GUÍA RÁPIDA: Git para QA (Control de versiones sin dolores de cabeza)

📌 1. ¿Por qué un QA necesita saber Git?

Git no es solo para los que escriben el código de la aplicación. Como QA, dominar Git te permite:

- **Probar ramas específicas:** Los desarrolladores trabajan en "ramas" (*branches*). Git te permite moverte a la rama exacta de una nueva funcionalidad para probarla antes de que se mezcle con el código principal.
- **Automatización:** Si escribes pruebas automatizadas (Cypress, Playwright, Selenium, etc.), necesitas Git para guardar tus avances y compartirlos con el equipo.
- **Saber qué probar:** Viendo el historial de cambios (*commits*), puedes deducir qué partes del sistema tienen más riesgo de fallar.

🔑 2. Los Conceptos Clave que debes entender

Para no perderte en las reuniones, debes dominar este vocabulario:

- **Repositorio (Repo):** La carpeta del proyecto donde se guarda todo el código y su historial.
- **Rama (Branch):** Una copia independiente del código. La rama principal suele llamarse `main` o `master`. Las nuevas funciones se hacen en ramas secundarias (ej. `feature/login`).
- **Commit:** Una "foto" o guardado de los cambios que has hecho. Cada commit tiene un mensaje explicando qué se hizo.
- **Pull Request (PR) / Merge Request:** Una propuesta para meter los cambios de una rama a la rama principal. ¡Aquí es donde el QA suele dar el visto bueno!

🔍 3. Los Comandos que usarás el 90% del tiempo

🟢 Para empezar a trabajar

Para traerte el proyecto a tu computadora por primera vez.

Bash
```
git clone https://url-del-repositorio.com/proyecto.git
```

🟡 Para actualizar tu código

Antes de empezar a probar, asegúrate de tener lo último que subieron los desarrolladores.

Bash
```
git pull
```

🔵 Para moverte entre ramas (Muy usado para probar historias de usuario)

Para cambiarte a la rama que contiene la nueva funcionalidad que vas a validar.

Bash
```
git checkout nombre-de-la-rama
```

🟣 Para ver en qué estado estás

¿No sabes si modificaste archivos o en qué rama estás parado? Este comando te lo dice todo.

Bash
```
git status
```

🟠 Para guardar tus scripts de automatización

Si hiciste cambios en tus pruebas y quieres guardarlos.

Bash
```
git add .
git commit -m "feat: agregadas pruebas para el módulo de login"
git push origin mi-rama-de-pruebas
```

🏋️ 4. Escenarios Prácticos para QAs

📝 Escenario 1: El desarrollador dice "Ya está listo para probar"

- **Tu misión:** Probar la funcionalidad de recuperación de contraseña que está en la rama `feature/recovery`.
- **Tus comandos:**

Bash
```
# 1. Asegúrate de estar al día
git fetch --all

# 2. Muévete a la rama del desarrollador
git checkout feature/recovery

# 3. Trae los últimos cambios de esa rama
```

```
git pull
```

📝 Escenario 2: Rompiste algo sin querer en tu automatización

- **Tu misión:** Modificaste unos archivos de prueba, todo empezó a fallar y quieres volver a como estaba todo antes de que tocaras nada.
- **Tus comandos:**

Bash
```
# CUIDADO: Esto borra tus cambios no guardados en los archivos
git restore .
```

⚠️ 5. Reglas de oro de Git para QA

- **Nunca trabajes directamente sobre la rama `main` o `master`:** Siempre crea una rama propia para tus scripts de pruebas.
- **Escribe mensajes de commit claros:** "Arreglado" no ayuda a nadie. Usa algo como "fix: corregido el selector CSS en el botón de compra".
- **Si tienes dudas, ¡no adivines!** Un comando mal ejecutado (como un `reset` forzado) puede borrar trabajo. Si no estás seguro, consulta con un desarrollador de tu equipo.