Organizar o Projeto

Aula 02 — Cada coisa no seu lugar

Aula 02 · 19/11/2026 · Indicadores 1, 3

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais

Cinco minutos, em pé

Na mesa do grupo, cada pessoa:

1. O que eu terminei desde a última aula?
2. O que está me travando?
3. O que eu vou fazer hoje?

Depois, abram o quadro de issues no milestone **Aula 02**.

O mapa de hoje

- **Bloco 1** — onde mora cada coisa: as camadas do sistema
- **Bloco 2** — as pastas, e o `import` que não acha o arquivo
- **Bloco 3** — a senha fora do código: o `.env`
- **Bloco 4** — mão na massa: o projeto do grupo, organizado — com a suíte verde antes e depois

Retomando a Aula 01

- Qual é o problema que o módulo de vocês vai resolver, em uma frase?
- Uma história de vocês, e um critério de aceitação dela
- Quantos arquivos o projeto tem hoje? Em quantos lugares diferentes?

Bloco 1

Onde mora cada coisa?

A pasta de um grupo, hoje

```
crud.ipynb              crud_final.ipynb        Untitled.ipynb
multa.py                test_multa.py           teste.py
conftest.py             config_local.py         banco.sql
banco_v2.sql            plano-de-teste.xlsx     RECUPERACAO.md
```

Palpite: o dono pediu para trocar o menu de `input()` por uma janela. Em quantos desses arquivos vocês vão mexer?

Tudo misturado numa função só

Uma função do menu, no notebook:

```
def alugar():
    id_cliente = int(input("Cliente: "))                    # tela
    dias = int(input("Dias: "))                             # tela
    if dias <= 0:                                           # regra
        print("Dias inválidos"); return
    cursor.execute("INSERT INTO emprestimo ... VALUES (%s, %s)",
                   (id_cliente, date.today()))              # banco
    conexao.commit()                                        # banco
    print("Aluguel registrado!")                            # tela
```

- Trocar o `input()` por uma janela é reescrever a função inteira. E testar o `dias <= 0` exige banco e alguém digitando.

As camadas de um sistema de informação

Interface
o que a pessoa vê e clica

Regras
o que pode e o que não pode

Dados
o SQL, e só ele

- Cada camada só conversa com a de **baixo**: a interface chama as regras e os dados; os dados nunca abrem janela.
- Troca uma camada, as outras nem percebem — desde que ela continue oferecendo as mesmas funções.
- E a regra se testa sozinha, sem banco e sem tela: é o `test_multa.py` da UC3.

⚠ Erro comum — a interface que escreve SQL

> O atalho tentador: o botão da janela faz o `cursor.execute` direto. Funciona na hora. Na semana seguinte, a mesma consulta está copiada em três botões, e um deles esqueceu o `commit()`.

- Regra de bolso: se a palavra `SELECT` aparece no arquivo da interface, alguma coisa está fora do lugar.
- A interface pede: `dados.listar_jogos(conexao)`. Quem sabe **como** listar é a camada de dados.

Checagem do Bloco 1

Na função `alugar()` do slide 7, diga a camada de cada linha:

- `dias = int(input("Dias: "))`
- `if dias <= 0:`
- `cursor.execute("INSERT INTO emprestimo ...")`
- `print("Aluguel registrado!")`

E: se a regra mudar para "no máximo 7 dias", que camada muda — e quais **não** mudam?



forms.gle/PAJqKLw4b4FCyqZRA

Intervalo

10 minutos

Bloco 2

As pastas

A locadora, organizada

```
main.py                    a janela — o ▶ é aqui
terminal.py                o mesmo sistema no terminal (e o terminal_rich.py)
locadora/
    __init__.py            diz ao Python que a pasta é um pacote
    janela.py, aba_*.py    interface: a janela, uma aba por arquivo
    menu.py, tela_*.py     interface: o menu do terminal
    regras.py              regras do negócio
    dados.py               acesso ao banco
    conexao.py             abre a conexão a partir do .env
tests/                     test_regras.py · test_dados.py · conftest.py
banco/                     criar_tabelas.sql · dados_de_exemplo.sql
README.md · requirements.txt · pyproject.toml · .env.exemplo · .gitignore
```

Baixem o `locadora-integrada.zip` do site e abram a pasta no VS Code.

Como um arquivo acha o outro

```python
# locadora/aba_jogos.py
from locadora import dados, regras

dados.listar_jogos(self.conexao)
regras.validar_jogo(titulo, ano, preco)
```

```python
# tests/test_regras.py
from locadora.regras import multa, validar_jogo
```

- `locadora` é o nome da **pasta**; `dados` e `regras`, os arquivos dentro dela.
- O `main.py` fica **fora** da pasta, na raiz: é dali que o Python começa a procurar.

⚠ Erro comum — o teste que não acha o pacote

Os testes saíram da raiz e foram para `tests/`. Rodando o `pytest`:

```
ModuleNotFoundError: No module named 'locadora'
```

- O teste dentro de `tests/` procura `locadora` **ao lado dele**, e não acha.
- O conserto é uma linha, num arquivo de configuração na raiz:

```toml
# pyproject.toml
[tool.pytest.ini_options]
pythonpath = ["."]
testpaths = ["tests"]
```

`requirements.txt` — o que o projeto precisa

```
psycopg2-binary==2.9.13
pytest==9.2.0
python-dotenv==1.2.4
```

- Quem clona instala tudo de uma vez: `%pip install -r requirements.txt`, numa célula.
- A versão vai junto: o projeto foi testado **com essa**. Daqui a um ano, a mais nova pode ter mudado.

Checagem do Bloco 2

Arrume esta pasta nas camadas e pastas da locadora:

```
crud.ipynb   test_multa.py   multa.py   conftest.py   banco.sql
```

- Para onde vai cada arquivo? E o que sai do `crud.ipynb`, e para onde?
- O `test_multa.py` dá `ModuleNotFoundError: No module named 'locadora'`. O que falta?

forms.gle/uHA4pk8F778bugrt9

Intervalo

10 minutos

Bloco 3

A senha fora do código: o `.env`

Do `config_local.py` ao `.env`

Na UC3, a senha foi para um `config_local.py`. O mercado usa a mesma ideia num arquivo de texto, o `.env` — que o Python, o Docker e o GitHub sabem ler:

```
PG_HOST=localhost
PG_USUARIO=postgres
PG_SENHA=a-senha-desta-maquina
PG_BANCO=locadora
PG_BANCO_DE_TESTE=locadora_teste
```

GLOSSÁRIO

`.env`	arquivo de ambiente — as configurações desta máquina, uma por linha
`environment variable`	variável de ambiente — um valor que o sistema entrega ao programa

Lendo o `.env`

```python
# locadora/conexao.py
import os
from dotenv import load_dotenv

load_dotenv()                          # lê o .env da pasta, se existir
senha = os.getenv("PG_SENHA")
banco = os.getenv("PG_BANCO")
```

- No repositório vai o `.env.exemplo`, com `PG_SENHA=troque-aqui`. Quem clona copia, renomeia e põe a dele.
- O `.gitignore` tem `.env` — antes do primeiro commit, como na UC3.
- Faltou o `.env`? A locadora avisa em português o que fazer, em vez de um erro de conexão.

Dois bancos, um `.env`

Variável	Quem usa	Pode apagar?
`PG_BANCO=locadora`	a janela, o `main.py`	**nunca**
`PG_BANCO_DE_TESTE=locadora_teste`	o `conftest.py`	a cada teste

- A mesma senha, a mesma função de conexão — muda só **qual** banco.
- O teste que apontar para `PG_BANCO` apaga os dados do sistema na primeira rodada.

Checagem do Bloco 3

- Escreva o `.env.exemplo` do projeto de vocês
- Um colega commitou o `.env` no repositório público. O que ele faz agora?
- Por que o `conftest.py` lê `PG_BANCO_DE_TESTE`, e não `PG_BANCO`?

`forms.gle/L73G2sjPG1DWofFs7`

Intervalo

10 minutos

MÃO NA MASSA

O projeto do grupo, organizado

1. Rodem a suíte da UC3 e anotem: **quantos verdes?** Esse número tem de voltar no fim
2. Criem as pastas: o pacote do sistema, `tests/`, `banco/`, e o `main.py` na raiz
3. Tirem as funções do CRUD do notebook para o `dados.py`, com a conexão por parâmetro e sem `print` nem `input`
4. As regras para o `regras.py`; o `.env`, o `.env.exemplo`, o `pyproject.toml` e o `requirements.txt`
5. A suíte de novo: o mesmo número de verdes. Um commit por passo, e as issues da Aula 02 fechadas com `fixes #N`

MÃO NA MASSA

O combinado desta entrega

GLOSSÁRIO

o link do repositório `github.com/<conta>/grupo03-uc3`, com o projeto organizado

O que eu confiro: Nenhum `SELECT` fora da camada de dados · Nenhum `input()` ou `print` nos dados nem nas regras · A suíte com o **mesmo número de verdes** de antes · O `.env` fora do repositório, o `.env.exemplo` dentro · O `README.md` com a estrutura das pastas

> Eu passo de mesa em mesa e vou pedir para rodar a suíte. Se o número de verdes caiu, a gente acha junto o que se perdeu na mudança.

PARA CASA · OPCIONAL

Antes da próxima aula

Testar em casa: clone o repositório numa pasta nova, crie o `.env` a partir do exemplo, instale o `requirements.txt` e rode a suíte. Se não ficar verde, falta algo no repositório.

Tarefa opcional: procure no projeto qualquer `SELECT` fora da camada de dados — no VS Code, **Ctrl+Shift+F**. Achou? Abra uma issue.

Para treinar: o `locadora-integrada.zip`, no site.

Para a próxima aula

Tudo no lugar, e a suíte verde. Mas a atendente no balcão não vai abrir o VS Code. Como ela usa o sistema?

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais