Sistema de control de versiones

Alumno: Jesús Espí Domenech

Una introducción teórica de qué es git, la metodología que vamos a utilizar para nuestros desarrollos (gitFlow, Github Flow, … o incluso una propuesta por vosotros) y por qué la utilizamos en nuestro desarrollo. Por supuesto, la metodología utilizada no puede ser un "todo a master". Al menos una rama para producción y otra para desarrollo debe existir. 2 puntos

Git es un sistema de control de versiones que ayuda en la gestion de codigo fuente a lo largo de un proyecto

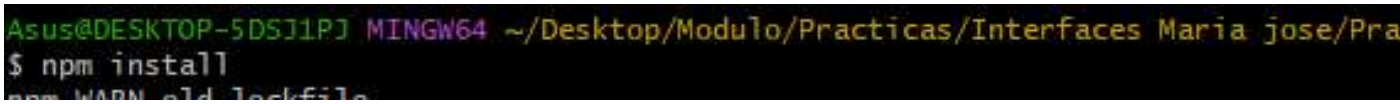
Para esta situacion utilizaremos github ya que es un proyeto pequeño y gracias a esto. Nos sera mas simple

El usuario 1 crea el repositorio con los primeros ficheros acordados (estructura inicial) y adecua este para poder utilizar la metodología elegida (creación de ramas iniciales). 1,5 puntos

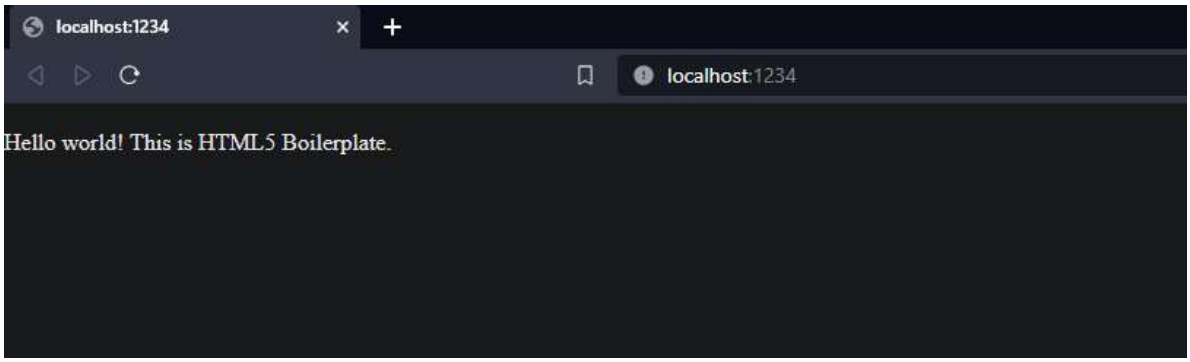
Lo primero que haremos sera clonar el repositorio y crear la rama develop

Una vez tenemos las ramas creadas pasamos a la instalacion del biolerplate

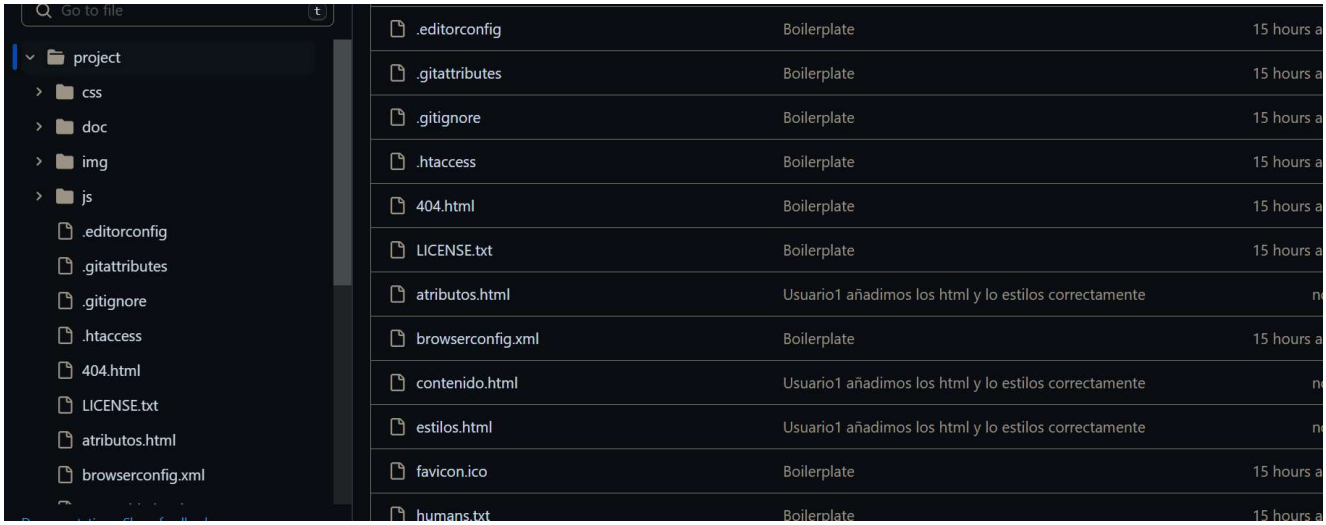
Utilizando el comando npx create-html5-boilerplate y empezando su instalacion



Una vez se instala lo ejecutamos el servidor local que en este caso de puerto tendra : 1234

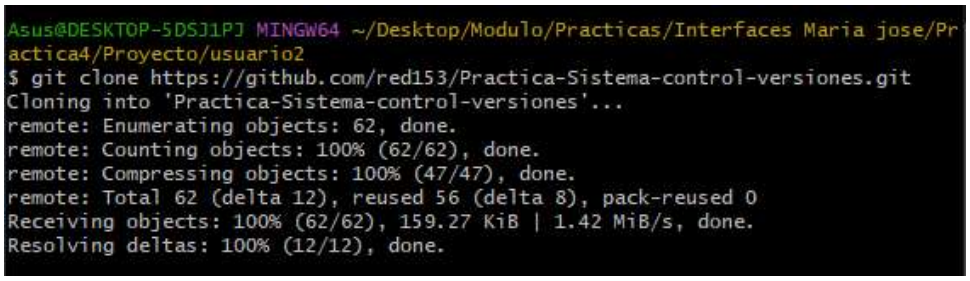


Una vez que nos aseguramos que funcione hacemos la pagina web. Hacemos los html y preparamos la tabla de estilos para que el usuario 2 y 3 puedan luego añadir su parte. Y lo subimos a github.

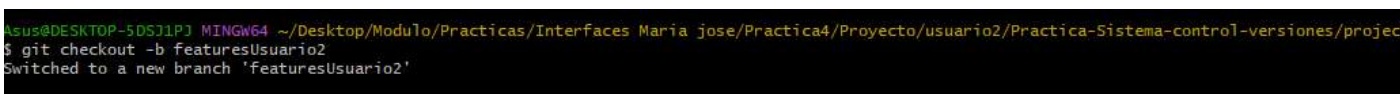


<u>El usuario 2, crea dos features (feature/contenidoHTML y feature/atributosHTML, 1 por cada sección que debe implementar).</u>

Ahora clonamos el repositorio en otro lugar para hacer el caso del usuario 2



Crearemos una rama para agregar los features (la crearemos desde develop)



Y realizaremos los cambios que se nos pide en el enunciado

Contenido:

```
<main>
  <h2>What Can JavaScript Do?</h2>

  <p id="demo">JavaScript can change HTML content.</p>

  <button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>
</main>
```

A continuación realizamos el siguiente comando

(Aquí realmente solo se esta modificando el contenido, el atibuto se cambia en el aparatado de abajo, porque me equivoque y puse los dos en la misma rama por una cofusion pero esta bien hecho en github)

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Pr
$ git push --set-upstream origin featuresUsuario2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'featuresUsuario2' on GitHub by visiting:
remote:      https://github.com/red153/Practica-Sistema-control-versiones/pull/n
ew/featuresUsuario2
remote:
To https://github.com/red153/Practica-Sistema-control-versiones.git
 * [new branch]      featuresUsuario2 -> featuresUsuario2
branch 'featuresUsuario2' set up to track 'origin/featuresUsuario2'.
```

Para crear la rama en remoto

Y realizamos un commit en el que previamente se habra hecho git add de atirbuto y contenido (esta captura la omitire ya que siempre que quiera hacer un commit se habra tenido que añadir previamente)

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Pr
actica4/Proyecto/usuario2/Practica-Sistema-control-versiones/project (featuresUs
uario2)
$ git commit -m "Usuario2 se implementan los features tanto en atributos.html co
mo en contenido.html"
[featuresUsuario2 30e163f] Usuario2 se implementan los features tanto en atribut
os.html como en contenido.html
 2 files changed, 8 insertions(+), 22 deletions(-)
```

(Aquí me di cuenta de que tenia que hacer un feature en cada rama asi que creo el otro que falta y añado unas cosas mas, ademas modifico el nombre de las ramas.)

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Pr
actica4/Proyecto/usuario2/Practica-Sistema-control-versiones/project (develop)
$ git branch -m featuresUsuario2 features/contenidoHTML
```

Creo el segundo featrure

```html
<main>
  <h1>Modificar Atributos de un Enlace HTML con JavaScript</h1>

  <p>Este es un enlace:</p>
  <a id="myLink" href="https://www.google.com" target="_blank">Ir a Google</a>

  <button type="button" onclick="cambiarEnlace()">Cambiar Enlace</button>

  <script>
      function cambiarEnlace() {
          document.getElementById('myLink').href = 'https://www.bing.com';
          document.getElementById('myLink').innerHTML = 'Ir a Bing';
      }
  </script>

</main>
```

Lo añadimos y hacemos el commit

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Practica4/Proyecto/usuario2/Practica-Sistema-control-versiones/project (feature/atributosHTML)
$ git commit -m "Usuario2 se implementa el feature de atributos.html"
```

```
$ git push -f origin feature/atributosHTML
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 674 bytes | 674.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'feature/atributosHTML' on GitHub by visiting:
remote:      https://github.com/red153/Practica-Sistema-control-versiones/pull/new/feature/atributosHTML
remote:
To https://github.com/red153/Practica-Sistema-control-versiones.git
 * [new branch]      feature/atributosHTML -> feature/atributosHTML
```

Y comprobamos que en github esta todas las ramas con sus cambios

El usuario 3, crea una feature (feature/estilosCSS) asociada al cambio para añadir su sección

En el caso del usuario3 clonaremos otra vez el boilerplate

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Practica4/Proyecto/usuario3
$ git clone https://github.com/red153/Practica-Sistema-control-versiones.git
Cloning into 'Practica-Sistema-control-versiones'...
remote: Enumerating objects: 62, done.
remote: Counting objects: 100% (62/62), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 62 (delta 12), reused 56 (delta 8), pack-reused 0
Receiving objects: 100% (62/62), 159.27 KiB | 1.50 MiB/s, done.
Resolving deltas: 100% (12/12), done.
```

Y creamos la rama

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Practica4/Proyecto/usuario3/Practica-Sistema-control-versiones (main)
$ git checkout -b feature/estilosCSS
Switched to a new branch 'feature/estilosCSS'
```

Y modificamos el estilos.html el feature

```
<main>
  <h2>What Can JavaScript Do?</h2>
  <p id="demo">JavaScript can change the style of an HTML element.</p>
  <button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!</button>
</main>
```

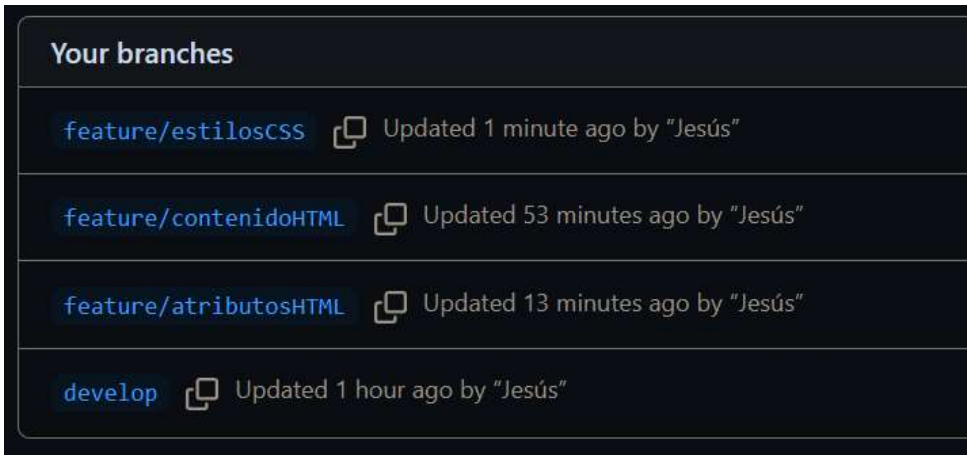
Hacemos el commit

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Practica4/Proyecto/usuario3/Practica-Sistema-control-versiones/project (feature/estilosCSS)
$ git commit -m "Usuario3 se implementa el feature estilos.html"
[feature/estilosCSS 9c6a731] Usuario3 se implementa el feature estilos.html
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Despues hacemos el push

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Practica4/Proyecto/usuario3/Practica-Sistema-control-versiones/project (feature/estilosCSS)
$ git push --set-upstream origin feature/estilosCSS
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 550 bytes | 550.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'feature/estilosCSS' on GitHub by visiting:
remote:      https://github.com/red153/Practica-Sistema-control-versiones/pull/new/feature/estilosCSS
remote:
To https://github.com/red153/Practica-Sistema-control-versiones.git
 * [new branch]      feature/estilosCSS -> feature/estilosCSS
branch 'feature/estilosCSS' set up to track 'origin/feature/estilosCSS'.
```

Y comprobamos que todo esta correcto



El usuario 1 que es el más experimentado, se encarga de realizar las pruebas. Por este motivo, en el momento que estén todos los cambios fusionados en la rama principal del proyecto, etiquetará la versión (v1.0) y creará una rama (test) que será la que supuestamente utilicen nuestros testers para probar el desarrollo.

Ahora vamos a la rama develop y juntaremos todo el proyecto

Haciendo un merge

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/
actica4/Proyecto/Practica-Sistema-control-versiones/project (develop|MERGING)
$ git merge feature/contenidoHTML
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.

Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/
actica4/Proyecto/Practica-Sistema-control-versiones/project (develop|MERGING)
$ git merge feature/atributosHTML
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.

Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/
actica4/Proyecto/Practica-Sistema-control-versiones/project (develop|MERGING)
$ git merge feature/estilosCSS
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
```

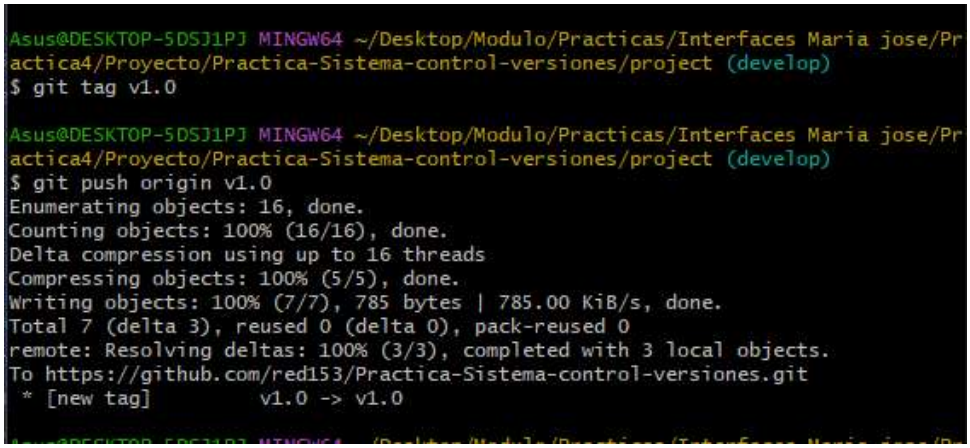
Cuando hice los merge tenian conflictos y los fui resolviendo

Una vez hecho el merge comprobamos que todos los features esten en la rama develop

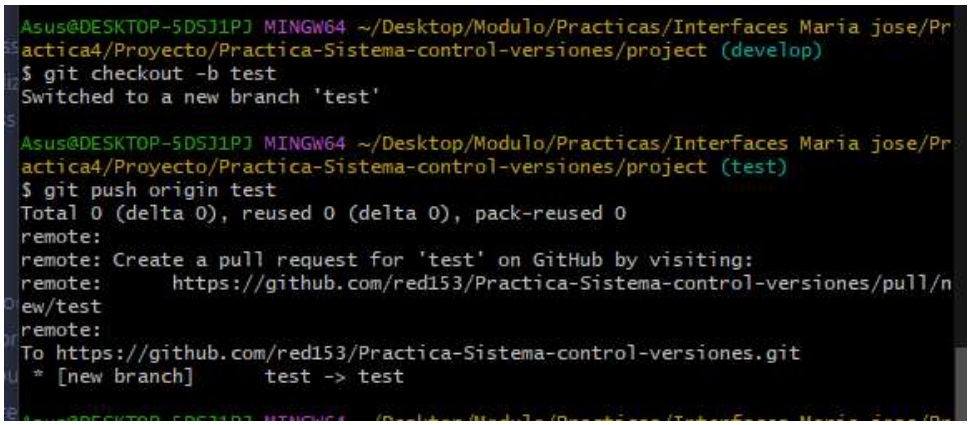
Hacemos un commit

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Pr
actica4/Proyecto/Practica-Sistema-control-versiones/project (develop|MERGING)
$ git commit -m "Usuario1 Se fusionan todas las ramas en la rama develop"
[develop c8a4b48] Usuario1 Se fusionan todas las ramas en la rama develop
```

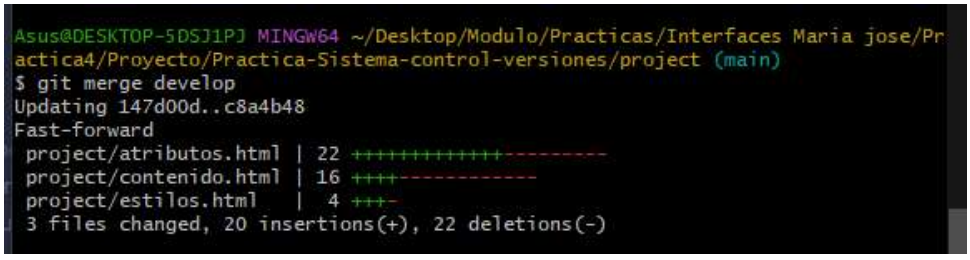
Acto seguido creamos un tag y le hacemos un push



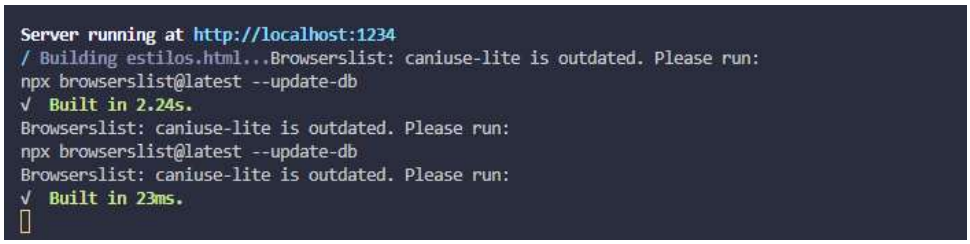
Y ahora hacemos la rama test y le hacemos un push



Y finalmente en el main hacemos un merge de la rama develop



Y ejecutamos la pagina para ver que funciona



Además, el usuario 1 ha decidido “automatizar” algunos procesos rutinarios. Por este motivo, decide crear los siguientes hooks de git: • Automatización de la instalación del proyecto. Cuando se realice un clonado/checkout del proyecto, se deberá recrear la carpeta de node_modules del proyecto. 0,5 puntos.

Implementamos el siguiente hook

Verificación del formato de mensaje de commit. Siempre que se realice un commit, el mensaje asociado deberá contener. 0,75 puntos:

• Un campo MOTIVO DEL COMMIT: Título asociado al commit

• Un campo IMPLEMENTACIÓN: con la explicación del desarrollo realizado

En este caso iremos a commit-msg y escribiremos lo siguiente

Y comprobamos que funcione

• Check caracteres extraños. Verificará que ninguno de los ficheros que se vaya a subir contenga texto con caracteres extraños como áâàéèêîìóòôúùû. 0,75 puntos

Aquí volveremos a usar el pre-commit

```
str='[áâàéèêíìîóòôúùû]'

files=$(git diff --cached --name-only --diff-filter=ACM | grep -E '\.(txt|md)$')

for file in $files; do
    if grep -q -P $str "$file"; then
        echo "ERROR HAS INTRODUCIDO CARACTERES EXTRAÑOS REPITALO SIN ELLOS"
        exit 1
    fi
done

exit 0
```

Previo a realizar el commit, se verificará el correcto formato en los ficheros html. Para ello, se ejecutará el linter eslint utilizando el siguiente plugin (link). 0,75 puntos Para que los anteriores hooks estén disponibles para el resto de miembros del equipo, el usuario 1 los subirá en una carpeta del proyecto denominada gitHooks. 0,3 puntos

Primero haremos que el usuario1 prepare los hooks para lo que haremos una carpeta y introduciremos los hooks. Luego lo subiremos al repositorio

```
$ git commit -m "MOTIVO DEL COMMIT: subir los hooks IMPLEMENTACIÓN: subir los ho
oks"
Pre-commit hook passed successfully.
[main be2b8b5] MOTIVO DEL COMMIT: subir los hooks IMPLEMENTACIÓN: subir los hook
s
 4 files changed, 35 insertions(+)
 create mode 100644 package-lock.json
 create mode 100644 project/gitHooks/commit-msg
 create mode 100644 project/gitHooks/post-checkout
 create mode 100644 project/gitHooks/pre-commit
```

Y hacemos el push

```
Asus@DESKTOP-5DSJ1PJ MINGW64 ~/Desktop/Modulo/Practicas/Interfaces Maria jose/Pr
actica4/Proyecto/Practica-Sistema-control-versiones/project (main)
$ git push origin
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 16 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (16/16), 1.85 KiB | 1.85 MiB/s, done.
Total 16 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 3 local objects.
To https://github.com/red153/Practica-Sistema-control-versiones.git
   147d00d..be2b8b5  main -> main
```

Al finalizar el trabajo, nuestro repositorio deberá contener, además de las ramas necesarias para poder realizar el desarrollo comentado, una rama (gh-pages) en la que se incluirá toda la documentación del proceso y que será la que publique github pages.

Creamos la rama gh-pages donde añadiremos la memoria realizada y subiremos a github