

¿Qué es git?

Git es un sistema de control de versiones distribuido, diseñado para manejar desde proyectos pequeños hasta muy grandes con velocidad y eficiencia. Fue creado por Linus Torvalds en 2005 para el desarrollo del núcleo Linux, con otros desarrolladores del núcleo contribuyendo a su desarrollo inicial.

¿Qué es el control de versiones? El control de versiones es un sistema que registra los cambios en un archivo o conjunto de archivos a lo largo del tiempo, de manera que puedas recuperar versiones específicas más tarde. Esto es particularmente útil para los desarrolladores de software ya que permite que el equipo mantenga un historial de quién, cuándo y por qué se realizaron ciertos cambios.

Características clave de Git:

1. **Distribuido:** A diferencia de los sistemas de control de versiones centralizados, cada usuario de Git tiene una copia completa del repositorio con todo el historial de cambios. Esto permite trabajar de forma autónoma y fusionar cambios con otros repositorios.
2. **Integridad de Datos:** Git usa una estructura de datos llamada "grafo acíclico dirigido" para almacenar los proyectos, donde cada cambio es capturado en 'snapshots' (instantáneas) del estado de todos los archivos en un momento dado. Cada snapshot es referenciado por un checksum SHA-1, asegurando la integridad del código fuente a lo largo del tiempo.
3. **Ramificaciones (Branching) y Fusiones (Merging):** Git maneja las ramificaciones de manera eficiente y permite la creación de múltiples 'branches' que son líneas independientes de desarrollo dentro de un proyecto. Los desarrolladores pueden trabajar en 'branches' sin afectar a otras, y luego fusionar los cambios cuando estén listos.
4. **Velocidad:** Git es conocido por ser extremadamente rápido. Dado que la mayoría de las operaciones son locales, no hay necesidad de red y, por lo tanto, no hay demoras. Git también minimiza la cantidad de datos necesarios para ser enviados a través de la red.
5. **Flexibilidad:** Git es flexible en varios aspectos: tipos de flujos de trabajo no lineales, plataformas y protocolos que soporta, y en su capacidad para manejar proyectos grandes y pequeños con eficiencia.
6. **Seguridad:** La naturaleza de su algoritmo SHA-1 hace que sea extremadamente difícil alterar el historial de cambios sin que sea detectado, proporcionando una seguridad robusta para el historial de revisiones.

Conceptos fundamentales en Git:

- **Repositorio:** Un directorio que está siendo monitoreado por Git. Puede ser local (en tu máquina) o remoto (como un repositorio en GitHub).
- **Commit:** Un 'commit' es como un 'checkpoint' que captura el estado de tus archivos en ese momento. Los 'commits' son inmutables y contienen un mensaje descriptivo de los cambios.
- **Branch:** Una rama es básicamente una línea independiente de desarrollo. Puedes cambiar entre ramas y trabajar en diferentes características o bugs.
- **Merge:** Unir dos o más historiales de desarrollo juntos. Git proporciona herramientas para fusionar ramas.
- **Pull:** Actualizar tu repositorio local con cambios desde un repositorio remoto.
- **Push:** Enviar tus cambios locales a un repositorio remoto para compartirlos con otros.
- **Clone:** Hacer una copia completa de un repositorio existente

Metodología

La adopción de una metodología para trabajar con repositorios puede ayudar a los equipos a manejar mejor los lanzamientos iterativos y planificados, pero también introduce cierta complejidad en la gestión de las ramas. En nuestro caso vamos a utilizar una metodología bastante similar a la de **gitFlow**, ya que esta es más adecuada para proyectos que tienen un ciclo de lanzamiento planificado y donde es importante mantener un estado estable de producción en todo momento.

GitFlow es una metodología de flujo de trabajo para Git, creada por Vincent Driessen en 2010, que define un modelo estricto de ramificación diseñado alrededor del lanzamiento del proyecto. Este enfoque establece un marco claro sobre cómo nombrar y usar las ramas y cómo se deben integrar los cambios en el repositorio principal, facilitando la colaboración entre desarrolladores y la gestión de múltiples versiones del software.

Componentes principales de GitFlow:

1. **Rama principal (master):** Es la rama que refleja el estado actual de producción, donde el código de la rama siempre está en un estado listo para ser desplegado.
2. **Rama de desarrollo (develop):** Es donde se desarrollan las nuevas características antes de estar listas para ser lanzadas. Todos los desarrollos de nuevas características se fusionan en esta rama.
3. **Ramas de características (feature branches):** Cada nueva característica se desarrolla en su propia rama, que se bifurca desde develop y se fusiona de nuevo en develop una vez que la característica está completa.
4. **Ramas de lanzamiento (release branches):** Cuando las características en develop están listas para un lanzamiento (una nueva versión del software), se bifurca una rama de lanzamiento desde develop. Las nuevas características se congelan en esta rama, permitiendo correcciones de errores y preparación del lanzamiento. Una vez que está lista, se fusiona en master y en develop.
5. **Ramas de corrección de errores (hotfix branches):** Si se descubre un error en producción, se crea una rama de corrección desde master. Una vez que el error se corrige, la rama de corrección se fusiona tanto en master como en develop (y en la rama de lanzamiento actual si hay una).
6. **Tags (tags):** Cuando se fusiona una rama de lanzamiento en master, se etiqueta con un número de versión para tener un punto de referencia claro para la producción actual.

En nuestro caso el flujo de trabajo ha sido el siguiente:

1. Hemos creado las ramas de feature desde develop para cada nueva característica.
2. Cuando la característica está completa, se fusiona de nuevo en develop.
3. Cuando se decide hacer un lanzamiento, en nuestro caso, lo hemos hecho directamente sobre la rama master. La rama release no la hemos utilizado en nuestra metodología.
4. Cuando se fusiona en master y en develop. master lleva una etiqueta de versión.
5. Si se encuentra un error en master, se crea una rama hotfix, que después de arreglar el problema, se fusiona de nuevo en master y develop. Pero en nuestro caso tampoco lo hemos hecho porque se ha creado una rama test para que los tester puedan ver los posibles problemas.

Proceso realizado

1.- Descargo el boilerplate en mi directorio de trabajo.

```
membr@Membrana MINGW64 ~/Desktop
$ mkdir Proyecto

membr@Membrana MINGW64 ~/Desktop
$ cd Proyecto

membr@Membrana MINGW64 ~/Desktop/Proyecto
$ npx create-html5-boilerplate new-site
npx: installed 237 in 37.169s
- Downloading html5-boilerplate version 'latest' to C:\Users\membr\Desktop\Proyecto\new-site
? Select language (Use arrow keys or type to search)
? Select language English (en)
- Downloading html5-boilerplate version 'latest' to C:\Users\membr\Desktop\Proyecto\new-site
✓ html5-boilerplate@latest copied to C:\Users\membr\Desktop\Proyecto\new-site in 321ms. Have fun!
```

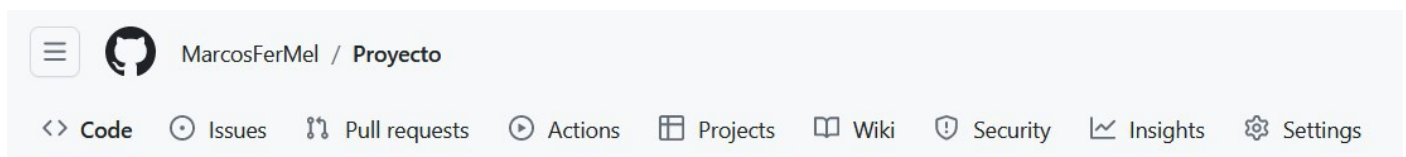
2.- Inicializamos el repositorio.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto
$ ls
new-site/

membr@Membrana MINGW64 ~/Desktop/Proyecto
$ git init
Initialized empty Git repository in C:/Users/membr/Desktop/Proyecto/.git/

membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$ ..
```

3.- Creamos el repositorio en GitHub.



4.-Añado el repositorio remoto al local.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$ git remote add origin https://github.com/MarcosFerMel/Proyecto.git

membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$ git remote -v
origin https://github.com/MarcosFerMel/Proyecto.git (fetch)
origin https://github.com/MarcosFerMel/Proyecto.git (push)

membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$ ..
```

5.- Creo un directorio para cada Usuario y así simular como actuarían cada uno desde su ordenador.

```
membr@Membrana MINGW64 ~/Desktop
$ mkdir Usuario1 Usuario2 Usuario3

membr@Membrana MINGW64 ~/Desktop
$ cd Usuario1

membr@Membrana MINGW64 ~/Desktop/Usuario1
$ git init
Initialized empty Git repository in C:/Users/membr/Desktop/Usuario1/.git/

membr@Membrana MINGW64 ~/Desktop/Usuario1 (master)
$ cd..
bash: cd..: command not found

membr@Membrana MINGW64 ~/Desktop/Usuario1 (master)
$ cd ..

membr@Membrana MINGW64 ~/Desktop
$ cd Usuario2

membr@Membrana MINGW64 ~/Desktop/Usuario2
$ git init
Initialized empty Git repository in C:/Users/membr/Desktop/Usuario2/.git/

membr@Membrana MINGW64 ~/Desktop/Usuario2 (master)
$ cd ..

membr@Membrana MINGW64 ~/Desktop
$ cd Usuario3

membr@Membrana MINGW64 ~/Desktop/Usuario3
$ git init
Initialized empty Git repository in C:/Users/membr/Desktop/Usuario3/.git/
```

6.- El Usuario1 crea la estructura inicial de la página para que los demás usuarios puedan ir agregando los cambios que se les piden.

```
<link rel="stylesheet" href="css/normalize.css">
<link rel="stylesheet" href="css/main.css">

<meta name="theme-color" content="#fafafa">
</head>

<body>

  <header>
    <h1>Curso de JavaScript Interactivo</h1>
  </header>

  <nav>
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="contenido.html">Modificar contenido HTML</a></li>
      <li><a href="estilos.html">Modificar estilos CSS</a></li>
      <li><a href="atributos.html">Modificar atributos HTML</a></li>
    </ul>
  </nav>

  <main>
    <section id="intro">
      <h1>Bienvenidos al Curso de JavaScript Interactivo</h1>
      <p>Este curso está diseñado para enseñarte cómo JavaScript puede hacer que tus páginas web sean más interactivas y dinámicas.</p>
    </section>
```

Curso de JavaScript Interactivo

[Home](#) [Modificar contenido HTML](#) [Modificar estilos CSS](#) [Modificar atributos HTML](#)

Bienvenidos al Curso de JavaScript Interactivo

Este curso está diseñado para enseñarte cómo JavaScript puede hacer que tus páginas web sean más interactivas y dinámicas.

© 2023 Curso de JavaScript Interactivo

7.- Creamos el primer Hook, que recrea la carpeta node_modules del proyecto cuando se realiza un clonado/checkout del proyecto.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ ls
applypatch-msg.sample*    pre-applypatch.sample*    pre-rebase.sample*        update.sample*
commit-msg.sample*        pre-commit.sample*        pre-receive.sample*
fsmonitor-watchman.sample* pre-merge-commit.sample*  prepare-commit-msg.sample*
post-update.sample*        pre-push.sample*          push-to-checkout.sample*

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ touch post-checkout

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ chmod +x post-checkout
post-checkout      post-update.sample

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ chmod +x post-checkout

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ nano post-checkout

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ cat post-checkout
#!/bin/sh

# Comprueba si el flag $3 está establecido en 1, lo que indica que es un checkout de una rama y no solo
if [ $3 -eq 1 ]; then

    # Navegar al directorio raíz del repositorio
    cd "$(git rev-parse --show-toplevel)" || exit

    # Opcionalmente, puedes verificar si realmente necesitas reinstalar

    if [ ! -d node_modules ]; then
        echo "Instalando dependencias..."

        npm install
    fi
fi

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ !
```


8.- Este Hook revisará los caracteres extraños en los ficheros antes de cada commit y ejecutará eslint en los archivos HTML.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ touch pre-commit

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ nano pre-commit

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ cat pre-commit
#!/bin/sh

# Ejecuta ESLint para archivos HTML
npm run lint

if [ $? -ne 0 ]; then
    echo "ESLint encontró problemas. Abortando commit."
    exit 1
fi

# Verificar la presencia de caracteres extraños
if git diff --cached --name-only | xargs grep --color='auto' -n '[áâãäåèéîïíóôõúü]'; then
    echo "Se encontraron caracteres extraños en los cambios a ser 'committed'."
    exit 1
fi

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ chmod +x pre-commit

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$
```

9.- Este Hook se ejecutará cada vez que se intente hacer un commit para asegurarse de que el mensaje del commit sigue el formato deseado.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ touch commit-msg

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ chmod +x commit-msg

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ nano commit-msg

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$ cat commit-msg
#!/bin/sh

# Captura el mensaje de commit desde el archivo proporcionado
COMMIT_MSG_FILE=$1
COMMIT_MSG=$(cat "$COMMIT_MSG_FILE")

# Define una expresión regular para tu formato de mensaje de commit
REQUIRED_PATTERN="^MOTIVO DEL COMMIT: .+\nIMPLEMENTACIÓN: .+"

if ! echo "$COMMIT_MSG" | grep -qE "$REQUIRED_PATTERN"; then
    echo "ERROR: Tu mensaje de commit no sigue el formato requerido."
    exit 1
fi

membr@Membrana MINGW64 ~/Desktop/Proyecto/.git/hooks (GIT_DIR!)
$
```

10.- El Usuario1 después de crear los nuevos Hooks los copia todos a una carpeta llamada gitHooks.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto (develop)
$ cp ~/Desktop/Proyecto/.git/hooks/* ~/Desktop/Proyecto/gitHooks/

membr@Membrana MINGW64 ~/Desktop/Proyecto (develop)
$ cd gitHooks/

membr@Membrana MINGW64 ~/Desktop/Proyecto/gitHooks (develop)
$ ls
applypatch-msg.sample*      pre-applypatch.sample*    pre-receive.sample*
commit-msg*                 pre-commit*               prepare-commit-msg.sample*
commit-msg.sample*         pre-commit.sample*        push-to-checkout.sample*
fsmonitor-watchman.sample*  pre-merge-commit.sample*  update.sample*
post-checkout*              pre-push.sample*          pre-rebase.sample*

membr@Membrana MINGW64 ~/Desktop/Proyecto/gitHooks (develop)
$ rm ~/Desktop/Proyecto/gitHooks/*.sample

membr@Membrana MINGW64 ~/Desktop/Proyecto/gitHooks (develop)
$ ls
commit-msg*  post-checkout*  pre-commit*

membr@Membrana MINGW64 ~/Desktop/Proyecto/gitHooks (develop)
$
```

11.- Creamos un README en nuestra carpeta de gitHooks donde explicamos la forma de configurarlos cuando sean descargados.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto/gitHooks (develop)
$ touch README.md

membr@Membrana MINGW64 ~/Desktop/Proyecto/gitHooks (develop)
$ nano README.md

membr@Membrana MINGW64 ~/Desktop/Proyecto/gitHooks (develop)
$ cat README.md
### Configuración de Hooks

Ejecuta el siguiente script:

```bash
#!/bin/sh
Copia o enlaza los hooks en .git/hooks
cp git_hooks/* .git/hooks/
chmod +x .git/hooks/*
```

membr@Membrana MINGW64 ~/Desktop/Proyecto/gitHooks (develop)
$
```

12.- Instalamos el plugin esLint para que cuando se ejecute el hook creado con anterioridad comprueba el formato de los HTML.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$ npm i eslint-plugin-html@6.0.1
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN Proyecto@1.0.0 No description
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\eslint-plugin-html\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ eslint-plugin-html@6.0.1
added 687 packages from 320 contributors and audited 7 packages in 35.885s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$
```


13.- Primer commit con el boilerPlate ya modificado con la estructura principal y la carpeta de Hooks.

```
membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$ git commit -m "Primer commit con el boilerPlate con la estructura inicial"
[master 5af6b1c] Primer commit con el boilerPlate con la estructura inicial
35 files changed, 11205 insertions(+)
create mode 100644 gitHooks/README.md
create mode 100644 gitHooks/commit-msg
create mode 100644 gitHooks/post-checkout
create mode 100644 gitHooks/pre-commit
create mode 100644 new-site/.editorconfig
create mode 100644 new-site/.gitattributes
create mode 100644 new-site/.gitignore
create mode 100644 new-site/.htaccess
create mode 100644 new-site/404.html
create mode 100644 new-site/LICENSE.txt
create mode 100644 new-site/browserconfig.xml
create mode 100644 new-site/css/main.css
create mode 100644 new-site/css/normalize.css
create mode 100644 new-site/doc/TOC.md
create mode 100644 new-site/doc/css.md
create mode 100644 new-site/doc/extend.md
create mode 100644 new-site/doc/faq.md
create mode 100644 new-site/doc/html.md
create mode 100644 new-site/doc/js.md
create mode 100644 new-site/doc/misc.md
create mode 100644 new-site/doc/usage.md
create mode 100644 new-site/favicon.ico
create mode 100644 new-site/humans.txt
create mode 100644 new-site/icon.png
create mode 100644 new-site/img/.gitignore
create mode 100644 new-site/index.html
create mode 100644 new-site/js/main.js
create mode 100644 new-site/js/plugins.js
create mode 100644 new-site/js/vendor/modernizr-3.11.2.min.js
create mode 100644 new-site/package-lock.json
create mode 100644 new-site/package.json
create mode 100644 new-site/robots.txt
create mode 100644 new-site/site.webmanifest
create mode 100644 new-site/tile-wide.png
create mode 100644 new-site/tile.png

membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$ git push origin master
Enumerating objects: 44, done.
Counting objects: 100% (44/44), done.
Delta compression using up to 8 threads
Compressing objects: 100% (41/41), done.
Writing objects: 100% (43/43), 124.66 KiB | 9.59 MiB/s, done.
Total 43 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MarcosFerMel/Proyecto.git
1f84a1e..5af6b1c master -> master
```

14.- El Usuario1 hace el push de la rama develop para tenerla también en el repositorio remoto y así pueda ser utilizada también por el resto de usuarios.

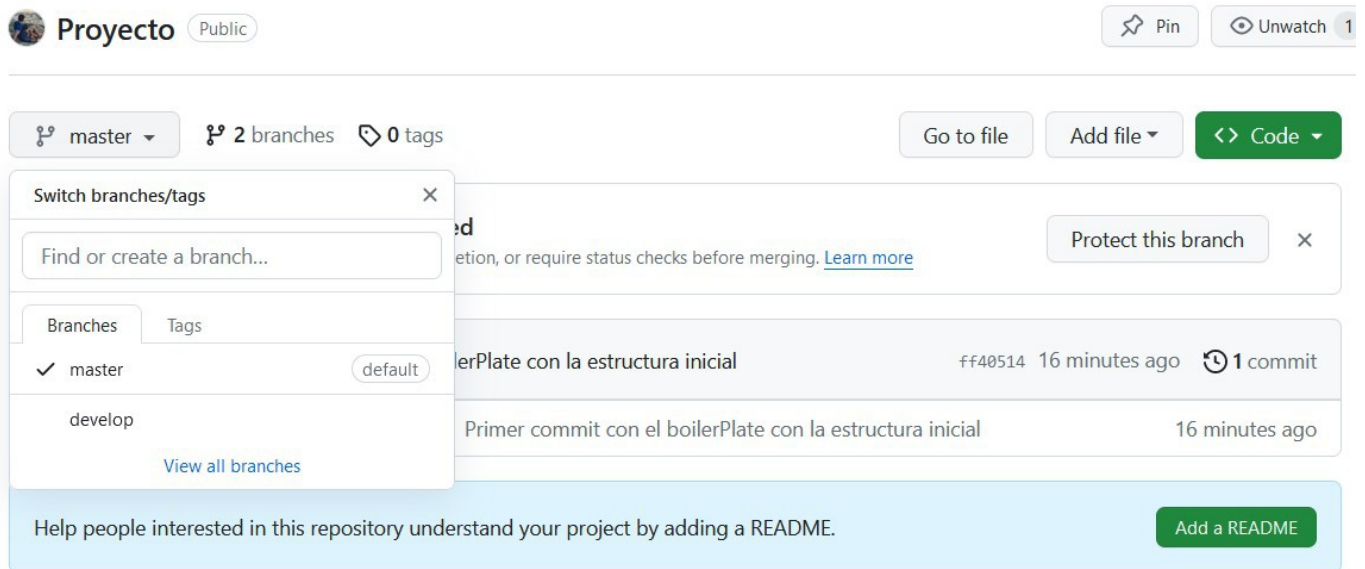
```
membr@Membrana MINGW64 ~/Desktop/Proyecto (master)
$ git checkout develop
Switched to branch 'develop'

membr@Membrana MINGW64 ~/Desktop/Proyecto (develop)
$ git add .

membr@Membrana MINGW64 ~/Desktop/Proyecto (develop)
$ git commit -m "Nueva rama develop, siguiendo la metodología GitFlow"
On branch develop
nothing to commit, working tree clean

membr@Membrana MINGW64 ~/Desktop/Proyecto (develop)
$ git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/MarcosFerMel/Proyecto/pull/new/develop
remote:
To https://github.com/MarcosFerMel/Proyecto.git
* [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

membr@Membrana MINGW64 ~/Desktop/Proyecto (develop)
$ !
```

15.- El Usuario2 clona el repositorio remoto, con la estructura creada por el Usuario1, en su repositorio local para empezar a trabajar en las tareas encomendadas.

```
membr@Membrana MINGW64 ~/Desktop/Usuario2 (master)
$ git clone https://github.com/MarcosFerMel/Proyecto.git
Cloning into 'Proyecto'...
remote: Enumerating objects: 45, done.
remote: Counting objects: 100% (45/45), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 45 (delta 0), reused 45 (delta 0), pack-reused 0
Receiving objects: 100% (45/45), 124.80 KiB | 919.00 KiB/s, done.

membr@Membrana MINGW64 ~/Desktop/Usuario2 (master)
$ ls
Proyecto/

membr@Membrana MINGW64 ~/Desktop/Usuario2 (master)
$ cd Proyecto/

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (master)
$ ls
gitHooks/  new-site/

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (master)
$
```

16.- El Usuario2 edita la estructura descargada de repositorio remoto y añade las secciones que se le encargaron.

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (master)
$ ls
gitHooks/  new-site/

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (master)
$ cd new-site/

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (master)
$ ls
404.html      contenido.html  humans.txt     js/            site.webmanifest
LICENSE.txt   css/           icon.png      package-lock.json  tile-wide.png
atributos.html  doc/          img/         package.json    tile.png
browserconfig.xml  favicon.ico  index.html   robots.txt
```

17.- El Usuario2 configura los hooks descargados en su repositorio local.

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/.git/hooks (GIT_DIR!)
$ ls
applypatch-msg.sample*      pre-applypatch.sample*      pre-receive.sample*
commit-msg.sample*          pre-commit.sample*          prepare-commit-msg.sample*
commit-msg.sample*          pre-commit.sample*          push-to-checkout.sample*
fsmonitor-watchman.sample*  pre-merge-commit.sample*    update.sample*
post-checkout*              pre-push.sample*
post-update.sample*         pre-rebase.sample*
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/.git/hooks (GIT_DIR!)
$ !
```

18.- El Usuario2 crea las dos features que se le piden y las sube al repositorio remoto.

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (master)
$ git checkout feature/contenidoHTML
Switched to branch 'feature/contenidoHTML'
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (feature/contenidoHTML)
$ ls
gitHooks/  new-site/
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (feature/contenidoHTML)
$ cd new-site/
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/contenidoHTML)
$ cd ..
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (feature/contenidoHTML)
$ git push -u origin feature/contenidoHTML
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/contenidoHTML' on GitHub by visiting:
remote:   https://github.com/MarcosFerMel/Proyecto/pull/new/feature/contenidoHTML
remote:
To https://github.com/MarcosFerMel/Proyecto.git
 * [new branch]      feature/contenidoHTML -> feature/contenidoHTML
branch 'feature/contenidoHTML' set up to track 'origin/feature/contenidoHTML'.
```

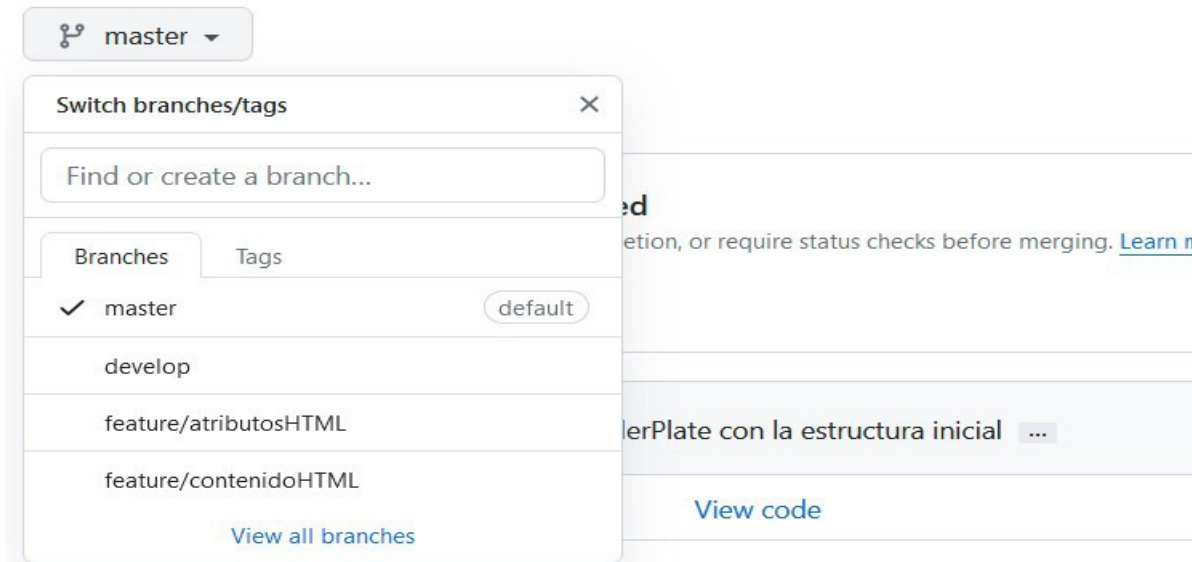
```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (feature/contenidoHTML)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (master)
$ git branch feature/atributosHTML
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (master)
$ git checkout feature/atributosHTML
Switched to branch 'feature/atributosHTML'
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (feature/atributosHTML)
$ git push -u origin feature/atributosHTML
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/atributosHTML' on GitHub by visiting:
remote:   https://github.com/MarcosFerMel/Proyecto/pull/new/feature/atributosHTML
remote:
To https://github.com/MarcosFerMel/Proyecto.git
 * [new branch]      feature/atributosHTML -> feature/atributosHTML
branch 'feature/atributosHTML' set up to track 'origin/feature/atributosHTML'.
```

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto (feature/atributosHTML)
$ !
```



19.- El Usuario2 hace el push de la sección Modificar Atributos HTML a su feature correspondiente, siguiendo en el commit la reglas establecidas por hook creado por el Usuario1.

```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/atributosHTML)
$ git add atributos.html

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/atributosHTML)
$ git commit -m "MOTIVO DEL COMMIT: Subida fichero atributos. IMPLEMENTACIÓN: sección modificar atributos HTML."
[feature/atributosHTML 2057ad0] MOTIVO DEL COMMIT: Subida fichero atributos. IMPLEMENTACIÓN: sección modificar atributos HTML.
1 file changed, 87 insertions(+)
create mode 100644 new-site/atributos.html

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/atributosHTML)
$ git push origin feature/atributosHTML
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.84 KiB | 1.84 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/MarcosFerMel/Proyecto.git
  5af6b1c..2057ad0  feature/atributosHTML -> feature/atributosHTML

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/atributosHTML)
$
```



20.- El Usuario2 hace el push de la sección Modificar Contenido HTML a su feature correspondiente, siguiendo en el commit la reglas establecidas por hook creado por el Usuario1.


```
membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/contenidoHTML)
$ git add contenido.html

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/contenidoHTML)
$ git commit -m "MOTIVO DEL COMMIT: Subida fichero contenido. IMPLEMENTACIÓN: sección modificar contenido HTML."
[feature/contenidoHTML 4f68866] MOTIVO DEL COMMIT: Subida fichero contenido. IMPLEMENTACIÓN: sección modificar contenido HTML.
1 file changed, 78 insertions(+)
create mode 100644 new-site/contenido.html


membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/contenidoHTML)
$ git push origin feature/contenidoHTML
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.75 KiB | 1.75 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/MarcosFerMel/Proyecto.git
5af6b1c..4f68866 feature/contenidoHTML -> feature/contenidoHTML

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/new-site (feature/contenidoHTML)
$ !
```

 feature/conten... ▼

 Branches
  Tags

This branch is **1 commit ahead** of master.
 [Contribute](#) ▼


MarcosFerMel MOTIVO DEL COMMIT: Subida fichero contenido. IMPLEMENT...
 2 minutes ago
🕒 3

[View code](#)

21.- El usuario3 clona el repositorio remoto, con la estructura creada por el usuario1, en su repositorio local para empezar a trabajar en las tareas encomendadas

```
membr@Membrana MINGW64 /
$ cd ~/Desktop/

membr@Membrana MINGW64 ~/Desktop
$ cd Usuario3

membr@Membrana MINGW64 ~/Desktop/Usuario3 (master)
$ ls

membr@Membrana MINGW64 ~/Desktop/Usuario3 (master)
$ git clone https://github.com/MarcosFerMel/Proyecto.git
Cloning into 'Proyecto'...
remote: Enumerating objects: 53, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 53 (delta 5), reused 50 (delta 2), pack-reused 0
Receiving objects: 100% (53/53), 126.87 KiB | 1.41 MiB/s, done.
Resolving deltas: 100% (5/5), done.

membr@Membrana MINGW64 ~/Desktop/Usuario3 (master)
$ ls
Proyecto/

membr@Membrana MINGW64 ~/Desktop/Usuario3 (master)
$ cd Proyecto/

membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto (master)
$ ls
gitHooks/ new-site/

membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto (master)
$ !
```


22.- El Usuario3 edita la estructura descargada de repositorio remoto y añade la sección de estilos que se le encargaron.

```
membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto (master)
$ ls
gitHooks/  new-site/

membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto (master)
$ cd new-site/

membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto/new-site (master)
$ ls
404.html      css/          favicon.ico   img/          package-lock.json  site.webmanifest
LICENSE.txt   doc/          humans.txt   index.html    package.json        tile-wide.png
browserconfig.xml  estilos.html  icon.png     js/           robots.txt         tile.png

membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto/new-site (master)
$
```

23.- El Usuario3 configura los hooks descargados en su repositorio local.

```
membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto/.git/hooks (GIT_DIR!)
$ ls
applypatch-msg.sample*  pre-applypatch.sample*  pre-receive.sample*
commit-msg.sample*      pre-commit*              prepare-commit-msg.sample*
commit-msg.sample*      pre-commit.sample*       push-to-checkout.sample*
fsmonitor-watchman.sample*  pre-merge-commit.sample*  update.sample*
post-checkout*           pre-push.sample*
post-update.sample*      pre-rebase.sample*

membr@Membrana MINGW64 ~/Desktop/Usuario2/Proyecto/.git/hooks (GIT_DIR!)
$
```

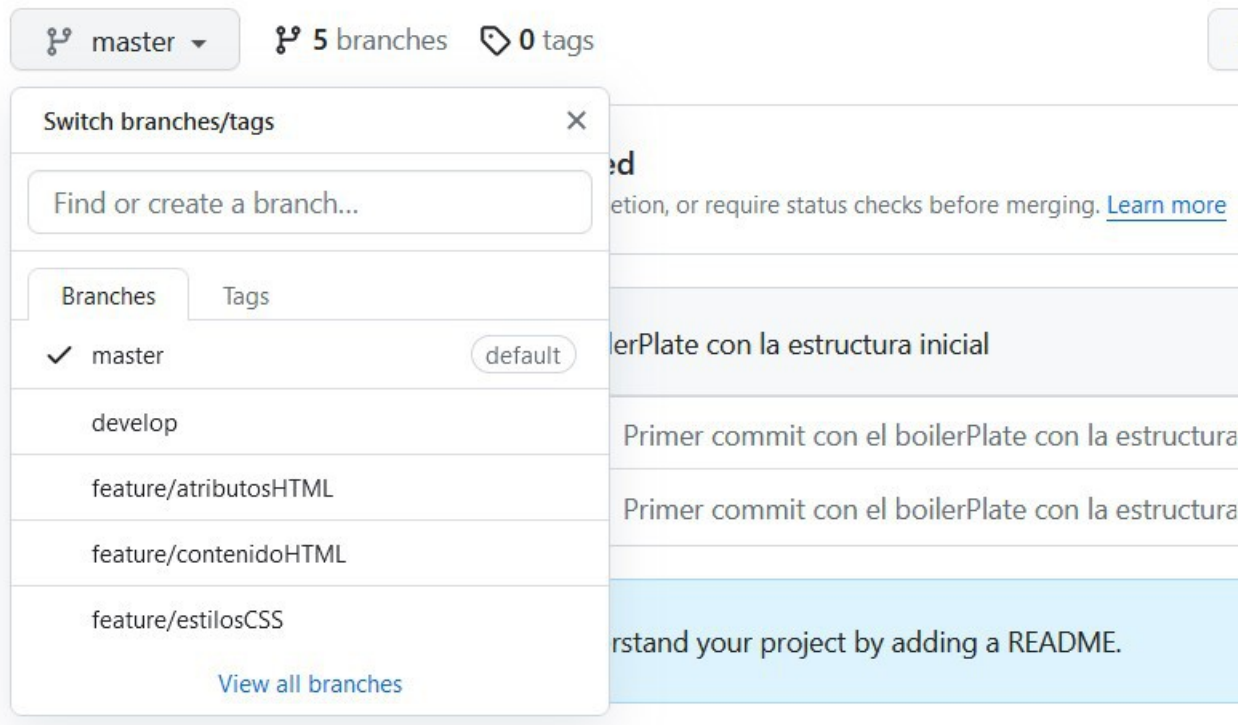
24.- El Usuario3 crea el feature de estilosCSS que se le pide y lo sube al repositorio remoto.

```
membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto/new-site (feature/estilosCSS)
$ ls
404.html      css/          favicon.ico   img/          package-lock.json  site.webmanifest
LICENSE.txt   doc/          humans.txt   index.html    package.json        tile-wide.png
browserconfig.xml  estilos.html  icon.png     js/           robots.txt         tile.png

membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto/new-site (feature/estilosCSS)
$ cd ..

membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto (feature/estilosCSS)
$ git push -u origin feature/estilosCSS
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/estilosCSS' on GitHub by visiting:
remote:   https://github.com/MarcosFerMel/Proyecto/pull/new/feature/estilosCSS
remote:
To https://github.com/MarcosFerMel/Proyecto.git
 * [new branch]      feature/estilosCSS -> feature/estilosCSS
branch 'feature/estilosCSS' set up to track 'origin/feature/estilosCSS'.

membr@Membrana MINGW64 ~/Desktop/Usuario3/Proyecto (feature/estilosCSS)
$
```



25.- El Usuario1 se encarga de fusionar todas las features en la rama Develop.

```
membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (develop)
$ git merge feature/estilosCSS
Merge made by the 'ort' strategy.
 new-site/estilos.html | 77 +++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 77 insertions(+)
 create mode 100644 new-site/estilos.html

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (develop)
$ git merge feature/atributosHTML
Merge made by the 'ort' strategy.
 new-site/atributos.html | 87 +++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 87 insertions(+)
 create mode 100644 new-site/atributos.html

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (develop)
$ ls
gitHooks/  new-site/

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (develop)
$ cd new-site/

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto/new-site (develop)
$ ls
404.html      contenido.html  favicon.ico    index.html     robots.txt
LICENSE.txt   css/           humans.txt    js/            site.webmanifest
atributos.html doc/           icon.png      package-lock.json tile-wide.png
browserconfig.xml estilos.html    img/          package.json   tile.png

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto/new-site (develop)
$ .....
```

26.- El Usuario1 sube el proyecto fusionado con las features diseñadas por los Usuarios 2 y 3 a la rama Develop.

```
membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (develop)
$ git push origin develop
Enumerating objects: 24, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 218.33 KiB | 21.83 MiB/s, done.
Total 15 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/MarcosFerMel/Proyecto.git
1f84a1e..944ef55 develop -> develop

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (develop)
$
```

27.- El Usuario1 sube todo a la rama principal y lo etiqueta con el tag v1.0

```
membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MarcosFerMel/Proyecto.git
5af6b1c..944ef55 master -> master

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (master)
$ git tag v1.0

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (master)
$ git push origin v1.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MarcosFerMel/Proyecto.git
* [new tag] v1.0 -> v1.0

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (master)
$
```

28.- El Usuario1 crea la rama test para que los testers prueben el desarrollo.

```
membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (test)
$ git push -u origin test
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'test' on GitHub by visiting:
remote: https://github.com/MarcosFerMel/Proyecto/pull/new/test
remote:
To https://github.com/MarcosFerMel/Proyecto.git
* [new branch] test -> test
branch 'test' set up to track 'origin/test'.

membr@Membrana MINGW64 ~/Desktop/Usuario1/Proyecto (test)
$
```

| Your branches | |
|-----------------------|--|
| test | Updated 33 minutes ago by MarcosFerMel |
| develop | Updated 33 minutes ago by MarcosFerMel |
| feature/estilosCSS | Updated 1 hour ago by MarcosFerMel |
| feature/contenidoHTML | Updated 20 hours ago by MarcosFerMel |
| feature/atributosHTML | Updated 20 hours ago by MarcosFerMel |