Integrar a Interface

Aula 03 — A janela que a atendente usa

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais

Cinco minutos, em pé

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

Depois, o milestone **Aula 03**: a primeira história, com a interface.

O mapa de hoje

- **Bloco 1** — a janela: as peças do Tkinter
- **Bloco 2** — a janela fina: ela chama as regras e os dados, e mostra o erro do banco
- **Bloco 3** — testar cada camada do jeito que ela se testa
- **Bloco 4** — mão na massa: a primeira história do grupo, com tela

> O Tkinter já vem com o Python: não tem nada para instalar. O ▶ no `main.py` abre a janela.

Retomando a Aula 02

- Em que camada mora o `SELECT`? E o `if dias <= 0`?
- O teste dentro de `tests/` não achava o pacote. O que resolveu?
- O que vai no `.env`, e o que vai no `.env.exemplo`?

Bloco 1

A janela

A atendente não abre o VS Code

- O CRUD funciona, os testes passam — e quem usa o sistema no balcão precisaria rodar célula de notebook e digitar o id no `input()`.
- Para quem usa, a interface **é** o sistema.
- A camada de interface é a única que faltava. As outras duas já estão prontas, e não vão mudar para ela existir.

Onde a gente quer chegar



É o `locadora-integrada.zip`: abram o `main.py` e ▶.

As peças do Tkinter

```
janela = tk.Tk()                          # import tkinter as tk; from tkinter import ttk
ttk.Label(janela, text="Título").grid(row=0, column=0)
campo = ttk.Entry(janela, width=30)
campo.grid(row=0, column=1)
ttk.Button(janela, text="Cadastrar", command=cadastrar).grid(row=1, column=1)
janela.mainloop()          # a janela fica aberta, esperando o clique
```

GLOSSÁRIO

`Label` · `Entry` · `Button`	rótulo · campo de digitar · botão
`grid(row, column)`	posição na grade: linha e coluna
`command=`	a função que roda quando o botão é clicado
`mainloop()`	o laço que mantém a janela viva

A tabela e a mensagem

```
tabela = ttk.Treeview(janela, columns=("titulo", "preco"), show="headings")
tabela.heading("titulo", text="Título")
tabela.insert("", "end", values=("Zelda", "R$ 8,50"))

from tkinter import messagebox
messagebox.showwarning("Confira os dados", "O título não pode ficar em branco.")
```

- O `Treeview` é a tabela; cada `insert` é uma linha.
- O `messagebox` é a janelinha de aviso — é por ela que o erro chega à pessoa, e não pelo terminal.

Para ir além: o vídeo Tkinter — tutorial completo de telas com Python (Hashtag Programação).

Checagem do Bloco 1

Escreva a janela mínima, com:

- um campo para o **nome do cliente**
- um botão **Buscar** que chama a função `buscar()`
- a linha que mantém a janela aberta

Por que o `print` não serve para avisar a atendente que o nome está em branco?

forms.gle/E5UYzeu2uxK7hcfL6

Intervalo

10 minutos

Bloco 2

A janela fina

O que o botão Cadastrar faz

```
def cadastrar(self):
    problemas, campos = regras.preparar_jogo(self.campo_titulo.get(), self.campo_genero.get(),
                                             self.campo_ano.get(), self.campo_preco.get())
    if problemas:                                                  # 1. a regra confere
        messagebox.showwarning("Confira os dados", "\n".join(problemas))
        return
    titulo, genero, ano, preco = campos
    dados.criar_jogo(self.conexao, titulo, genero, ano, preco)   # 2. os dados gravam
    self.atualizar()                                               # 3. a tela mostra
```

- A janela **pede**: à regra, que confira; aos dados, que gravem. Ela mesma não decide nada nem escreve SQL.
- O `preparar_jogo` confere e converte o que foi digitado ("8,50" vira `8.50`), e devolve os problemas **em português** — e se testa sem janela.

⚠ O banco recusou. Quem avisa?

No notebook, o CRUD fazia `print(e)` dentro do `except`. Na janela, esse `print` vai para um terminal que a atendente nem vê.

```
# dados.py — desfaz, e devolve o erro para cima
    except pg.DatabaseError:
        conexao.rollback()
        raise
```

```
# aba_jogos.py — é aqui que se sabe o que dizer à pessoa
    except pg.errors.UniqueViolation:
        messagebox.showerror("Jogo repetido", "Já existe um jogo com esse título.")
```

⚠ Erro comum — o erro engolido

> `except: pass`, ou um `print` no lugar do aviso: o botão não faz nada, a tela não muda, e a atendente clica de novo, e de novo. O defeito existe — só ninguém fica sabendo.

- Erro de banco: desfaz e devolve para cima (`raise`) na camada de dados.
- Na janela: vira uma mensagem que a pessoa entende — "Já existe um jogo com esse título", e não `duplicate key value violates unique constraint`.

O `main.py` liga as peças

```python
from locadora.conexao import conectar
from locadora.janela import JanelaDaLocadora

def main():
    conexao = conectar()
    try:
        JanelaDaLocadora(conexao).mainloop()
    finally:
        conexao.close()          # fecha mesmo se algo der errado no meio

if __name__ == "__main__":
    main()
```

- A janela recebe a conexão pronta. O `terminal.py` só troca a janela pelo `Menu`.

Checagem do Bloco 2

```
def apagar(self):
    cursor = self.conexao.cursor()
    cursor.execute("DELETE FROM jogo WHERE id_jogo = %s", (self.id_escolhido,))
    self.conexao.commit()
    print("apagado")
```

- Que **duas** coisas estão fora do lugar nesta função da janela?
- Reescreva usando a camada de dados e um aviso para a pessoa

forms.gle/WuS7XnSKFkvjnyV76

Intervalo

10 minutos

Bloco 3

Testar cada camada do seu jeito

Cada camada se testa de um jeito

Camada	Como se testa	O que já existe
regras	`pytest`, sem banco nem tela — milissegundos	`test_regras.py`: a multa, o `validar_jogo`
dados	`pytest` no banco de teste	`test_dados.py`, com o `conftest.py`
interface	um **roteiro** que uma pessoa segue, na tela	o caso de teste da Aula 06 da UC3

- Tela é difícil de testar automaticamente — por isso ela tem de ser fina: quanto menos ela decide, menos sobra sem teste.

A regra nova ganha teste antes da tela

```
def test_titulo_em_branco_e_recusado():
    assert validar_jogo("   ", "2017", "8,50") == ["O título não pode ficar em branco."]

def test_ano_antes_de_1971_e_recusado():
    assert validar_jogo("Pong", "1970", "1") == ["O ano tem de ser depois de 1970."]
```

- A mensagem que a atendente vai ler está **no teste**: se alguém mudar o texto sem querer, o teste avisa.
- O `1970` é a mesma fronteira do `CHECK` da tabela, na Aula 11 da UC2 — agora avisada em português, antes de chegar ao banco.

Opcional — o mesmo CRUD com um ORM

Um ORM escreve o SQL a partir de uma classe. No zip: `dados.py` × `dados_peewee.py`.

```python
from peewee import Model, CharField, DecimalField

class Jogo(Model):
    titulo = CharField(unique=True)
    preco_diaria = DecimalField()

Jogo.create(titulo="Zelda", preco_diaria=8.5)      # o INSERT
Jogo.select().where(Jogo.preco_diaria < 10)        # o SELECT ... WHERE
```

- **Não é obrigatório**: o projeto fecha com o psycopg2. Para quem quiser ver: Banco de Dados & Peewee ORM em Python, em 10 minutos (Programador Python).
- No mercado, o mais usado é o **SQLAlchemy** — a mesma ideia, com mais peças.

Checagem do Bloco 3

- Escreva o teste de regra para: preço com letra (`"oito"`) é recusado
- Escreva o roteiro manual para testar o botão **Apagar o selecionado**: pré-condição, passos e resultado esperado
- Por que a janela tem pouco teste automático, e isso não é um problema?



forms.gle/AFwf7Yz7Re7o6t8h7

Intervalo

10 minutos

MÃO NA MASSA

A primeira história, com tela

1. A história do milestone **Aula 03**: os critérios de aceitação dela viram testes de regra — escrevam antes da tela
2. O que faltar nos dados, no `dados.py`, com teste no banco de teste
3. A janela: campos, botão, tabela — chamando só a regra e os dados
4. O `main.py` abrindo a janela, e o erro do banco virando aviso
5. Suíte verde, um commit por passo, e as issues da Aula 03 fechadas

MÃO NA MASSA

O combinado desta entrega

GLOSSÁRIO

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

O que eu confiro: O ▶ no `main.py` abre a janela · Nenhum SQL na janela · O erro do banco chega à pessoa como **aviso**, não como `print` · Os critérios de aceitação da história com **teste** · A suíte verde

> Eu passo de mesa em mesa e vou usar a janela de vocês tentando quebrar: campo em branco, texto no lugar do número, item repetido.

PARA CASA · OPCIONAL

Antes da próxima aula

Testar em casa: tente quebrar a janela do grupo: campo vazio, letra no número, o mesmo item duas vezes, apagar sem escolher. Cada coisa que quebrar vira uma issue.

Tarefa opcional: o vídeo de Tkinter do Bloco 1 — e uma melhoria na janela que ele ensinou.

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

Para a próxima aula

Funciona na máquina de vocês, com quem fez. Uma pessoa que nunca viu o sistema consegue usar sozinha? E como vocês mostram que a versão está pronta?

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais