

**Centro Federal de Educação Tecnológica Celso Suckow da Fonseca -  
CEFET/RJ  
Campus Maria da Graça**

**João Tentis, Pedro Lyrio, Jorge Harbes e Eduardo Torres.**

**Dashboard das Commodities**

Descrição e funcionalidades

**Rio de Janeiro  
2023**

<b>1. Introdução .....</b>	<b>5</b>
<b>2. Tecnologias .....</b>	<b>5</b>
<b>3. Funcionalidades.....</b>	<b>6</b>
3.1 Barra lateral.....	6
3.1.1 Filtro.....	6
3.1.2 Botão de heatmap .....	6
3.1.3 Botão de tamanho .....	7
3.2 Abas .....	7
3.2.1 Primeira aba.....	8
3.2.2 Segunda aba.....	9
3.2.3 Terceira aba .....	11
3.2.4 Quarta aba .....	12
3.3 Outras funções .....	12
<b>4. Interface do aplicativo.....</b>	<b>14</b>
<b>5. Conclusão .....</b>	<b>16</b>

## 1. Introdução

O aplicativo “Dashboard das Commodities”, foi desenvolvido para servir como uma forma de monitoramento e acompanhamento da variância das *commodities* (mercadorias primárias vendidas em larga escala e negociadas em bolsas de valores).

## 2. Tecnologias

- **Python** - Linguagem de programação utilizada e escolhida por 2 motivos: é uma boa opção quando se trata de manipulação de dados e trabalha muito bem com outras tecnologias que foram escolhidas.
- **Streamlit** - Biblioteca de Python que facilita muito a vida para a criação de aplicativos web para ciência de dados e *machine learning*, sendo fácil de utilizar, altamente responsivo e interativo.
- **Yfinance** - Biblioteca que serve para acessar as commodities e ativos presentes no Yahoo Finance.
- **Pandas** - Biblioteca de Python que é comumente usada para ciência de dados, análises e *machine learning*.
- **Matplotlib** - Biblioteca de Python usada de diversos jeitos para visualização de gráficos e análise de dados no geral.
- **Datetime** - Biblioteca utilizada para controle e manipulação de datas.
- **PIL (Python Imaging Library)** - Biblioteca de Python que serve para manipulação de imagens.

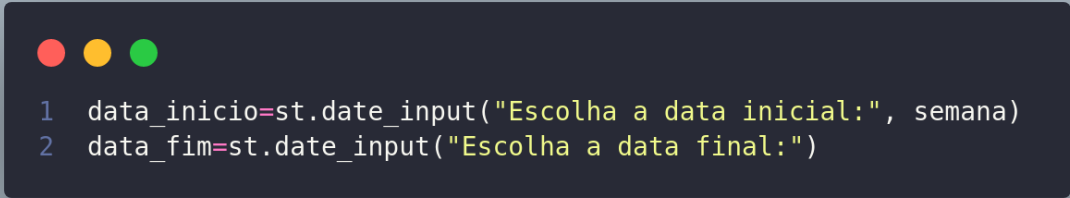
### 3. Funcionalidades

#### 3.1 Barra lateral

Presente no aplicativo como uma central de comando, a barra lateral detém funcionalidades como filtro, botão de heatmap e botão de tamanho. As opções escolhidas em cada função, afetam o aplicativo por inteiro.

##### 3.1.1 Filtro

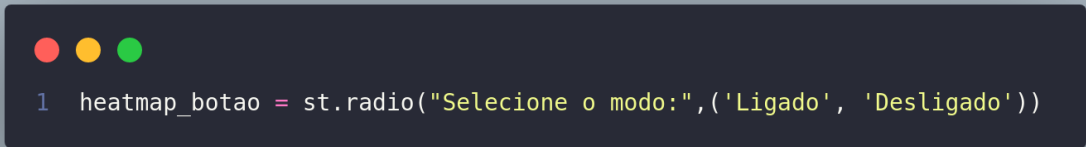
O filtro do aplicativo é um *input* de data, que serve para selecionar a data final e a data inicial em que os dados serão resgatados do *yfinance*. Representado no aplicativo pela linha de código abaixo:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains two lines of Python code.

```
1 data_inicio=st.date_input("Escolha a data inicial:", semana)
2 data_fim=st.date_input("Escolha a data final:")
```

##### 3.1.2 Botão de *heatmap*

O botão de heatmap serve como uma opção para ativação ou desativação do mapa de calor das tabelas que correlacionam commodities e ativos. Representado no aplicativo pela linha de código abaixo:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains one line of Python code.

```
1 heatmap_botao = st.radio("Selecione o modo:", ('Ligado', 'Desligado'))
```

Condicionado na tabela da aba 3, exemplificado abaixo:

```
1 if heatmap_botao == 'Ligado':
2     if tabela_botao == 'Grande':
3         cmap=plt.cm.get_cmap('RdYlGn')
4         st.table(corr_commodities_tudo.style.background_gradient(cmap=cmap,vmin=(-1),vmax=1, axis=None))
5     else:
6         cmap=plt.cm.get_cmap('RdYlGn')
7         st.dataframe(corr_commodities_tudo.style.background_gradient(cmap=cmap,vmin=(-1),vmax=1, axis=None))
8     else:
9         if tabela_botao == 'Compacta':
10            st.dataframe(r_pd_commodities_tudo.corr())
11        else:
12            st.table(r_pd_commodities_tudo.corr())
```

### 3.1.3 Botão de tamanho

Botão de tamanho serve para mudar entre 2 estilos de tabela: compacta e grande. Esta função foi adicionada com o intuito de facilitar a visualização dos dados na tabela. Representado no aplicativo pela linha de código mostrada anteriormente, no tópico acima.

## 3.2 Abas

O aplicativo foi dividido em abas, contendo o total de 4 abas. Cada uma delas detém um conteúdo exclusivo e independente, uma não interfere com a outra.

### 3.2.1 Primeira aba

A primeira aba do aplicativo, chamada de “Gráfico Geral”, dá início a coleta de dados. Como o nome já diz, aqui temos uma visão geral de todas as commodities, de acordo com a data escolhida. Representado no aplicativo pela linha de código abaixo:

```

1  with tab1:
2      st.header("TABELA")
3      if tabela_botao == 'Grande':
4          st.table(r_pd_commodities_tudo)
5      else:
6          st.dataframe(r_pd_commodities_tudo)
7
8      st.divider()
9
10     # plotando
11     st.header("GRÁFICO")
12     grafico_botao=st.radio("Selecione o modo do gráfico:",('Completo', 'Único'))
13     if grafico_botao == 'Completo':
14         st.line_chart(r_pd_commodities_tudo)
15     else:
16         grafico_all=r_pd_commodities_tudo.columns.tolist()
17         options_key = "_".join(grafico_all)
18         selecao_grafico=st.multiselect('Selecione as commodities para plotar:', options=grafico_all)
19         st.write("")
20         st.line_chart(r_pd_commodities_tudo[selecao_grafico])
21
22
23     with st.expander("Ver explicação"):
24         st.write("O gráfico acima mostra a variação de preço (em :green[U$]), das :orange[COMMODITIES].")

```

### 3.2.2 Segunda aba

A segunda aba do aplicativo se chama “Report Semanal”, que traz algumas informações importantes sobre as commodities. Representado por uma tabela, tem-se: preço de abertura, fechamento, preço mais alto, mais baixo e a diferença percentual do preço de abertura com o de fechamento. Representado no aplicativo pela linha de código abaixo:

```

1  with tab2:
2      st.header("REPORT SEMANAL")
3      week_texto=data_fim-timedelta(days=7)
4      st.write(week_texto, 'a', data_fim)
5      st.write("")
6      tickers_report=r_tickers_hist.stack(level=1).rename_axis(['Data', 'Tickers']).reset_index(level=1)
7      report_semanal=tickers_report.drop(['Dividends', 'Volume', 'Stock Splits'], axis=1)
8      report_semanal['Resultado']=(report_semanal['Close'] - report_semanal['Open'])/report_semanal['Open']*100
9      report_semanal=report_semanal[['Tickers', 'Open', 'High', 'Low', 'Close', 'Resultado']]
10     report_semanal.rename(columns={'Tickers':'Commodities'}, inplace=True)
11     report_semanal=report_semanal.sort_values('Resultado', ascending=True)
12     download_report=report_semanal
13     if tabela_botao == 'Compacta':
14         st.dataframe(report_semanal)
15         st.divider()
16     else:
17         st.table(report_semanal)
18         st.divider()

```

Logo abaixo, as linhas de código do gráfico, que mostra apenas os dados da aba “Resultado”, que consiste na diferença percentual explicada acima.

```
1 data = yf.download(list(dif_percentual.keys()), start=data_inicio, end=data_fim)['Close']
2
3 weekly_returns = data.pct_change( periods=1 ) * 100
4 sorted_returns = weekly_returns.iloc[-1].sort_values(ascending=False)
5 sorted_indices = sorted_returns
6
7 positive_returns = sorted_returns[sorted_returns >= 0]
8 negative_returns = sorted_returns[sorted_returns < 0]
9
10 fig, ax = plt.subplots(figsize=(5, 8))
11 ax.barh([dif_percentual[idx] for idx in positive_returns.index], positive_returns, color='green')
12 ax.barh([dif_percentual[idx] for idx in negative_returns.index], negative_returns, color='red')
13 ax.axvline(x=0, color='white', linestyle='--')
14 plt.xlabel('Variação Percentual')
15 plt.ylabel('Commodities')
16 fig.set_figwidth(15)
17 ax.set_facecolor('#0e1117')
18 fig.set_facecolor('#0e1117')
19 ax.tick_params(colors='white')
20 ax.xaxis.label.set_color('white')
21 ax.yaxis.label.set_color('white')
22 ax.spines['bottom'].set_color('w')
23 ax.spines['top'].set_color('w')
24 ax.spines['left'].set_color('w')
25 ax.spines['right'].set_color('w')
26
27 # Adiciona as porcentagens ao lado de cada barra
28 for i, (index, value) in enumerate(zip(positive_returns.index, positive_returns)):
29     ax.text(0, i, f'{value:.2f}%', ha='right', va='center', color='white', fontweight='bold')
30     y = i
31 for i, (index, value) in enumerate(zip(negative_returns.index, negative_returns)):
32     ax.text(0, y+1+i, f'{value:.2f}%', ha='left', va='center', color='white', fontweight='bold')
33
34 st.pyplot(plt.show())
```

### 3.2.3 Terceira aba

A terceira aba do aplicativo, tem como função a correlação das *commodities*, que serve para uma melhor visualização do que aumentou e o que diminuiu, baseando-se uma nas outras. Representado no aplicativo pela linha de código abaixo:

```

1 with tab3:
2     st.header("CORRELAÇÃO")
3
4     # mostrando o dataframe da correlação e colocando heatmap
5     corr_commodities_tudo=r_pd_commodities_tudo.corr()
6     download_all=r_pd_commodities_tudo.corr()
7     if heatmap_botao == 'Ligado':
8         if tabela_botao == 'Grande':
9             cmap=plt.cm.get_cmap('RdYlGn')
10            st.table(corr_commodities_tudo.style.background_gradient(cmap=cmap,vmin=(-1),vmax=1, axis=None))
11        else:
12            cmap=plt.cm.get_cmap('RdYlGn')
13            st.dataframe(corr_commodities_tudo.style.background_gradient(cmap=cmap,vmin=(-1),vmax=1, axis=None))
14    else:
15        if tabela_botao == 'Compacta':
16            st.dataframe(r_pd_commodities_tudo.corr())
17        else:
18            st.table(r_pd_commodities_tudo.corr())
19
20 with st.expander("Ver explicação"):
21     st.write("O DataFrame acima mostra a correlação das :orange[COMMODITIES].")

```

### 3.2.4 Quarta aba

A quarta e última aba, tem uma função parecida com a terceira aba, que é correlacionar commodities. Aqui, por outro lado, temos uma correlação selecionada, onde o usuário escolhe as *commodities* para correlacionar, adicionando ativos na correlação se desejar. Representado no aplicativo pela linha de código abaixo:

```

1 with tab4:
2     st.header("CORRELAÇÃO SELECIONADA")
3     st.write("Selecione pelo menos 2 :orange[COMMODITIES] para correlação!")
4     st.text("")
5     # fazendo o multiselect
6     todas_colunas = r_pd_commodities_tudo2.columns.tolist()
7     options_key = " ".join(todas_colunas)
8     colunas_selecionadas = st.multiselect('', options=todas_colunas)
9
10    if colunas_selecionadas:
11        # r_pd_commodities_tudo2[colunas_selecionadas]
12        colunas_corr = r_pd_commodities_tudo2[colunas_selecionadas]
13        color_change_colunas_corr=colunas_corr.corr()
14        if heatmap_botao == 'Ligado':
15            if tabela_botao == 'Grande':
16                cmap=plt.cm.get_cmap('RdYlGn')
17                st.table(color_change_colunas_corr.style.background_gradient(cmap=cmap,vmin=(-1),vmax=1, axis=None))
18            else:
19                cmap=plt.cm.get_cmap('RdYlGn')
20                st.dataframe(color_change_colunas_corr.style.background_gradient(cmap=cmap,vmin=(-1),vmax=1, axis=None))
21        else:
22            if tabela_botao == 'Compacta':
23                st.dataframe(color_change_colunas_corr)
24            else:
25                st.table(color_change_colunas_corr)
26    else:
27        st.write('')

```



### 3.3 Outras funções

O aplicativo também conta com algumas funções presentes por ele todo, como *expander*, botão de download e o hiperlink da fonte dos dados.

- **Expander** - O *expander* tem como função adicionar uma mini explicação do que cada aba é responsável por fazer. Facilita o entendimento do usuário quanto ao que aparece. Abaixo, a linha de código do *expander* da primeira aba como exemplo:

```
1 with st.expander("Ver explicação"):
2     st.write("Acima, uma tabela e um gráfico que mostram a variação de preço (em :green[US$]), das :orange[COMMODITIES].")
```

- **Botão de download** - O botão de download, como o nome já diz, serve para baixar os dados que o usuário visualiza em cada aba. Caso exista um filtro na aba, o download será feito com o filtro aplicado. Abaixo, a linha de código do botão de download da primeira aba como exemplo:

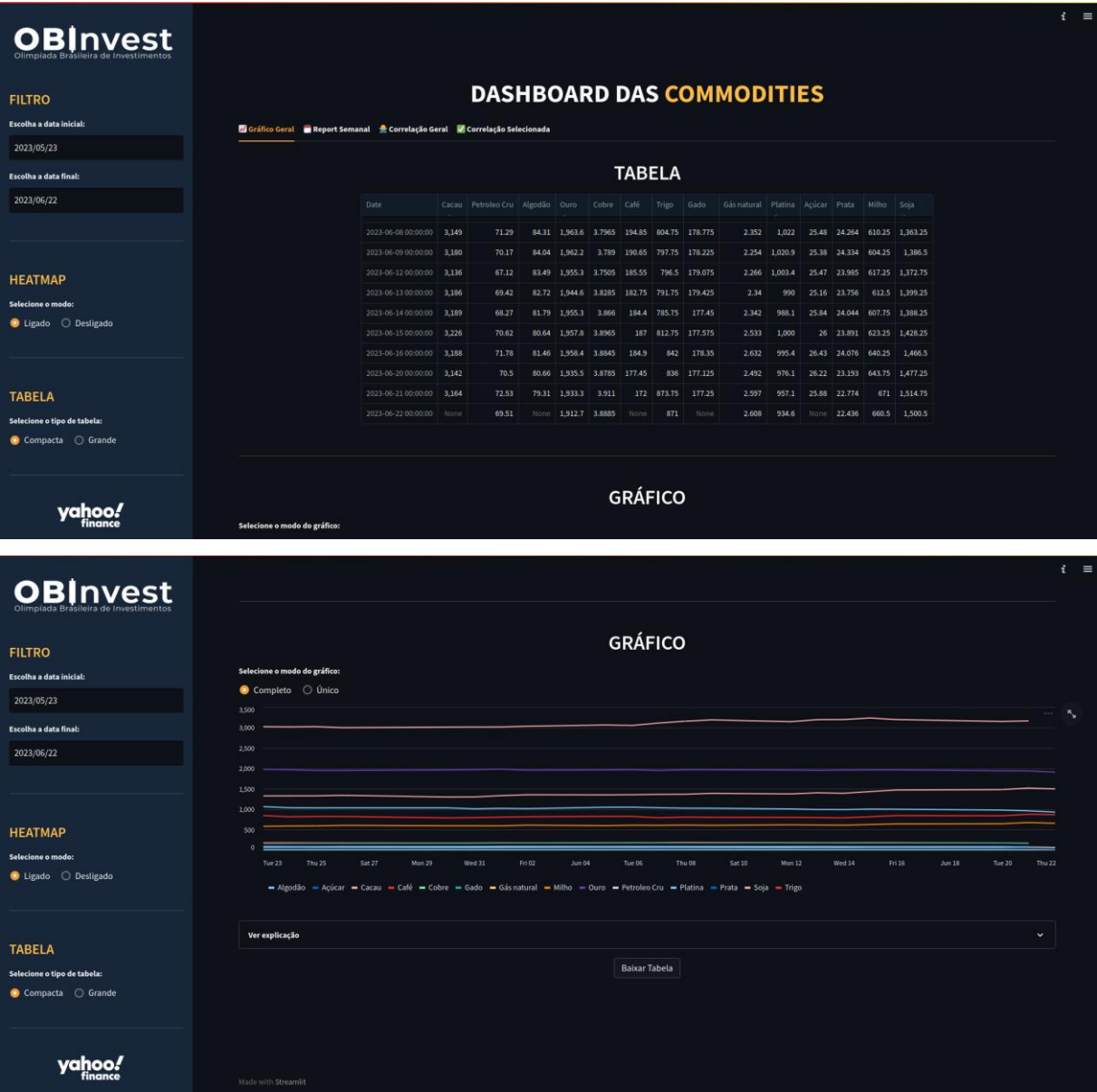
```
1 st.download_button("Baixar Tabela",
2                     r_pd_commodities_tudo.to_csv(),
3                     file_name='commodities_dados.csv',
4                     mime='text/csv')
```

- **Hyperlink da fonte** - Presente no rodapé da barra lateral, temos uma imagem que serve como hyperlink, levando para o site de onde foi tirado os dados das *commodities*. Abaixo, a linha de código desta funcionalidade:

```
1 st.markdown("[!Fonte](https://public.flourish.studio/uploads/4e293af7-8464-45d7-9428-a96963909e42.png)](https://finance.yahoo.com/commodities/)")
```

## 4. Interface do aplicativo

Agora, a interface em si do aplicativo, separado pelas abas.



## FILTRO

Escolha a data inicial:

2023/05/23

Escolha a data final:

2023/06/22

## HEATMAP

Selecione o modo:

☒ Ligado ☐ Desligado

## TABELA

Selecione o tipo de tabela:

☒ Compacta ☐ Grande

## DASHBOARD DAS COMMODITIES

☒ Gráfico Geral ☒ Report Semanal ☐ Correlação Geral ☐ Correlação Seleccionada

### REPORT SEMANAL

2023-06-15 a 2023-06-22

Data	Commodities	Open	High	Low	Close	Resultado
2023-06-19 00:00:00	Açúcar	26.25	26.38	24.9	24.93	-5.0286
2023-06-19 00:00:00	Café	178.9	178.95	168.95	171.7	-4.0246
2023-06-19 00:00:00	Platina	957.1	957.1	934.6	934.6	-2.3509
2023-06-19 00:00:00	Petroleo Cru	78.9	72.72	68.93	69.51	-1.9605
2023-06-19 00:00:00	Algodão	80.66	80.92	78	79.29	-1.6985
2023-06-19 00:00:00	Prata	22.774	22.774	22.436	22.436	-1.4841
2023-06-19 00:00:00	Ouro	1,935.6	1,936.8	1,912.1	1,912.7	-1.1831
2023-06-19 00:00:00	Cacau	3,131	3,182	3,104	3,138	0.2236
2023-06-19 00:00:00	Gado	177.2	177.9	176.25	177.65	0.2539
2023-06-19 00:00:00	Cobre	3.86	3.95	3.86	3.885	0.7383

## FILTRO

Escolha a data inicial:

2023/05/24

Escolha a data final:

2023/06/23

## HEATMAP

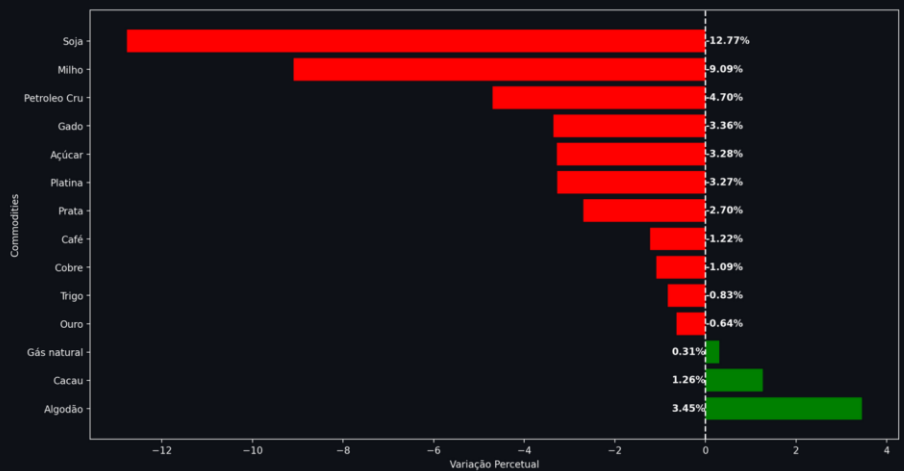
Selecione o modo:

☒ Ligado ☐ Desligado

## TABELA

Selecione o tipo de tabela:

☒ Compacta ☐ Grande



Ver explicação

Baixar Tabela

## FILTRO

Escolha a data inicial:

2023/05/23

Escolha a data final:

2023/06/22

## HEATMAP

Selecione o modo:

☒ Ligado ☐ Desligado

## TABELA

Selecione o tipo de tabela:

☒ Compacta ☐ Grande

## DASHBOARD DAS COMMODITIES

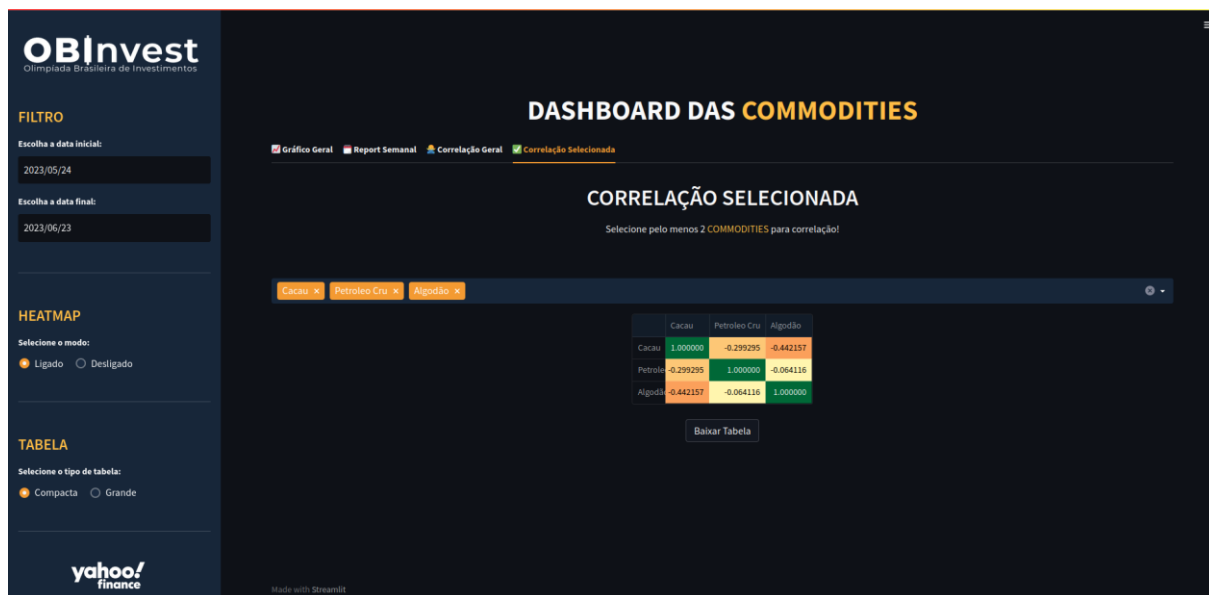
☐ Gráfico Geral ☐ Report Semanal ☒ Correlação Geral ☐ Correlação Seleccionada

### CORRELAÇÃO

	Cacau	Petroleo Cru	Algodão	Ouro	Cobre	Café	Trigo	Gado	Gás natural	Platina	Açúcar	Prata	Milho	Soja
Cacau	1.000000	-0.295801	-0.446126	-0.284463	-0.870173	0.222765	0.069074	0.726360	0.616138	-0.642998	0.536116	0.513026	0.649867	0.777398
Petroleo Cru	-0.295801	1.000000	-0.052273	0.035197	-0.257132	0.089707	0.396076	-0.311422	0.079340	0.408643	-0.084555	-0.282626	-0.114453	-0.051831
Algodão	-0.446126	-0.052273	1.000000	0.578329	-0.307005	0.261056	0.085588	-0.787949	0.607436	-0.645993	0.378262	-0.550186	-0.601404	-0.601404
Ouro	-0.284463	0.035197	0.578329	1.000000	-0.421720	0.508944	-0.496552	-0.177450	-0.554775	0.706334	-0.129353	0.650663	-0.706437	-0.649506
Cobre	0.870173	-0.257132	-0.307005	-0.421720	1.000000	-0.082170	0.361720	0.807389	0.614453	-0.693905	0.462728	0.197073	0.813812	0.870427
Café	0.222765	0.089707	0.261056	0.508944	-0.082170	1.000000	-0.319778	0.102617	-0.136966	0.437085	0.017197	0.671620	-0.410240	-0.233095
Trigo	0.069074	0.396076	-0.436359	-0.496552	0.361720	-0.319778	1.000000	-0.065024	0.629312	-0.441503	0.399866	-0.539962	0.661258	0.671162
Gado	0.726360	-0.311422	0.085588	-0.177450	0.807389	0.102617	-0.065024	1.000000	0.200969	-0.448064	-0.013528	0.558254	0.569188	0.583609
Gás natural	0.616138	0.079340	0.767949	-0.514776	0.614453	-0.136966	0.629312	0.200969	1.000000	-0.683954	0.741818	-0.296948	0.764286	0.812483
Platina	-0.642998	0.408643	0.007436	0.708334	-0.693905	0.437085	-0.441503	-0.448064	-0.683954	1.000000	-0.548660	0.269379	-0.863348	-0.826409

Ver explicação

Baixar Tabela



## 5. Conclusão

Como mostrado e exemplificado no corpo do trabalho, o aplicativo executa e exerce o que é proposto. Nele, consegue-se ver: a variância das commodities, junto ao gráfico de linha para melhor visualização; Consegue-se ver o fechamento dos preços das *commodities* junto ao gráfico de diferença percentual; Consegue-se ver a correlação de *commodities* entre si e entre ativos.