



Mantenimiento de software

Unidad de aprendizaje 2

Actividad 4 - Utilizando sistemas de control de versiones

Hernán Ricardo Loaiza Doncel  
David Felipe Trejos Mirando

Facultad de ingeniería

Docente tutor. Rogelio Vasquez  
Ingeniería de software

Corporación Universitaria Iberoamericana  
Bogotá – Colombia  
Mayo 25 de 2025

## Tabla de Contenido

1. Introducción .....	2
2. Historia, Promoción y Objetivo de GitHub .....	2
2.1 Historia y Evolución .....	2
2.1.1 Fundación y Crecimiento Inicial (2008-2009) .....	2
2.1.2 Expansión y Consolidación (2010-2013) .....	2
2.1.3 Adquisición por Microsoft (2018) .....	2
2.1.4 Impacto Actual .....	2
2.2 Promoción de GitHub .....	3
2.3 Objetivo de GitHub .....	3
3. Procesos Realizados .....	4
4. Trabajo Colaborativo .....	6
5. Comandos Utilizados .....	6
6. Evidencias del Proceso .....	7
7. Conclusiones .....	11
8. Referencias Bibliográficas .....	11

## Introducción

El presente documento describe el proceso de implementación de un proyecto utilizando Git y GitHub como herramientas de control de versiones y trabajo colaborativo. La actividad tiene como objetivo desarrollar competencias en el manejo básico de Git, desde la creación de un repositorio hasta el trabajo en equipo mediante ramas e implementación de historias de usuario.

### Historia, Promoción y Objetivo de GitHub

#### Historia y Evolución

##### 1. Fundación y Crecimiento Inicial (2008-2009)

GitHub fue fundada en 2008 por Chris Wanstrath, PJ Hyett, Tom Preston-Werner y Scott Chacon. Desde sus inicios, destacó por su enfoque en la colaboración y facilidad de uso. En su primer año ya contaba con más de 46,000 repositorios y 100,000 usuarios.

##### 2. Expansión y Consolidación (2010-2013)

La plataforma creció rápidamente, alcanzando el millón de repositorios en 2010 y superando a competidores como SourceForge y Google Code. En 2013 ya tenía más de 3 millones de usuarios y 5 millones de repositorios.

##### 3. Adquisición por Microsoft (2018)

Microsoft adquirió GitHub por 7.500 millones de dólares. A pesar de las dudas iniciales, GitHub mantuvo su enfoque en el código abierto y se integró con servicios como Azure DevOps, fortaleciendo su ecosistema.

##### 4. Impacto Actual

Hoy, GitHub es el centro global del desarrollo colaborativo, con más de 100 millones de usuarios y millones de repositorios activos. Es clave en proyectos de software, inteligencia artificial, ciencia de datos, aplicaciones móviles y más.

#### Promoción de GitHub

GitHub creció sin campañas publicitarias masivas, gracias a estrategias centradas en la comunidad y el desarrollador:

Eventos tecnológicos: Participación en conferencias y encuentros clave del sector.

Código abierto: Su enfoque atrajo a comunidades influyentes, generando un efecto multiplicador.

Interfaz amigable: Su diseño intuitivo facilitó la adopción por parte de desarrolladores de todos los niveles.

Redes sociales y comunidad: El apoyo en plataformas como Twitter, Reddit y blogs técnicos impulsó su difusión.

Estas acciones fomentaron un crecimiento orgánico, posicionando a GitHub como referente mundial en colaboración para el desarrollo de software.

### Objetivo de GitHub

El objetivo principal de GitHub es facilitar la colaboración eficiente, segura y estructurada en el desarrollo de software. Para ello, ofrece herramientas como:

Control de versiones: Basado en Git, permite gestionar cambios, versiones y errores.

Colaboración distribuida: Funciones como pull requests, issues y code reviews permiten trabajo simultáneo en equipo.

Automatización (CI/CD): Con GitHub Actions, se automatizan pruebas, despliegues y tareas repetitivas.

Seguridad integrada: Detecta vulnerabilidades y sugiere parches automáticos.

Documentación accesible: Archivos README, wikis y páginas estáticas mejoran la comprensión de los proyectos.

## Procesos realizados

### Comprobamos si tenemos git instalado

```
C:\Users\hernan>git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
          [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
          [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
          <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  backfill   Download missing objects in a partial clone
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

Miramos que version tenemos en la cual tenemos la version 2.49.0 en la cual es la ultima version.

```
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

C:\Users\hernan> git --version
git version 2.49.0.windows.1

C:\Users\hernan>_
```

## Pasos para realizar la instalacion

Se realizó la instalación de Git en los equipos de cada uno de los integrantes del grupo desde el sitio oficial <https://git-scm.com/>. Versión instalada: git --version Ejemplo de salida:

```
git version 2.49.0
```

## Creacion de cuentas en GitHub

Cada integrante creó su cuenta en GitHub y compartió su usuario para ser añadido como colaborador.

## Creacion del repositorio

Uno de los integrantes Hernan Loaiza creó el repositorio llamado mi- proyecto-equipo Enlace al repositorio: <https://github.com/Hernansoft96/mi-proyecto-equipo>

## Subida del proyecto

Se utilizó un proyecto previamente desarrollado en otro módulo. El proyecto consiste en la cual es una app web “ficticio” es un formulario de contactos en la cual se guarda su registro

El equipo de trabajo

Un formulario funcional “aunque no envía datos, sirve como maqueta”

Galería “aunque los íconos no cargaron eso se puede ajustar a futuro ”

Tabla “en la cual se encuentra mas abajo del proyecto ”



The screenshot displays a web application with a light gray background. At the top, there is a section titled "Equipo de Trabajo" (Team Work) with the text "Juan Pérez, María Gómez, Carlos Ruiz, etc." below it. Below this is a section titled "Formulario de Contacto" (Contact Form). The form contains three input fields: "Tu nombre" (Your name), "Tu correo" (Your email), and "Escribe tu mensaje aquí..." (Write your message here...). Below the third field is an "Enviar" (Send) button. At the bottom, there is a section titled "Galería de Imágenes" (Image Gallery) with three image placeholders labeled "Imagen 1", "Imagen 2", and "Imagen 3".

## Historia de usuario

Como administrador, quiero ver una lista de usuarios registrados, para poder gestionarlos eficientemente.

Como usuario, quiero editar mi perfil, para mantener mi información actualizada.

Como visitante, quiero registrarme en la plataforma, para acceder a las funcionalidades disponibles.

Como usuario, quiero ver un historial de mis actividades, para llevar un seguimiento de mi uso.

Como administrador, quiero eliminar usuarios inactivos, para mantener la base de datos limpia.

### Trabajo colaborativo

Cada integrante tomó una historia de usuario y trabajó en su propia rama usando los siguientes comandos:

```
git checkout -b historia-usuario-x
git add .git commit -m "Implementación de historia x" git push origin historia-usuario-x
```

Los cambios fueron revisados y luego fusionados al repositorio principal mediante pull requests.

### Comandos que vamos a utilizar.

Comando	Uso
<b>git init</b>	Inicializar repositorio local
<b>git add</b>	Añadir archivos al área de staging
<b>git commit</b>	Crear un snapshot del proyecto
<b>git branch</b>	Crear o listar ramas
<b>git remote</b>	Enlazar repositorio remoto
<b>git push</b>	Subir cambios al repositorio remoto
<b>git status</b>	Ver estado actual del repositorio
<b>git log</b>	Ver historial de commits
<b>git clone</b>	Clonar un repositorio remoto
<b>git ignore</b>	Ignorar archivos específicos con .gitignore

### Evidencias de los pasos

inicializaste un repositorio Git local “git init”

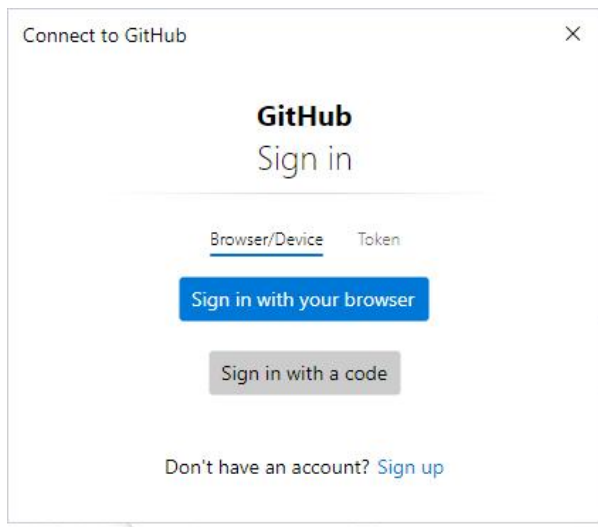
Configuraste tu usuario Git

Agregaste tu repositorio remoto de GitHub

Subiste tu primer commit con todos tus archivos “git push”

Cambiaste la rama por defecto de master a main “una buena práctica moderna”

Git necesita conectarse a tu cuenta de GitHub para poder **subir (push)** tu proyecto al repositorio remoto.



### Comandos utilizados:

Git push y Git Branch



```

MINGW64: c:/Users/hernan/OneDrive/Escritorio/mi-proyecto-equipo
herman@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (master)
$ git branch -M main

herman@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1004 bytes | 167.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Hernansoft96/mi-proyecto-equipo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

herman@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git push -u origin main
branch 'main' set up to track 'origin/main'.
Everything up-to-date

herman@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$

```

## Git status

```

herman@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

herman@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$

```

Aquí utilizamos los comandos **git add index.html** lo que nos indica es que esta listo para incluir proximo commit

el siguiente commando utilizado es **git commit -m "Modifique el index.html"**

en la cual nos indica que en ese commit añadieron 85 lineas nuevas y se eliminaron 3

```

$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

herman@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git commit -m "Modifique el index.html"
[main abce742] Modifique el index.html
 1 file changed, 85 insertions(+), 3 deletions(-)

herman@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ |

```

Este commando **git remote** quiere decir que estamos conectados con el repositorio local con el remote GitHub.

```
hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git remote -v
origin https://github.com/Hernansoft96/mi-proyecto-equipo.git (fetch)
origin https://github.com/Hernansoft96/mi-proyecto-equipo.git (push)

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$
```

Se utilizó el comando **git push -u origin main** para subir los cambios realizados en el repositorio local al repositorio remoto alojado en GitHub. Esto permitió que el historial de commits y los archivos del proyecto se sincronizaran con la nube.

```
hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.14 KiB | 233.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Hernansoft96/mi-proyecto-equipo.git
 35a856e..abce742 main -> main
branch 'main' set up to track 'origin/main'.

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ |
```

Este commando **git log --oneline** nos sirve para ver el historial de commits:

```
hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git log --oneline
abce742 (HEAD -> main, origin/main) Modifique el index.html
35a856e subimos la base del proyecto HTML y CSS del equipo

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ |
```

Para organizar el trabajo en el proyecto, se utilizó el comando **git branch** para visualizar y gestionar las ramas. Primero, se confirmó que la rama principal (main) estaba activa. Luego, se creó una nueva rama llamada nueva-rama con el comando **git branch nueva-rama**. Finalmente, se verificó que ambas ramas existían en el repositorio local, lo que permite desarrollar nuevas características sin afectar la rama principal.

```
hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git branch
* main

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git branch nueva-rama

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ git branch
* main
  nueva-rama

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$
```

Se utilizó el comando **git clone** para obtener una copia exacta del repositorio remoto desde GitHub hacia el entorno local. Esto permite a cualquier integrante del equipo acceder al proyecto y colaborar.

```
hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/mi-proyecto-equipo (main)
$ cd "C:/Users/hernan/OneDrive/Escritorio/la copia"

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/la copia
$ git clone https://github.com/Hernansoft96/mi-proyecto-equipo.git
Cloning into 'mi-proyecto-equipo'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 1), reused 7 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/Escritorio/la copia
$ |
```

Se creó un archivo **.gitignore** con el fin de evitar que archivos innecesarios, como archivos de log o carpetas de dependencias, se suban al repositorio. Esto ayuda a mantener limpio el historial del proyecto y evita conflictos.

```
MINGW64:/c:/Users/hernan/OneDrive/escritorio/mi-proyecto-equipo

hernan@DESKTOP-0E1GF90 MINGW64 ~
$ cd "C:/Users/hernan/OneDrive/escritorio/mi-proyecto-equipo"

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/escritorio/mi-proyecto-equipo (main)
$ git add .gitignore

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/escritorio/mi-proyecto-equipo (main)
$ git commit -m "Archivo .gitignore para ignorar archivos innecesarios"
[main c3a49d5] Archivo .gitignore para ignorar archivos innecesarios
1 file changed, 3 insertions(+)
create mode 100644 .gitignore

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/escritorio/mi-proyecto-equipo (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 405 bytes | 202.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Hernansoft96/mi-proyecto-equipo.git
abce742..c3a49d5  main -> main

hernan@DESKTOP-0E1GF90 MINGW64 ~/OneDrive/escritorio/mi-proyecto-equipo (main)
$ |
```

Todo queda listo y actualizado al entorno remote de GitHub.

The screenshot shows a GitHub repository interface. At the top, the repository name 'mi-proyecto-equipo' is displayed with a 'Público' (Public) label. Below this, there are navigation tabs: 'principal', '1 sucursal', and '0 etiquetas'. A search bar 'Ir al archivo' and buttons for 'Agregar archivo' and 'Código' are visible. The file list shows:
 

- `.gitignore`: Archivo .gitignore para ignorar archivos innecesarios (14 minutes ago)
- `README.md`: Actualizar README.md (3 minutes ago)
- `indice.html`: Modifique el index.html (1 hour ago)
- `estilo.css`: subimos la base del proyecto HTML y CSS del equipo (yesterday)

 The 'README.md' file is selected and its content is displayed below, featuring the heading 'Proyecto de GitHub - Equipo'. On the right side, there are statistics and links for the repository.

## Conclusiones

El uso de Git y GitHub facilita el trabajo colaborativo y el control de versiones. Aprendimos a manejar ramas para trabajar en paralelo sin conflictos.- El uso de comandos básicos permitió una correcta gestión del proyecto. Trabajar como equipo nos ayudó a dividir tareas, mejorar la comunicación y aprender herramientas del mundo real del desarrollo de software.

## Referencias Bibliograficas

**Lardinois, F. (2022, octubre 26). Four years after being acquired by Microsoft, GitHub keeps doing its thing. TechCrunch. <https://techcrunch.com/2022/10/26/four-years-after-being-acquired-by-microsoft-github-keeps-doing-its-thing/>**

**Microsoft. (2018, junio 4). Microsoft + GitHub = Empowering Developers. The Official Microsoft Blog. <https://blogs.microsoft.com/blog/2018/06/04/microsoft-github-empowering-developers/>**

**Microsoft Learn. (s.f.). Aprendizaje de GitHub. Microsoft Learn. <https://learn.microsoft.com/es-es/training/github/>**

## Enlace del repositorio

<https://github.com/Hernansoft96/mi-proyecto-equipo>