



**Nombre:**

Oliver Adames Beato

**Matricula:**

2022-1215

**Carrera:**

Desarrollo de software

**Materia**

Prog 3

**Profesor:**

Kelyn Tejada Belliard

**Tema:**

Tarea

## Parte 1 – Cuestionario (30%)

### Portada

**Nombre del Estudiante:** Oliver Adames Beato

**Materia:** Desarrollo de Software

**Profesor:** Carlos Daniel Fernández Rodríguez

**Fecha:** [Fecha de entrega]

**Instituto:** Instituto Tecnológico de las Américas (ITLA)

---

### Índice

1. ¿Qué es Git?
2. ¿Para qué sirve el comando git init?
3. ¿Qué es una rama en Git?
4. ¿Cómo saber en cuál rama estoy trabajando?
5. ¿Quién creó Git?
6. ¿Cuáles son los comandos esenciales de Git?
7. ¿Qué es Git Flow?
8. ¿Qué es el desarrollo basado en trunk (Trunk Based Development)?

#### 1. ¿Qué es Git?

Git es una herramienta de control de versiones distribuido. En pocas palabras, es un sistema que nos ayuda a llevar un registro de todos los cambios que se hacen a un proyecto, como si fuera un diario de todos los momentos importantes que se dan en el desarrollo del software. Esto es útil cuando trabajas en equipo, porque permite que varias personas colaboren sin interferir en el trabajo de los demás. Lo bueno de Git es que no necesitas tener todo el proyecto en una sola computadora. Cada persona tiene su propia copia y puede trabajar en lo que quiera, y luego se combinan los cambios de una forma segura y eficiente.

#### 2. ¿Para qué sirve el comando git init?

Cuando ejecutas git init en tu proyecto, básicamente le estás diciendo a Git: “Voy a empezar a usar este sistema para controlar los cambios en mi proyecto.” Es el primer paso que das para que Git empiece a seguir los cambios que hagas en los archivos. Este

comando crea una carpeta oculta llamada .git en tu proyecto, donde Git guarda toda la información sobre los cambios. Así, con solo este comando ya puedes empezar a trabajar y hacer seguimiento de lo que haces.

### 3. ¿Qué es una rama en Git?

Una rama en Git es como un camino paralelo dentro de nuestro proyecto. Imagina que estás trabajando en una nueva funcionalidad o en una corrección de error, y no quieres que tus cambios afecten la versión que ya está funcionando en producción. Para eso, puedes crear una rama. De esa forma, trabajas en una copia del proyecto y, cuando terminas, puedes fusionar esa rama con la principal sin problemas. Las ramas te permiten trabajar de manera independiente y en paralelo, y luego unir todo cuando está listo. Es como si tu proyecto tuviera varias versiones a la vez, cada una haciendo cosas diferentes pero al mismo tiempo.

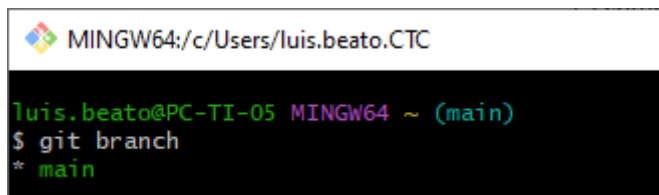
### 4. ¿Cómo saber en cuál rama estoy trabajando?

Cuando quieres saber en qué rama estás trabajando, puedes usar el comando:

`git branch`

Este comando te muestra todas las ramas que existen en el proyecto, y te señala con un asterisco (\*) en cuál de ellas estás trabajando en ese momento. También puedes usar `git status` para tener una idea más clara de qué está pasando en tu repositorio en ese instante, incluyendo la rama activa.

Ejemplo:



```
MINGW64:/c/Users/luis.beato.CTC
luis.beato@PC-TI-05 MINGW64 ~ (main)
$ git branch
* main
```

### 5. ¿Quién creó Git?

Git fue creado por **Linus Torvalds** en 2005. Linus es más conocido por ser el creador de **Linux**, el sistema operativo. La historia detrás de Git es que, cuando estaban trabajando en el desarrollo del kernel de Linux, el sistema de control de versiones que usaban no estaba funcionando bien. Así que, Linus decidió crear uno propio, que fuera más rápido, más confiable y más adecuado para el trabajo en equipo. Y así nació Git.

### 6. ¿Cuáles son los comandos esenciales de Git?

Los comandos básicos de Git que todo desarrollador debe conocer son:

- **git init**: Inicia un nuevo repositorio en tu proyecto.
- **git clone <url>**: Clona un repositorio remoto a tu computadora.
- **git status**: Muestra el estado de los archivos en tu proyecto, si están modificados, si hay archivos nuevos, etc.
- **git add <archivo>**: Prepara los archivos para ser guardados (commit) en el repositorio.
- **git commit -m "mensaje"**: Guarda los cambios con un mensaje que describe lo que hiciste.
- **git pull**: Descarga los últimos cambios del repositorio remoto y los fusiona con tu trabajo local.
- **git push**: Envía tus cambios al repositorio remoto para que los demás puedan verlos.
- **git branch**: Muestra las ramas de tu proyecto.
- **git merge <rama>**: Fusiona los cambios de otra rama en tu rama actual.

## 7. ¿Qué es Git Flow?

Git Flow es una estrategia para organizar las ramas en un proyecto usando Git. Básicamente, define reglas y patrones sobre cómo deben crearse, usarse y fusionarse las ramas. Git Flow es especialmente útil en equipos grandes. La idea es tener una rama principal, llamada master, que siempre debe estar en producción. Luego, se crean ramas develop para el desarrollo continuo, y ramas de tipo feature para trabajar en nuevas funcionalidades. También hay ramas release para preparar nuevas versiones y hotfix para corregir errores críticos. Con Git Flow, todo el trabajo está organizado y puedes gestionar las versiones y nuevas funcionalidades de manera clara y ordenada.

## 8. ¿Qué es el desarrollo basado en trunk (Trunk Based Development)?

El **Trunk Based Development (TBD)** es una forma de trabajar en proyectos donde todos los desarrolladores trabajan directamente sobre la rama principal (también llamada trunk o main). La idea es hacer pequeños cambios constantemente y fusionarlos de inmediato a la rama principal, en lugar de crear ramas separadas que puedan durar mucho tiempo. Esto permite que el equipo de desarrollo siempre esté trabajando sobre la versión más actualizada del proyecto, evitando conflictos grandes a la hora de hacer las fusiones. Es un enfoque que promueve la integración continua, donde los cambios se validan de inmediato para mantener el proyecto en buen estado.

## Bibliografía

- Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress.

- Git Documentation. (2021). *Git User Manual*. Recuperado de <https://git-scm.com/doc>
- Torvalds, L. (2005). *Git: A system for distributed version control*. Recuperado de <https://git-scm.com/>