

**Estrutura de pastas:**

* [**Banco\_de\_dados**](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**:** scripts e banco SQLite.
* [**Coleta\_de\_dados**](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**:** scripts de coleta, APIs externas e planilhas.

Dentro da pasta Coleta\_de\_dados:

* **Apis:** Scripts para cada API
* [**IA**](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**:** modelo de machine learning.
* I[**nterface**](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**:** interface gráfica.
* [**legacy**](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**:** scripts antigos, testes e experimentos.
* [**backend**](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**:** API Flask para servir dados e receber palpites.

**ApostaPro/**

**├── run.py**

**├── relatorio\_geral\_coleta.csv**

**├── erros\_coleta.log**

**├── \_\_init\_\_.py**

**│**

**├── Banco\_de\_dados/**

**│   ├── criar\_banco.py**

**│   ├── verificar\_dados.py**

**│   ├── \_\_init\_\_.py**

**│   └── \_\_pycache\_\_/**

**│**

**├── Coleta\_de\_dados/**

**│   ├── agendador.py**

**│   ├── coletar\_tudo.py**

**│   ├── processar\_temporarios.py**

**│   ├── \_\_init\_\_.py**

**│   ├── logs/**

**│   │   ├── coleta.log**

**│   │   ├── erros.log**

**│   ├── apis/**

**│   │   ├── api\_football.py**

**│   │   ├── football\_data\_org.py**

**│   │   ├── sofascore\_scraper.py**

**│   │   ├── statsbomb\_api.py**

**│   │   ├── thesportsdb\_api.py**

**│   │   ├── \_\_init\_\_.py**

**│   │   ├── fbref/**

**│   │   │   ├── coletar\_dados\_partidas.py**

**│   │   │   ├── coletar\_estatisticas\_detalhadas.py**

**│   │   │   ├── fbref\_criar\_tabela\_rotulada.py**

**│   │   │   ├── fbref\_integrado.py**

**│   │   │   ├── fbref\_utils.py**

**│   │   │   ├── gerar\_relatorio\_final.py**

**│   │   │   ├── orquestrador\_coleta.py**

**│   │   │   ├── \_\_init\_\_.py**

**│   │   │   └── \_\_pycache\_\_/**

**│   │   └── \_\_pycache\_\_/**

**│   └── \_\_pycache\_\_/**

**│**

**├── backend/**

**│   ├── backend\_rest.py**

**│   └── \_\_init\_\_.py**

**│**

**├── interface/**

**│   ├── interface.py**

**│   └── \_\_init\_\_.py**

**│**

**├── IA/**

**│   ├── modelo\_ia\_apostapro.py**

**│   └── \_\_init\_\_.py**

**│**

**├── utils/**

**│   ├── log\_utils.py**

**│   ├── logs/**

**│   │   ├── coleta.log**

**│   │   ├── erros.log**

**│   └── \_\_pycache\_\_/**

**│**

**├── legacy (nao usar)/**

**│   ├── vários scripts antigos e testes**

**│**

**├── testes (nao usar)/**

**│   ├── vários scripts de teste e verificação**

**│**

**├── logs/**

**│   ├── coletar\_tudo.log**

**│   ├── thesportsdb.log**

**│   └── \_\_init\_\_.py**

**│**

**└── .git/ (estrutura interna do Git)**

**Sugestões para evoluir:**

* Continue migrando scripts de coleta para a subpasta [apis](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) (um arquivo por API/fonte).
* Centralize a orquestração da coleta em um script principal.
* Quando um script do [legacy](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) for substituído, pode movê-lo para backup ou excluir.
* Documente cada módulo com README ou docstrings para facilitar manutenção.

Ampliar sua coleta de dados com múltiplas APIs de estatísticas de clubes, ligas, jogadores e outros esportes, siga estas recomendações:

**1. Crie uma estrutura modular:**

* Crie uma pasta ou módulo Coleta\_de\_dados/apis\_externas/ para separar scripts de integração com cada API.
* Para cada API, crie um arquivo Python dedicado (ex: sofascore\_api.py, api\_football.py, the\_sports\_db.py).
* Cada API ou site de scraping tem seu próprio módulo Python na pasta [apis](vscode-file://vscode-app/c:/Program%20Files/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).
* Cada módulo expõe funções padronizadas, como coletar\_jogos(), coletar\_jogadores(), coletar\_ligas(), etc.

**2. Padronize a coleta:**

* Implemente funções com interface semelhante para cada API (ex: coletar\_jogos(), coletar\_jogadores(), coletar\_ligas()).
* Use um script principal para orquestrar a coleta e consolidar os dados no seu banco.
* Um script principal (ex: coletar\_tudo.py) importa todos os módulos e chama suas funções.
* Ele pode rodar tudo em sequência, em paralelo (usando threads/processos), ou agendado (cron, scheduler).

**3. Adapte o banco de dados:**

* Crie tabelas para clubes, jogadores, ligas e outros esportes, se ainda não existirem.
* Considere normalizar dados para evitar duplicidade.
* odas as coletas salvam no mesmo banco (aposta.db), em tabelas normalizadas: jogos\_historicos, jogos\_futuros, jogadores, clubes, ligas, etc.
* Cada registro tem um campo fonte para saber de onde veio o dado.

**4. Trate limites e autenticação:**

* Implemente rotação de chaves/tokens e tratamento de limites de requisição para cada API.

**5. Documente e registre a fonte dos dados:**

* Salve no banco de qual API veio cada registro, para rastreabilidade.

#### **6. **Backend/API****

* O backend Flask serve os dados já integrados, permitindo consultas por fonte, liga, jogador, etc.
* Pode expor endpoints para dados brutos ou processados (ex: odds, estatísticas agregadas).

### ****7. IA e Interface****

* O módulo de IA consome os dados do banco para treinar e fazer previsões.
* A interface (desktop/web) consome a API Flask para exibir dados e resultados.

### **8. **Logs e Monitoramento****

* Cada módulo de coleta gera logs de sucesso/erro.
* O orquestrador central pode enviar alertas se alguma fonte falhar.

### **9. **Documentação****

* Documente cada módulo e o fluxo geral para facilitar manutenção e expansão.

****Resumo visual:****

[APIs/Scraping] --> [Módulos de coleta] --> [Banco de dados unificado] <-- [Backend Flask/API] <-- [Interface/IA]

Fluxo de desenvolvimento recomendado para cada fonte/serviço, com observação sobre gratuidade:

### **1. **SofaScore (scraping)****

* Não tem API oficial gratuita. Use scraping (como seu script faz).
* Fluxo:
  1. Identifique a URL do time/jogador.
  2. Implemente função de scraping (BeautifulSoup, requests).
  3. Parseie e salve os dados no banco.
  4. Trate mudanças de layout do site.

### **2. **API-Football (api-sports.io)****

* Tem plano gratuito (limite de requisições/dia).
* Fluxo:
  1. Crie conta e obtenha API key.
  2. Implemente módulo Python para autenticação e requisições.
  3. Crie funções para coletar jogos, jogadores, ligas, etc.
  4. Salve dados no banco.

### **3. **TheSportsDB****

* Tem plano gratuito para uso pessoal.
* Fluxo:
  1. Crie conta e obtenha API key.
  2. Implemente módulo Python para buscar jogadores, times, ligas.
  3. Salve dados no banco.

### **4. **Football-Data.org****

* Tem plano gratuito, mas limitado.
* Fluxo:
  1. Crie conta e obtenha API key.
  2. Implemente módulo Python para buscar dados de jogos, ligas, times.
  3. Salve dados no banco.

### **5. **SportMonks****

* Tem plano gratuito, mas com poucos endpoints liberados.
* Fluxo:
  1. Crie conta e obtenha API key.
  2. Implemente módulo Python para buscar dados disponíveis.
  3. Salve dados no banco.

### **6. **API-Basketball (api-sports.io)****

* Tem plano gratuito (limite de requisições/dia).
* Fluxo igual ao API-Football.

### **7. **API-Tennis (api-sports.io)****

* Tem plano gratuito (limite de requisições/dia).
* Fluxo igual ao API-Football.

### **8. **ESPN API (não oficial)****

* Não tem API oficial pública. Só wrappers/scraping, pode ser instável.
* Fluxo:
  1. Pesquise wrapper Python atualizado (ex: espn-api).
  2. Implemente funções para buscar dados desejados.
  3. Se não funcionar, considere remover.

### **9. **StatsBomb API****

* Tem API gratuita para dados públicos (futebol feminino, Copa do Mundo, etc).
* Fluxo:
  1. Solicite acesso gratuito no site.
  2. Use biblioteca Python oficial (statsbombpy).
  3. Importe e salve dados no banco.

### **10. **mplsoccer library****

* Biblioteca Python para análise/visualização, não é API de dados.
* Fluxo:
  1. Use para processar/visualizar dados já coletados.
  2. Não precisa de integração de coleta.

### **11. **Scraping Flashscore e outros sites****

* Não tem API oficial, só scraping.
* Fluxo:
  1. Identifique URLs e estrutura dos sites.
  2. Implemente scraping (BeautifulSoup, Selenium, etc).
  3. Trate mudanças de layout e respeite termos de uso.

### **12. **APIs do RapidAPI****

* A maioria tem plano gratuito, mas com limites. Verifique cada uma:
  + **today-football-prediction**: Gratuito limitado.
  + **soccer-football-info**: Gratuito limitado.
  + **sportspage-feeds**: Gratuito limitado.
  + **football-prediction**: Gratuito limitado.
  + **pinnacle-odds**: Gratuito limitado.
  + **bet365-futebol-virtual**: Gratuito limitado.
  + **transfermarkt-db**: Gratuito limitado.
  + **football-pro (Sportmonks)**: Gratuito limitado.
  + **football98**: Gratuito limitado.
* **Fluxo:**
  + Crie conta no RapidAPI.
  + Assine cada API e obtenha chave.
  + Implemente módulo Python para cada API.
  + Trate limites de requisição.

Se alguma dessas APIs não tiver opção gratuita (ou o wrapper ESPN não funcionar), recomendo remover do fluxo principal e focar nas demais.

Para usar todas essas APIs e fontes (incluindo scraping), siga estas recomendações para um projeto robusto, escalável e organizado:

### **1. Estruture sua coleta em módulos**

* Crie uma pasta Coleta\_de\_dados/apis/ e, para cada API ou fonte, crie um arquivo Python separado (ex: sofascore\_api.py, api\_football.py, thesportsdb\_api.py, flashscore\_scraper.py etc).
* Em cada módulo, implemente funções padronizadas como coletar\_jogos(), coletar\_jogadores(), coletar\_ligas(), etc.

### **2. Centralize a orquestração**

* Crie um script principal (ex: coletar\_tudo.py) que chama as funções de cada módulo e salva os dados no banco.
* Use logs para saber qual API/fonte está sendo usada e se houve erro.

### **3. Adapte o banco de dados**

* Crie tabelas para clubes, jogadores, ligas, odds, estatísticas, etc.
* Adicione um campo “fonte” para saber de onde veio cada dado.

### **4. Trate limites e autenticação**

* Implemente rotação de chaves/tokens para cada API.
* Respeite limites de requisição e trate erros de rate limit.

### **5. Scraping**

* Use bibliotecas como BeautifulSoup, requests, selenium ou playwright para scraping.
* Sempre cheque os Termos de Uso dos sites antes de fazer scraping.

### **6. RapidAPI**

* Para as APIs do RapidAPI, use o header X-RapidAPI-Key e siga a documentação de cada uma.

### **7. Documentação e manutenção**

* Documente cada módulo com exemplos de uso e endpoints.
* Mantenha um requirements.txt com todas as dependências.

### **Lista de Dados Essenciais para Treinamento de Modelos de Apostas Esportivas:**

#### ****Dados de Desempenho Histórico**:**

1. ****Resultados de Partidas****:
   * Placares finais (vitória, derrota, empate).
   * Diferença de gols.
   * Sequências recentes (ex: vitórias/contra/empates nas últimas 5 partidas).
2. ****Estatísticas por Temporada****:
   * Posição na tabela histórica.
   * Pontos conquistados por temporada.
   * Número de vitórias/derrotas/empates.
3. ****Desempenho em Casa vs. Fora****:
   * Porcentagem de vitórias em casa e fora.
   * Média de gols marcados e sofridos em cada tipo de jogo.
4. ****Histórico de Confrontos Diretos****:
   * Resultados nos últimos encontros entre duas equipes.
   * Gols marcados e sofridos nos duelos passados.

#### ****Dados Táticas e Estatísticas Avançadas**:**

1. ****Posse de Bola****:
   * Porcentagem média de posse por jogo.
2. ****Criação de Chances****:
   * Finalizações (total, dentro da, área de fora).
   * xG (Expected Goals - probabilidade de gol em cada finalização).
3. ****Defesa****:
   * Intercepções, desarmes, faltas cometidas.
   * Média de gols sofridos por jogo.
4. ****Estilo de Jogo****:
   * Passes por jogo, cruzamentos, escanteios.
   * Velocidade média do jogo (níveis pressão, transição).

#### ****Dados Contextuais**:**

1. ****Lesões e Suspensões****:
   * Jogadores titulares ou chavelesionados/suspendidos.
   * Tempo estimado de recuperação de lesionados.
2. ****Formação Tática****:
   * Alinhamento mais utilizado (ex: 4-3-3, 3-5-2).
   * Mudanças recentes na escalação.
3. ****Condições Climáticas****:
   * Temperatura, chuva, vento no dia da partida.
4. ****Motivação da Equipe****:
   * Posição na tabela (luta contra rebaixamento ou briga por títulos).
   * Torneios simultâneos (ex: Copa Libertadores + Campeonato Nacional).

#### ****Dados de Jogadores**:**

1. ****Estatísticas Individuais****:
   * Gols, assistências, passes certos, finalizações.
   * xG (Expected Goals) e xA (Expected Assists).
   * Desempenho defensivo (desarmes, roubadas de bola).
2. ****Forma Física****:
   * Minutos jogados nos últimos jogos (risco de fadiga).
   * Retornos de lesões.
3. ****Histórico contra Equipes Específicas****:
   * Jogadores com histórico de boas atuações contra um adversário.

#### ****Dados de Apostas**:**

1. ****Linhas de Apostas Históricas****:
   * Odds oferecidas por casas de apostas em partidas passadas.
   * Volume de apostas em cada resultado.
2. ****Desfecho vs. Odds****:
   * Frequência em que resultados com baixas odds (favoritos) ou altas odds (underdogs) ocorrem.

#### ****Fontes Complementares**:**

* ****Notícias Esportivas****: Transferências recentes, mudanças de treinador.
* ****Dados de Saída de Bolas**** (ex: escanteios, faltas laterais).
* ****Estatísticas de Chuteira**** (quem cria as chances de gol).

### **Recomendações para Coleta de Dados:**

* Use APIs ou web scraping (respeitando políticas de uso do site).
* Combine dados de múltiplas fontes (ex: FBRef +WhoScored + SofaScore).
* Verifique a consistência histórica (padronização de métricas ao longo dos anos).