Controle de Versão

Aula 07 — Voltar para a versão que funcionava

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais

O mapa de hoje

- **Bloco 1** — a pasta copiada, e o que ela não guarda
- **Bloco 2** — o histórico: gravar, ler e voltar no tempo
- **Bloco 3** — o que nunca entra no repositório
- **Bloco 4** — mão na massa: o projeto de vocês no GitHub

> O Git vocês já viram na UC1, com o professor João: `add`, `commit`, `push`. Hoje ele deixa de ser "o jeito de mandar para o GitHub" e vira a máquina do tempo do projeto.

Retomando a Aula 06

- O teste achou o defeito, vocês corrigiram — e quebrou outra coisa. Como voltar?
- Quem fez a tarefa de copiar a pasta antes de mexer: quantas cópias tem agora?
- O que um relatório de defeito precisa ter para outra pessoa reproduzir?

Bloco 1

A pasta `uc3-final-agora-vai`

Quatro pastas, uma pergunta

```
uc3/
uc3 - Copia/
uc3-final/
uc3-final-2-agora-vai/
```

- Qual delas tem a versão que **passava em todos os testes**?
- O que mudou de uma para a outra?
- Quem do grupo mexeu, e por quê?

⚠ O que a cópia não guarda

A pergunta	A pasta copiada responde?
o que mudou	não — só olhando arquivo por arquivo
quando	mais ou menos — pela data da pasta
quem	não
por quê	não
e se duas pessoas mexeram no **mesmo arquivo**?	quem copiar por último apaga o trabalho do outro

E o pendrive com as quatro pastas fica na mochila de uma pessoa só.

Controle de versão

- Cada vez que o grupo grava, fica registrado: **o que** mudou, **quem**, **quando** e — se a mensagem for boa — **por quê**.
- Cada gravação é uma **versão**, e qualquer versão pode ser recuperada.
- A pasta em que cada um trabalha é o **espaço de trabalho**; o lugar onde as versões ficam guardadas é o **repositório**.

GLOSSÁRIO

`version control`	controle de versão — guardar e recuperar cada versão do projeto
`repository`	repositório — o histórico guardado de todas as versões
`working directory`	espaço de trabalho — a pasta onde cada um edita
`commit`	gravação — uma versão, com autor, data e mensagem

Centralizado e distribuído

	Centralizado	Distribuído — o Git
onde fica o histórico	num servidor só	**inteiro** em cada máquina
sem internet	não dá para gravar versão	grava normalmente
o servidor quebrou	o histórico se foi	qualquer cópia recupera tudo

- O **GitHub não é o Git**. É uma cópia a mais do repositório, num servidor, que o grupo inteiro alcança.

Checagem do Bloco 1

- Das quatro perguntas da cópia (o quê, quando, quem, por quê), qual o Git só responde se o **grupo** fizer a parte dele?
- O notebook de um colega do grupo queimou. Com o Git, o que ele perdeu? E com as pastas copiadas no pendrive dele?
- Verdadeiro ou falso: "sem internet, não dá para usar o Git"

forms.gle/De5Emr3WzZZe9G7Z9

Intervalo

10 minutos

Bloco 2

O histórico

Configurar quem está gravando

No **Git Bash**, dentro da pasta do projeto:

```
git config user.name "Ana do grupo 3"
git config user.email "ana@email.com"
```

No laboratório, **sem** o `--global` da UC1. A máquina é de todos: com `--global`, os commits de quem sentar aqui amanhã saem com o nome de vocês.

O repositório vai ser **público**: usem o nome e o e-mail que vocês aceitam que fiquem públicos.

Mudar, preparar, gravar

O que se quer	Painel do VS Code (Ctrl+Shift+G)	Git Bash
ver o que mudou	a lista **Changes**	`git status`
ver a mudança linha a linha	clicar no arquivo	`git diff`
escolher o que vai na versão	o **+** ao lado do arquivo	`git add multa.py`
gravar a versão	escrever a mensagem e **Commit**	`git commit -m "..."`
ver as versões	**Source Control Graph**	`git log --oneline`

- O painel faz o dia a dia. Os comandos são os mesmos da UC1 — o painel só aperta os botões por vocês.

⚠ Erro comum — a mensagem que não diz nada

Assim, não

```
ajustes
final
agora vai
asdf
```

Assim, sim

```
Multa passa a cobrar a partir do 3o dia
Remove senha do conftest; agora vem do config_local
Corrige busca por titulo: titlo -> titulo
```

Daqui a um mês, a mensagem é o que vocês vão ler procurando o commit que quebrou. "ajustes" não ajuda ninguém a achar nada.

O histórico da locadora

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

1. Rodem os testes no painel Testing — um fica vermelho
2. Abram o **Source Control Graph**: seis versões, de duas pessoas, em uma semana
3. Leiam só as **mensagens**

> Palpite, antes de abrir qualquer arquivo: só pelas mensagens, qual commit quebrou o teste?

Achando o commit que quebrou

No Git Bash, dentro da pasta:

```
git log --oneline                # as versões, da mais nova para a mais velha
git log -p multa.py              # cada mudança que o multa.py já sofreu
```

```
-def multa(dias_atraso):
-    """Multa pela devolução atrasada, em reais."""
-    if dias_atraso >= 3:
+def multa(dias_atraso, cliente_fiel=False):
+    """Multa pela devolução atrasada, em reais. Cliente fiel paga metade."""
+    if dias_atraso > 3:
```

- O desconto para cliente fiel trouxe junto o `>` no lugar do `>=` — a fronteira da Aula 02. No painel, clicar na versão no **Source Control Graph** mostra o mesmo.

Voltar sem apagar a história

Comando	O que faz	Quando usar
`git revert <commit>`	cria um commit **novo** que desfaz aquele	a versão errada já foi para o GitHub
`git restore --source=<commit> multa.py`	traz o arquivo como era naquela versão	quero só um arquivo de volta, e depois gravo

- Os dois deixam a suíte verde. Os dois **guardam** a versão errada no histórico — de propósito: ela explica o que aconteceu.
- O que **não** se faz: apagar a pasta e copiar a do pendrive.

GLOSSÁRIO

`revert`	reverter — desfazer com um commit novo, que fica registrado
`restore`	restaurar — trazer um arquivo de outra versão

Checagem do Bloco 2

No `locadora-historico`:

- Qual commit quebrou o teste? Como você descobriu?
- Escreva o comando que desfaz esse commit sem apagar a história
- Reescreva a mensagem "Desconto de metade da multa para cliente fiel" para avisar o que mais ela mudou

`forms.gle/xvak6uwxPYcMdUQn8`

Intervalo

10 minutos

Bloco 3

O que nunca entra no repositório

A senha sumiu?

No mesmo histórico, a Ana gravou os dados do banco — e quinze minutos depois:

```
Remove a senha do repositorio
Ops, a senha nao podia estar aqui.
```

O `config_local.py` não está mais na pasta.

> Palpite: se esse repositório fosse público no GitHub, a senha estaria segura?

⚠ Apagar não basta

```
git log -p -S "SENHA" --oneline     # todo commit que pôs ou tirou "SENHA"
```

```
... Remove a senha do repositorio
-SENHA = "Locadora#2026"
... Dados de conexao com o banco de teste
+SENHA = "Locadora#2026"
```

- O commit que **pôs** a senha continua no histórico — é para isso que o histórico existe.
- Repositório público: qualquer pessoa clona e roda esse comando. Robôs fazem isso o dia inteiro no GitHub.
- Senha que entrou num commit público é senha **vazada**: o conserto é **trocar a senha**, não apagar o arquivo.

`.gitignore` — antes do primeiro commit

```
# .gitignore
config_local.py
__pycache__/
.pytest_cache/
```

```
# conftest.py
from config_local import SENHA          # o arquivo fica só na máquina

BANCO_DE_TESTE = { ..., "password": SENHA }
```

- No repositório vai um `config_local.exemplo.py`, com `SENHA = "troque-aqui"`. Quem clona copia, renomeia e põe a própria senha.
- O `conftest.py` da Aula 05 tinha a senha escrita no arquivo. Hoje ela sai de lá.

Para o GitHub

1. No painel do VS Code: **Publish Branch** → **Publish to GitHub public repository**
2. O navegador pede para entrar no GitHub — a conta da UC1
3. Antes de confirmar, olhem a lista de arquivos: o `config_local.py` **não** pode estar nela
4. Depois, cada commit novo sobe com **Sync Changes** — ou `git push`, no Git Bash

> No fim da aula, **saiam do GitHub** no VS Code e no navegador. Na máquina do laboratório, o login fica salvo para quem sentar aqui depois.

Checagem do Bloco 3

- Um colega commitou a senha num repositório público, e em seguida apagou o arquivo e commitou de novo. O que ele precisa fazer agora?
- Escreva o `.gitignore` do projeto de vocês
- Por que o `config_local.exemplo.py` vai para o repositório, e o `config_local.py` não?



`forms.gle/z3t6JB8wdXYt22616`

Intervalo

10 minutos

MÃO NA MASSA

O projeto de vocês, com histórico

1. Na pasta `uc3` do grupo: **Initialize Repository** no painel, e o `git config` sem `--global`
2. **Antes** do primeiro commit: o `.gitignore`, o `config_local.py` e o `config_local.exemplo.py`; o `conftest.py` passa a ler a senha de lá
3. Um commit por assunto, com mensagem que diz o quê e por quê: os testes de cada aula, o plano de teste, a senha que saiu do conftest
4. Rodem a suíte inteira: verde antes de publicar
5. Publiquem no GitHub como repositório **público**, conferindo a lista de arquivos

MÃO NA MASSA

O combinado desta entrega

GLOSSÁRIO

o link do repositório — `github.com/<conta>/grupo03-uc3`

O que eu confiro: O `config_local.py` **não** aparece em nenhum commit — nem no primeiro · O `.gitignore` existe e foi commitado · As mensagens dizem **o quê e por quê** · O histórico tem mais de um commit, e de mais de uma pessoa do grupo · Clonando do GitHub numa pasta nova, com o `config_local.py` criado, a suíte fica **verde**

> Eu passo de mesa em mesa e vou rodar o `git log -p -S "SENHA"` no repositório de vocês. Se aparecer, a gente troca a senha juntos — antes de ir embora.

PARA CASA · OPCIONAL

Antes da próxima aula

Testar em casa: clone o repositório do grupo numa pasta nova, crie o `config_local.py` e rode a suíte. Se não ficar verde, falta alguma coisa no repositório.

Tarefa opcional: no `locadora-historico`, desfaça o commit que quebrou a multa com `git revert`, e veja o histórico depois. Quantos commits tem agora?

Para treinar: o `locadora-historico.zip` está no site, na página da aula.

Para ver acontecer: demonstrações da Aula 07 — achar no histórico o commit que quebrou a multa, e ver a senha apagada continuar lá — e oito perguntas com resposta na hora.

Para a próxima aula

Duas pessoas do grupo precisam mexer no mesmo arquivo ao mesmo tempo — uma conserta um defeito, a outra faz uma regra nova — sem estragar a versão que funciona. Como?

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais