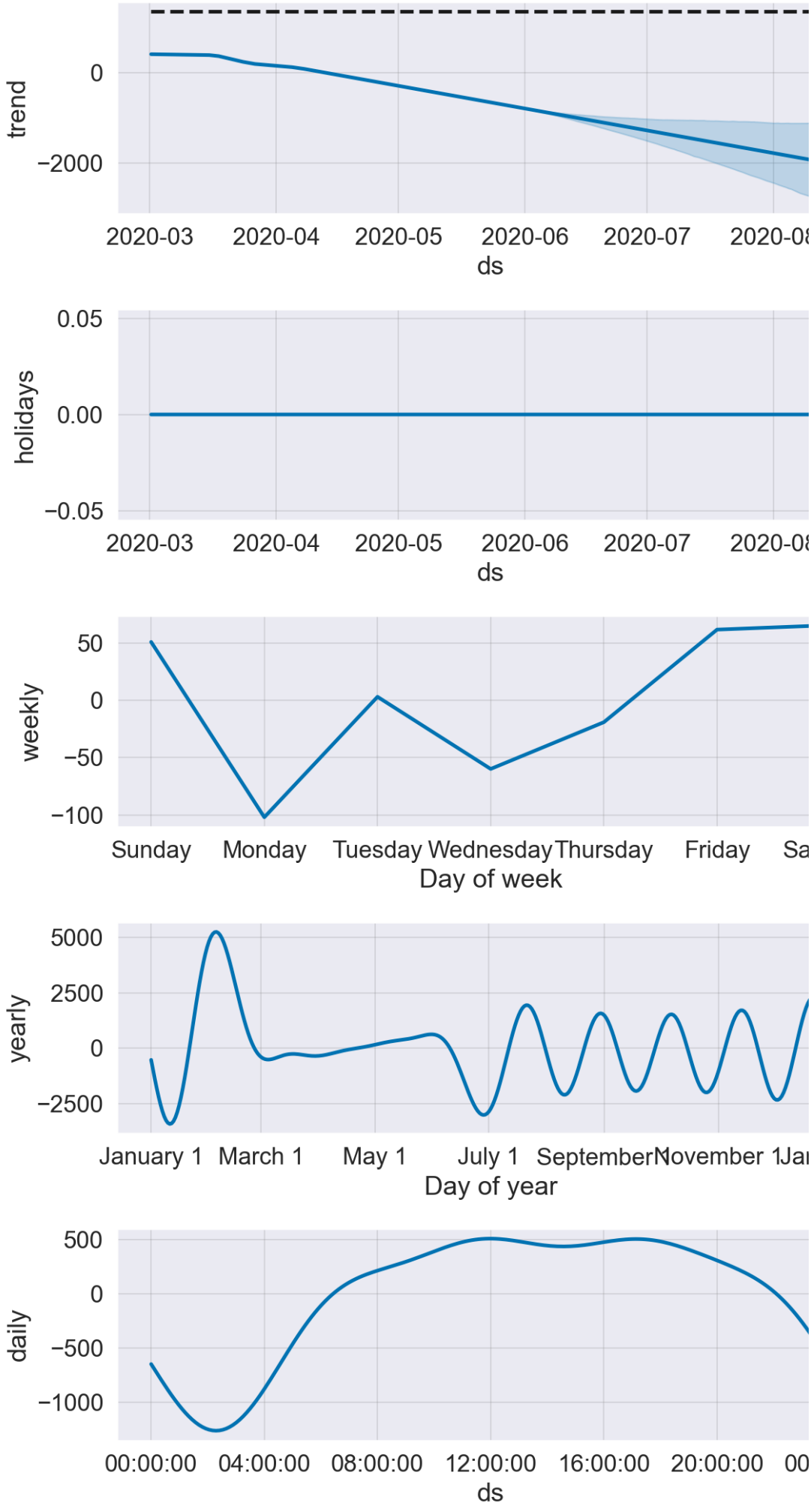
Visualizando os trends changepoints de venda

In [58]:
```
figtrend_changepoints_diario_sale = m_274_time_sale_linear.p
trend_changepoints_diario_sale = add_changepoints_to_plot(fi
```



Visualizando os componentes da série

In [59]:
```
fig_comp_time_sale_a = m_274_time_sale_linear.plot_component
```



Visualizando os dados previstos para o futuro

In [60]:
```
future_no_art_sales_pred_l = forecast_time_sale_l.loc[foreca
future_no_art_sales_pred_l = future_no_art_sales_pred_l[['ds
future_no_art_sales_pred_l.sample(5)
```

Out[60]:

	ds	yhat	yhat_lower	yhat_upper
3553	2020-08-02 11:30:00	-2131.297804	-2864.912910	-1388.811329
2951	2020-06-25 17:30:00	-3556.837560	-3871.927060	-3255.453392
2907	2020-06-22 21:30:00	-3613.207610	-3899.518521	-3330.634798
3534	2020-08-01 08:30:00	-1889.034164	-2601.967943	-1120.354268
2798	2020-06-16 08:30:00	-2002.112367	-2254.069399	-1746.185322

Saturação linear para quantidade de clientes/hora

In [61]:
```
m_274_time_cli_linear = Prophet(growth='linear',
                          interval_width=0.95,
                          changepoint_prior_scale=0.095,
                          yearly_seasonality=True,
                          weekly_seasonality=True,
                          holidays=datacleaner.get_Holiday())
m_274_time_cli_linear.add_country_holidays(country_name='BR'
m_274_time_cli_linear.fit(df_274_time_cli)
m_274_time_cli_linear.train_holiday_names
```

Out[61]:
```
0                       parana
1                New Year's Day
2                    Tiradentes
3                  Worker's Day
4               Independence Day
5    Our Lady of the Apparition
6                All Souls' Day
7     Republic Proclamation Day
8                      Christmas
dtype: object
```

In [62]:
```
future_274_time_cli_l = m_274_time_cli_linear.make_future_da
future_274_time_cli_l['cap'] = 67
teste_future_274_time_cli_l = future_274_time_cli_l.loc[futu
teste_future_274_time_cli_l
```

Out[62]:

	ds	cap
2558	2020-06-01 00:30:00	67
2559	2020-06-01 01:30:00	67
2560	2020-06-01 02:30:00	67
2561	2020-06-01 03:30:00	67
2562	2020-06-01 04:30:00	67
...	...	...
4350	2020-08-14 16:30:00	67
4351	2020-08-14 17:30:00	67
4352	2020-08-14 18:30:00	67
4353	2020-08-14 19:30:00	67
4354	2020-08-14 20:30:00	67

1797 rows × 2 columns

In [63]:
```
df_cli_test_a = df_274_time_cli.copy()
df_cli_test_a['ds'] = pd.to_datetime(df_cli_test_a['ds'])
df_cli_test_a = df_cli_test_a.set_index(pd.DatetimeIndex(df_
df_cli_test_a.loc[df_cli_test_a['ds'].between_time('06:00','
```

Out[63]:

	ds	y	cap
ds			
2020-03-02 07:00:00	2020-03-02 07:00:00	3	67
2020-03-03 07:00:00	2020-03-03 07:00:00	22	67
2020-03-04 07:00:00	2020-03-04 07:00:00	9	67
2020-03-05 07:00:00	2020-03-05 07:00:00	5	67
2020-03-06 07:00:00	2020-03-06 07:00:00	21	67
2020-03-09 07:00:00	2020-03-09 07:00:00	18	67
2020-03-10 07:00:00	2020-03-10 07:00:00	18	67
2020-03-11 07:00:00	2020-03-11 07:00:00	18	67
2020-03-12 07:00:00	2020-03-12 07:00:00	14	67
2020-03-13 07:00:00	2020-03-13 07:00:00	24	67
2020-03-17 07:00:00	2020-03-17 07:00:00	33	67
2020-03-18 07:00:00	2020-03-18 07:00:00	15	67
2020-03-19 07:00:00	2020-03-19 07:00:00	8	67
2020-03-20 07:00:00	2020-03-20 07:00:00	7	67
2020-03-22 07:00:00	2020-03-22 07:00:00	1	67
2020-03-27 07:00:00	2020-03-27 07:00:00	1	67
2020-04-03 07:00:00	2020-04-03 07:00:00	1	67
2020-05-04 06:30:00	2020-05-04 06:30:00	1	67
2020-05-05 06:00:00	2020-05-05 06:00:00	1	67
2020-05-08 07:00:00	2020-05-08 07:00:00	1	67
2020-05-12 06:30:00	2020-05-12 06:30:00	1	67
2020-05-29 06:30:00	2020-05-29 06:30:00	1	67
2020-05-30 06:00:00	2020-05-30 06:00:00	2	67

In [64]:
```
future_274_time_cli_adjusted_l = future_274_time_cli_l.copy(
future_274_time_cli_adjusted_l['ds'] = pd.to_datetime(future_
future_274_time_cli_adjusted_l = future_274_time_cli_adjuste
future_274_time_cli_adjusted_l = future_274_time_cli_adjuste
#future_274_time_cli_adjusted_l.reset_index(drop=True)
future_274_time_cli_adjusted_l
```

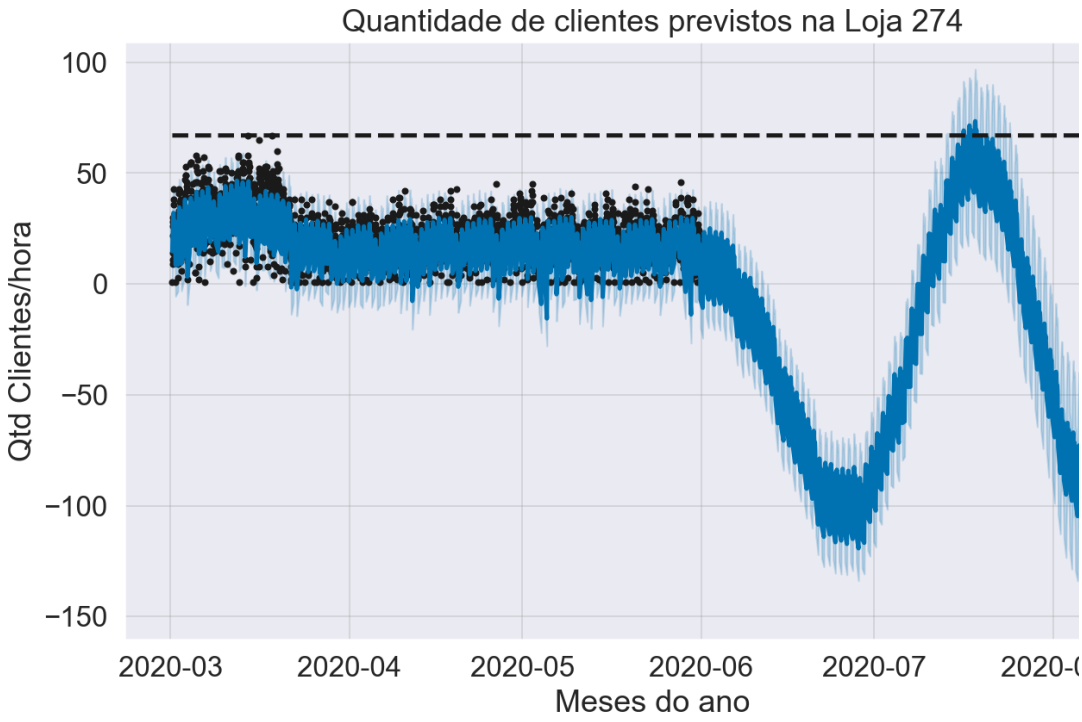
Out[64]:

	ds	cap
ds		
2020-03-01 07:30:00	2020-03-01 07:30:00	67
2020-03-01 08:00:00	2020-03-01 08:00:00	67
2020-03-01 08:30:00	2020-03-01 08:30:00	67
2020-03-01 09:00:00	2020-03-01 09:00:00	67
2020-03-01 09:30:00	2020-03-01 09:30:00	67
...	...	...
2020-08-14 16:30:00	2020-08-14 16:30:00	67
2020-08-14 17:30:00	2020-08-14 17:30:00	67
2020-08-14 18:30:00	2020-08-14 18:30:00	67
2020-08-14 19:30:00	2020-08-14 19:30:00	67
2020-08-14 20:30:00	2020-08-14 20:30:00	67

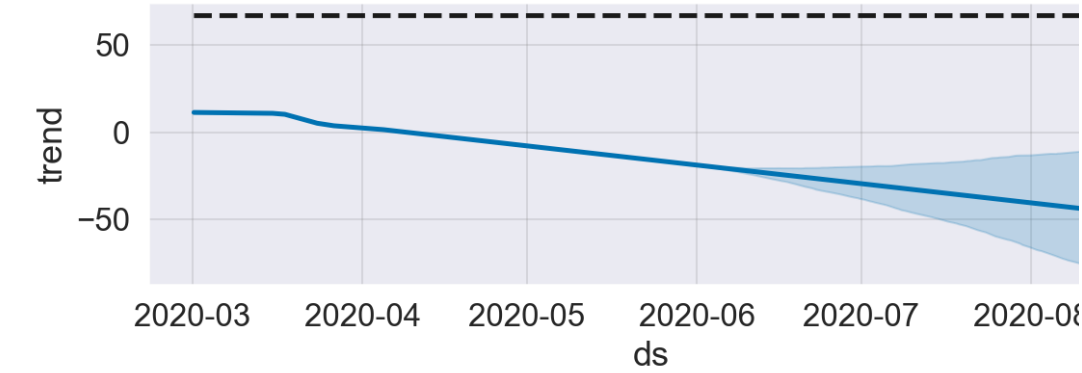
3755 rows × 2 columns

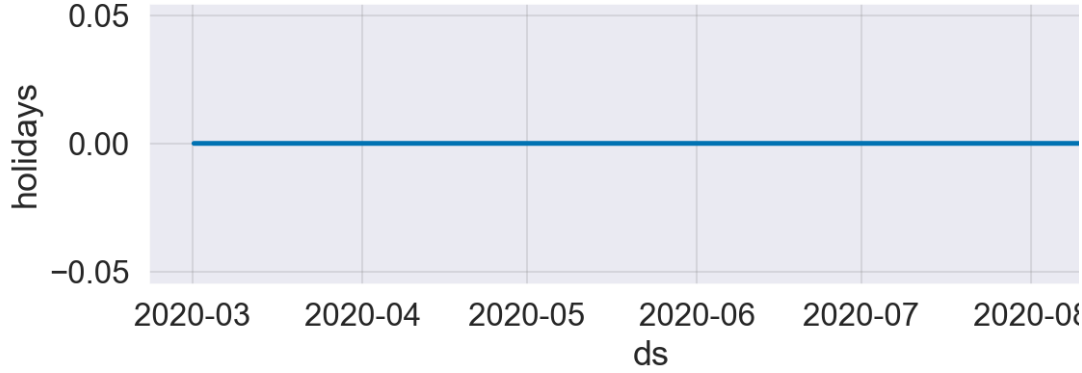
In [65]:
```
forecast_time_cli_l = m_274_time_cli_linear.predict(future_2
fig_time_cli_a = m_274_time_cli_linear.plot(forecast_time_cl
plt.title('Quantidade de clientes previstos na Loja 274')
```

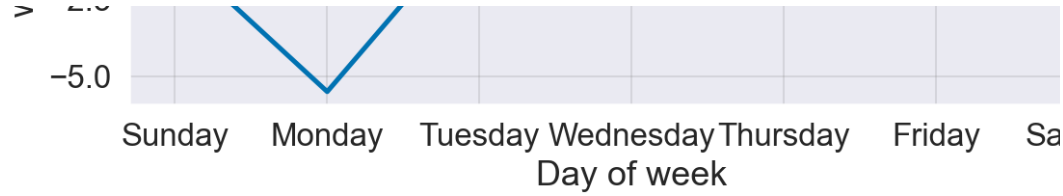
Out[65]:
```
Text(0.5, 1.0, 'Quantidade de clientes previstos na Loja 274
```

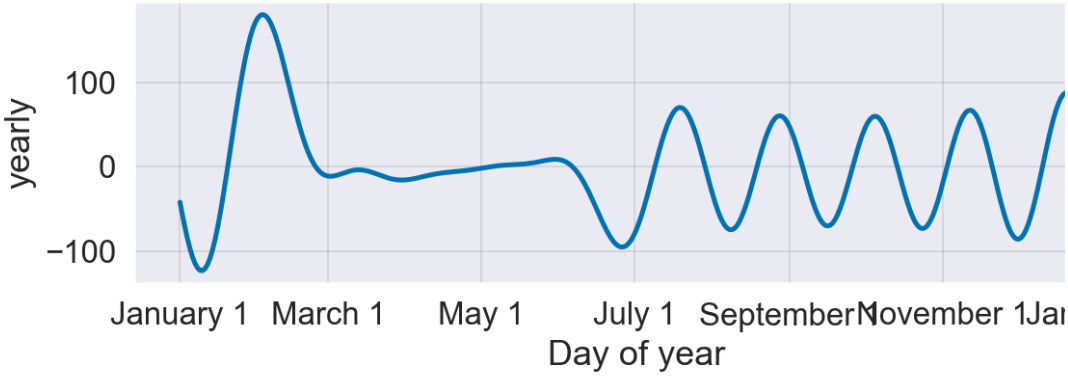


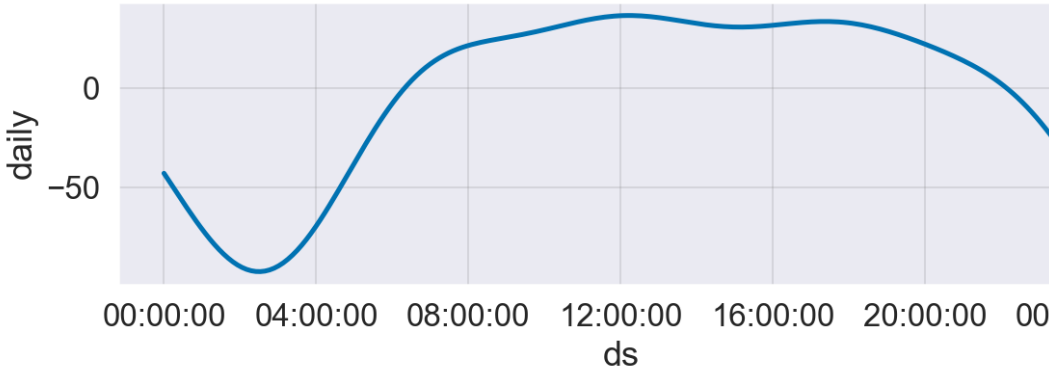
In [66]:
```
fig_comp_time_cli_a = m_274_time_cli_linear.plot_components(
```











Visualizando os dados previstos para o futuro

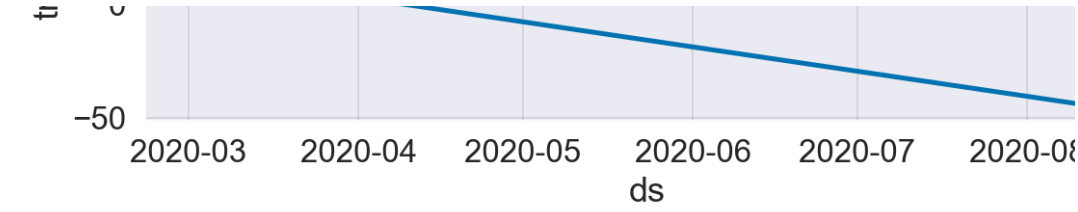
In [67]:
```
future_no_cli_pred_l = forecast_time_cli_l.loc[forecast_time_
future_no_cli_pred_l = future_no_cli_pred_l[['ds', 'yhat', '
future_no_cli_pred_l.sample(5)
```

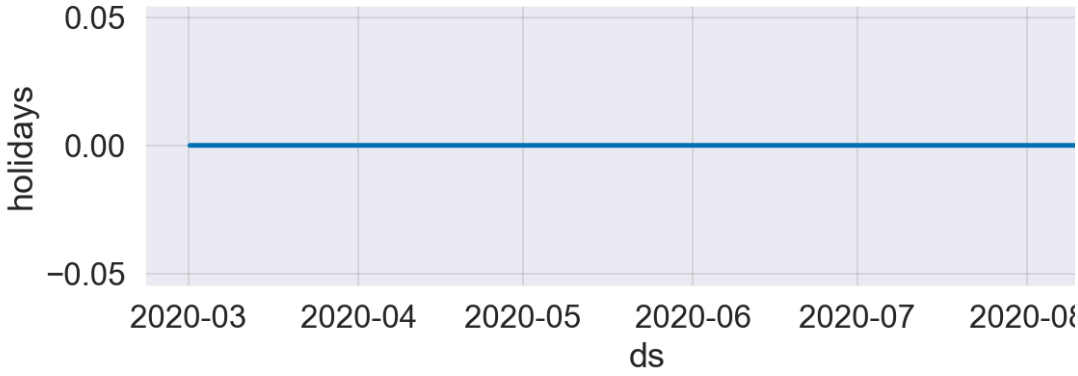
Out[67]:

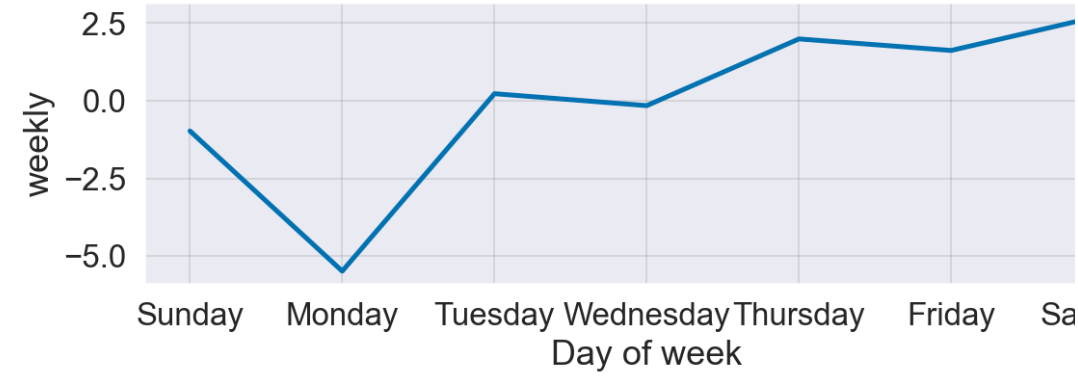
	ds	yhat	yhat_lower	yhat_upper
2863	2020-06-20 09:30:00	-74.179197	-87.283416	-60.258193
2764	2020-06-14 06:30:00	-62.777725	-75.328535	-49.645391
2853	2020-06-19 15:30:00	-66.961611	-80.268589	-53.551589
3160	2020-07-08 18:30:00	-1.698264	-18.063591	15.564830
2988	2020-06-28 06:30:00	-119.182329	-133.792152	-103.629670

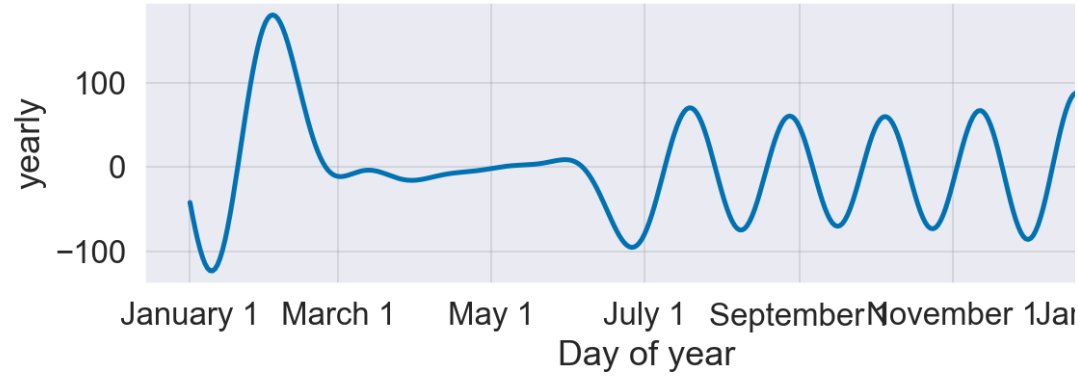
Analisando o gráfico sem o parâmetro de incerteza

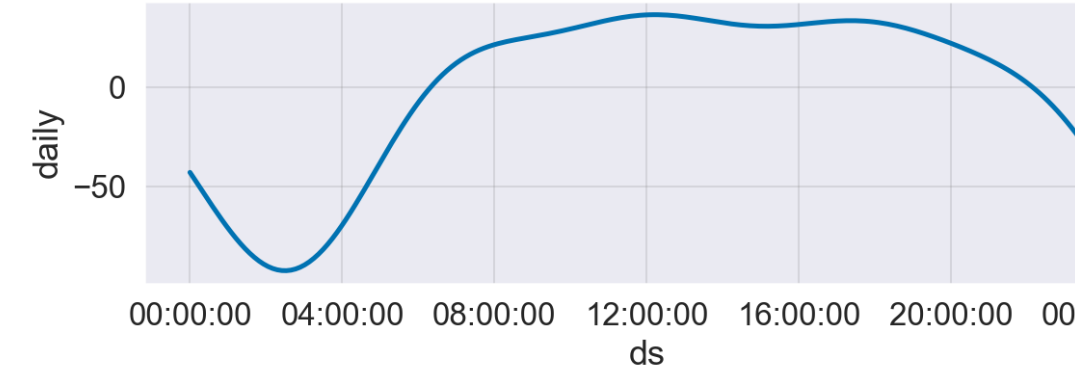
In [68]:
```
fig_comp_time_cli_a_uncertainty_false = m_274_time_cli_linea
```











Visualizando a previsão para 5 anos

In [69]:
```
future_274_time_cli_l = m_274_time_cli_linear.make_future_da
future_274_time_cli_l['cap'] = 67
teste_future_274_time_cli_l = future_274_time_cli_l.loc[futu
teste_future_274_time_cli_l
```

Out[69]:

	ds	cap
2555	2020-06-01 20:30:00	67
2556	2020-06-02 20:30:00	67
2557	2020-06-03 20:30:00	67
2558	2020-06-04 20:30:00	67
2559	2020-06-05 20:30:00	67
...	...	...
4350	2025-05-01 20:30:00	67
4351	2025-05-02 20:30:00	67
4352	2025-05-03 20:30:00	67
4353	2025-05-04 20:30:00	67
4354	2025-05-05 20:30:00	67

1800 rows × 2 columns

In [70]:
```
future_274_time_cli_adjusted_l = future_274_time_cli_l.copy(
future_274_time_cli_adjusted_l['ds'] = pd.to_datetime(future_
future_274_time_cli_adjusted_l = future_274_time_cli_adjuste
future_274_time_cli_adjusted_l = future_274_time_cli_adjuste
#future_274_time_cli_adjusted_l.reset_index(drop=True)
future_274_time_cli_adjusted_l
```

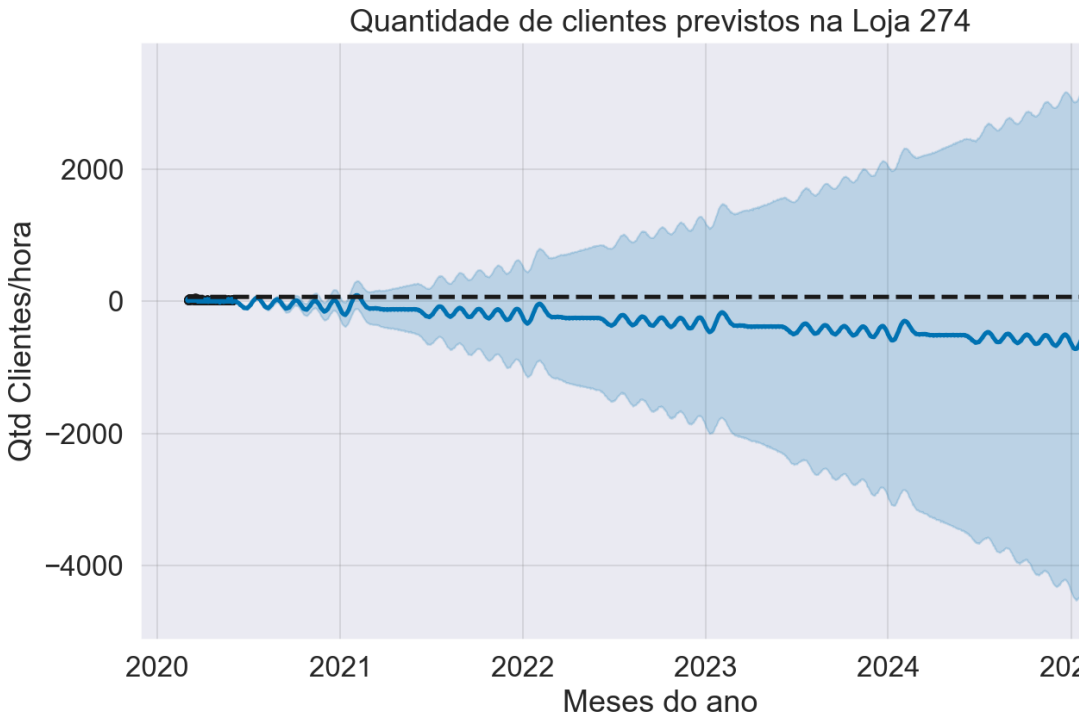
Out[70]:

	ds	cap
ds		
2020-03-01 07:30:00	2020-03-01 07:30:00	67
2020-03-01 08:00:00	2020-03-01 08:00:00	67
2020-03-01 08:30:00	2020-03-01 08:30:00	67
2020-03-01 09:00:00	2020-03-01 09:00:00	67
2020-03-01 09:30:00	2020-03-01 09:30:00	67
...	...	...
2025-05-01 20:30:00	2025-05-01 20:30:00	67
2025-05-02 20:30:00	2025-05-02 20:30:00	67
2025-05-03 20:30:00	2025-05-03 20:30:00	67
2025-05-04 20:30:00	2025-05-04 20:30:00	67
2025-05-05 20:30:00	2025-05-05 20:30:00	67

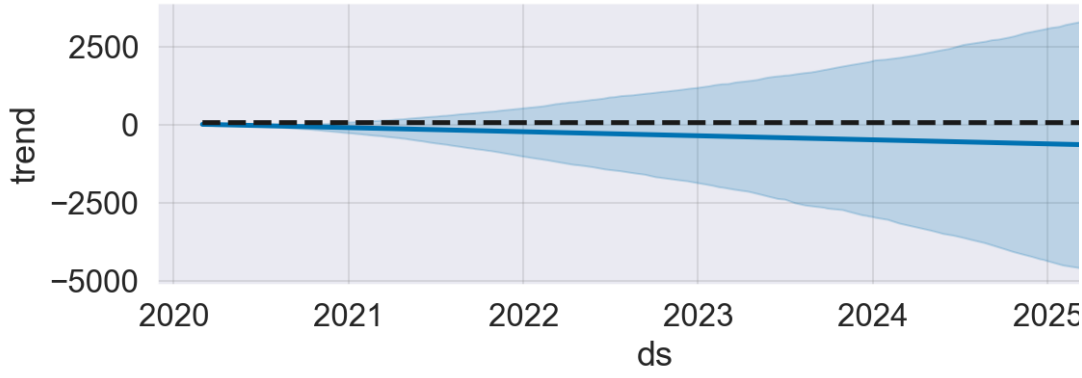
4355 rows × 2 columns

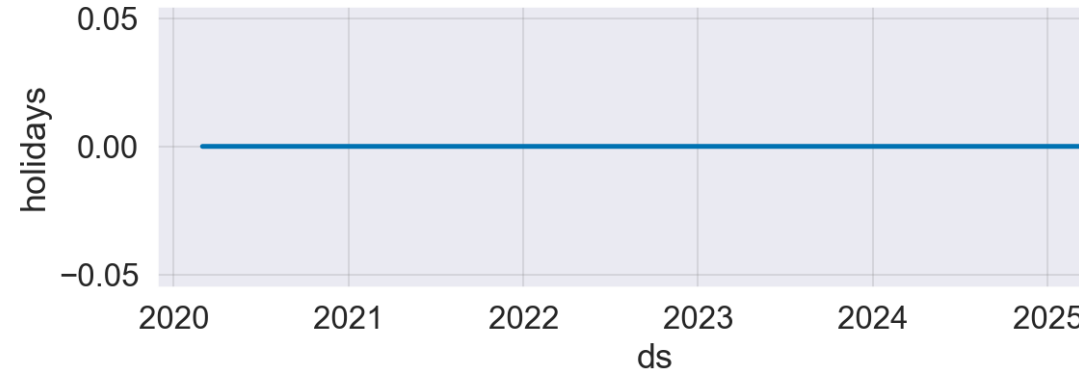
In [71]:
```
forecast_time_cli_l = m_274_time_cli_linear.predict(future_2
fig_time_cli_a = m_274_time_cli_linear.plot(forecast_time_cl
plt.title('Quantidade de clientes previstos na Loja 274')
```

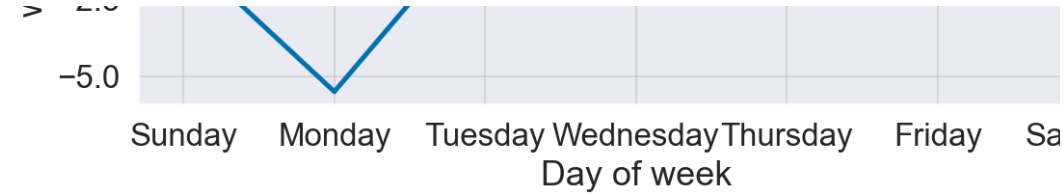
Out[71]:
```
Text(0.5, 1.0, 'Quantidade de clientes previstos na Loja 274
```

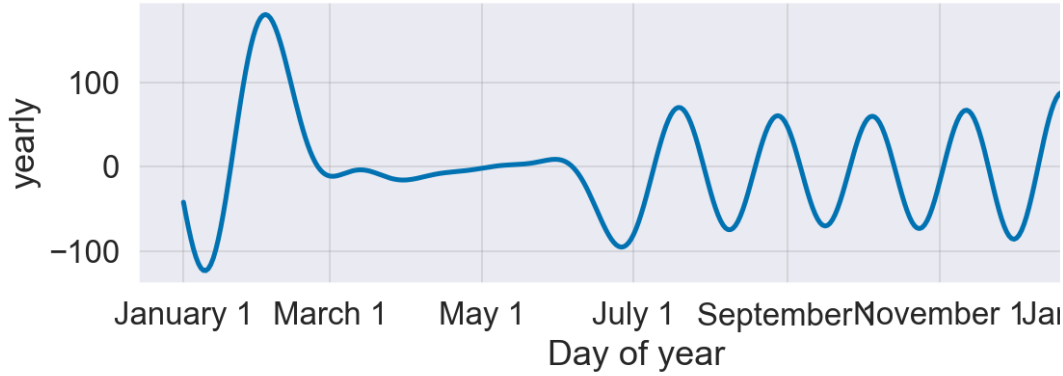


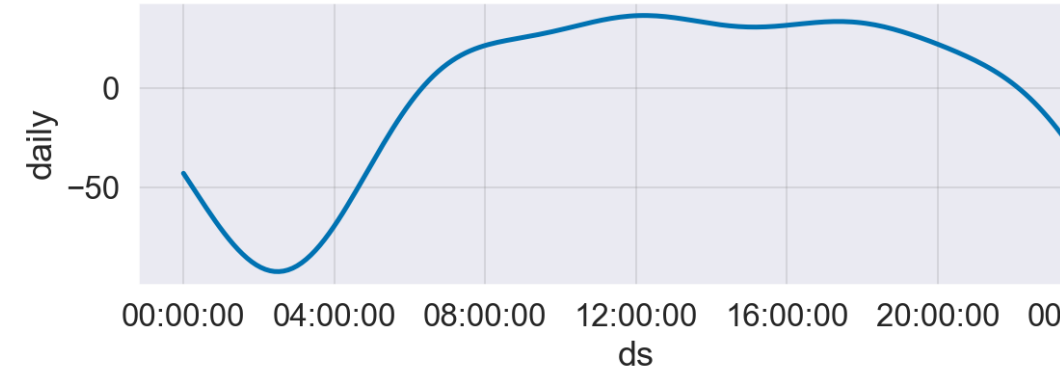
In [72]:
```
fig_comp_time_cli_a = m_274_time_cli_linear.plot_components(
```











Visualizando os dados previstos para o futuro

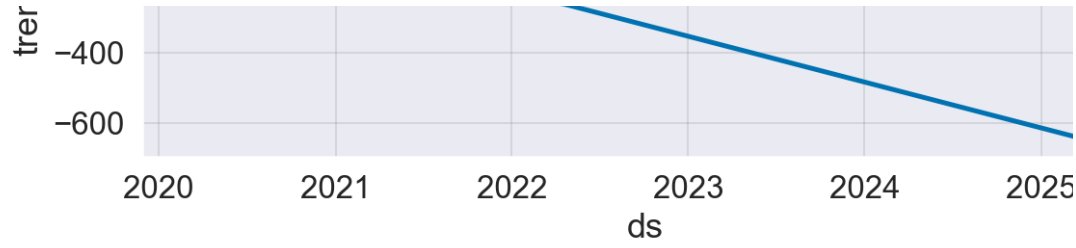
In [73]:
```
future_no_cli_pred_l = forecast_time_cli_l.loc[forecast_time_
future_no_cli_pred_l = future_no_cli_pred_l[['ds', 'yhat', '
future_no_cli_pred_l.sample(5)
```

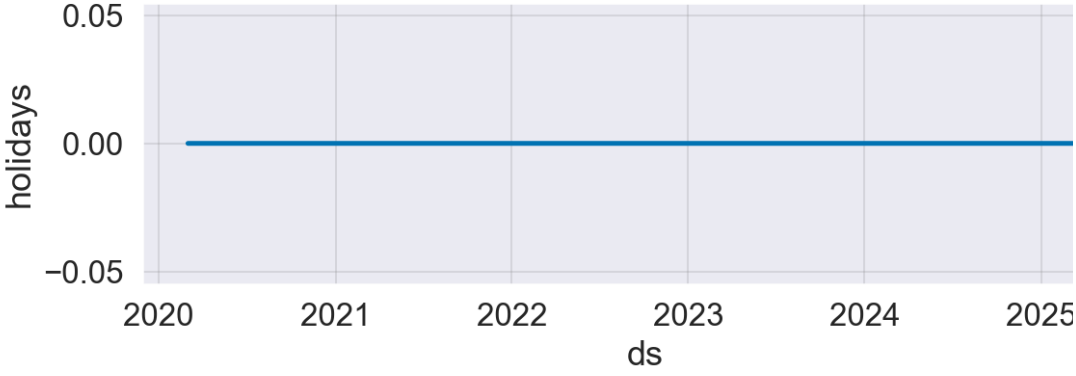
Out[73]:

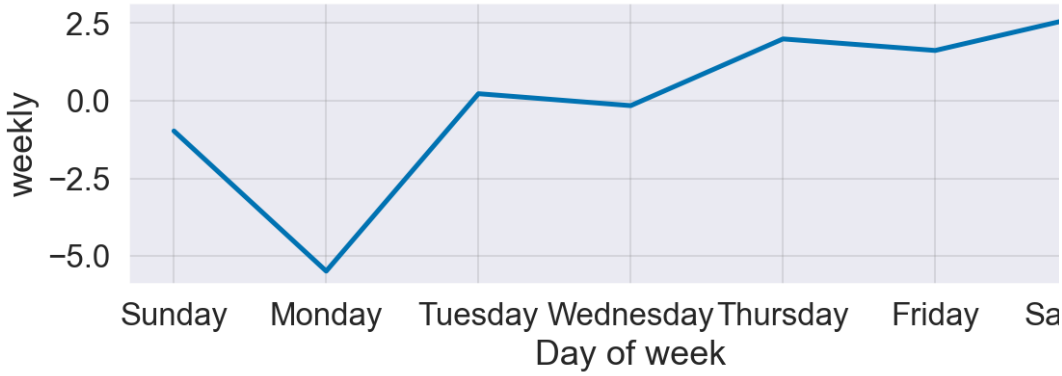
	ds	yhat	yhat_lower	yhat_upper
2871	2021-04-13 20:30:00	-120.920686	-404.334406	195.051554
2681	2020-10-05 20:30:00	12.995277	-62.823963	87.698411
4158	2024-10-21 20:30:00	-641.141190	-4125.764010	2802.292204
3232	2022-04-09 20:30:00	-251.488222	-1244.814470	744.652201
3761	2023-09-20 20:30:00	-482.216721	-2689.713407	1739.071594

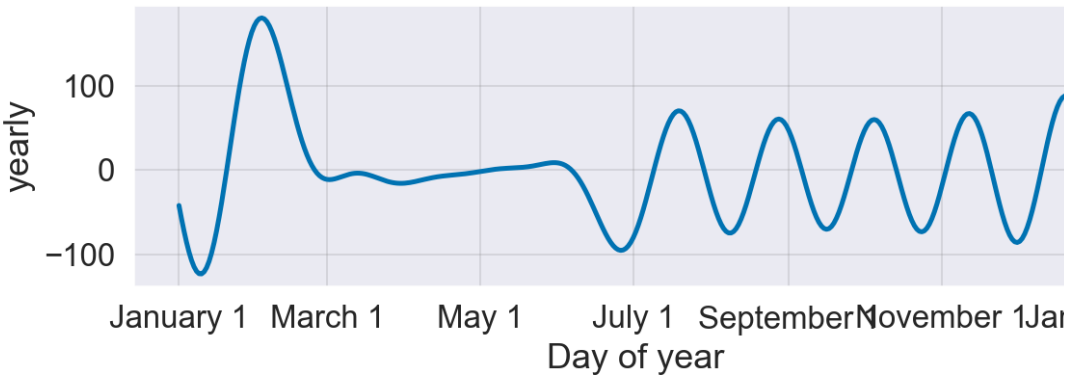
Analisando o gráfico sem o parâmetro de incerteza

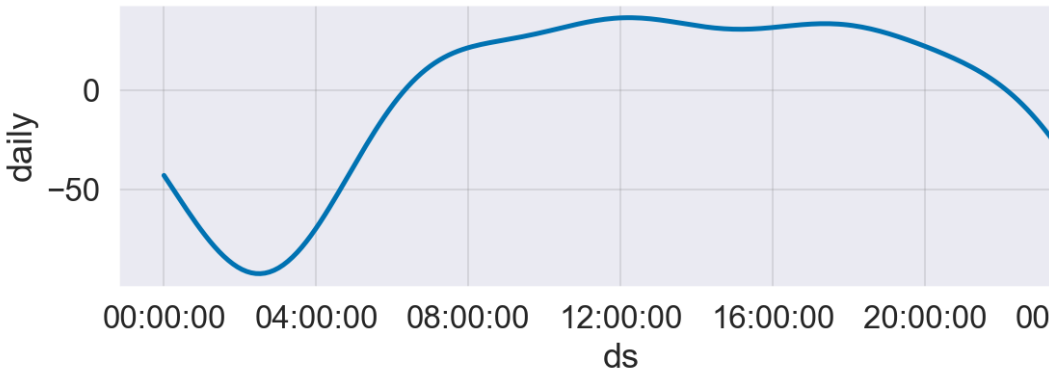
In [74]:
```
fig_comp_time_cli_a_uncertainty_false = m_274_time_cli_linea
```


trer
−400
−600
2020
2021
2022
2023
2024
2025
ds


holidays
0.05
0.00
−0.05
2020
2021
2022
2023
2024
2025
ds


weekly
2.5
0.0
−2.5
−5.0
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Sa
Day of week


yearly
100
0
−100
January 1
March 1
May 1
July 1
September
November 1
Jar
Day of year


daily
0
−50
00:00:00
04:00:00
08:00:00
12:00:00
16:00:00
20:00:00
00
ds

In [ ]: