

// laboratório codex 04 · economia · micro e macro

python + ia + app

>_ Laboratório Codex 04

Os modelos no mundo real, com IA

O Laboratório de Modelos Econômicos no GitHub, com testes automáticos, o IPCA, a Selic e o dólar do Banco Central ao lado dos modelos, publicado na internet com um link, feito em equipe e no seu portfólio.

// apostila do aluno IV · o projeto em etapas

01 Git e GitHub

03 Dados reais

05 Trabalho em equipe

07 GitHub Actions

02 Testes automáticos

04 Publicar o app

06 Portfólio

08 Apresentação



Prof. Caio Haffner

Mestre em Desenvolvimento Econômico · Unicamp

1 Como usar esta apostila

Esta é a **Apostila IV de Economia (Micro e Macro)**, a última da trilha: ela continua a **Apostila III (Laboratório de Modelos Econômicos)**. Agora você vai levar o **Laboratório de Modelos Econômicos (Apostila III)** para o mundo: guardado no GitHub, com testes automáticos, dados reais, publicado na internet com um link e apresentado no seu portfólio.

1. Guardar Git e GitHub	2. Conferir Testes automáticos	3. Conectar Dados reais	4. Publicar O app na internet	5. Mostrar Equipe e portfólio
-----------------------------------	--	-----------------------------------	---	---



O PROJETO: OS MODELOS NO MUNDO REAL

Hoje o seu app só funciona no seu computador. Ao final desta apostila, qualquer pessoa vai poder abri-lo pelo celular, com um link. Ele vai usar o IPCA, a Selic e o dólar do Banco Central ao lado dos modelos, conferir as próprias contas antes de publicar e mostrar quem fez cada parte.

Ao final, você deverá conseguir

Habilidade	Como saber que consegui	Ok
Versionar	O projeto está no GitHub, com o histórico de cada mudança.	☐
Testar	Os testes rodam com um comando e passam.	☐
Usar dados reais	O app busca dados oficiais sozinho e mostra a data e a fonte.	☐
Publicar	Tenho um link do app que abre no celular de qualquer pessoa.	☐
Trabalhar em equipe	Revisei a mudança de um colega antes de ela entrar no projeto.	☐
Apresentar	O projeto tem página, vídeo de 1 minuto e post no LinkedIn.	☐



REGRA DE OURO

Você é responsável pelo código que entrega, mesmo quando a IA o escreveu. **Não entregue código que você não consegue explicar.** Aqui vale mais uma: **nada vai para a internet sem passar nos testes.** Publicar é responsabilidade.

Antes de começar



LEIA TUDO, NA ORDEM

Leia esta apostila **desde a primeira palavra** e faça tudo o que ela manda, na ordem. Cada etapa prepara a seguinte, e publicar algo errado na internet é pior do que não publicar.

A Apostila IV começa onde a III termina: **o app da Apostila III precisa estar funcionando** no seu computador. Se ainda não está, termine a III antes.

O que você precisa	Ok
O app da Apostila III abrindo com <code>python -m streamlit run app.py</code>	☐
VS Code, Python e Codex funcionando (Apostila I)	☐
Um e-mail para criar a conta gratuita no GitHub	☐
Uma dupla ou trio para a etapa 5 (trabalho em equipe)	☐

Ao longo das etapas, a pasta do projeto vai ganhar três novidades:

TERMINAL

```
seu_projeto/  
├─ dados/  
├─ calculos.py  
├─ graficos.py  
├─ app.py  
├─ tests/           ← novo: testes automáticos  
├─ requirements.txt ← novo: a lista de bibliotecas  
└─ README.md       ← novo: a página do projeto
```



TUDO GRATUITO

GitHub, Git, pytest e Streamlit Community Cloud são gratuitos para estudantes e projetos públicos. Nenhum cartão de crédito é necessário. Se algum site pedir pagamento, pare e pergunte ao professor.



LEMBRETES DAS APOSTILAS ANTERIORES

Peça sempre um **plano** antes do código · teste com contas feitas por você · peça o **diagnóstico antes da correção** · só aprove o que entender · nunca coloque senhas no código.

3 Etapa 1 • Guardar: Git e GitHub

Até aqui, você guardava versões copiando a pasta. Profissionais usam o **Git**, que registra cada mudança, e o **GitHub**, que guarda tudo na nuvem.

- 1 Crie uma conta gratuita em github.com com o seu e-mail de estudante.
- 2 Instale o Git em git-scm.com (no Windows, aceite as opções padrão). Feche e abra o VS Code.
- 3 No VS Code, clique no ícone de **Controle do Código-Fonte** (os três círculos ligados, na barra da esquerda) e entre com a sua conta do GitHub.

PROMPT 1

Quero guardar o meu projeto no GitHub. Antes de fazer qualquer coisa, me explique em 5 linhas o que são Git, commit e repositório. Depois:

1. crie um arquivo `.gitignore` para Python (sem dados pessoais nem senhas);
2. inicie o Git nesta pasta e faça o primeiro commit com a mensagem "Primeira versão do app";
3. me diga como publicar no GitHub pelo botão "Publish Branch" do VS Code, como repositório PÚBLICO.



CHECKPOINT

O projeto aparece em `github.com/seu-usuario/nome-do-projeto`, com todos os arquivos.



NÃO FAÇA

Nunca coloque no GitHub senhas, chaves, planilhas com dados pessoais ou dados reais de empresas. O repositório público pode ser visto por qualquer pessoa. Use só os dados fictícios das apostilas.

4

Etapa 2 · Conferir: testes automáticos

Nas apostilas anteriores, você conferia as contas com a calculadora. Agora o próprio projeto vai conferir, com um comando, antes de cada publicação.

PROMPT 2

Instale o pytest (`python -m pip install pytest`) e crie a pasta `tests` com o arquivo `test_calculos.py`.
 Escreva testes para as funções principais do meu `calculos.py`, usando os números que conferi nas Apostilas II e III (tabelas de conferência). Cada teste deve ter um nome que explique o que confere.
 Antes de escrever, liste quais funções e quais números você vai testar.

TERMINAL

```
python -m pytest
```



CHECKPOINT

O terminal mostra os testes em verde, com a palavra `passed`.



TESTE QUE FALHA DE PROPÓSITO

Mude um número de um teste para um valor errado e rode de novo: ele precisa ficar vermelho. Um teste que nunca falha não testa nada. Depois desfaça a mudança.



NÚMEROS PARA TESTAR (APOSTILAS II E III)

Equilíbrio de $Q_d = 100 - 2P$ e $Q_s = 20 + 2P$: **P = 20, Q = 60** · elasticidade no ponto estudado: **-1,22** (elástica) · multiplicador com PMgC 0,8: **k = 5**.

Teste	O que confere	Passou?
Equilíbrio do mercado		
Elasticidade		
Multiplicador		

5 Etapa 3 • Conectar: dados reais

O app deixa de usar só números digitados e passa a buscar **dados oficiais** sozinho: o IPCA, a Selic e o dólar do Banco Central ao lado dos modelos.

PROMPT 3

Crie dados_reais.py com uma função que busque uma série do Banco Central (SGS) pela API:
`https://api.bcb.gov.br/dados/serie/bcdata.sgs.CODIGO/dados/ultimos/N?formato=json`
Ela deve: devolver uma tabela com data e valor (vírgula brasileira tratada); guardar o último resultado num arquivo para usar se a internet falhar (plano B); e informar a data do dado.
No app, mostre o valor com a frase "Fonte: Banco Central, série CODIGO, dado de DATA".
Use as séries do IPCA mensal (código 433), da Selic (código 432) e do dólar (código 1).
No app, mostre ao lado de cada modelo o dado real correspondente: a inflação real ao lado do modelo de preços, a Selic ao lado da Regra de Taylor e o dólar com a data da cotação.



CHECKPOINT

O app mostra um dado real com a fonte e a data, e continua funcionando sem internet (plano B).



CONFIRA A FONTE

Abra o mesmo dado no site do Banco Central (www3.bcb.gov.br/sgspub) e compare com o que o app mostra. A IA às vezes troca o código da série, a unidade (% ao mês × % ao ano) ou a data.

Dado real	Valor no app	Data	Fonte confere?
IPCA do mês			
Selic			
Dólar			

6

Etapa 4 · Publicar: o app na internet

O **Streamlit Community Cloud** publica de graça apps guardados no GitHub. Ao final, você terá um link para mandar a qualquer pessoa.

PROMPT 4

Prepare o meu projeto para publicar no Streamlit Community Cloud:

1. crie o `requirements.txt` com as bibliotecas que o app usa (sem versões desnecessárias);
2. confira se o app não depende de arquivos fora da pasta do projeto;
3. faça um commit "Prepara a publicação" e envie ao GitHub.

Depois me explique, em passos curtos, como criar o app em `share.streamlit.io` a partir do repositório.

- 1 Entre em share.streamlit.io com a sua conta do GitHub.
- 2 Clique em **Create app**, escolha o repositório, a branch `main` e o arquivo `app.py`.
- 3 Clique em **Deploy** e espere alguns minutos.



CHECKPOINT

O app abre pelo link `https://nome.streamlit.app` no navegador do celular.

Link do meu app: _____

7

Etapa 5 · Trabalhar em equipe

Em empresas, ninguém mexe direto na versão principal. Cada pessoa trabalha numa **branch** e pede para juntar a mudança com um **pull request**, que um colega revisa.

Formem duplas ou trios. Um colega é convidado para o repositório do outro (Settings → Collaborators).

PROMPT 5

Quero propor uma melhoria no app do meu colega sem mexer na versão principal.

1. Crie uma branch chamada melhoria-grafico;
2. [descreva a melhoria: por exemplo, um modelo novo ou o deslocamento de uma curva na tela];
3. rode os testes; se passarem, faça o commit e envie a branch;
4. me diga como abrir o pull request no GitHub e o que escrever na descrição.



REVISAR O CÓDIGO DO COLEGA

No pull request do colega, leia o que mudou, rode o app e os testes, e deixe pelo menos um comentário útil (uma dúvida, um risco ou uma melhoria). Só aprove o que você entendeu.



CHECKPOINT

Abri um pull request, um colega revisou, e a mudança entrou na versão principal.

Etapa 6 · Mostrar: portfólio

Agora o projeto existe para o mundo. Falta contar a história dele.

PROMPT 6

Escreva um rascunho de README.md para o meu projeto, para recrutadores: o que o app faz, o link do app no ar, um print, as tecnologias (Python, Streamlit, dados do Banco Central), como rodar no computador, o que aprendi e a frase "Feito com ajuda de IA, revisado por mim". Use linguagem simples. Eu vou revisar antes de publicar.

- ☐ README revisado e publicado no GitHub
- ☐ Vídeo de 1 minuto: o problema, o app no ar pelo link e um dado real
- ☐ Post no LinkedIn com o vídeo e o link do app



UMA FRASE PARA O POST

"Construí em Python, com ajuda de IA, um laboratório de modelos econômicos com dados reais do Banco Central. Está no ar: [link]. O código está no GitHub: [link]."

9

Desafio extra · Atualização automática

Com o **GitHub Actions**, o próprio GitHub pode rodar os testes a cada mudança e atualizar os dados sozinho, toda semana.

PROMPT 7

desafio

Crie um fluxo do GitHub Actions (.github/workflows/testes.yml) que rode o pytest a cada push.
Depois, crie um segundo fluxo que rode toda segunda-feira às 8h (cron) para atualizar o arquivo de dados reais e fazer um commit automático. Explique cada linha do arquivo antes de criar.

**CHECKPOINT**

Na aba Actions do GitHub, os testes aparecem com um sinal verde.

10

Use o Codex como revisor do projeto

REVISAR A ORGANIZAÇÃO

Revise o meu projeto como um programador experiente revisaria o de um iniciante. Não faça alterações. Aponte: funções muito longas, código repetido, nomes pouco claros e cálculos misturados com a tela. Separe o que é obrigatório do que é opcional.

REVISAR O CONTEÚDO

Revise os cálculos do meu projeto como um professor de Economia (Micro e Macro). Não faça alterações. Confira as fórmulas, os dados reais (data, fonte e unidade) e se os testes conferem o que importa e liste o que pode estar errado.

**GRÁFICO BONITO NÃO É GRÁFICO CERTO**

Confira pelo menos um número de cada gráfico com a calculadora ou com a Apostila II. A IA erra eixos, unidades e somas com frequência.

Entrega: apresente o seu projeto

Um projeto só existe quando alguém o vê funcionando. Prepare uma **apresentação de 3 minutos** e um registro para o seu portfólio.

Roteiro de 3 minutos

Tempo	O que mostrar
30 s	O problema que o app resolve e para quem.
1 min 30 s	O app funcionando: mexa nos controles e mostre o app no ar, pelo link, com os modelos ao lado dos dados reais.
30 s	Um erro que a IA cometeu e como você descobriu.
30 s	O que você faria a seguir, se tivesse mais tempo.

O que entregar

- ☐ A pasta do projeto, com todos os arquivos .py e os dados
- ☐ O relatório em Excel gerado pelo app
- ☐ Três prints do app (telas principais)
- ☐ Um arquivo LEIAME.md explicando o que o app faz e como rodar (peça um rascunho ao Codex e revise)
- ☐ O diário de prompts: pelo menos 5 prompts importantes e o que aconteceu



PARA O SEU PORTFÓLIO

Um post no LinkedIn com um vídeo curto do app funcionando ("Construí em Python, com ajuda de IA, um app de economia") mostra a recrutadores algo que poucos estudantes têm: **uma ferramenta real feita por você.**

Reflexão individual

1. Qual etapa foi a mais difícil, e como você a resolveu?

2. Em que momento a IA errou, e como você percebeu?

3. O que você aprendeu de Economia (Micro e Macro) construindo o app que não tinha aprendido só com a teoria?

12 Como você será avaliado

Critério	0	1	2	3
O app funciona	Não abre	Abre com erros	Funciona	Funciona e trata entradas inválidas
Conteúdo	Cálculos errados	Alguns certos	Quase todos certos	Todos certos e conferidos
Gráficos	Nenhum	Sem títulos ou eixos	Corretos	Corretos e fáceis de ler
Relatório	Não gera	Gera incompleto	Gera completo	Completo e organizado em abas
Uso do Codex	Não usou	Só copiou	Usou com compreensão	Usou de forma crítica
Apresentação	Não apresentou	Confusa	Clara	Clara, no tempo e com o erro da IA

Total máximo: 18 pontos, convertidos para a escala da disciplina.

13 Consulta rápida

Comandos

TERMINAL

```
python -m pytest           # roda
os testes
git status                 # o
que mudou
git add . ; git commit -m "... " #
guarda uma versão
git push                   #
envia ao GitHub
python -m streamlit run app.py # abre
o app
```

Streamlit

<code>st.title · st.header</code>	Títulos
<code>st.number_input · st.slider</code>	Entradas de número
<code>st.selectbox · st.file_uploader</code>	Escolha e upload
<code>st.metric · st.dataframe</code>	Indicador e tabela
<code>st.pyplot(fig)</code>	Mostrar um gráfico
<code>st.tabs · st.sidebar</code>	Abas e barra lateral
<code>st.download_button</code>	Botão para baixar o Excel

Glossário

Biblioteca

Código pronto que você instala para ganhar recursos (gráficos, telas).

pip

O instalador de bibliotecas do Python.

DataFrame

Tabela do pandas, com linhas e colunas.

App web

Programa que roda no navegador.

Git

Programa que guarda o histórico de versões do projeto.

GitHub

Site que guarda projetos Git na nuvem.

Commit

Uma "foto" do projeto num momento, com uma frase explicando a mudança.

Repositório

A pasta do projeto no GitHub.

Branch

Uma cópia de trabalho para mexer sem estragar a versão principal.

Pull request

Pedido para juntar uma mudança à versão principal, com revisão.

API

Endereço de onde um programa busca dados de outro sistema.

Deploy

Publicar o app na internet.

1

LEMBRE-SE

A IA é uma ferramenta. Conhecimento, julgamento, validação e responsabilidade continuam sendo de quem programa. Nada nesta apostila é recomendação profissional: é aprendizado.