Lista de comandos de git 💻

Comando

Descripción

git init

Inicializa un nuevo repositorio Git en un directorio.

git clone <url>

Clona un repositorio remoto en tu máquina local.

git status

Muestra el estado de los archivos en el repositorio (modificados, en staging, sin seguimiento, etc.).

git add <archivo>

Agrega un archivo específico al área de preparación (staging).

git add .

Agrega todos los archivos modificados al área de preparación.

git commit -m "Mensaje"

Guarda los cambios en el historial del repositorio con un mensaje descriptivo.

git push origin <rama>

Sube los commits locales a un repositorio remoto en la rama especificada.

git pull origin <rama>

Descarga los cambios desde el repositorio remoto y los fusiona en la rama actual.

git fetch

Descarga los cambios del repositorio remoto pero no los fusiona automáticamente.

git merge <rama>

Fusiona otra rama con la actual.

git branch

Lista todas las ramas en el repositorio.

git branch <nombre>

Crea una nueva rama con el nombre especificado.

git checkout <rama>

Cambia a otra rama existente.

git checkout -b <rama>

Crea y cambia a una nueva rama.

git reset <archivo>

Deshace la adición de un archivo al área de preparación.

git reset --hard <commit>

Revierte el estado del repositorio a un commit específico y descarta todos los cambios posteriores.

git log

Muestra el historial de commits del repositorio.

git reflog

Muestra el historial de comandos ejecutados en el repositorio.

git rebase <rama>

Reaplica commits de una rama sobre otra de manera más limpia que merge.

git cherry-pick <commit>

Aplica un commit específico en otra rama sin fusionar todo el historial.

🚀 Guía de Uso de Git para Proyectos Web

Caso 1: Un Programador Trabajando Solo

Crear el Repositorio en local git init Esto inicializa un nuevo repositorio en la carpeta del proyecto.

Configurar el Repositorio Remoto (Opcional): Si el repositorio está en GitHub, GitLab o Bitbucket, agrégalo como remoto: git remote add origin <URL\_DEL\_REPO>

Realizar el Primer Commit en la Rama main

git add .

git commit -m "Primer commit: estructura base"

Subir los Cambios a GitHub git push -u origin main

Crear la Rama de Desarrollo (dev o develop) git checkout -b develop

Crear Ramas por Feature desde develop o dev. Para cada nueva funcionalidad o mejora, se recomienda crear una nueva rama desde develop: git checkout -b feature/nueva-funcionalidad

Desarrollar la Funcionalidad y Realizar Commits Frecuentes

git add .

git commit -m "Implementada funcionalidad X"

Fusionar la Rama feature/nueva-funcionalidad en develop

git checkout develop

git merge feature/nueva-funcionalidad

git branch -d feature/nueva-funcionalidad Borrar la rama de feature (opcional)

Subir los Cambios a GitHub git push origin develop

Cuando se Completen varias o todas las funcionalidades, Fusionar develop en main

git checkout main

git merge develop

git push origin main

Caso 2: Proyecto colaborativo

Un integrante del equipo crea el repositorio en GitHub, luego lo inicializa en forma local y sube el repo.

git init

git remote add origin <URL\_DEL\_REPO>

git commit -m "Primer commit"

git checkout -b develop

git push origin main

git push origin develop

Cada Compañero Clona el Repositorio

git clone <URL\_DEL\_REPO>

cd <nombre-del-repositorio>

git checkout develop

Cada Desarrollador Crea una Rama para su Feature

git checkout -b feature/nueva-funcionalidad

Hacer Commits y Subirlos a GitHub

git add .

git commit -m "Implementando X"

git push origin feature/nueva-funcionalidad

Crear un Pull Request en GitHub

Ir al repositorio en GitHub.

Seleccionar la rama feature/nueva-funcionalidad y hacer un Pull Request hacia develop.

Un Compañero Revisa y Aprueba el Pull Request

Si hay errores, se piden correcciones.

Si está todo correcto, se aprueba y se hace merge en develop.

Cada Desarrollador Actualiza su Repositorio Local

git checkout develop

git pull origin develop

Cuando el Proyecto Está Listo, Fusionar develop en main

git checkout main

git merge develop

git push origin main

💡 Extras para Buenas Prácticas en Equipos

Antes de empezar una nueva funcionalidad, asegurarse de tener develop actualizado:

git checkout develop

git pull origin develop

Usar nombres descriptivos para las ramas (feature/nueva-ui, bugfix/error-login, release/error404).

No hacer commits grandes, dividir los cambios en pequeños commits con mensajes claros.