

```
# finsaude_ai.py
```

```
import streamlit as st
```

```
import yfinance as yf
```

```
import pandas as pd
```

```
import requests
```

```
import matplotlib.pyplot as plt
```

```
from datetime import datetime
```

```
from transformers import pipeline
```

```
from reportlab.pdfgen import canvas
```

```
# Configuração da página
```

```
st.set_page_config(page_title="FinSaúde AI", layout="wide")
```

```
# Título
```

```
st.title("📊 FinSaúde AI — Monitoramento Financeiro Inteligente")
```

```
# Sidebar para seleção de empresa
```

```
st.sidebar.header("🔍 Selecione uma empresa")
```

```
ticker = st.sidebar.text_input("Código da ação (ex: PETR4.SA)", value="PETR4.SA")
```

```
# Função para buscar dados financeiros
```

```
def get_financial_data(ticker):
```

```
    stock = yf.Ticker(ticker)
```

```
    info = stock.info
```

```
    balance = stock.balance_sheet
```

```
    income = stock.financials
```

```
    cashflow = stock.cashflow
```

```
return info, balance, income, cashflow
```

```
# Função para calcular indicadores
```

```
def calculate_indicators(info):
```

```
    try:
```

```
        roe = info['returnOnEquity']
```

```
        debt_ratio = info['debtToEquity']
```

```
        current_ratio = info['currentRatio']
```

```
        return roe, debt_ratio, current_ratio
```

```
    except:
```

```
        return None, None, None
```

```
# Função para gerar score de atratividade
```

```
def investment_score(roe, debt_ratio, current_ratio):
```

```
    score = 0
```

```
    if roe and roe > 0.15:
```

```
        score += 1
```


```
    if debt_ratio and debt_ratio < 1.5:
```

```
        score += 1
```


```
    if current_ratio and current_ratio > 1.2:
```

```
        score += 1
```

```
    if score == 3:
```

```
        return "Alta atratividade —  Comprar", "green"
```

```
    elif score == 2:
```

```
        return "Atratividade moderada —  Acompanhar", "orange"
```

```
    else:
```

```
        return "Baixa atratividade —  Evitar", "red"
```

Função para buscar notícias resumidas

```
def get_news_summary(ticker):
```

```
    summarizer = pipeline("summarization", model="sshleifer/distilbart-cnn-12-6")
```

```
    query = f"{ticker} notícias financeiras"
```

```
    url =
```

```
    f"https://newsapi.org/v2/everything?q={query}&apiKey=YOUR_NEWSAPI_KEY"
```

```
    response = requests.get(url).json()
```

```
    if response["status"] == "ok":
```

```
        articles = response["articles"][:3]
```

```
        summaries = []
```

```
        for article in articles:
```

```
            summary = summarizer(article["description"], max_length=50,
min_length=25, do_sample=False)[0]["summary_text"]
```

```
            summaries.append(f"📰 {article['title']}\n{summary}")
```

```
        return summaries
```

```
    else:
```

```
        return ["Não foi possível buscar notícias no momento."]
```

Função para exportar relatório em PDF

```
def export_pdf(ticker, score_text, roe, debt_ratio, current_ratio):
```

```
    c = canvas.Canvas(f"{ticker}_relatorio.pdf")
```

```
    c.drawString(100, 800, f"Relatório Financeiro — {ticker}")
```

```
    c.drawString(100, 780, f"Indicador de Investimento: {score_text}")
```

```
    c.drawString(100, 760, f"ROE: {roe:.2%}" if roe else "ROE: N/A")
```

```
    c.drawString(100, 740, f"Dívida/Patrimônio: {debt_ratio:.2f}" if debt_ratio else
"Dívida/Patrimônio: N/A")
```

```
    c.drawString(100, 720, f"Liquidez Corrente: {current_ratio:.2f}" if current_ratio
else "Liquidez Corrente: N/A")
```

```
c.save()

st.success("📄 PDF gerado com sucesso!")
```

```
# Função para gráfico de indicadores
```

```
def plot_indicators(roe, debt_ratio, current_ratio):

    fig, ax = plt.subplots()

    indicators = ['ROE', 'Dívida/Patrimônio', 'Liquidez Corrente']

    values = [roe or 0, debt_ratio or 0, current_ratio or 0]

    ax.bar(indicators, values, color=['green', 'red', 'blue'])

    st.pyplot(fig)
```

```
# Função para histórico de preço
```

```
def plot_price_history(ticker):

    stock = yf.Ticker(ticker)

    hist = stock.history(period="6mo")

    st.line_chart(hist['Close'])
```

```
# Função radar de oportunidades
```

```
def radar_oportunidades(tickers):

    resultados = []

    for t in tickers:

        info, _, _, _ = get_financial_data(t)

        roe, debt_ratio, current_ratio = calculate_indicators(info)

        score_text, _ = investment_score(roe, debt_ratio, current_ratio)

        resultados.append((t, score_text))

    return resultados
```

```
# Executar funções principais
```

```
info, balance, income, cashflow = get_financial_data(ticker)
roe, debt_ratio, current_ratio = calculate_indicators(info)
score_text, score_color = investment_score(roe, debt_ratio, current_ratio)
```

```
# Indicador de investimento
```

```
st.markdown(f"### 🧠 Indicador de Investimento: ")
st.markdown(f"<span style='color:{score_color}; font-size:24px'>{score_text}</span>", unsafe_allow_html=True)
```

```
# KPIs
```

```
st.markdown("### 📊 Indicadores Financeiros")
col1, col2, col3 = st.columns(3)
col1.metric("ROE", f"{roe:.2%}" if roe else "N/A")
col2.metric("Dívida/Patrimônio", f"{debt_ratio:.2f}" if debt_ratio else "N/A")
col3.metric("Liquidez Corrente", f"{current_ratio:.2f}" if current_ratio else "N/A")
```

```
# Gráfico de indicadores
```

```
plot_indicators(roe, debt_ratio, current_ratio)
```

```
# Histórico de preço
```

```
st.markdown("### 📈 Histórico de Preço da Ação")
plot_price_history(ticker)
```

```
# Notícias
```

```
st.markdown("### 📰 Notícias Recentes")
for news in get_news_summary(ticker):
    st.write(news)
```

```
# Dados contábeis
```

```
with st.expander("📊 Balanço Patrimonial"):
```

```
    st.dataframe(balance)
```

```
with st.expander("📊 Demonstração de Resultados"):
```

```
    st.dataframe(income)
```

```
with st.expander("📊 Fluxo de Caixa"):
```

```
    st.dataframe(cashflow)
```

```
# Exportar relatório
```

```
if st.button("📄 Exportar relatório em PDF"):
```

```
    export_pdf(ticker, score_text, roe, debt_ratio, current_ratio)
```

```
# Radar de oportunidades
```

```
st.markdown("### 📊 Radar de Oportunidades")
```

```
tickers_lista = ["PETR4.SA", "VALE3.SA", "ITUB4.SA"]
```

```
for empresa, indicativo in radar_oportunidades(tickers_lista):
```

```
    st.write(f"🔍 {empresa}: {indicativo}")
```

```
# Rodapé
```

```
st.markdown("---")
```

```
st.caption("App pessoal desenvolvido com Streamlit e IA. Dados via Yahoo Finance e NewsAPI.")
```