Validar e Fechar a Versão

Aula 04 — Alguém de fora consegue usar?

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?

Hoje é a última aula de desenvolvimento: o que não fechar hoje, não entra na demonstração.

O mapa de hoje

- **Bloco 1** — alguém que nunca viu o sistema tenta usar
- **Bloco 2** — o plano de novo: o que fechou, o que fica de fora, e por quê
- **Bloco 3** — a versão `v2.0.0`, e uma menção honrosa: o GitHub testando sozinho
- **Bloco 4** — mão na massa: a rodada de teste cruzado, os consertos e a versão

Retomando a Aula 03

- Quem decide o texto do aviso que a atendente lê — a janela, a regra ou os dados?
- O que a janela de vocês fez quando vocês tentaram quebrar?
- Que parte do sistema não tem teste automático, e como ela se testa?

Bloco 1

Alguém de fora tenta usar

Quem fez não enxerga

- Quem fez sabe que o ano vai sem ponto, que o preço aceita vírgula, que tem de clicar na linha antes de apagar.
- A atendente não sabe nada disso. E não vai ler manual.
- Palpite: uma pessoa de outro grupo, sem ajuda, consegue cadastrar um item na janela de vocês na primeira tentativa?

O teste com uma pessoa de verdade

1. Escrevam **uma tarefa** em linguagem do balcão: "cadastre o jogo Hollow Knight, de 2017, a R$ 6,50 a diária"
2. Uma pessoa de **outro grupo** senta na máquina, lê a tarefa e tenta — sozinha
3. Vocês só observam, **calados**, e anotam no papel: onde ela parou, o que ela tentou, quanto tempo levou
4. No fim, uma pergunta: o que foi mais difícil?

> Quem está sendo testado é o **sistema**, não a pessoa. Se ela travou, o defeito é da tela — e a anotação fica com o grupo, não vai para a turma.

A janela para todo mundo

O que conferir	Por quê
a letra é **grande** o bastante para ler de longe	balcão, projetor, quem usa óculos
a tecla **Tab** passa pelos campos na ordem da tela	quem não usa mouse, e quem digita rápido
Enter faz a ação principal	a mão não sai do teclado
o aviso diz **o que fazer**, em português	"o ano tem de ser um número", e não `ValueError`
nada depende **só de cor**	quem não distingue vermelho de verde

Checagem do Bloco 1

- Escreva a tarefa do teste de vocês, em linguagem do balcão — sem dizer qual botão clicar
- O que vocês vão anotar enquanto a pessoa tenta?
- A pessoa travou no campo do preço. É problema dela ou da janela? Por quê?

forms.gle/Wguh1UqWt2AAwuGy8

Intervalo

10 minutos

Bloco 2

O plano de novo

O que o quadro mostra hoje

Abram o quadro de issues e contem, no milestone de cada aula:

	Aula 02	Aula 03	Aula 04
planejadas			
fechadas			
ainda abertas			

- O que ficou aberto: termina hoje, ou sai da versão?
- É a **ação–reflexão–ação** do Projeto Integrador: fazer, olhar o que aconteceu, e decidir o próximo passo com o que se aprendeu.

Corta a história, não a data

- A demonstração é amanhã, e a data não muda.
- O que não fica pronto hoje **sai da versão** — inteiro. Meia história na tela é pior que nenhuma: parece que funciona.
- A issue cortada não é apagada: ganha o milestone **Depois** e um comentário dizendo por quê.

> Cortar com motivo escrito também cumpre o plano. O Indicador 1 olha se o grupo conduziu o plano — e conduzir inclui decidir o que fica de fora.

⚠ Erro comum — a noite antes da entrega

> "Amanhã de manhã a gente termina tudo." A história nova entra às 23h, sem teste, quebra a que funcionava, e a demonstração começa com um erro na tela.

- A regra da versão: **só entra o que está verde na suíte e foi usado por alguém de fora**.
- Depois da tag, a `main` não recebe mais nada até a demonstração.

Checagem do Bloco 2

São 20h de hoje. Ainda abertas:

- "Tela de devolução" — começada, sem teste
- "Aviso de título repetido" — pronta, falta o commit
- "Relatório de jogos parados" — não começada

O que entra na versão, o que sai, e o que vocês escrevem na issue que sai?



forms.gle/fizjNXgijNQeZ1Yb8

Intervalo

10 minutos

Bloco 3

A versão `v2.0.0`

Por que 2.0.0, e não 1.1.0

Da versão 1.0.0 (UC3) para hoje	Muda
a pessoa usa por uma **janela**, não pelo notebook	quem usa precisa reaprender
o projeto mudou de **estrutura** — pastas, `.env`, `main.py`	quem clona precisa refazer a configuração

- É a regra da UC3, Aula 08: mudou o jeito de usar, o primeiro número sobe.

```
git tag v2.0.0
git push --tags
```

O que vai junto com a versão

1. **README.md** — o problema, a estrutura das pastas, como rodar, como testar
2. **requirements.txt** e **.env.exemplo** — tudo o que uma máquina nova precisa
3. O **plano de teste** da UC3, atualizado com os casos das telas
4. No GitHub, um **Release** da tag, com três linhas: o que entrou, o que ficou para depois, o defeito conhecido

O teste final é o da UC3: clonar numa pasta nova e seguir só o README.

Menção honrosa — o GitHub testando sozinho

```yaml
# .github/workflows/testes.yml
name: testes
on: push
jobs:
  testes:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: "3.12"
      - run: pip install -r requirements.txt
      - run: pytest tests/test_regras.py
```

- A cada `push`, uma máquina limpa instala o projeto e roda os testes de regra. Quebrou? O commit aparece com um **X vermelho**.
- Isso é **integração contínua (CI)**. O passo seguinte, **entrega contínua (CD)**, publica a versão sozinho quando tudo está verde.
- Não é obrigatório — o exemplo da locadora já traz o arquivo.

Checagem do Bloco 3

- Depois da `v2.0.0`, o grupo conserta o texto de um aviso. Qual é a próxima versão? E se entrar a tela de devolução?
- Escreva as três linhas do Release de vocês
- No arquivo do GitHub Actions, por que o passo `pip install -r requirements.txt` vem antes do `pytest`?



`forms.gle/Bcs5onNexdXBR6mQ9`

Intervalo

10 minutos

MÃO NA MASSA

Teste cruzado, consertos e a versão

1. **Rodada de teste cruzado (20 min):** uma pessoa do grupo vai testar a janela de outro grupo, com a tarefa escrita; o resto do grupo recebe outra pessoa e anota
2. Cada travada anotada vira uma **issue** — e as que cabem hoje, viram conserto
3. O quadro revisto: o que entra, o que sai com motivo escrito
4. README, `requirements.txt`, `.env.exemplo` e plano de teste atualizados; clone numa pasta nova
5. Suíte verde: a tag `v2.0.0`, o `push --tags` e o Release no GitHub

MÃO NA MASSA

O combinado desta entrega

GLOSSÁRIO

o link do repositório `github.com/<conta>/grupo03-uc3`, com a tag `v2.0.0` e o Release

O que eu confiro: A anotação do teste com uma pessoa de fora, e o que mudou por causa dela · O quadro com o que **entrou** e o que **saiu**, com motivo · A suíte verde na `v2.0.0` · O clone em pasta nova funcionando só com o README · O Release com as três linhas

> Eu passo de mesa em mesa. A pergunta que eu vou fazer: "o que vocês mudaram depois de ver alguém de fora usar?".

PARA CASA · OPCIONAL

Antes da próxima aula

Testar em casa: clone a `v2.0.0` numa pasta nova e siga o README ao pé da letra. É exatamente o que a demonstração de amanhã vai fazer.

Tarefa opcional: coloque o `.github/workflows/testes.yml` no repositório e veja o primeiro ✓ verde no GitHub.

Para treinar: o `locadora-integrada.zip`, que já traz o arquivo do GitHub Actions.

Para a próxima aula

Amanhã é a demonstração. O que vocês vão mostrar — e o que vocês aprenderam no caminho, que não aparece na tela?

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais