UNIDAD 2 — CONTROL DE VERSIONES CON GIT Y GITHUB

SESIÓN 6: Comandos esenciales — git init, git add, git commit, git push, git pull

1. INTRODUCCIÓN

En el desarrollo de software moderno, Git es una herramienta fundamental para:

- gestionar versiones
- trabajar en equipo
- mantener la historia completa de un proyecto.

El objetivo es dominar el uso de los comandos fundamentales:

- git init
- git add
- git commit
- git push
- git pull

Cada uno representa una etapa en el ciclo de vida del código.

Comprender su función es clave para todo programador o estudiante de informática.

2. INICIALIZAR EL REPOSITORIO: git init

Todo proyecto bajo control de versiones comienza con un repositorio.

El comando **git init** activa Git dentro de una carpeta local y permite comenzar a registrar cambios.

Este comando crea un entorno de trabajo bajo control de versiones en el equipo local.

Genera una carpeta oculta llamada .git, que contiene:

- La base de datos de versiones.
- Las ramas.
- La configuración del proyecto.

Ejemplo práctico:

mkdir MiSitioWeb

cd MiSitioWeb

git init

Después de ejecutar el comando, al escribir ls -a, se observa la carpeta .git, lo que indica que Git está listo para rastrear los cambios.

Conclusión

git init es el punto de partida de todo proyecto con Git.
Permite trabajar incluso sin conexión y marca el inicio del historial del proyecto.

3. AÑADIR ARCHIVOS: git add

Una vez inicializado el repositorio, debemos indicar qué archivos queremos rastrear.

Git no registra automáticamente todos los cambios; el desarrollador decide qué incluir en la siguiente versión.

El comando **git add** mueve archivos desde el área de trabajo (Working Directory) al área de preparación (Staging Area).

Solo los archivos añadidos serán incluidos en el siguiente commit.

Ejemplos

git add index.html

git add .

git add *.css

Estado del archivo	Comando	Descripción
Nuevo o modificado	git add archivo.txt	Añadido al área de preparación
Eliminado	git rm archivo.txt	Eliminado del control de versiones
Revertido	git reset archivo.txt	Devuelto al estado anterior

El comando **git add** representa una decisión consciente del desarrollador sobre qué cambios incluir.
Permite un control detallado y evita subir errores o archivos innecesarios.

4. CONFIRMAR CAMBIOS: git commit

Después de preparar los archivos, se debe registrar una versión oficial del proyecto.

El comando **git commit** crea un punto de control dentro del historial del desarrollo.

Un commit guarda una fotografía exacta del proyecto e incluye:

- Qué se cambió.
- Quién lo cambió.
- Cuándo lo cambió.
- Por qué se cambió.

Ejemplo

git commit -m "Agrega validación de formulario"

git log –oneline

Buenas prácticas en mensajes de commit

Correcto	Incorrecto
Corrige error en login	Arreglado lo del login
Añade botón de registro	Commit final
Optimiza carga de imágenes	Cambios varios

git commit es el corazón del versionado.
Cada commit forma parte del historial y permite volver atrás si se detectan errores.
Un mensaje claro y descriptivo refleja profesionalismo.

5. SUBIR CAMBIOS: git push

git push permite subir los commits locales al repositorio remoto, compartiendo el trabajo con el resto del equipo.

Este comando sincroniza los commits locales con un servidor remoto (por ejemplo, GitHub, GitLab o Bitbucket).

Es el paso en el que el trabajo individual se integra con el global del equipo.

Ejemplo práctico

git push origin main

- origin: nombre del remoto.
- main: rama principal del proyecto.

Ejemplo completo del flujo

git init

git add .

git commit -m "Primer commit"

git remote add origin https://github.com/luisgomez/ProyectoWeb.git

git push -u origin main

Problemas comunes

Mensaje de error	Significado	Solución
rejected non-fast-forward	Hay cambios en el remoto que no tienes	git pull origin main antes del push
no upstream branch	No se ha configurado la rama remota	git push -u origin main

git push sincroniza los cambios locales con el trabajo global del equipo.
Subir cambios de forma frecuente evita conflictos y mejora la colaboración.

6. TRAER CAMBIOS: git pull

Así como puedes subir cambios, otros también pueden hacerlo.

git pull actualiza tu repositorio local con las modificaciones más recientes del remoto.

Combina dos operaciones:

- git fetch : Descarga las actualizaciones.
- git merge : Fusiona los cambios.

Ejemplo

git pull origin main

Caso de conflicto

Si dos personas editan la misma línea, Git mostrará algo así:

<<<<<<< HEAD

color: red;

=======

color: blue;

>>>>>>> origin/main

Debe resolverse manualmente y luego ejecutar:

git add .

git commit -m "Resuelve conflicto de estilos"

git push origin main

- git pull mantiene el proyecto actualizado y evita sobrescribir el trabajo de otros.
- Es recomendable ejecutarlo siempre antes de comenzar a trabajar.

7. CASO REAL: EMPRESA PIXELFORGE

Contexto

Seis programadores colaboran en un proyecto web.
Antes de usar Git, compartían archivos por correo, generando versiones duplicadas y pérdida de trabajo.

Implementación

- El líder crea un repositorio remoto en GitHub.
- Cada miembro lo clona en su equipo:

git clone https://github.com/pixelforge/web-corporativa.git

- Cada desarrollador trabaja en su módulo:

git add .

git commit -m "Crea componente de contacto"

git push origin main

- Al iniciar el día, todos actualizan:

git pull origin main

Resultados

- Productividad duplicada.
- Trazabilidad completa de cambios.
- Copias de seguridad automáticas.
- Sin pérdida de trabajo ni conflictos graves.

Conclusión

- Git transformó la colaboración en PixelForge.
- Pasaron de un entorno desorganizado a un flujo sincronizado y profesional.

8. RESUMEN

Comando	Etapa	Descripción	Ejemplo
git init	Inicio	Crea el repositorio local	git init
git add	Preparación	Selecciona archivos a confirmar	git add .
git commit	Confirmación	Guarda una versión con mensaje	git commit -m "mensaje"
git push	Publicación	Sube los cambios al remoto	git push origin main
git pull	Sincronización	Trae y fusiona cambios remotos	git pull origin main

- Dominar estos comandos constituye la base de todo flujo de trabajo con Git.
- Son los cimientos para comprender temas más avanzados como ramas (branching), fusiones (merge) y flujos colaborativos (Git Flow, GitHub Flow).
- Git no es solo una herramienta técnica: es una forma de pensar en la colaboración y en la responsabilidad compartida del código.

9. EJERCICIOS FINALES DE REPASO Y PROYECTOS

Ejercicio 1:
Crea un proyecto sencillo (por ejemplo, HTML o Java) e inicializa un repositorio con git init.
Realiza al menos tres commits con mensajes descriptivos.

Ejercicio 2:
Sube tu proyecto a GitHub con git push y verifica los archivos en línea.

Ejercicio 3:
En parejas, simulad un conflicto editando la misma línea de un archivo.
Resuélvelo usando git pull y git merge.

Ejercicio 4:
Publica tu proyecto en GitHub Pages o comparte el enlace de tu repositorio con el profesor.

Ejercicio 1 — Crear proyecto, iniciar Git y hacer 3 commits

Objetivo

Crear un repositorio local, versionar archivos y ver el historial.

Pasos

1. **Crea la carpeta del proyecto y entra**

```
mkdir MiProyectoGit
cd MiProyectoGit
```

2. **Crea archivos iniciales**

```
echo "<!doctype html><html><head><meta charset='utf-8'><title>Inicio</title></head><body><h1>Hola</h1></body></html>" > index.html

echo "body { font-family: Arial, sans-serif; }" > estilos.css
```

En PowerShell, si el echo da problemas con comillas, puedes abrir y guardar con tu editor (VS Code: code .).

3. **Inicializa Git**

```
git init
```

Verifica:

```
git status
```

Deberías ver untracked files: index.html, estilos.css.

4. **Commit 1: estructura base**

```
git add index.html estilos.css

git commit -m "Estructura base: HTML + CSS inicial"
```

5. **Modifica el HTML y haz el commit 2**

```
echo "<p>Sección de noticias</p>" >> index.html

git add index.html

git commit -m "Añade sección de noticias a la portada"
```

6. **Crea un README y haz el commit 3**

```
echo "# MiProyectoGit" > README.md

git add README.md

git commit -m "Crea README con descripción"
```

7. **Revisa el historial**

```
git log --oneline
```

Deberías ver **3 commits** con tus mensajes.

Errores comunes y solución

- “Please tell me who you are”: configura tu usuario de Git:

git config --global user.name "Tu Nombre"

git config --global user.email "tuemail@ejemplo.com"

- Nada que confirmar: asegúrate de haber modificado/añadido archivos y ejecuta git add antes del commit.

Ejercicio 2 — Subir el proyecto a GitHub con git push

Objetivo

Conectar el repo local con GitHub y subir tus commits.

Pasos

1. **Crea un repositorio nuevo en GitHub**

- Entra en GitHub → **New** → Nombre: MiProyectoGit → **Create repository**.
- Copia la **URL HTTPS** (ej.: https://github.com/tuusuario/MiProyectoGit.git).

2. **Conecta tu repo local con el remoto**

git remote add origin https://github.com/tuusuario/MiProyectoGit.git

git remote -v

Debe listar origin (fetch/push) con tu URL.

3. **Asegura que tu rama se llama main**

git branch -M main

4. **Sube al remoto (primera vez)**

git push -u origin main

5. **Verifica en GitHub**

- Recarga la página del repo → verás tus archivos y commits.

Ejercicio 3 — Simular conflicto y resolverlo

Objetivo

Generar un conflicto real, resolverlo y dejar el repo sincronizado.

Preparación

- Dos personas: **A** y **B**.
- Ambos clonan el mismo repo de GitHub.

Para A y B:

```
git clone https://github.com/tuusuario/MiProyectoGit.git
cd MiProyectoGit
git checkout main
```

Pasos

1. **A edita la misma línea que B editará**

- A abre index.html y cambia el <h1> por:

```
<h1>Hola desde A</h1>
```

- A confirma y sube:

```
git add index.html
git commit -m "A: cambia título a 'Hola desde A'"
git push origin main
```

2. **B edita en su copia local (aún desactualizada)**

- B abre index.html y cambia el mismo <h1> por:

```
<h1>Hola desde B</h1>
```

- B intenta subir:

```
git add index.html
git commit -m "B: cambia título a 'Hola desde B'"
```

git push origin main

Resultado esperado: error rejected (hay cambios en remoto).

3. **B trae cambios del remoto y aparece el conflicto**

git pull origin main

Git intentará fusionar y **marcará conflicto** en index.html. El archivo mostrará:

<<<<<<< HEAD

<h1>Hola desde B</h1>

=======

<h1>Hola desde A</h1>

>>>>>>> origin/main

4. **B resuelve el conflicto**

- Edita index.html y deja **una sola versión final** (acordada). Ejemplo:

<h1>Hola equipo: A y B</h1>

- Guarda y confirma la resolución:

git add index.html

git commit -m "Resuelve conflicto de título combinando aportes A y B"

git push origin main

5. **A actualiza su copia**

git pull origin main

Ya ambos están sincronizados con la versión resuelta.

Consejos y buenas prácticas

- Antes de empezar el día: git pull origin main.
- Conflictos en varios archivos: repite el proceso en cada archivo marcado.
- Si te lías, puedes **abortar la fusión** y empezar de nuevo:

git merge --abort

Ejercicio 4 — Publicar en GitHub Pages (opcional) o compartir repo

Objetivo

Poner el proyecto accesible desde una URL o validar que está online.

Opción A: Publicar con GitHub Pages (sitios estáticos)

1. **Asegúrate de tener index.html en la raíz**
   (ya lo tienes del ejercicio 1).
2. **Sube los cambios (si hiciste alguno)**

git add .

git commit -m "Prepara contenido para GitHub Pages"

git push origin main

3. **Activa GitHub Pages**

- En GitHub → tu repo → **Settings** → **Pages**.
- **Source**: selecciona **Deploy from a branch**.
- **Branch**: elige **main** y carpeta **/** (root).
- Guarda.

4. **Abre la URL pública**

- GitHub mostrará una URL tipo:
  https://tuusuario.github.io/MiProyectoGit/
  Puede tardar 1–2 minutos en estar disponible.

Opción B: Compartir el repositorio con el profesor

- Copia la URL del repo: https://github.com/tuusuario/MiProyectoGit
- Asegúrate de que sea **público** o añade al profesor como **collaborator** (Settings → Collaborators).

Checklist de verificación rápida (para cada estudiante)

- Tengo **3 commits** con mensajes claros.
- git log --oneline muestra mi historial correctamente.
- El repo está conectado a GitHub (git remote -v).
- He hecho **push** y los archivos están en GitHub.
- Sé hacer git pull y resolver un conflicto básico.
- (Opcional) Mi sitio carga en GitHub Pages o he compartido la URL del repo.