

PROJETO WIKBANK





API UTILIZADA

- **YahooFinance**

Possui uma biblioteca (`yfinance`) no Python que faz a requisição e trata os dados em um DataFrame do Pandas



COMO FUNCIONA?

- Importamos a biblioteca YFinance

```
# Importando Yfinance para obter cotação das ações
import yfinance
```

- Chamamos a Função cotacao, que pegará o banco solicitado e verificará se ele tem no dicionario bancos_yahoo_finance, após isso ele pegará seu Simbolo De Ação correspondente e habilitará o botão de ver as informações das cotações.
- Após isso ele pegará a data inicial e final dos calendarios da interface e irá converter para que os mesmos fiquem na formatação certa para o YFinance

```
def cotacao(banco):  
  
    bancos_yahoo_finance = {  
        "nubank": "NU",  
        "itau": "ITUB4.SA",  
    }  
  
    if banco in bancos_yahoo_finance:  
  
        botao_cotacao.place_forget()  
        botao_cotacao.place(x=1168, y=75)  
        botao_cotacao.config( command=lambda: mostrar_tela_info("tela_cotacao"))  
  
        data_final = datetime.datetime.strptime(calendario_final.get_date(), "%d/%m/%Y").strftime("%Y-%m-%d")  
        data_inicial = datetime.datetime.strptime(calendario_inicial.get_date(), "%d/%m/%Y").strftime("%Y-%m-%d")  
  
        cotacao_acao = yfinance.download(bancos_yahoo_finance[banco], start=data_inicial, end=data_final)  
  
        cotacao_acao_grafico(cotacao_acao)  
  
        tabela_cotacao(cotacao_acao)  
    else:  
        botao_cotacao.place_forget()  
        mostrar_tela_info("tela_infoPrincipal")
```



- E enfim usamos o Yfinance para baixar a cotação das ações, dando a ele o simbolo das ações, data inicial e final, armazenando os dados em uma variavel

```
cotacao_acao = yfinance.download(bancos_yahoo_finance[banco], start=data_inicial, end=data_final)
```

- Após terminar de coletar os dados, ele chama outras duas funções, a de montar a tabela e o grafico, passando as informações coletadas.

```
def tabela_cotacao(cotacao_acao):
    global tabela_cotacaoAcoes

    if tabela_cotacaoAcoes == None:
        tabela_cotacaoAcoes = ttk.Treeview(frame_cotacao)
        tabela_cotacaoAcoes.place(relx=0.5, y=505, height=210, anchor='center')
    else:
        tabela_cotacaoAcoes.destroy()
        tabela_cotacaoAcoes = ttk.Treeview(frame_cotacao)
        tabela_cotacaoAcoes.place(relx=0.5, y=505, height=210, anchor='center')

    tabela_cotacaoAcoes["columns"] = tuple(['Data'] + list(cotacao_acao.columns))
    tabela_cotacaoAcoes["show"] = "headings"

    tabela_cotacaoAcoes.heading("Data", text="Data")
    for column in cotacao_acao.columns:
        tabela_cotacaoAcoes.heading(column, text=column)

    for column in tabela_cotacaoAcoes["columns"]:
        tabela_cotacaoAcoes.column(column, width=150)

    for index, row in cotacao_acao.iterrows():
        tabela_cotacaoAcoes.insert("", "end", values=tuple([index.strftime('%Y-%m-%d')] + list(row)))
```

```
info_legenda = None

def cotacao_acao_grafico(cotacao_acao):
    global info_legenda

    if info_legenda == None:
        info_legenda = Label(frame_cotacao, text="Data: | Fechamento: ", bg="white", relief="flat", font=("Ivy 12 bold"))
        info_legenda.place(relx=0.5, rely=0.5, anchor='center')

    graficoCotacao = plt.figure(figsize=(6, 4))
    ax = graficoCotacao.add_subplot(111)
    ax.plot(cotacao_acao.index, cotacao_acao['Close'], label='Fechamento')

    ax.xaxis.set_major_locator(AutoDateLocator())
    graficoCotacao.autofmt_xdate()

    canvas_graficoCotacao = FigureCanvasTkAgg(graficoCotacao, master=frame_cotacao)
    canvas_graficoCotacao.draw()
    canvas_graficoCotacao.get_tk_widget().place(relx=0.5, rely=0.23, anchor='center')

    info_legenda.config(text="Data: | Fechamento: ")
    info_legenda.lift()

    calendario_inicial.lift()
    calendario_final.lift()
    titulo_calendario_inicial.lift()
    titulo_calendario_final.lift()

    def cotacao_info(event):
        if event.inaxes:
            x = event.xdata
            mouse_timestamp = pd.Timestamp(num2date(x)).date()
            index_proximo = encontrar_indice_proximo(mouse_timestamp, cotacao_acao.index.date)
            data_proxima = cotacao_acao.index[index_proximo].strftime('%Y-%m-%d')
            fechamento_proximo = cotacao_acao['Close'].iloc[index_proximo]
            info_legenda.config(text=f'Data: {data_proxima} | Fechamento: {fechamento_proximo:.2f}')

    graficoCotacao.canvas.mpl_connect('motion_notify_event', cotacao_info)

def encontrar_indice_proximo(valor, cotacao):
    return np.abs(cotacao - valor).argmin()
```

CORREÇÕES DE PROBLEMAS

Elementos do TKINTER

Nosso programa utiliza o TKINTER para criação da interface, quando faziamos buscas com novos bancos, ao inves dele sobrepor as informações antigas, ele apenas colocava novas uma em cima da outra desnecessariamente!

Datas mostradas no Eixo X do gráfico das cotações das ações

Antes, as datas ficavam muito coladas e se sobrepondo caso o periodo colocado nos calendarios fossem muito curtas ou muito longas, agora o programa rotaciona as datas para elas não entram dentro da outra e coloca apenas algumas datas para não ficar gigantesco.



DEMONSTRANDO O
FUNCIONAMENTO
NA PRÁTICA E
COMPARANDO OS
DADOS.

