DÍA 6 – GIT INTERMEDIO: RAMAS, MERGE, CONFLICTOS Y GITHUB

Objetivo del día

- Entender qué es una **rama (branch)** en Git
- Crear, cambiar y fusionar ramas (**git branch**, **checkout**, **merge**)
- Resolver conflictos cuando dos ramas modifican lo mismo
- Subir cambios al repositorio remoto de GitHubLS
- Aplicar todo con práctica real sobre tu script **perfil.py**

¿Qué es una rama en Git?

> **Analogía:** Imagina que estás escribiendo un libro.
> Para probar una nueva idea, no borras el original: haces una **copia aparte** y pruebas allí. Eso es una **rama**.

```
main  ← línea principal (oficial)
│
├── experimento  ← una nueva idea, o funcionalidad en desarrollo
```

COMANDOS BÁSICOS EN GIT

Acción	Comando
Ver ramas existentes	`git branch`
Crear nueva rama	`git branch nombre-rama`
Crear y cambiar a nueva rama	`git checkout -b nombre-rama`
Cambiar de rama	`git checkout nombre-rama`
Volver a la rama principal (main)	`git checkout main`
Fusionar rama con main	`git merge nombre-rama`
Borrar rama	`git branch -d nombre-rama`
Ver en qué rama estás	`git status`
Ver historial de commits	`git log --oneline --graph --all`
Ver diferencias entre ramas	`git diff rama1..rama2`
Subir rama actual a GitHub	`git push -u origin nombre-rama`
Subir cambios nuevos	`git push`
Clonar repositorio de GitHub	`git clone URL-del-repo`
Añadir archivo remoto (una vez)	`git remote add origin URL-del-repo`
Ver qué remoto tienes configurado	`git remote -v`
Descargar cambios del repositorio	`git pull`
Ver resumen de cambios con ramas	`git log --oneline --decorate --graph --all`
Guarda archivos ya creados, no nuevos	`git commit -am`

🌐 SUBIR TU REPO A GITHUB

1. Conectar con GitHub

Crea un repo vacío en GitHub con el mismo nombre del proyecto (ej: **mi_proyecto**), luego:

```
git remote add origin https://github.com/tuusuario/mi_proyecto.git
git branch -M main
git push -u origin main
```

Subir cambios posteriores:

```
git add .
git commit -m "Comentario"
git push
```

EJEMPLO GUIADO PASO A PASO

```
cd ruta/de/tu/proyecto
git init
git add perfil.py
git commit -m "Versión inicial de perfil"
```

1. Crear una rama nueva

```
git checkout -b experimento
```

```
# perfil.py (modificación en rama experimento)
print("Hola, soy la versión experimental.")
```

2. Hacer commit en la rama

```
git add perfil.py
git commit -m "Cambios en rama experimento"
```

3. Volver a main y fusionar

```
git checkout main
git merge experimento
```

⚔ ¿Qué es un conflicto en Git?

> **Analogía:** Dos personas editan el mismo párrafo de un documento.
> Git se queda bloqueado y **te pide decidir**.

⚠ **Ejemplo de conflicto**

```
<<<<<<< HEAD
print("Hola, soy el original.")
=======
print("Hola, soy la nueva versión.")
>>>>>>> experimento
```

Eliminas los símbolos y decides:

```
print("Hola, soy la nueva versión.")
```

```
git add perfil.py
git commit -m "Resuelto conflicto"
```

RETO GUIADO

1. Crear rama **experimento**
2. Hacer cambios en **perfil.py**
3. Simular conflicto modificando lo mismo en **main**
4. Hacer merge y resolverlo manualmente

---

5 EJERCICIOS INTERACTIVOS

1. Crear rama **feature-x**, editar **perfil.py**, volver a main
2. Fusionar **feature-x** en **main** sin conflicto
3. Crear conflicto modificando misma línea en ambas ramas
4. Resolver conflicto y hacer commit de fusión
5. Crear 2 ramas hijas y fusionarlas antes de volver a **main**

FRASE PARA RECORDAR

> “Una rama en Git es como un laboratorio: pruebas sin romper nada.
> Un conflicto es solo una decisión pendiente.”