Sesión 6: Ramas (CLI/GUI)

Trabaja con ramas para aislar cambios, fusionar con control y resolver conflictos.

Qué vas a lograr

- Crear/cambiar ramas rápido.
- Fusionar con merge o rebase según convenga.
- Resolver conflictos de forma segura.

Comandos esenciales (multiplataforma)

- Crear y cambiar de rama

```
git switch -c feature/login   # o: git checkout -b feature/login
git switch main               # o: git checkout main
```

- Publicar y sincronizar

```
git push -u origin feature/login
git fetch --all --prune
```

- Fusionar

```
# Desde main, fusiona la rama de feature
git switch main
git merge --no-ff feature/login -m "merge: integra feature/login"
```

- Rebase (alternativa)

```
git switch feature/login
git rebase main
# Si hay conflictos, resuélvelos y luego:
git rebase --continue
```

- Resolver conflictos

```
git status
# Edita archivos con <<<<<<<, =======, >>>>>>>
git add <archivo>
# Si estabas en un merge:
git commit
# Si estabas en rebase:
git rebase --continue
```

GUI recomendadas

- GitHub Desktop (Win/Mac), GitKraken (Win/Mac/Linux), Sourcetree (Win/Mac), VS Code Source Control.

Estrategias comunes

- Feature branches: una rama por funcionalidad.
- Merge sin fast-forward para mantener historia explícita en main.
- Rebase local para limpiar antes de publicar.

Ejercicio

- Crea dos ramas `feature/a` y `feature/b` que toquen el mismo archivo, provoca un conflicto y resuélvelo por CLI y por GUI.

Problemas comunes

- Muchos merge commits: usa `--no-ff` solo para merges a main; para ramas cortas, fast-forward puede ser suficiente.
- Reescritura de historia publicada: evita rebase de ramas ya compartidas; prefiere `git revert`.