

// laboratório codex 03 · princípios de gestão

python + ia + app

>_ Laboratório Codex 03

Painel do gestor em Python, com IA

SWOT colorida, plano 5W2H editável, organograma desenhado, liderança e indicadores com semáforo: as quatro funções da gestão num app feito por você.

// apostila do aluno III · o projeto em etapas

01 Dados da empresa



03 SWOT e semáforo



05 Acompanhamento



07 Radar dos 7S



02 Regras em funções



04 O app em abas



06 Relatório em Excel



08 Apresentação



Prof. Caio Haffner

Mestre em Desenvolvimento Econômico · Unicamp

1 Como usar esta apostila

✖ LEIA TUDO, NA ORDEM

Leia esta apostila **desde a primeira palavra** e faça tudo o que ela manda, na ordem. Não pule direto para os exercícios: cada página prepara a seguinte. Se ainda não fez a **Apostila I** (inclusive a página "PARE. Leia antes de tudo"), comece por ela.

Esta é a **Apostila III**, que continua a I (instalação e primeiros passos) e a II (exercícios no terminal). Agora você vai construir **um projeto completo**: um **painel do gestor** com as quatro funções da gestão: SWOT e plano 5W2H editáveis, organograma desenhado, diagnóstico de liderança e indicadores com semáforo, num app com relatório em Excel.

1. Dados

Os arquivos da empresa

2. Funções

As regras testadas

3. Gráficos

SWOT, semáforo e organograma

4. App

Uma aba por função

5. Relatório

Acompanhamento e Excel

★ O PROJETO: PAINEL DO GESTOR

Cada aba é uma função do gestor. Em **Planejar**, a SWOT vira uma matriz colorida e o plano 5W2H é editado na tela. Em **Organizar**, o organograma é desenhado a partir da lista da equipe. Em **Dirigir**, o questionário de Likert mostra o perfil de liderança. Em **Controlar**, os indicadores aparecem com o semáforo e as ações atrasadas ficam em destaque. No fim, um relatório em Excel.

Ao final, você deverá conseguir

Habilidade	Como saber que consegui	Ok
Organizar os dados da empresa	Monto os arquivos de SWOT, ações, equipe e indicadores.	☐
Programar as regras	Funções testadas com os números da Apostila II.	☐
Mostrar a SWOT	Matriz colorida com a postura estratégica.	☐
Desenhar a estrutura	Organograma gerado a partir da lista da equipe.	☐
Controlar	Indicadores com semáforo e ações atrasadas em destaque.	☐
Relatar	Relatório em Excel com uma aba por função.	☐

1

REGRA DE OURO

Você é responsável pelo código que entrega, mesmo quando a IA o escreveu. **Não entregue código que você não consegue explicar.** No projeto, vale mais uma: **a cada etapa, o app precisa continuar funcionando.** Teste antes de seguir.

2 Antes de começar

Você precisa do que montou nas Apostilas I e II: VS Code, Python e Codex funcionando. Se faltar algo, volte à **Apostila I, seções 3 a 7**. A novidade são três **bibliotecas** que dão ao Python gráficos, telas e planilhas.

Instale as bibliotecas (uma vez só)

No terminal do VS Code, digite:

TERMINAL

```
python -m pip install streamlit matplotlib pandas openpyxl
```

Biblioteca	Para que serve
pandas	Trabalhar com tabelas (como uma planilha dentro do Python).
matplotlib	Fazer gráficos: linhas, barras, pizza, dispersão.
streamlit	Transformar o programa num aplicativo que abre no navegador.
openpyxl	Salvar e ler arquivos do Excel (.xlsx).

Para conferir, digite:

TERMINAL

```
python -c "import streamlit, matplotlib, pandas, openpyxl; print('tudo certo')"
```



CHECKPOINT

O terminal mostrou `tudo certo`.



SE DER ERRO

Se aparecer "pip não é reconhecido" ou "python não é reconhecido", troque `python` por `py` (como na Apostila I). Se a instalação demorar, é normal: são bibliotecas grandes. Se travar, copie a mensagem e use o prompt universal de diagnóstico.

Crie a pasta do projeto

Crie a pasta `Laboratorio_Codex_PG3` em Documentos e abra no VS Code. Ao longo das etapas, ela vai ficar assim:

TERMINAL

```
Laboratorio_Codex_PG3/  
├─ dados/           ← arquivos de dados (CSV)  
├─ calculos.py      ← as contas (funções)  
├─ graficos.py      ← os gráficos  
├─ app.py           ← a tela (Streamlit)  
└─ relatorio.xlsx   ← gerado pelo app
```



GUARDE VERSÕES

Ao terminar cada etapa com o app funcionando, copie a pasta inteira com o nome da etapa (por exemplo, `Laboratorio_Codex_PG3_etapa2`). Se algo quebrar depois, você volta.

3

Seu primeiro app

Antes do projeto, um teste rápido das ferramentas novas. Crie um arquivo `teste_app.py` com o código abaixo. Você pode digitar ou pedir ao Codex.

```
import streamlit as st  
import matplotlib.pyplot as plt  
  
st.title("Meu primeiro app")  
valor = st.number_input("Valor aplicado (R$)", value=1000.0)  
taxa = st.slider("Taxa ao mês (%)", 0.0, 2.0, 1.0)  
meses = st.slider("Meses", 1, 60, 12)  
  
saldos = [valor * (1 + taxa / 100) ** m for m in range(meses + 1)]  
fig, ax = plt.subplots()  
ax.plot(saldos)  
ax.set_xlabel("Mês")  
ax.set_ylabel("R$")  
st.pyplot(fig)  
st.metric("Valor final", f"R$ {saldos[-1]:,.2f}")
```

Para abrir o app, digite no terminal:

TERMINAL

```
python -m streamlit run teste_app.py
```



CHECKPOINT

O navegador abriu um app com o título "Meu primeiro app", dois controles deslizantes e um gráfico. Mexa nos controles: o gráfico muda na hora.

Complete a tela

TERMINAL

Valor aplicado: 1000 · Taxa ao mês: 1,00 · Meses: 12

Valor final: R\$ _____



ARMADILHA: O FORMATO DO NÚMERO

O app mostra o valor no formato americano (vírgula no milhar e ponto nos centavos). Peça ao Codex: *"mostre os valores no formato brasileiro, com ponto no milhar e vírgula nos centavos"*. Você vai usar isso em todo o projeto.



COMO O STREAMLIT FUNCIONA

Cada vez que você mexe num controle, o Streamlit roda o programa inteiro de novo, de cima para baixo. Por isso o gráfico se atualiza sozinho. Para fechar o app, volte ao terminal e aperte `Ctrl` + `C`.



LEMBRETES DAS APOSTILAS ANTERIORES

Peça sempre um **plano** antes do código · teste com contas feitas por você · peça o **diagnóstico antes da correção** · aceite vírgula ou ponto · só aprove o que entender.

4

Etapa 1 · Os dados da empresa

O painel começa pelos dados do Café Bom Dia, os mesmos da Apostila II, agora em arquivos.

PROMPT 1

Crie a pasta dados com quatro arquivos CSV do Café Bom Dia (use os dados da Apostila II):

- swot.csv: quadrante, item, nota;
- acoes.csv: o_que, por_que, onde, quando, quem, como, quanto, situacao (a fazer, em andamento, concluída);
- equipe.csv: nome, cargo, chefe;
- indicadores.csv: nome, meta, realizado, sentido (maior ou menor).

Em calculos.py, crie ler_dados() com pandas, que devolve os quatro arquivos.



CHECKPOINT

A função lê os quatro arquivos e mostra as tabelas no terminal.

As ferramentas da Apostila II, agora como funções testadas.

PROMPT 2

Em `calculos.py`, crie:

1. `postura_swot(df)`: saldos interno e externo e a postura;
2. `semaforo(meta, realizado, sentido)`: atingimento e cor;
3. `margens(equipe)`: margem de controle de cada gestor, níveis e alertas;
4. `sistema_likert(respostas)`: média e sistema de 1 a 4;
5. `atrasadas(acoes, hoje)`: ações não concluídas com prazo vencido.

Escreva testes com `assert` usando os números da Apostila II.

Teste	Esperado	A função devolveu
SWOT do Café	interno +2, externo +2: desenvolvimento	
Reclamações (meta 10, real 14)	71,4%: vermelho	
Equipe do Café	4 níveis, alerta no Bruno	
Likert 3·3·4·3·2·3·3·4	3,125: Sistema 3	

Etapa 3 · SWOT, semáforo e organograma em gráficos

Três imagens que o dono entende de primeira.

PROMPT 3

Crie graficos.py com:

1. grafico_swot: os 4 quadrantes coloridos (verde, vermelho, azul e laranja) com os itens e as notas;
2. grafico_indicadores: barras horizontais com o atingimento de cada indicador, pintadas com a cor do semáforo e uma linha vertical em 100%;
3. organograma_dot(equipe): o texto no formato DOT (Graphviz) do organograma, para o Streamlit desenhar com st.graphviz_chart.



CHECKPOINT

No gráfico de indicadores, a barra das reclamações está vermelha e para em 71%.

Cada função do gestor numa aba.

PROMPT 4

Crie app.py com Streamlit e quatro abas:

1. "Planejar": a matriz SWOT, a postura estratégica e o plano 5W2H editável (st.data_editor);
2. "Organizar": o organograma (st.graphviz_chart), as margens de controle e os alertas;
3. "Dirigir": as 8 afirmações de Likert com st.radio (1 a 4), a média e o sistema;
4. "Controlar": st.metric com quantos indicadores estão em cada cor e o gráfico do semáforo.

TERMINAL

```
python -m streamlit run app.py
```

Teste no app	Resultado
Ameaças com nota 5: nova postura	
lara passa a responder à Ana: alertas	
Reclamações caem para 9: nova cor	

Etapa 5 · O acompanhamento do plano

Planejar não basta: o painel mostra o andamento das ações e liga o controle ao planejamento.

PROMPT 5

Acrescente à aba "Controlar":

1. o percentual de ações concluídas e a lista das atrasadas em vermelho;
2. para cada indicador vermelho ou amarelo sem ação no plano, um botão "Criar ação" que adiciona uma linha nova no 5W2H (com o_que e por_que preenchidos);
3. salvar as alterações de volta em acoes.csv.



CHECKPOINT

Ao criar a ação para as reclamações, ela aparece na aba Planejar.

Etapa 6 · O relatório em Excel

O app termina com um relatório para o dono guardar e discutir com a equipe.

PROMPT 6

Acrescente um botão "Baixar relatório do gestor" com as abas:
"Resumo" (texto de 4 a 6 frases escrito pelo app: postura, alertas da estrutura, perfil
de liderança, indicadores vermelhos e ações atrasadas), "SWOT", "Plano 5W2H",
"Equipe"
e "Indicadores".



CHECKPOINT

O resumo cita os números do app e o Excel tem 5 abas.

10 Desafio extra · O radar dos 7S

Mostre o alinhamento da organização num **gráfico de radar** com os 7S da McKinsey.

PROMPT 7

desafio

Acrescente uma aba "7S": sliders de 1 a 5 para cada S, um gráfico de radar (matplotlib, eixos polares) com as notas, a média, o S mais fraco e o aviso de desalinhamento (amplitude de 2 ou mais). Explique o gráfico de radar antes de escrever o código.

11 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 Princípios de Gestão. Não faça alterações. Confira as regras de cada ferramenta, os limites do semáforo, o sentido dos indicadores e as datas do plano 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 um indicador mudando de cor e a ação aparecendo no plano.
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 ferramentas de gestão para pequenas empresas") 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 Princípios de Gestão construindo o app que não tinha aprendido só com a teoria?

13 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
Registro da auditoria humana (opcional, vale ponto extra)	☐			

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

14 Consulta rápida

Comandos

TERMINAL

```
python -m pip install streamlit  
matplotlib pandas openpyxl  
python -m streamlit run app.py    # abre  
o app  
Ctrl + C                          #  
fecha 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.

Painel (dashboard)

Tela que reúne os principais números para decidir.

PODC

Planejar, organizar, dirigir e controlar.

Ciclo PDCA

Planejar, fazer, checar e agir, em ciclos.

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.

Todo material produzido com IA precisa de auditoria humana. Sempre. A IA escreve rápido e com segurança, mas pode errar uma conta, inventar uma regra ou usar um dado desatualizado. E ela escreve o erro com a mesma confiança do acerto.

Esta apostila foi escrita com ajuda de IA e revisada pelo professor, conta por conta. Mesmo assim, algum erro pode ter escapado. Agora o auditor é você: encontre, comprove e relate.



REGRA DO LABORATÓRIO: IA SEMPRE PASSA POR AUDITORIA HUMANA

A IA não sabe quando errou: o texto errado sai tão convincente quanto o certo.

Quem assina o resultado é uma pessoa: você no trabalho, o analista no relatório, o professor na apostila.

Auditar não é desconfiar da ferramenta. É a etapa que transforma um rascunho da IA num material confiável.

Vale para tudo: código, contas, textos, gráficos e esta apostila.

O que procurar

Tipo de erro	Exemplo	Como conferir
Conta ou número	Um total que não fecha, uma porcentagem errada	Refaça na calculadora
Regra da disciplina	Fórmula trocada, conceito mal explicado	Compare com o livro ou o material da aula
Referência	"Veja a seção 9", quando o assunto está na 11	Vá até a seção indicada
Prompt	Um prompt que faz o Codex errar sempre do mesmo jeito	Rode o prompt duas vezes
Dado desatualizado	Selic, IPCA, regra de imposto	Confira no site oficial (Banco Central, IBGE, Receita Federal)
Texto	Frase confusa, erro de português	Leia o trecho em voz alta

Como auditar com o Codex

PROMPT DE AUDITORIA

use com cuidado

Vou colar um trecho da minha apostila de [DISCIPLINA].

Atue como revisor. Não reescreva o texto.

1. refaça cada conta e mostre o cálculo;
2. aponte afirmações que podem estar erradas ou desatualizadas;
3. para cada problema, diga como eu posso confirmar sozinho.

Trecho: [COLE AQUI]



A IA TAMBÉM ERRA NA AUDITORIA

O Codex pode apontar erros que não existem. Só relate o que você confirmou com uma conta, uma fonte oficial ou um teste.

Registro da auditoria

#	Seção e página	O que está escrito	O que deveria ser	Como confirmei
1				
2				
3				

Como enviar

- 1 Preencha a tabela acima.
- 2 Envie cada erro pelo formulário: **laboratorio-codex.pages.dev/auditoria**
- 3 Aguarde a confirmação do professor.



O QUE VOCÊ GANHA

Cada erro confirmado vale **+1 ponto extra** na avaliação (até 3 pontos), e o seu nome entra na **lista de revisores** da próxima versão da apostila. Se duas pessoas relatarem o mesmo erro, vale o primeiro envio.