

Repository colaborativ cu recuperare controlată

Fișă publică de proiect · exerciții și criterii de acceptare

Lucrezi într-un repository fictiv, creezi branch-uri și rezolvi un conflict intenționat. Exersezi recuperarea fără a risca istoricul unui proiect activ.

Livrabile

- Repository
- Branch
- Conflict rezolvat
- Pull request
- Exercițiu reflog

Criterii de acceptare

- Nu sunt secrete în commit
- Diff-ul este revizuit
- Commitul recuperat este păstrat pe un branch

1. De ce Git

Creează două versiuni ale unui fișier și inspectează diferența.

Verificare orientativă

git diff arată modificarea înainte de commit; git log păstrează istoricul după commit.

De evitat

Git nu este backup pentru fișiere ignorate și secrete locale.

2. Primul repository

Inițializează un repository cu README și .gitignore.

Verificare orientativă

Verifică git status înainte de primul commit. Fișierul .env trebuie ignorat, iar .env.example să conțină doar valori demonstrative.

De evitat

Adăugarea ulterioară în .gitignore nu șterge un secret din istoric.

3. Branch-uri

Creează un branch pentru schimbarea unei pagini.

Verificare orientativă

Fă commit pe branch și revino pe ramura principală. Verifică că modificarea nu apare până la integrare.

De evitat

Lucrul necomis poate fi transportat între branch-uri în anumite situații.

4. Merge și conflicte

Produce un conflict controlat în același rând din README.

Verificare orientativă

Citește ambele variante, rezolvă sensul, elimină marker-ele și rulează verificarea proiectului.

De evitat

Acceptarea automată a unei părți poate pierde modificări necesare.

5. Rebase pe scurt

Compară merge cu rebase într-un repository de laborator.

Verificare orientativă

Desenează istoricul înainte și după. Folosește rebase doar pe istoricul pentru care poți coordona rescrierea.

De evitat

Nu rescrie istoricul partajat fără acordul echipei.

6. Pull requests pe GitHub

Pregătește un pull request cu problemă, schimbare și verificare.

Verificare orientativă

Include un exemplu înainte/după și rezultatele testelor. Diferența trebuie să rămână concentrată pe o singură problemă.

De evitat

O descriere lungă nu compensează o modificare greu de verificat.

7. Recuperarea din greșeli

Recuperează un commit folosind reflog într-un repository fictiv.

Verificare orientativă

Identifică hash-ul și creează un branch nou care îl păstrează. Pentru anularea publică preferă revert când este potrivit.

De evitat

Nu experimenta cu `reset --hard` pe munca nesalvată.

Atelier final: Conflict de Git cu recuperare

Timp de lucru estimat: 40 de minute. Folosește datele de laborator de mai jos și salvează rezultatul separat de orice proiect real.

Date de pornire

Repository de laborator cu README. Branchul main schimbă instrucțiunea de instalare, branchul feature schimbă aceeași linie și adaugă un exemplu. Nu folosi un repository de producție.

1. Pregătire — 5 minute

Extrage intrările și condițiile din enunț. Notează ce rezultat trebuie produs și ce informații nu sunt confirmate. Pregătește fișierele sau tabelul în care vei păstra dovada lucrului.

2. Construire — 15 minute

Creează conflictul, citește ambele intenții și păstrează instrucțiunea corectă plus exemplul. Înregistrează commiturile, apoi exersează recuperarea unui commit pe un branch separat folosind `reflog`.

3. Variație și verificare — 15 minute

Prezintă `diff`-ul ca într-un pull request: problemă, soluție și verificare. Arată ce fișiere temporare trebuie excluse înainte de commit.

4. Autoevaluare — 5 minute

Compară rezultatul cu reperele de mai jos. Înregistrează cel puțin o eroare identificată sau un caz-limită verificat, plus corectura ori explicația comportamentului observat.

Repere de rezolvare

Conflictul este rezolvat fără marcaje rămase și fără pierderea exemplului. Commitul recuperat este referit de un branch. Istoricul partajat nu este rescris pentru a simplifica exercițiul.

Ce predai

- Repository
- Branch
- Conflict rezolvat
- Pull request
- Exercițiu `reflog`

Criterii de acceptare

- Nu sunt secrete în commit
- `Diff`-ul este revizuit
- Commitul recuperat este păstrat pe un branch

Păstrează o versiune inițială și una revizuită, cu explicația diferențelor. Dacă un criteriu nu este îndeplinit, revino la lecția relevantă și repetă verificarea. Durata este orientativă; instalarea mediului și aprofundarea pot

necesita timp suplimentar.

Registrul de lucru

Pentru fiecare etapă notează intrarea, rezultatul, dovada verificării, eroarea găsită și corectura. Nu introduce date personale sau secrete în fișa partajată.

Documentație și surse

- Pro Git — Git

<https://git-scm.com/book/en/v2>