

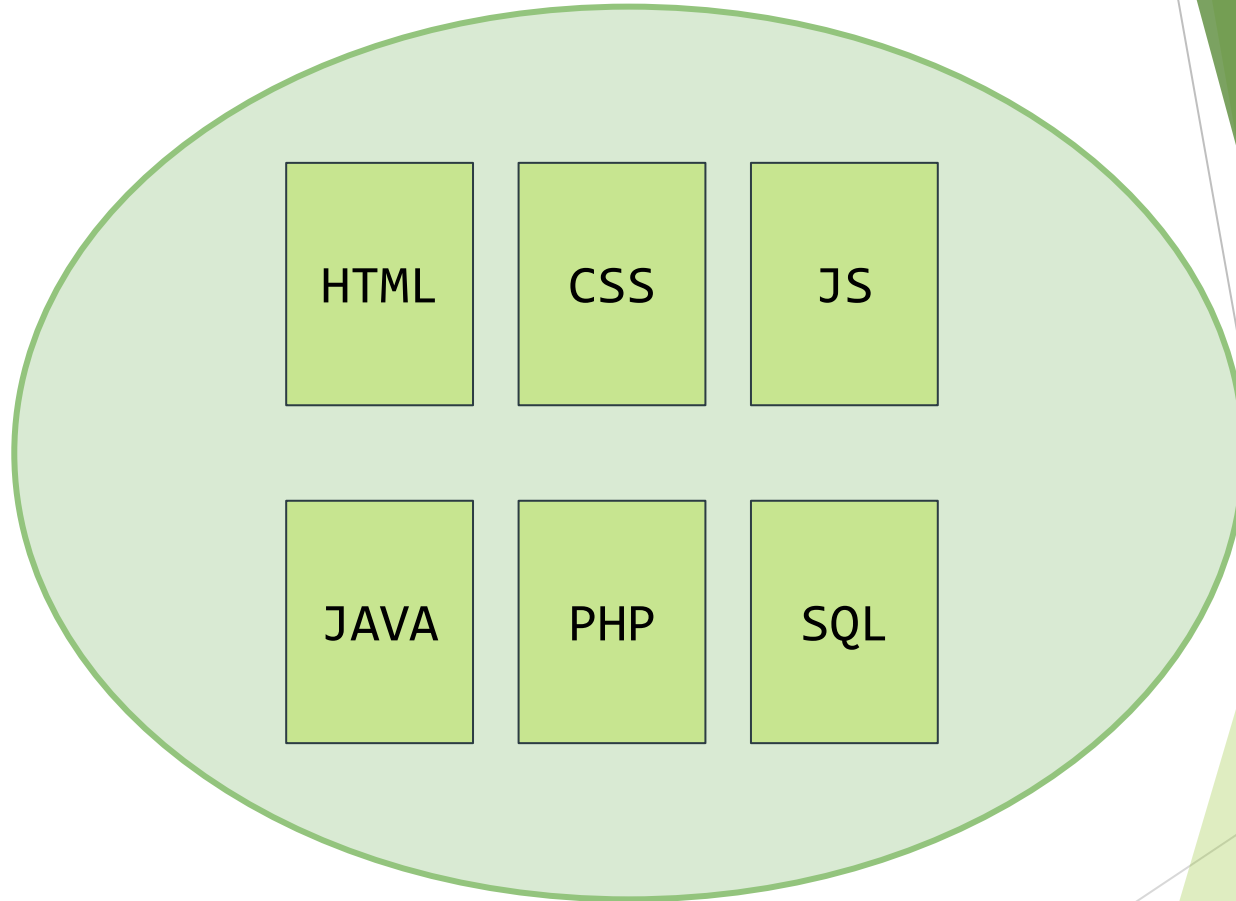
Versionado de Software

Tecnología de la Información en las Organizaciones
2023

Objetivos

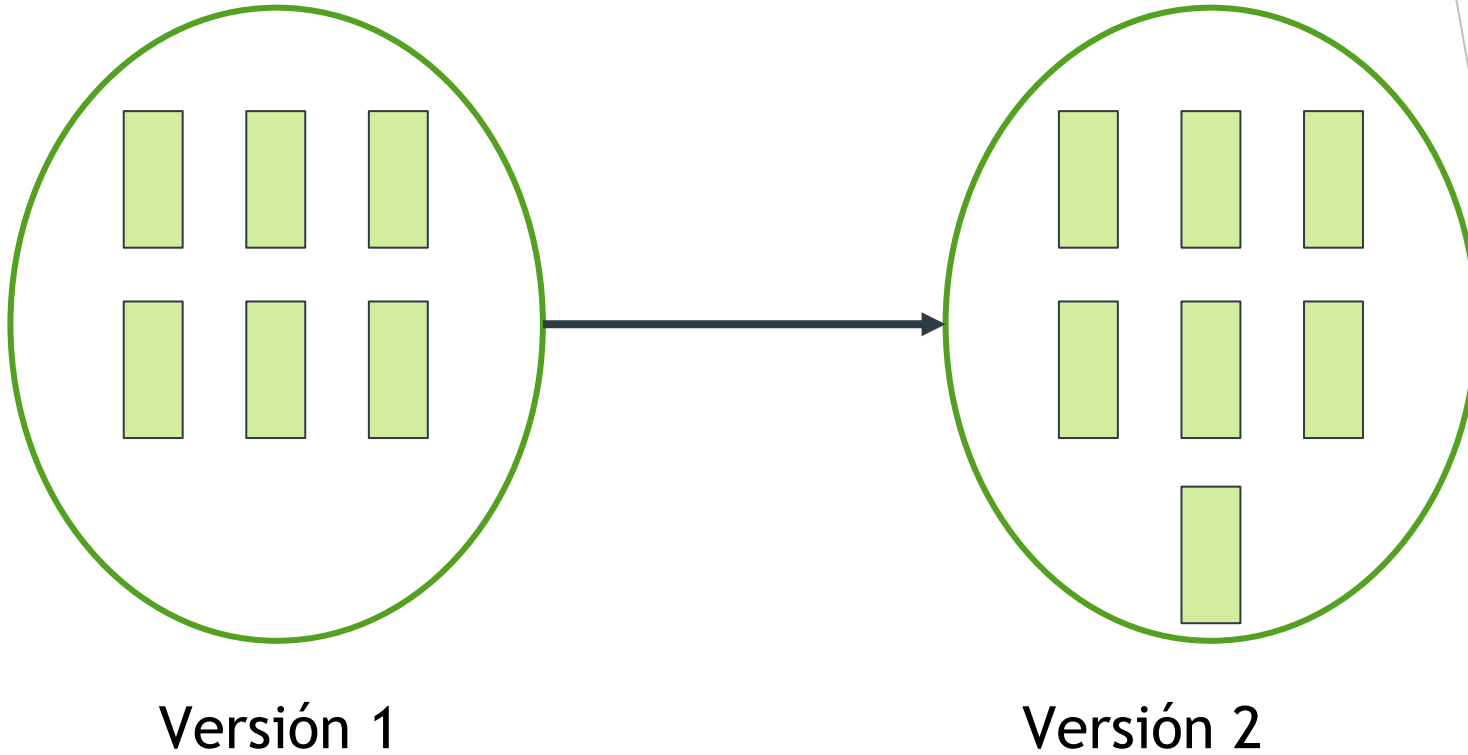
- ▶ Aprender a gestionar un repositorio local con Git
 - ▶ Crear repositorios
 - ▶ Agregar un seguimiento de archivo al proyecto y confirmar cambios
 - ▶ Realizar un seguimiento de la versión del proyecto

Proyecto

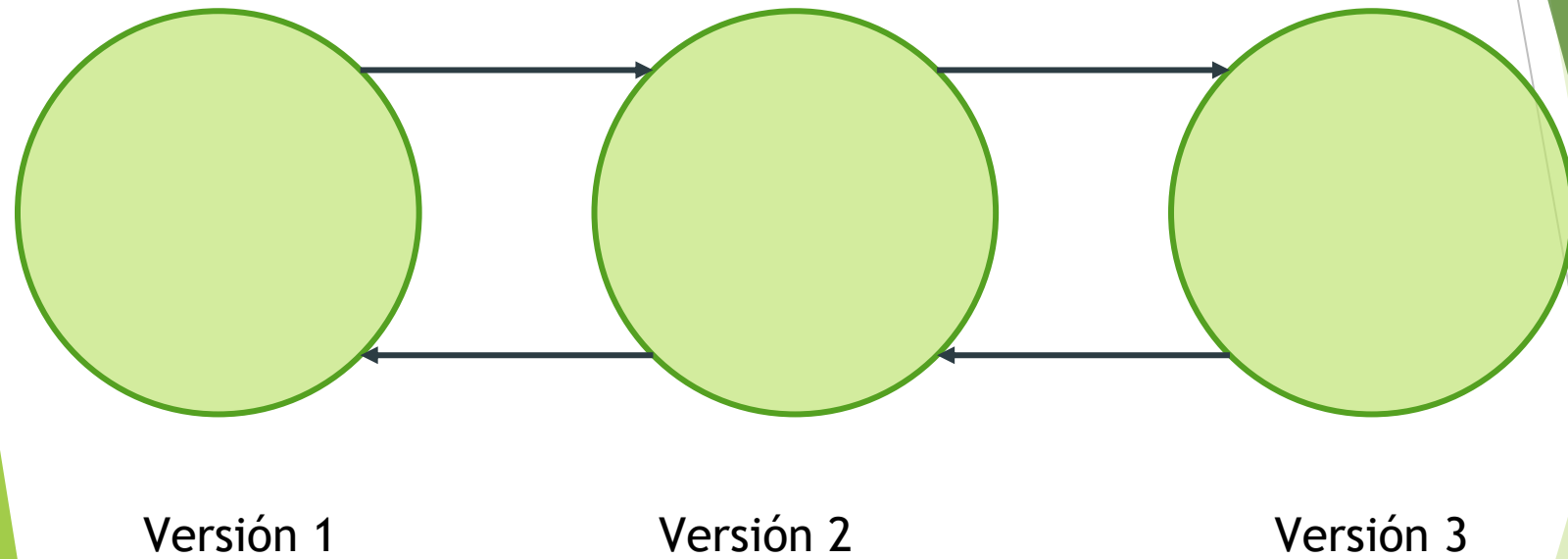


Versión 1

Proyecto



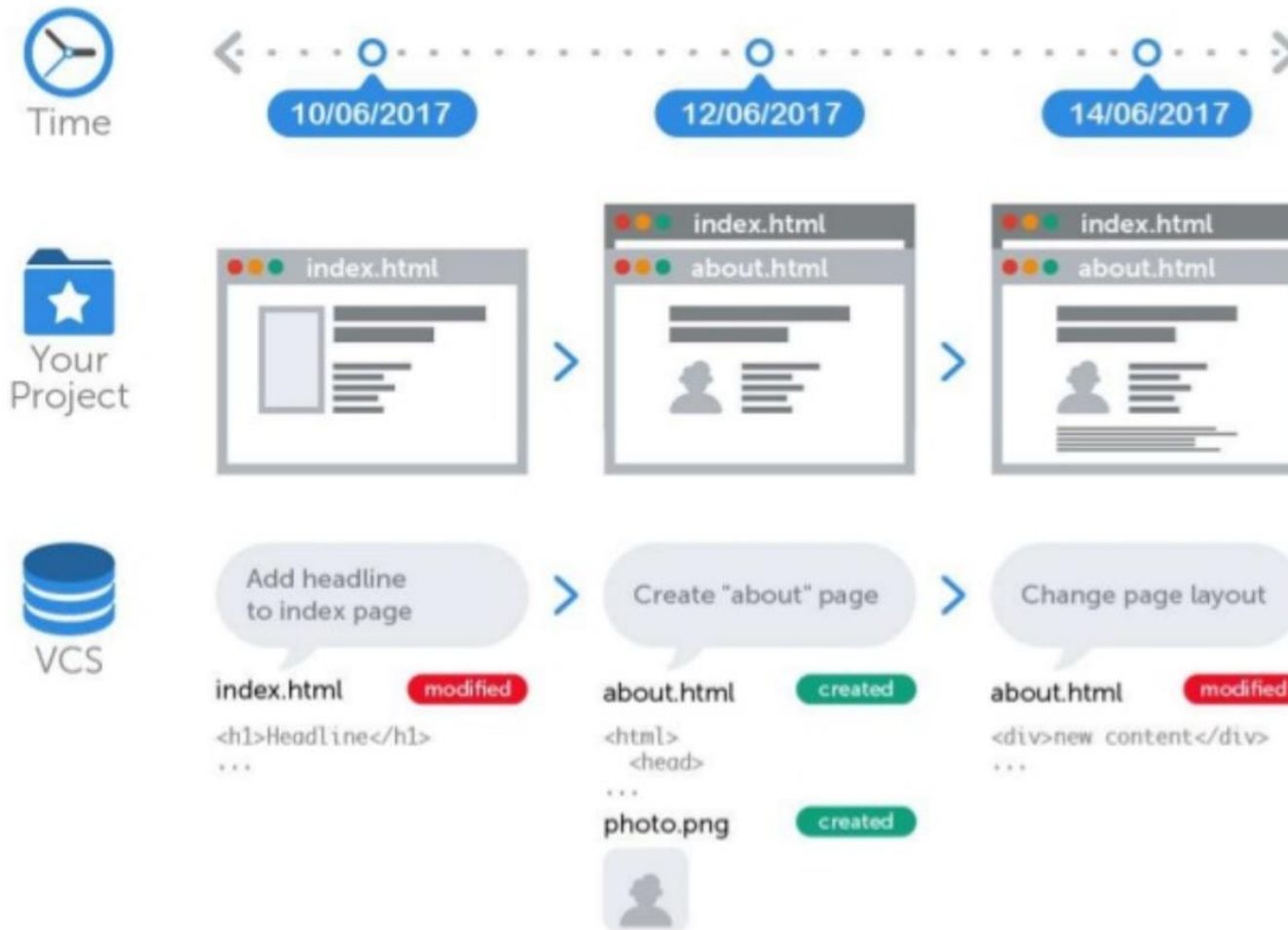
Proyecto



Sistemas de Control de Versiones

- Registrar cambios realizados en archivos
- Visualizar qué se modificó
- Identificar quién realizó qué cambios
- Registrar cuándo se realizó una modificación
- Volver a versiones anteriores

Sistemas de Control de Versiones



Sistemas de Control de Versiones

▶ Ventajas

- ▶ Poseer histórico de los proyectos
- ▶ Mayor flexibilidad ante modificaciones
- ▶ Generar estadísticas
- ▶ Facilita el trabajo en equipo
- ▶ Facilita administración del proyecto
- ▶ Mejora el compromiso del equipo

▶ Desventajas

- ▶ Curva de aprendizaje elevada

Sistemas de Control de Versiones

Ejemplos



PERFORCE



Sistemas de Control de Versiones

Tipos



PERFORCE



Pueden ser "centralizados" o "distribuidos",
veamos...

Sistemas de Control de Versiones Centralizados

- ▶ Único servidor contiene los archivos versionados
- ▶ Clientes descargan última versión y cargan archivos en dicho servidor
- ▶ Se requiere de conexión con el servidor para realizar registros nuevos
- ▶ Estándar del control de versiones durante muchos años
- ▶ Ejemplos: SVN, Perforce
- ▶ Desventajas:
 - ▶ Centralización del Servidor
 - ▶ Baja Tolerancia a Fallos
 - ▶ Recuperación Compleja

Sistemas de Control de Versiones Distribuidos

- ▶ Los clientes no solo descargan la última copia instantánea de los archivos, sino que se replica completamente el repositorio.
- ▶ Cada clon es realmente una copia completa de todo el repositorio.
- ▶ Pueden registrarse cambios aún estando sin conexión. Estos cambios cuando se recupere la conexión deberán ser enviados al servidor.
- ▶ Facilita la recuperación y aumenta la tolerancia a fallos.
- ▶ Si un servidor deja de funcionar, cualquiera de los repositorios disponibles en los clientes puede ser copiado al servidor con el fin de restaurarlo.
- ▶ Ejemplos: Git, Mercurial

Git



- ▶ Velocidad
- ▶ Diseño sencillo
- ▶ Fuerte apoyo al desarrollo no lineal (miles de ramas paralelas)
- ▶ Completamente distribuido
- ▶ Capaz de manejar grandes proyectos de manera eficiente (velocidad y tamaño de datos)



git



¿Quién usa Git?



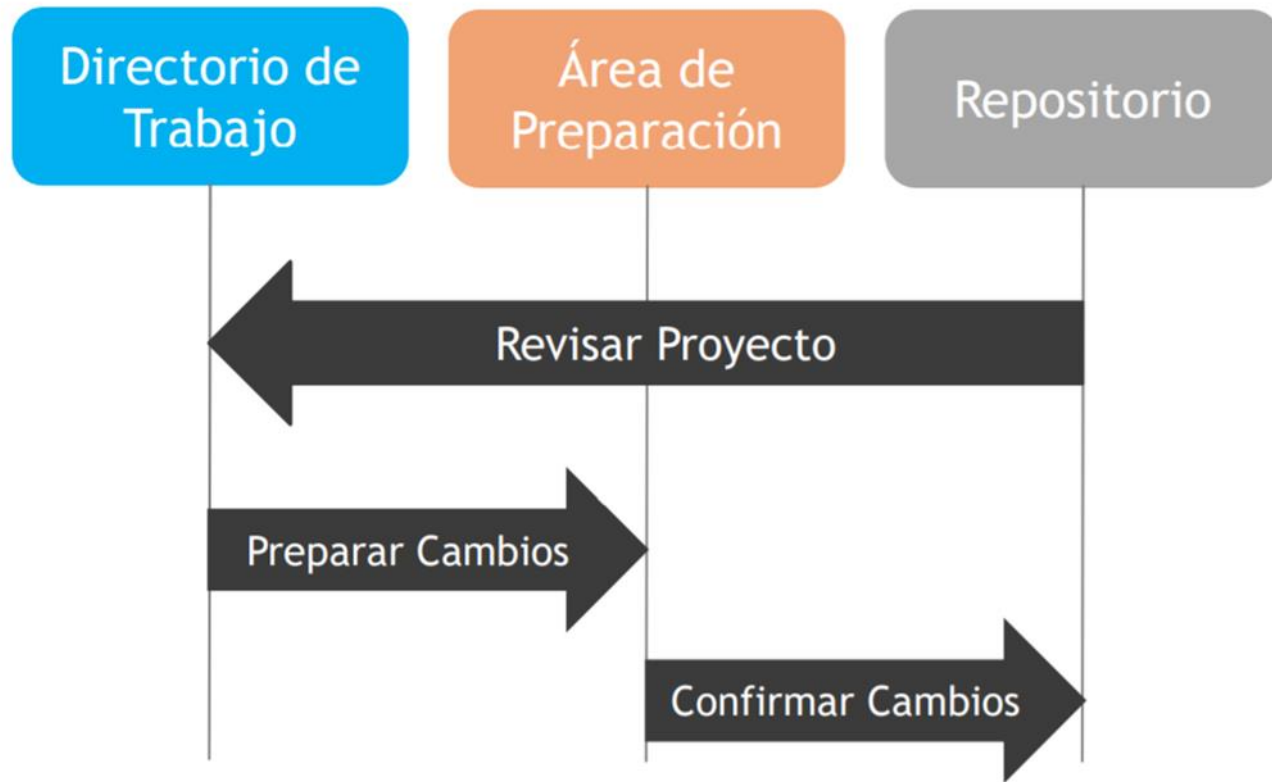
Entre otros muchos proyectos más

¿Cómo funciona Git?

- ▶ Conserva una única versión de cada archivo
- ▶ Toma una “foto” de los cambios periódicamente (commit)
- ▶ Esas “fotos” quedan registradas en un historial de cambios

Git

Flujo de trabajo



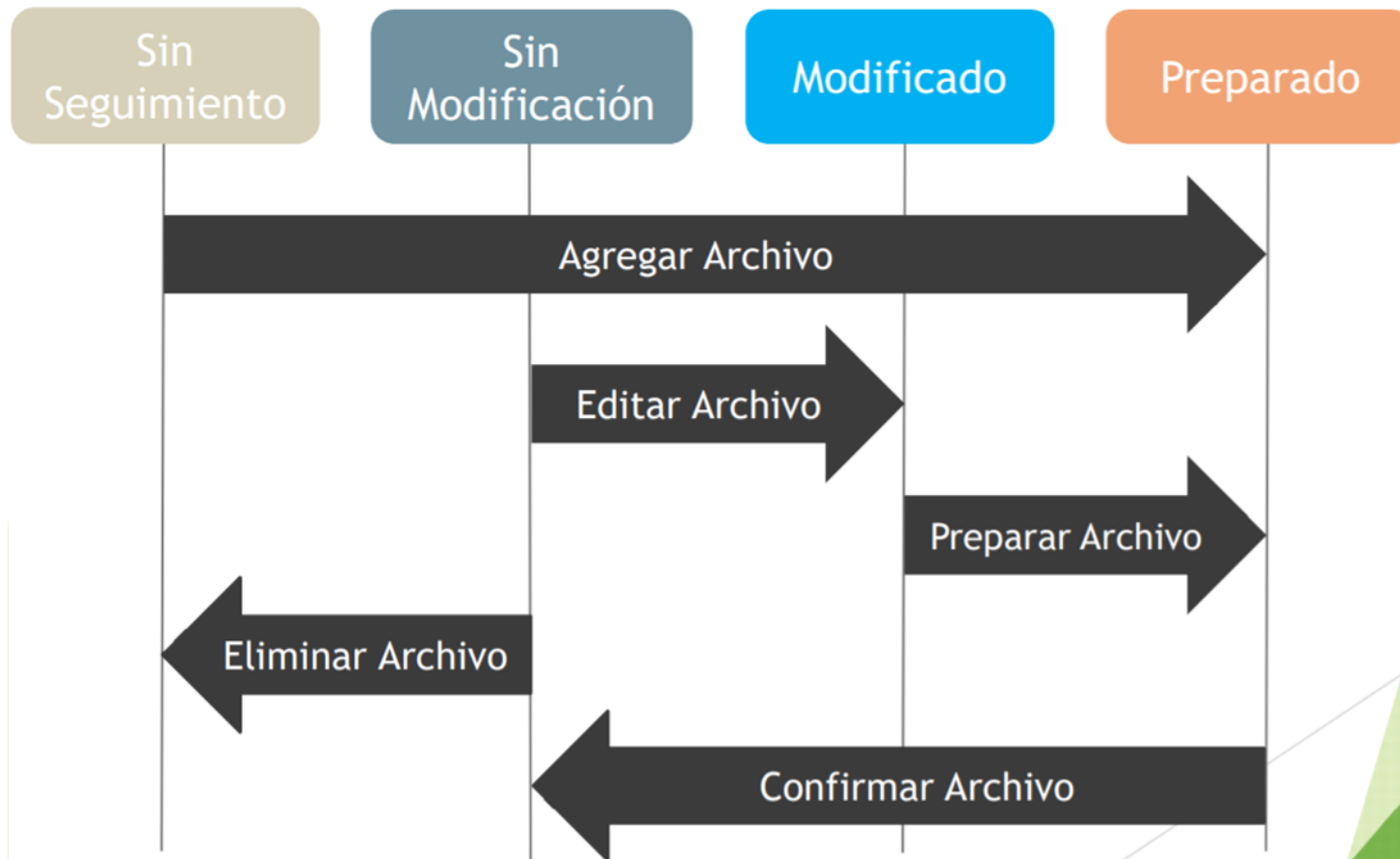
Git

Flujo de trabajo



Git

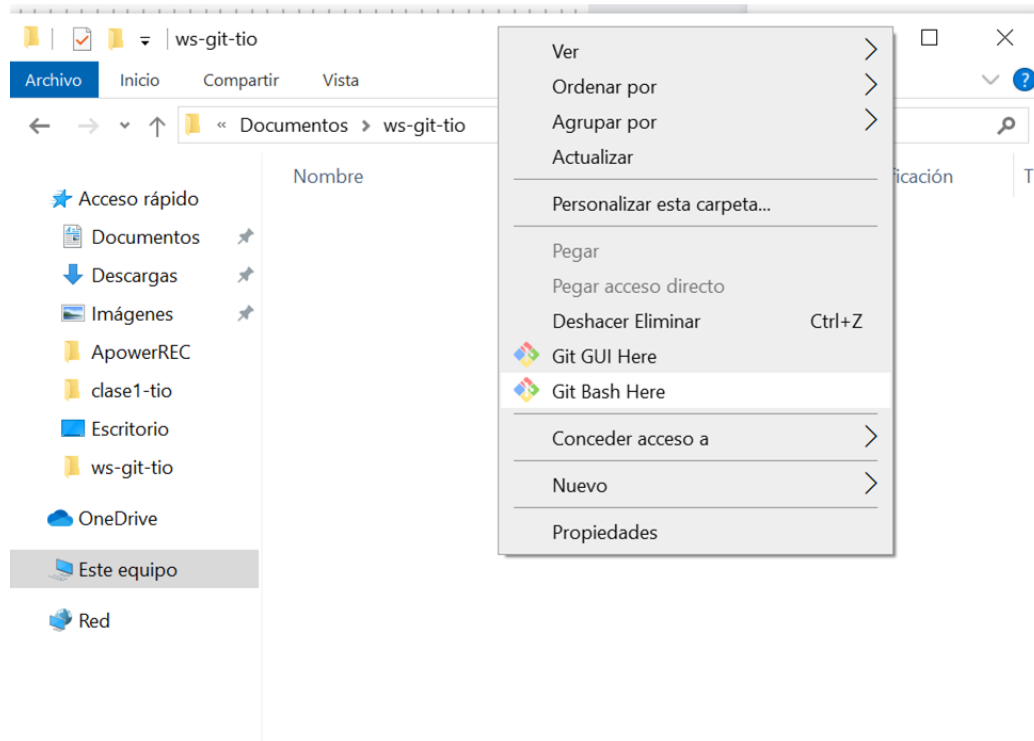
Ciclo de vida de los archivos



Comenzamos a trabajar

Para comenzar a trabajar deben crear una carpeta en su computadora para guardar los proyectos de git. En este caso, la vamos a llamar “ws-git-tio”

Dentro de la carpeta, clickeamos con el botón derecho y seleccionamos “Git Bash Here”



Git Config

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git
$ git config --global user.name "Florencecia Rodriguez"

Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio (master)
$ git config --global user.email "rodrig.florencia@gmail.com"

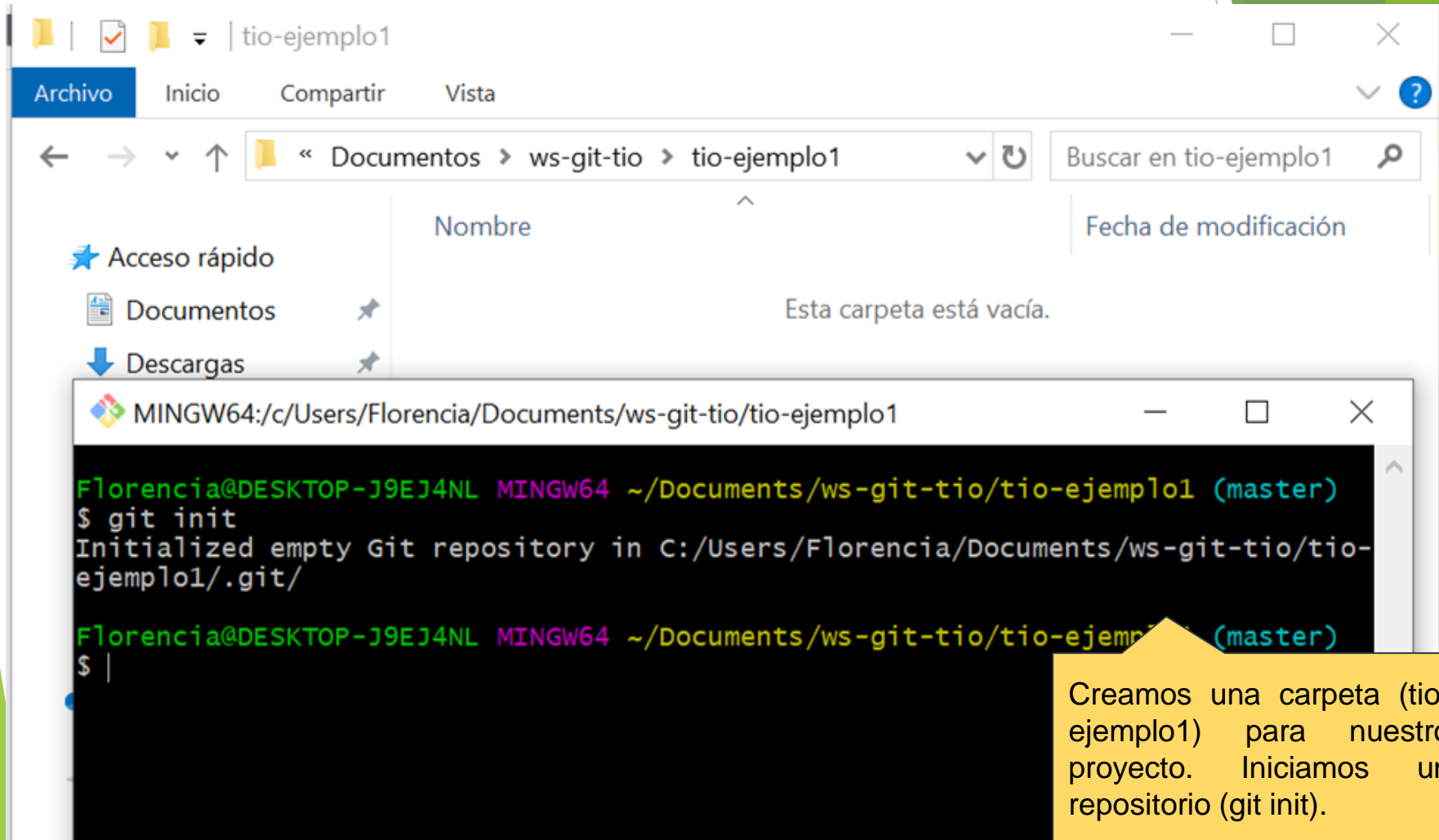
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio (master)
$ git config --global color.ui auto

Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio (master)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
credential.helper=manager
user.name=Florencia Rodriguez
user.email=rodrig.florencia@gmail.com
color.ui=auto
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
gui.wmstate=normal
gui.geometry=1322x693+114+114 254 315

Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio (master)
$
```

Configuramos nombre de usuario y email y activamos el coloreado de la salida. Luego mostramos la configuración.

Git init



Git status

```
MINGW64:/c:/Users/Florencia/Documents/ws-git-tio/tio-ejemplo1

Florecia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git init
Initialized empty Git repository in C:/Users/Florencia/Documents/ws-git-tio/tio-ejemplo1/.git/

Florecia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

Florecia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ |
```

Comprobamos el estado del nuevo repositorio.

Git Add

Git Commit

MINGW64:/c/Users/Florencia/Documents/ws-git-tio/tio-ejemplo1

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git add 'test.txt'

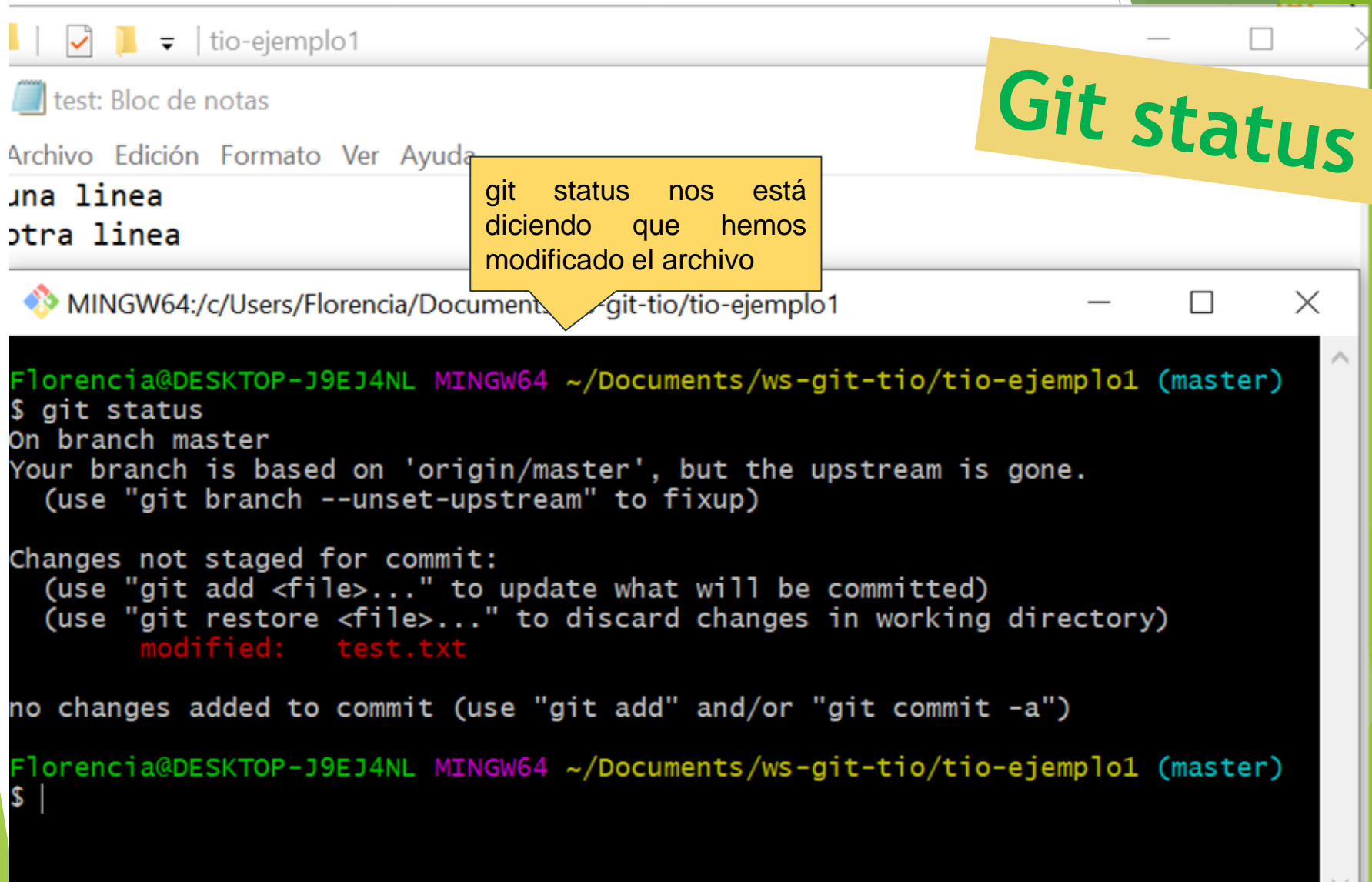
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git commit -m "agrego un archivo"
[master (root-commit) de174c8] agrego un archivo
1 file changed, 1 insertion(+)
create mode 100644 test.txt

Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ |
```

Con git add **agregamos** el archivo al **área de preparación**

Con git commit **confirmamos** el archivo que está en el área de preparación

Modifico el archivo 'texto A'



The image shows a Windows file explorer window titled 'tio-ejemplo1' with a file named 'test: Bloc de notas'. Below it, a terminal window titled 'MINGW64:/c/Users/Florencia/Documents/git-tio/tio-ejemplo1' displays the output of the 'git status' command. A yellow callout bubble points to the terminal output, stating: 'git status nos está diciendo que hemos modificado el archivo'. A yellow banner with the text 'Git status' is also present in the upper right area of the terminal window.

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

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:   test.txt

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

Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ |
```

Git status

MINGW64:/c/Users/Florencia/Documents/ws-git-tio/tio-ejemplo1

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git add 'test.txt'
```

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git status
```

```
On branch master
```

```
Your branch is based on 'origin/master', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)
```

```
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)
    modified:   test.txt
```

ahora nos está diciendo que hemos agregado el archivo, pero no lo confirmamos (falta commit)

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ |
```

Git status

MINGW64:/c/Users/Florencia/Documents/ws-git-tio/tio-ejemplo1

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git commit -m "modifique el archivo"
[master c6b2cf7] modifique el archivo
1 file changed, 2 insertions(+), 1 deletion(-)
```

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)
```

```
nothing to commit, working tree clean
```

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ |
```

Git Log

MINGW64:/c/Users/Florencia/Documents/ws-git-tio/tio-ejemplo1

```
Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ git log
commit c6b2cf7c83d31c364bf25940e75c82d5e2c7d62e (HEAD -> master)
Author: Florencia Rodriguez <rodrig.florencia@gmail.com>
Date:   Mon Apr 20 15:26:44 2020 -0300

    modifique el archivo

commit de174c809ea7da2ced90af5f274dd27df8dbee59
Author: Florencia Rodriguez <rodrig.florencia@gmail.com>
Date:   Mon Apr 20 15:18:09 2020 -0300

    agrego un archivo

Florencia@DESKTOP-J9EJ4NL MINGW64 ~/Documents/ws-git-tio/tio-ejemplo1 (master)
$ |
```

Con git log vemos el historial de confirmaciones en el repositorio

Órdenes Básicas

`git init` Iniciar un repositorio vacío en una carpeta específica

`git status` Revisamos el estado de los archivos

`git add 'nombre_de_archivo'` Añadir un archivo específico al área de preparación

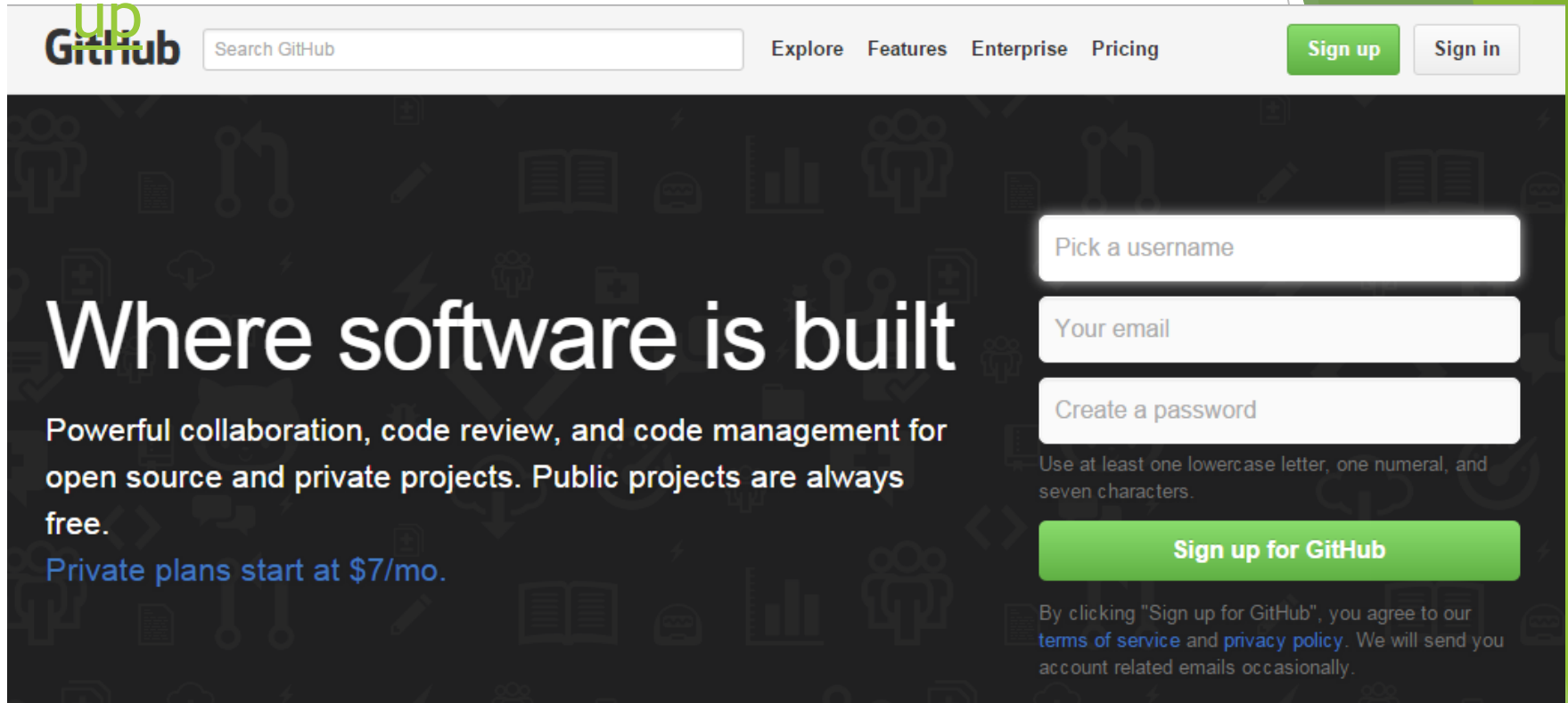
`git add .` Añadir todos los archivos del directorio

`git commit -m "mensaje"` Confirmar los cambios realizados

`git log` Muestra el historial de confirmaciones

`git diff` Muestra los cambios con respecto a la última versión guardada en el repositorio

<https://github.com/sign>

A screenshot of the GitHub sign-up page. The page has a dark background with a pattern of small icons. At the top, there is a navigation bar with the GitHub logo, a search bar, and links for Explore, Features, Enterprise, and Pricing. On the right side of the navigation bar are 'Sign up' and 'Sign in' buttons. The main content area features the heading 'Where software is built' in large white text. Below this heading is a paragraph: 'Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free.' followed by 'Private plans start at \$7/mo.' in blue text. To the right of this text is a sign-up form with three input fields: 'Pick a username', 'Your email', and 'Create a password'. Below the 'Create a password' field is a note: 'Use at least one lowercase letter, one numeral, and seven characters.' At the bottom of the form is a green button labeled 'Sign up for GitHub'. Below the button is a disclaimer: 'By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We will send you account related emails occasionally.'

Creamos una cuenta en Github y nos logueamos

Bibliografía

<https://git-scm.com/book/es/v1>

Leer:

1.3 Empezando - Fundamentos de Git
1.5 Empezando
1.6 Empezando - Obteniendo ayuda
2.1 - 2.4 Fundamentos de Git

Para practicar:

https://learngitbranching.js.org/?locale=es_AR