

# Actividad evaluable 2 - Git y GitHub

Documento realizado por Roberto Delgado Sánchez - Alumno de Despliegue de Aplicaciones Web - DAW

## Actividad evaluable 2 - Git y GitHub

1. Consideraciones previas
2. Trabajo en local
  - 2.1 Cuestión 1 - Inicialización y primeros pasos
  - 2.2 Cuestión 2 - Gitignore
  - 2.3 Cuestión 3 - Creamos algunos archivos
  - 2.4 Cuestión 4 - Creación de una nueva rama
  - 2.5 Cuestión 5 - Regreso a la rama main
  - 2.6 Cuestión 6 - Fusión de las ramas
3. Trabajo en remoto
  - 3.1 Cuestión 1 - Añadir el repositorio local a SourceTree
  - 3.2 Cuestión 2 - Crear repositorio remoto en GitHub y asociarlo al local
  - 3.3 Cuestión 3 - Creación de una nueva rama y cambiamos un archivo
  - 3.4 Cuestión 4 - Regreso a main y fusión con la rama antes creada
  - 3.5 Cuestión 5 - Hacemos cambios en otro archivo
  - 3.6 Cuestión 6 - Deshacer cambios antes hechos
  - 3.7 Cuestión 7 - Creación de otra rama
  - 3.8 Cuestión 8 - Subir los cambios de todas las ramas al remoto
  - 3.9 Cuestión 9 - Regreso a main y fusión con la última rama creada
4. Conflictos
  - 4.1 Cuestión 1 - Crear una nueva rama y añadir código a un fichero
  - 4.2 Cuestión 2 - Volver a main y modificar ese mismo fichero
  - 4.3 Cuestión 3 - Fusionar ambas ramas y resolver el conflicto
5. Subida de documentación al repositorio
6. Enlaces de interés
  - 6.1 Repositorio en GitHub
  - 6.2 Vídeo solicitado para la práctica

## 1. Consideraciones previas

En esta actividad se pondrán en práctica los conocimientos adquiridos en esta asignatura sobre el sistema de control de versiones **Git**, realizando la primera parte en nuestra máquina local utilizando el intérprete de comandos **Git Bash** y la segunda en la plataforma **GitHub** en combinación con la herramienta **SourceTree**, que permite trabajar con **Git** en modo gráfico. Todas las cuestiones planteadas en la actividad serán debidamente documentadas con la inclusión de los comandos ejecutados y los resultados de dicha ejecución.

En la parte final de este documento se incluyen enlaces tanto al repositorio remoto creado en **GitHub** como al vídeo creado con los contenidos requeridos.

## 2. Trabajo en local

Antes de empezar con nuestra tarea, además de revisar la configuración de nuestro repositorio local, ejecutaremos el siguiente comando para que el historial de ramas de **Git** se mantenga como un árbol:

```
$ git config --global merge.ff false
```

```
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit
$ git config --global merge.ff false

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit
$ |
```

La configuración de partida de nuestro repositorio local se comprueba con el comando que se muestra a continuación y es la siguiente:

```
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit
$ 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/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Roberto Delgado Sanchez
user.email=egl33817@educastur.es
color.ui=auto
merge.ff=false
```

En los cuadros rojos se pueden ver tanto mis datos personales (nombre, apellidos y correo de **educastur**) como que el primer comando que se ejecutó para el tema del historial de ramas se ejecutó correctamente.

Una vez tenemos listo el entorno, podemos empezar a resolver las cuestiones.

### 2.1 Cuestión 1 - Inicialización y primeros pasos

Inicia un nuevo repositorio Git en una carpeta llamada **labdist** y agrega los archivos proporcionados en el aula virtual. Renombra la rama master a **main**, si es necesario, realiza el primer **commit** y, finalmente, muestra el log del repositorio.

El primer paso es crear el repositorio **Git**, para lo cual nos situamos en la carpeta deseada (en mi caso será la situada en **D:\Despliegue\PracticaGit**) y ejecutamos el siguiente comando, que creará una carpeta llamada **labdist** en cuyo interior inicializará nuestro repositorio:

```
$ git init labdist
```

```

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit
$ pwd
/d/Despliegue/Practicagit

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit
$ git init labdist
Initialized empty Git repository in D:/Despliegue/PracticaGit/labdist/.git/

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit
$ ls
labdist/

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit
$ cd labdist

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (master)
$ ls -al
total 4
drwxr-xr-x 1 rober 197609 0 Feb 13 09:48 .
drwxr-xr-x 1 rober 197609 0 Feb 13 09:48 ../
drwxr-xr-x 1 rober 197609 0 Feb 13 09:48 .git/

```

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (master)

Como se puede ver en la captura, se ejecutan otros comandos con el siguiente objetivo:

- `pwd` : para que quede claro en qué carpeta estamos trabajando.
- `ls` : para comprobar que, efectivamente, se ha creado la carpeta de nombre `labdist`.
- `ls -al` : para verificar que tenemos una carpeta oculta de nombre `git`, lo que certifica que el repositorio se ha creado correctamente.

Ahora copiamos los archivos facilitados en el aula virtual con el explorador de archivos de Windows, mostrando en la siguiente captura de pantalla que, efectivamente, se han copiado correctamente:

```

$ ls -al

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (master)
$ ls -al
total 12
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 .
drwxr-xr-x 1 rober 197609 0 Feb 13 09:48 ../
drwxr-xr-x 1 rober 197609 0 Feb 13 09:48 .git/
-rw-r--r-- 1 rober 197609 1188 Nov 14 09:18 contacto.html
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 css/
-rw-r--r-- 1 rober 197609 1237 Nov 14 09:19 index.html
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 js/

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (master)
$ 

```

Cambiamos el nombre de la rama `master` a `main`, ya que cuando pasemos a la parte del trabajo en remoto será necesario que se llame así, pues en **GitHub** la rama principal tiene esa denominación. Ese cambio de nombre se hace con este comando:

```
$ git branch -M main
```

En la imagen se puede observar cómo, efectivamente, la rama `master` pasa a llamarse `main`:

```
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (master)
$ git branch -M main
```

```
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ |
```

Hacemos ahora nuestro primer `commit`, para lo cual movemos en primer lugar nuestros ficheros del directorio de trabajo a la llamada `staging area` y luego ejecutamos el `commit`. Los comandos implicados en este proceso son los que se muestran a continuación:

```
$ git status
$ git add .
$ git status
$ git commit -m "Realizamos el primer commit con los archivos de partida de nuestra web"
$ git status
```

```
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    contacto.html
    css/
    index.html
    js/

nothing added to commit but untracked files present (use "git add" to track)

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git add .

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   contacto.html
    new file:   css/estilos.css
    new file:   index.html
    new file:   js/script.js

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git commit -m "Realizamos el primer commit con los archivos de partida de nuestra web"
[main (root-commit) eb2121] Realizamos el primer commit con los archivos de partida de nuestra web
 4 files changed, 195 insertions(+)
 create mode 100644 contacto.html
 create mode 100644 css/estilos.css
 create mode 100644 index.html
 create mode 100644 js/script.js

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
nothing to commit, working tree clean

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ |
```

En principio la ejecución del comando `git status` no sería necesaria, pero considero interesante que quede claro el camino que siguen los archivos que hemos subido al repositorio desde el directorio de trabajo hasta su inclusión en el primer `commit`. Se marca en rojo el comando más importante aquí, `git commit`.

Ya por último mostramos el log del repositorio:

```
$ git log  
  
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)  
$ git log  
commit eb212172a4a1b37782ce518e967a9304cf6935b (HEAD -> main)  
Author: Roberto Delgado Sanchez <egl33817@educastur.es>  
Date:   Thu Feb 13 10:03:28 2025 +0100  
  
    Realizamos el primer commit con los archivos de partida de nuestra web  
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)  
$ |
```

Podemos ver en la captura detalles del `commit` como su código Hash completo, rama en la que se ha hecho, datos del autor del mismo y el mensaje añadido.

## 2.2 Cuestión 2 - Gitignore

Incluye un fichero `.gitignore` para que los ficheros `README.md`, `LICENCE.txt` y `passwords.txt` sean ignorados por el control de versiones. Realiza el commit y muestra los logs del repositorio en una línea.

A veces tendremos algún tipo de archivo que no queremos que sea controlado por **Git**, como por ejemplo archivos temporales, binarios, aquellos que son generados por el IDE que estemos usando, etc. En el archivo `.gitignore` se especifican los datos de aquellos archivos que queremos que sean ignorados (como su propio nombre indica) por el sistema de control de versiones, en nuestro caso son los tres indicados en el enunciado: `README.md`, `LICENCE.txt` y `passwords.txt`.

El archivo `.gitignore` se crea en la carpeta raíz de nuestro repositorio e introducimos en él los nombres de esos archivos que deben ser ignorados. El proceso implica la ejecución de los siguientes comandos y se puede ver en la captura de pantalla:

```
$ pwd  
$ ls -al  
$ vi .gitignore  
$ cat .gitignore  
$ ls -al
```

```

rober@surfacepro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ pwd
/d/Despliegue/PracticaGit/labdist

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ ls -al
total 12
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 .
drwxr-xr-x 1 rober 197609 0 Feb 13 09:48 ../
drwxr-xr-x 1 rober 197609 0 Feb 13 10:03 .git/
-rw-r--r-- 1 rober 197609 1188 Nov 14 09:18 contacto.html
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 css/
-rw-r--r-- 1 rober 197609 1237 Nov 14 09:19 index.html
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 js/

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ vi .gitignore

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ cat .gitignore
README.md
LICENCE.txt
passwords.txt

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ ls -al
total 17
drwxr-xr-x 1 rober 197609 0 Feb 13 10:29 .
drwxr-xr-x 1 rober 197609 0 Feb 13 09:48 ../
drwxr-xr-x 1 rober 197609 0 Feb 13 10:03 .git/
-rw-r--r-- 1 rober 197609 36 Feb 13 10:29 .gitignore
-rw-r--r-- 1 rober 197609 1188 Nov 14 09:18 contacto.html
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 css/
-rw-r--r-- 1 rober 197609 1237 Nov 14 09:19 index.html
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 js/

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ ...

```

Con los comandos ejecutados se demuestra que el archivo `.gitignore` no existía inicialmente y cómo se procedió a su creación con el editor `vi`. Su contenido se limita al nombre de los tres archivos que queremos ignorar.

Hacemos el `commit` para que el archivo `.gitignore` sea controlado por `Git`, ejecutando `git status` para ver los pasos que sigue dicho fichero:

```

$ git status
$ git add .
$ git status
$ git commit -m "Agregamos el fichero .gitignore a nuestro repositorio"
$ git status

```

```

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore

nothing added to commit but untracked files present (use "git add" to track)

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git commit -m "Agregamos el fichero .gitignore a nuestro repositorio"
[main bbd746f] Agregamos el fichero .gitignore a nuestro repositorio
 1 file changed, 3 insertions(+)
 create mode 100644 .gitignore

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
nothing to commit, working tree clean

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ |

```

Ya por último mostramos los logs del repositorio en una única línea:

```

$ git log --oneline

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git log --oneline
bbd746f (HEAD -> main) Agregamos el fichero .gitignore a nuestro repositorio
eb21217 Realizamos el primer commit con los archivos de partida de nuestra web

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ |

```

## 2.3 Cuestión 3 - Creamos algunos archivos

En el repositorio, crea los archivos `README.md`, `LICENCE.txt` y `passwords.txt` con algún contenido. Muestra el estado del repositorio. Muestra el listado de archivos ignorados.

La creación de los archivos no tiene ningún misterio, se lleva a cabo con `vi` y en la siguiente imagen se puede ver su contenido:

```

$ vi README.md
$ vi LICENCE.txt
$ vi passwords.txt

```

```

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ vi README.md

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ vi LICENCE.txt

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ vi passwords.txt

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ cat README.md
Este es un texto de ejemplo para el archivo README.md
Contiene dos líneas con palabras aleatorias.
Un saludo.

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ cat LICENCE.txt
Aquí dentro deberían venir detalles relativos a la licencia de nuestra web.
NO me extenderé mucho en este archivo.

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ cat passwords.txt
Este archivo contiene contraseñas necesarias para nuestra aplicación.
No se controla con Git ya que no interesa que salga de nuestra máquina local.
Así que aquí se queda.

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ 

```

Cuando comprobamos el estado del repositorio es cuando vemos realmente el trabajo que realiza `.gitignore`, ya que aunque hay tres ficheros nuevos en el repositorio, al figurar en la lista de aquellos que deben ser ignorados al hacer un `git status` vemos que el directorio de trabajo permanece vacío:

```

$ ls -al
$ git status

```

```

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ ls -al
total 20
drwxr-xr-x 1 rober 197609 0 Feb 13 10:45 .
drwxr-xr-x 1 rober 197609 0 Feb 13 09:48 ..
drwxr-xr-x 1 rober 197609 0 Feb 13 10:50 .git/
-rw-r--r-- 1 rober 197609 36 Feb 13 10:29 .gitignore
-rw-r--r-- 1 rober 197609 118 Feb 13 10:44 LICENCE.txt
-rw-r--r-- 1 rober 197609 111 Feb 13 10:43 README.md
-rw-r--r-- 1 rober 197609 1188 Nov 14 09:18 contacto.html
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 css/
-rw-r--r-- 1 rober 197609 1237 Nov 14 09:19 index.html
drwxr-xr-x 1 rober 197609 0 Feb 13 09:52 js/
-rw-r--r-- 1 rober 197609 176 Feb 13 10:45 passwords.txt

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
nothing to commit, working tree clean

```

Por último, para ver la lista de archivos que están siendo ignorados por Git gracias a `.gitignore` se usa el comando que se muestra a continuación:

```

$ git status --ignored

```

```

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status --ignored
On branch main
Ignored files:
  (use "git add -f <file>..." to include in what will be committed)
    LICENCE.txt
    README.md
    passwords.txt

nothing to commit, working tree clean
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ ...

```

## 2.4 Cuestión 4 - Creación de una nueva rama

Crea una rama de nombre `feature-estilos`, cámbiate a ella y lleva a cabo las siguientes tareas:

- modifica el archivo `estilos.css` añadiendo los siguientes estilos:
  - propiedad `color` del `body` y de `h2` : `#2a2a2a`
  - propiedad `background-color` de `header` y `footer` : `#2a75ff`
- comprueba el estado del repositorio. Añade los cambios, realiza un `commit` con el mensaje "actualizados estilos a azules".

En primer lugar creamos la rama `feature-estilos` con el siguiente comando:

```

$ git branch feature-estilos
$ git branch

```

```

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git branch feature-estilos

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git branch
  feature-estilos ←
* main

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ ...

```

El segundo comando ejecutado, `git branch`, nos sirve para ver el listado de ramas que tenemos en el repositorio y que la activa es `main` (es la que tiene el asterisco al lado de su nombre). Nos cambiamos a la rama recién creada con el siguiente comando y comprobamos (de nuevo con `git branch`) que es la rama que tenemos activa ahora:

```

$ git checkout feature-estilos
$ git branch

```

```
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git checkout feature-estilos
Switched to branch 'feature-estilos'

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ git branch
* feature-estilos
  main

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$
```

Una vez situados en esta rama, hacemos los cambios indicados en el archivo `estilos.css` con el **IDE Visual Studio Code**, incluyendo capturas de pantalla de dichos cambios y de cómo se mostraba nuestra web antes y después de los mismos:

The screenshot shows the homepage of the Labowebs website. The header features the company name "Labowebs" in a large, bold, black font, with the tagline "Confección de sitios web modernos y accesibles para pequeñas empresas" in a smaller, gray font below it. The main content area has a light blue background and contains two sections: "¿Quiénes Somos?" and "Nuestros Servicios". The "¿Quiénes Somos?" section includes a brief description of their services and a "Read more" button. The "Nuestros Servicios" section lists four services: Diseño de sitios web personalizados, Desarrollo de tiendas en línea, Optimización SEO para motores de búsqueda, and Soporte y mantenimiento continuo. At the bottom of the page, there is a footer with copyright information and links to "Inicio" and "Contacto".

**Labowebs**

Confección de sitios web modernos y accesibles para pequeñas empresas

## ¿Quiénes Somos?

En Labowebs nos especializamos en el diseño y desarrollo de sitios web adaptados a las necesidades de pequeñas empresas, brindando soluciones accesibles, atractivas y funcionales.

## Nuestros Servicios

- Diseño de sitios web personalizados
- Desarrollo de tiendas en línea
- Optimización SEO para motores de búsqueda
- Soporte y mantenimiento continuo

© 2024 Labowebs - Gijón, Asturias

[Inicio](#) | [Contacto](#)

EXPLORADOR

LABDIST

css

# estilos.css

> js

.gitignore

contacto.html

index.html

LICENCE.txt

passwords.txt

README.md

# estilos.css

```
* {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}  
  
body {  
    font-family: Arial, sans-serif;  
    color: #000;  
    line-height: 1.6;  
    background-color: #e6f0ff;  
    display: flex;  
    flex-direction: column;  
    min-height: 100vh;  
}
```

EXPLORADOR

LABDIST

css

# estilos.css

> js

.gitignore

contacto.html

index.html

LICENCE.txt

passwords.txt

README.md

# estilos.css

```
* {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}  
  
body {  
    font-family: Arial, sans-serif;  
    color: #2a2a2a;  
    line-height: 1.6;  
    background-color: #e6f0ff;  
    display: flex;  
    flex-direction: column;  
    min-height: 100vh;  
}
```

The screenshot shows a code editor interface with a dark theme. In the top navigation bar, the tabs are Archivo, Editar, Selección, Ver, Ir, Ejecutar, ..., and back/forward arrows. The left sidebar is titled 'EXPLORADOR' and shows a folder structure under 'LABDIST': 'css' (selected), '# estilos.css', 'js', '.gitignore', 'contacto.html', and 'index.html'. The main editor area displays the contents of '# estilos.css'. The code includes:

```
css > # estilos.css > header
35 }
36
37 h2 {
38 | color: □#000;
39 | margin-bottom: 15px;
40 }
41
```

A red rectangular box highlights the 'color: □#000;' line.

This screenshot is similar to the first one, showing the same code editor interface and file structure. The main editor area displays the contents of '# estilos.css'. The code includes:

```
css > # estilos.css > ...
28 section.about, section.cor
34 | box-shadow: 0 0 10px
35 }
36
37 h2 {
38 | color: □#2a2a2a;
39 | margin-bottom: 15px;
40 }
41
```

A green rectangular box highlights the 'color: □#2a2a2a;' line.

This screenshot shows the same code editor interface and file structure. The main editor area displays the contents of '# estilos.css'. The code includes:

```
css > # estilos.css > header h1
15 }
16
17 header, footer {
18 | background-color: □#bbb;
19 | color: □#2a2a2a;
20 | text-align: center;
21 | padding: 1em 0;
22 }
23
```

A red rectangular box highlights the 'background-color: □#bbb;' line.

```
# estilos.css
css > # estilos.css > header h1
7 body {
14     min-height: 100vh;
15 }
16
17 header, footer {
18     background-color: #2a75ff;
19     color: #2a2a2a;
20     text-align: center;
21     padding: 1em 0;
22 }
```

No forma parte de la práctica, pero se quiere dejar constancia de que, una vez que se guardan los cambios realizados en el archivo `estilos.css`, el propio IDE nos informa de que esos cambios han sido detectados por Git, de tal forma que ahora aparece una letra `M` a la derecha del nombre del archivo:

```
EXPLORADOR ...
```

✓ LABDIST

- ✓ css
- # estilos.css M**
- > js
- ❖ .gitignore
- ⌚ contacto.html
- ⌚ index.html
- 🔑 LICENCE.txt
- ≡ passwords.txt
- ⓘ README.md

Para finalizar con este apartado, toca comprobar el estado del repositorio, añadir los cambios y hacer un `commit` con el mensaje `"Actualizados estilos a azules"`. Como todas esas tareas ya las hemos hecho con anterioridad, se muestran todos los comandos del tirón para no ser repetitivos, así como su resultado y la aplicación de esos nuevos estilos a nuestra web:

```
$ git status
$ git add .
$ git status
$ git commit -m "Actualizados estilos a azules"
$ git status
```

```
rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ git status
on branch feature-estilos
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:   css/estilos.css

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

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ git add .

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ git status
on branch feature-estilos
changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   css/estilos.css

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ git commit -m "Actualizados estilos a azules"
[feature-estilos d069db0] Actualizados estilos a azules
  1 file changed, 3 insertions(+), 3 deletions(-)

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ git status
on branch feature-estilos
nothing to commit, working tree clean

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ |
```

# Labowebs

Confección de sitios web modernos y accesibles para pequeñas empresas

## ¿Quiénes Somos?

En Labowebs nos especializamos en el diseño y desarrollo de sitios web adaptados a las necesidades de pequeñas empresas, brindando soluciones accesibles, atractivas y funcionales.

## Nuestros Servicios

- Diseño de sitios web personalizados
- Desarrollo de tiendas en línea
- Optimización SEO para motores de búsqueda
- Soporte y mantenimiento continuo

© 2024 Labowebs - Gijón, Asturias

[Inicio](#) | [Contacto](#)

Son cambios muy sutiles y casi no se observa diferencia.

## 2.5 Cuestión 5 - Regreso a la rama main

Vuelve a la rama `main` y en el archivo `index.html` añade un comentario donde se indique tu nombre como autor de la página. Comprueba el estado del repositorio. Añade los cambios, realiza un `commit` con el mensaje "`Añadido autor en index`". Muestra los logs del repositorio en una línea, gráficamente y con 'decoración'.

En primer lugar hacemos el cambio de rama con el comando siguiente:

```
$ git checkout main
```

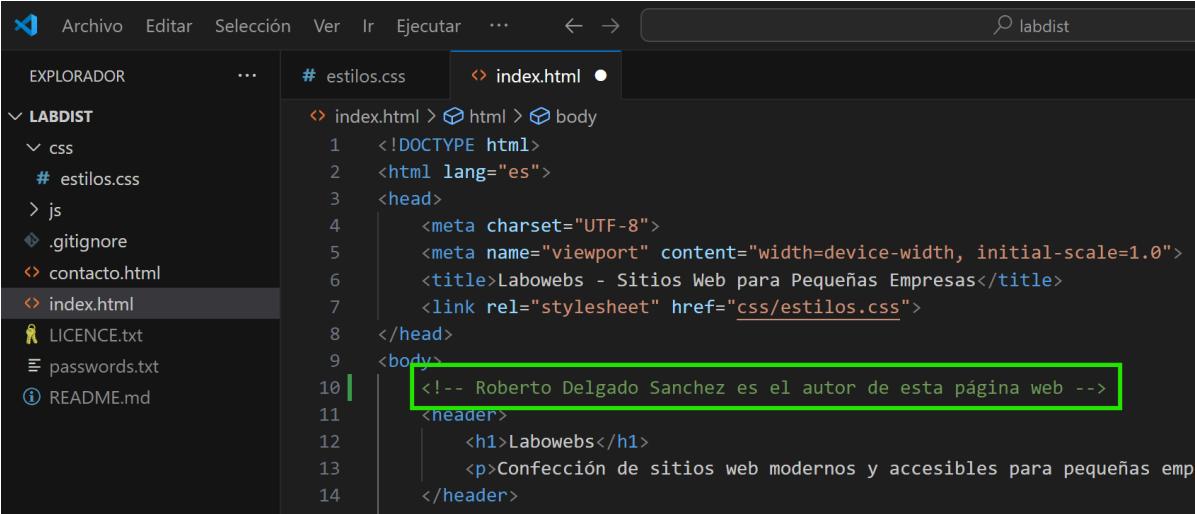
```
rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ git branch
* feature-estilos
  main

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (feature-estilos)
$ git checkout main
Switched to branch 'main'

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git branch
  feature-estilos
* main

rober@surfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$
```

Como se puede ver en las capturas de pantalla, con el comando `git branch` se comprueba cómo, efectivamente, se hace ese cambio entre ramas. Volvemos a continuación a **Visual Studio Code** y añadimos el comentario solicitado, tal y como se puede ver en esta captura de pantalla:



The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with files like # estilos.css, .gitignore, contacto.html, index.html (which is selected), LICENCE.txt, passwords.txt, and README.md. The main editor area shows the content of index.html:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Labowebs - Sitios Web para Pequeñas Empresas</title>
    <link rel="stylesheet" href="css/estilos.css">
</head>
<body>
    <!-- Roberto Delgado Sanchez es el autor de esta página web -->
    <header>
        <h1>Labowebs</h1>
        <p>Confección de sitios web modernos y accesibles para pequeñas emp
```

Comprobamos el estado del repositorio, añadimos los cambios y hacemos el `commit`, todo ello aplicando los comandos que ya hemos visto a lo largo de la actividad:

```
$ git status
$ git add .
$ git status
$ git commit -m "Añadido autor en index"
$ git status
```

```

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch 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")

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git add .

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   index.html

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git commit -m "Añadido autor en index"
[main a6f196e] Añadido autor en index
  1 file changed, 1 insertion(+)

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git status
On branch main
nothing to commit, working tree clean

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ |

```

Por último, mostramos los logs del repositorio en una línea, en modo gráfico y con decoración:

```

$ git log --oneline --graph --decorate

```

```

rober@SurfacePro9 MINGW64 /d/Despliegue/Practicagit/labdist (main)
$ git log --oneline --graph --decorate
* a6f196e (HEAD -> main) Añadido autor en index
* bbd746f Agregamos el fichero .gitignore a nuestro repositorio
* eb21217 Realizamos el primer commit con los archivos de partida de nuestra web
rober@SurfacePro9 MINGW64 /d/Despliegue/Practicagit/labdist (main)
$ |

```

## 2.6 Cuestión 6 - Fusión de las ramas

Fusione la rama `feature-estilos` en la rama `main`. Muestra los logs del repositorio en una línea, gráficamente y con 'decoración'.

Al estar ya situados en la rama `main` (era necesario para resolver la cuestión anterior), la fusión de la rama llamada `feature-estilos` se hace con este comando:

```

$ git merge feature-estilos

```

Como se hace un `commit` de manera automática al fusionar las ramas, se nos abre el editor `vi` con un mensaje por defecto para ese `commit`. Dejamos un comentario personalizado, aunque no se indique en el enunciado:

```
MINGW64:/d/Despliegue/PracticaGit/labdist
Merge branch 'feature-estilos'
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
```

```
MINGW64:/d/Despliegue/PracticaGit/labdist
Fusionamos la rama 'feature-estilos' con main.
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
```

```
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git merge feature-estilos
Merge made by the 'ort' strategy.
 css/estilos.css | 6 +++---
 1 file changed, 3 insertions(+), 3 deletions(-)

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$
```

Una vez fusionadas las ramas, mostramos los logs del repositorio con las opciones solicitadas:

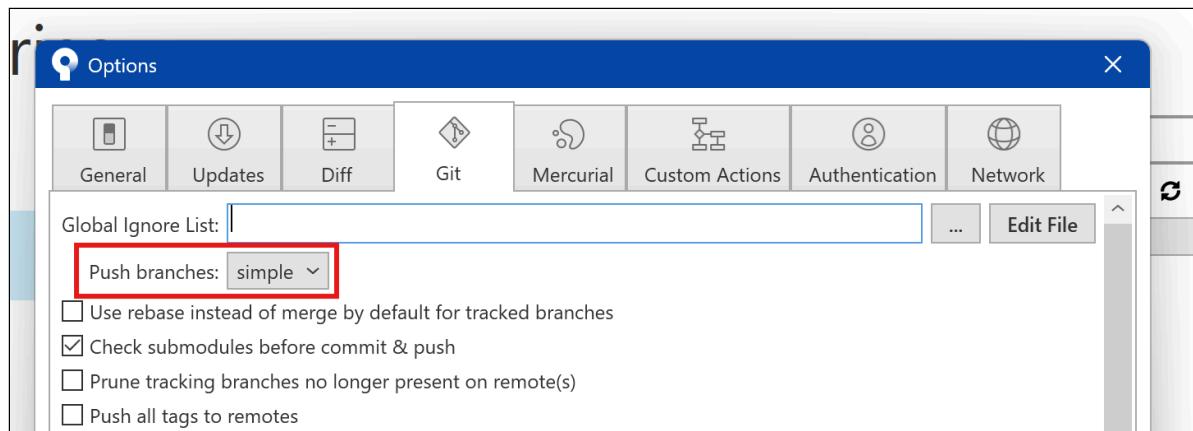
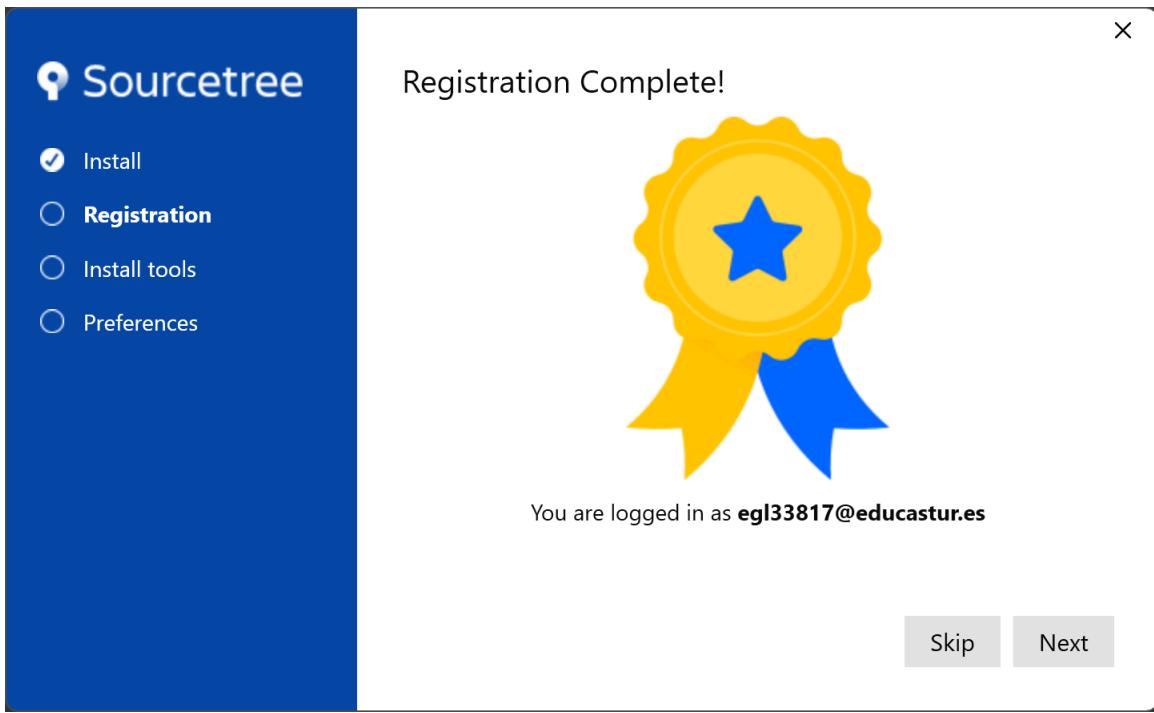
```
$ git log --oneline --graph --decorate
```

```
rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$ git log --oneline --graph --decorate
*   80a09d6 (HEAD -> main) Fusionamos la rama 'feature-estilos' con main.
|\ 
| * d069db0 (feature-estilos) Actualizados estilos a azules
| | a6f196e Añadido autor en index
|/
* bbd746f Agregamos el fichero .gitignore a nuestro repositorio
* eb21217 Realizamos el primer commit con los archivos de partida de nuestra web

rober@SurfacePro9 MINGW64 /d/Despliegue/PracticaGit/labdist (main)
$
```

### 3. Trabajo en remoto

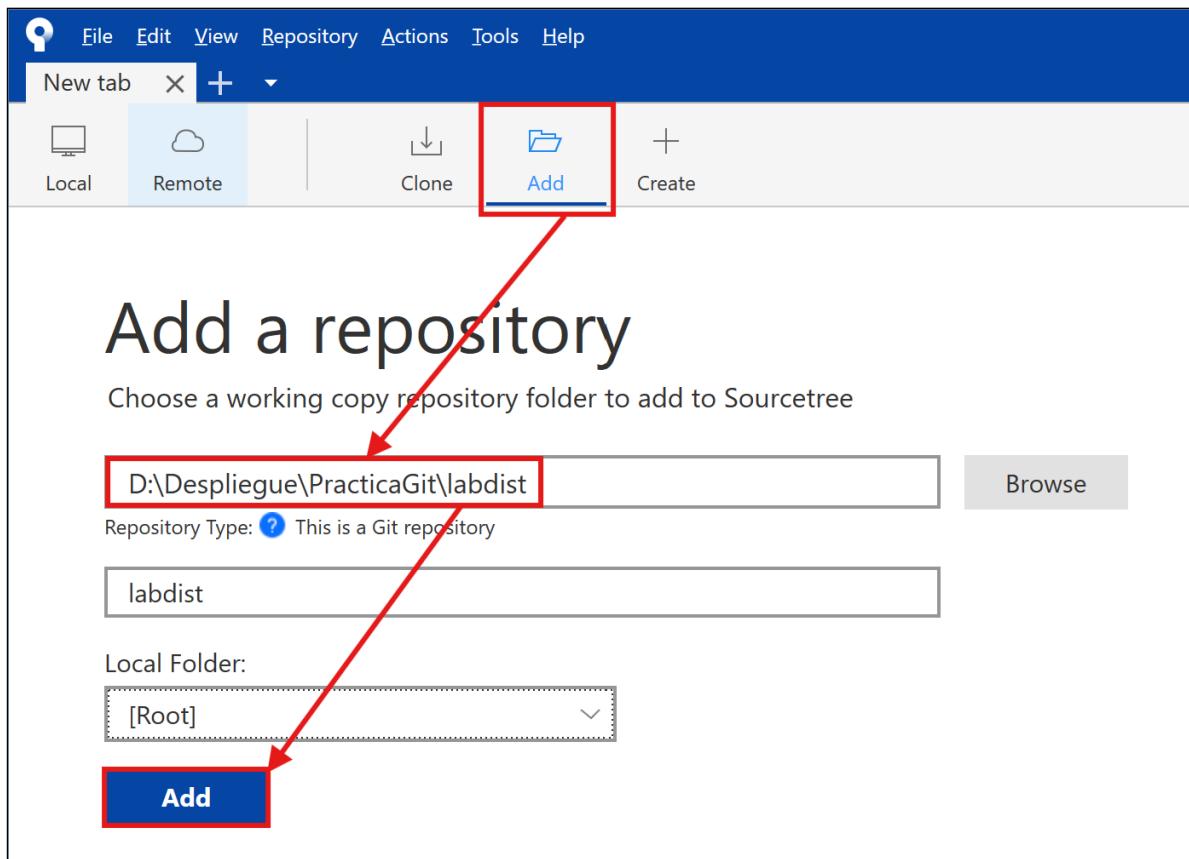
Antes de empezar con nuestra tarea, nos registramos con nuestra cuenta de educastur para poder usar SourceTree y luego comprobamos que esta herramienta está configurada con las opciones indicadas en el enunciado:



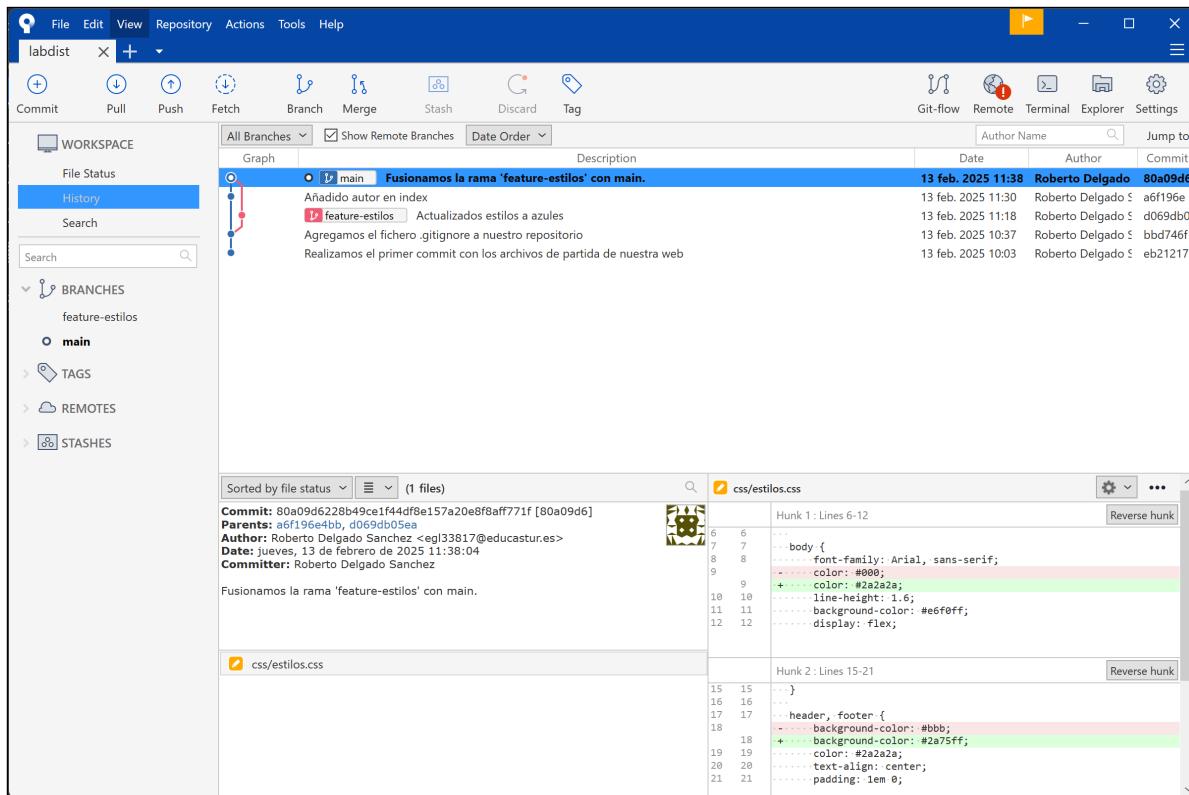
### 3.1 Cuestión 1 - Añadir el repositorio local a SourceTree

Continúa con el repositorio **labdist**. Añade el repositorio a SourceTree.

Para gestionar nuestro repositorio local **labdist** con SourceTree, lo único que tenemos que hacer es ir a la opción **Add** e incluir la localización del directorio donde está alojado nuestro repositorio, tal y como se puede ver a continuación:

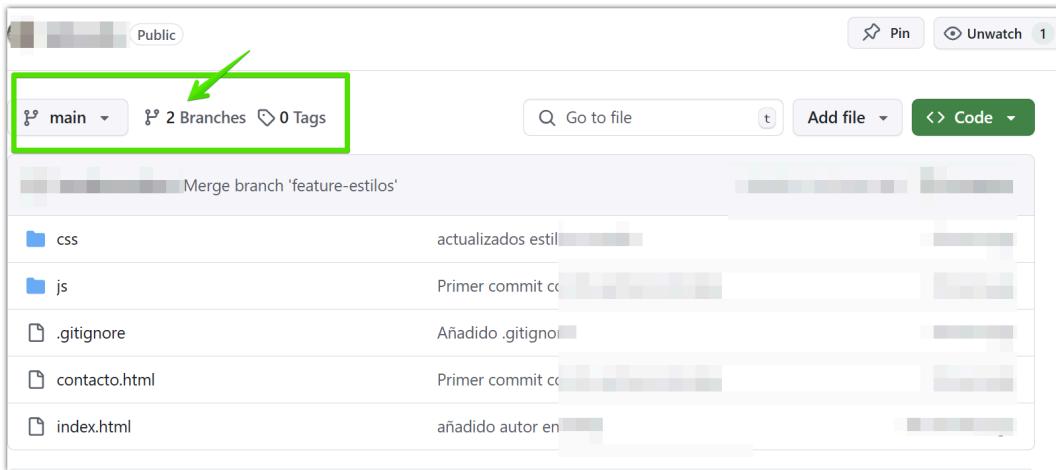


Una vez añadido podemos ver de forma gráfica los diversos aspectos relacionados con nuestro repositorio, como son los **commits** hechos con sus detalles, ramas creadas y fusionadas, etc.:



### 3.2 Cuestión 2 - Crear repositorio remoto en GitHub y asociarlo al local

En tu cuenta de GitHub, crea un repositorio remoto y sube al remoto los ficheros de tu repositorio local. Debes subir todas las ramas. Muestra, además, la captura de pantalla donde se vean en GitHub, algo similar a esto:



En primer lugar, nos vamos a **GitHub** y, tras loguearnos con nuestra cuenta de **educastur**, creamos un repositorio remoto de nombre **labdist** y de acceso público:

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

**Owner \***  egl33817 / **Repository name \*** labdist  labdist is available.

Great repository names are short and memorable. Need inspiration? How about [fictional-system](#) ?

**Description** (optional)

 **Public**  
Anyone on the internet can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**

**Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

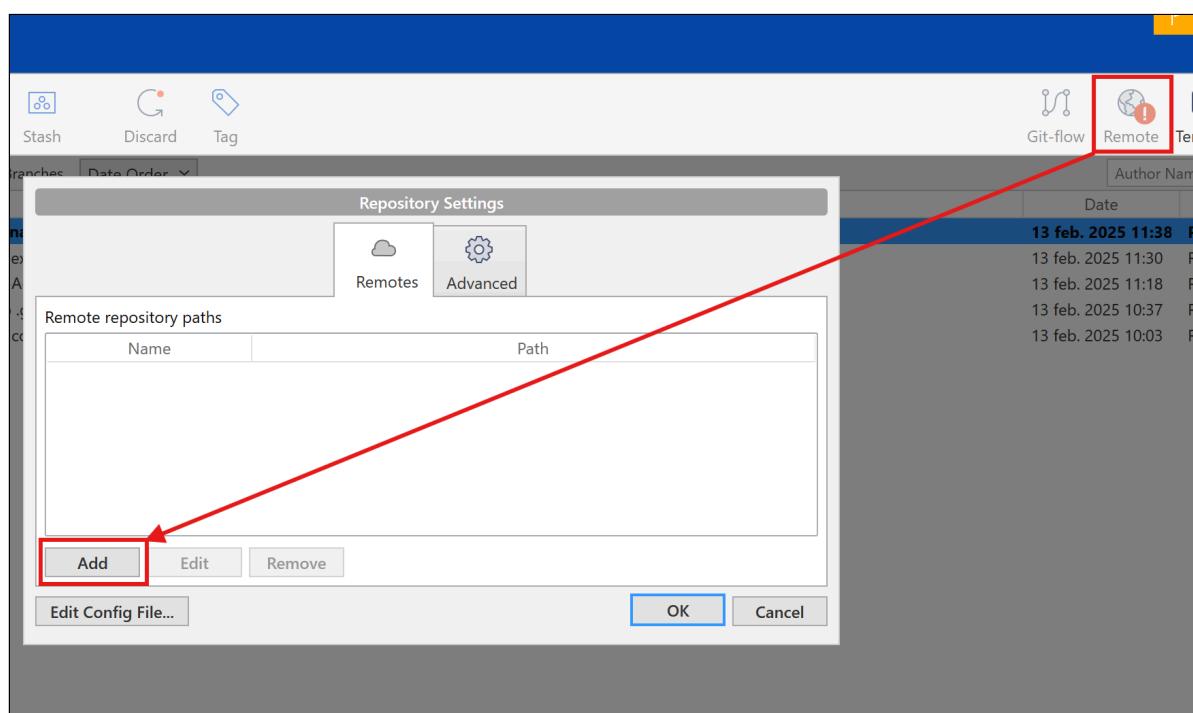
The screenshot shows a GitHub repository page for 'labdist'. At the top, there's a navigation bar with links like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The repository name 'labdist' is displayed with a 'Public' badge. A sidebar on the right has a 'Collaborators' section with a '+ Add collaborator' button and a search field. The main content area features a 'Set up GitHub Copilot' section with a 'Get started with GitHub Copilot' button. Below it is a 'Quick setup — if you've done this kind of thing before' section. It provides two ways to set up the repository: 'Set up in Desktop' (selected) or 'HTTPS' or 'SSH', with the URL <https://github.com/egl33817/labdist.git>. It also suggests starting by creating a new file or uploading an existing one, and recommends including a README. Further down, there's a section for '...or create a new repository on the command line' with a block of terminal commands:

```
echo "# labdist" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/egl33817/labdist.git
git push -u origin main
```

Finally, there's a section for '...or push an existing repository from the command line' with another block of terminal commands:

```
git remote add origin https://github.com/egl33817/labdist.git
git branch -M main
git push -u origin main
```

Una vez creado el repositorio en **GitHub**, de vuelta en **SourceTree** asociamos nuestro repositorio local con el remoto siguiendo los pasos que se muestran en las siguientes imágenes:



**Remote details**

Required information

Remote name: origin  Default remote

URL / Path: https://github.com/egl33817/labdist.git

Optional extended integration

Remote Account:

Generic Account   
Generic Host

Legacy Account Settings:

Host Type: GitHub

Host Root URL: https://www.github.com

Username: egl33817

Extended integration is used to enable deeper integration with hosting providers such as Bitbucket, including locating existing clones when following links from sites and creating pull requests.

OK Cancel

**Repository Settings**

Remotes Advanced

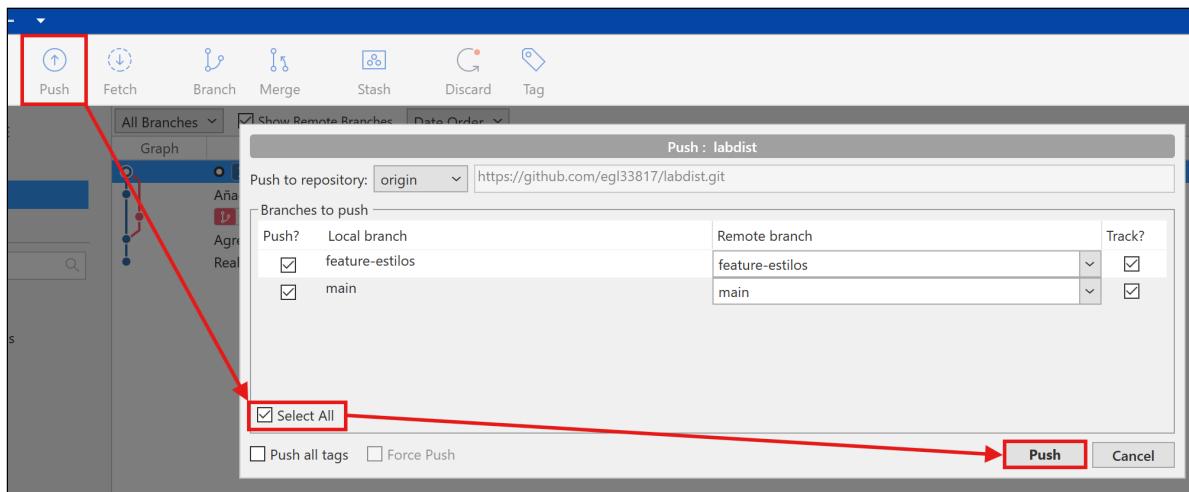
Remote repository paths

Name	Path
origin	https://github.com/egl33817/labdist.git

Add Edit Remove

Edit Config File... OK Cancel

Luego subimos todas las ramas que tenemos definidas en nuestro repositorio local al remoto con la opción **Push**:



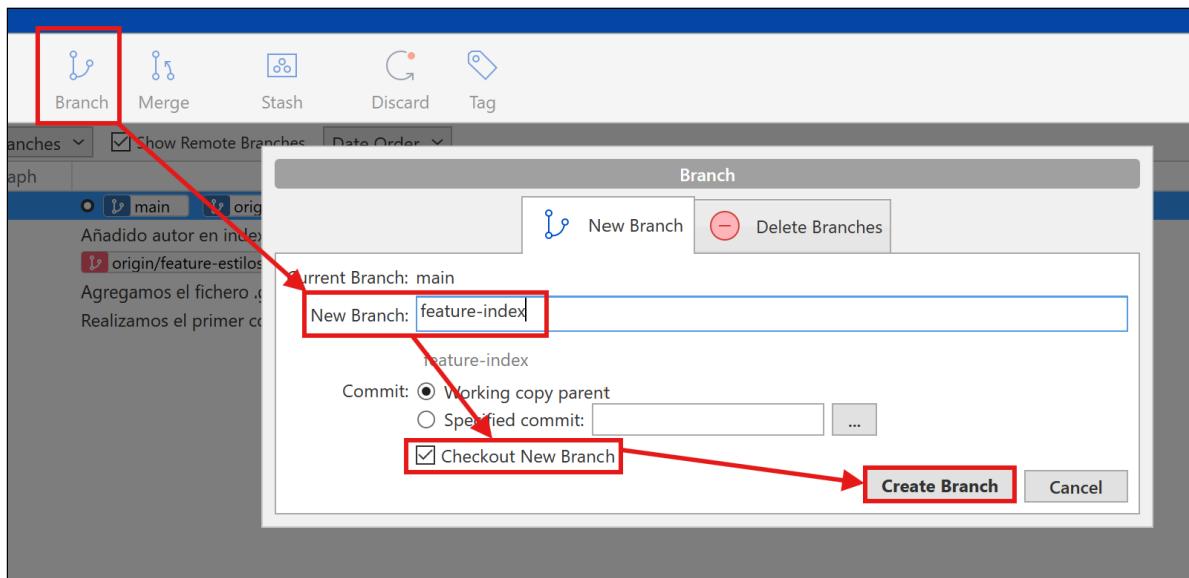
Y comprobamos que aparece correctamente en GitHub:

### 3.3 Cuestión 3 - Creación de una nueva rama y cambiamos un archivo

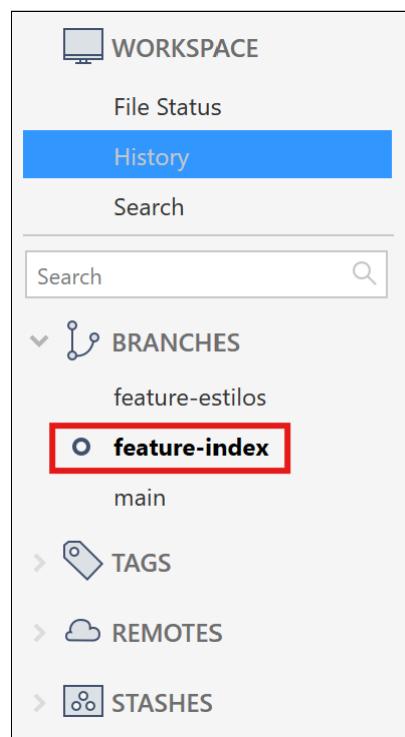
En el repositorio local, crea una rama **feature-index**. Añade el siguiente código dentro de la **<section class='about'>**. Añade los cambios y crea un commit con el mensaje "Añadido párrafo equipo en index.html". Sube los cambios al remoto.

```
<h2>Conoce a Nuestro Equipo</h2>
      <p>labdist está formado por un equipo de diseñadores y desarrolladores apasionados que trabajan juntos para ofrecer soluciones tecnológicas de alta calidad. Cada miembro de nuestro equipo aporta experiencia en diseño, programación, y soporte técnico, asegurando que nuestros clientes reciban un servicio completo y personalizado.</p>
      <p>Nuestro objetivo es que cada cliente se sienta acompañado en su aventura digital, con un equipo profesional que entiende sus necesidades y trabaja para hacer crecer su presencia en línea.</p>
```

Para crear una nueva rama en nuestro repositorio local, hacemos clic en la opción **Branch** de SourceTree, luego le asignamos el nombre indicado a la rama y al estar marcado el checkbox **checkout new branch** en cuanto se crea pasa a ser la rama activa.



En la siguiente captura de pantalla se puede ver que la rama activa es **feature-index**:



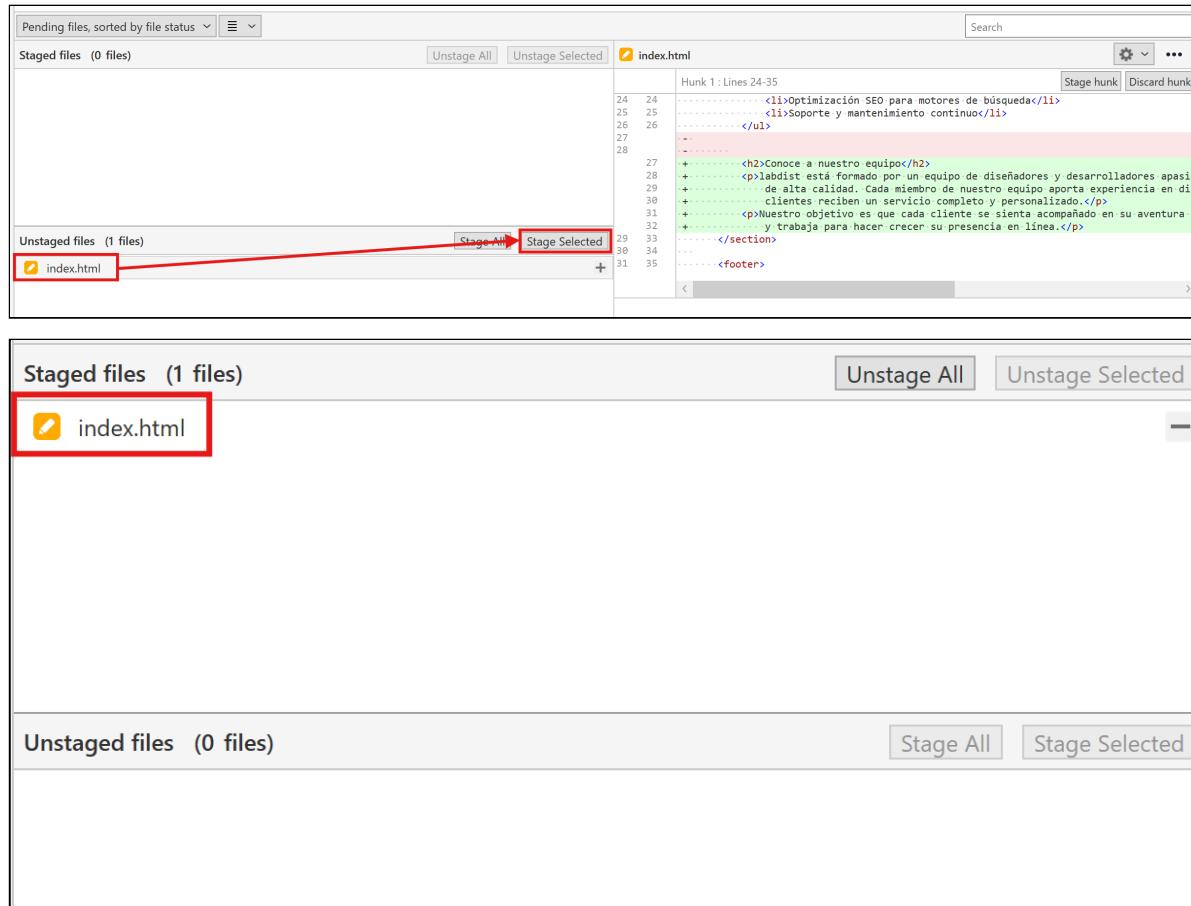
Ahora añadimos el código facilitado en el archivo **index.html** usando **Visual Studio Code**:

```

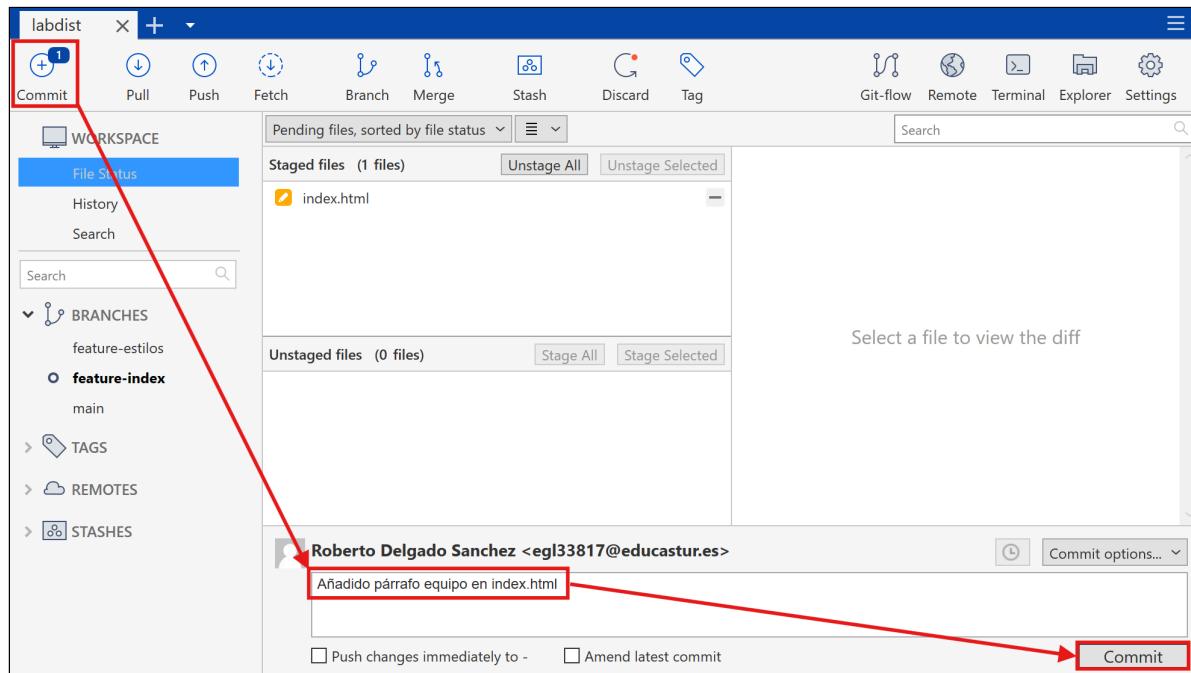
# estilos.css
index.html M X
index.html > html > body > section.about > p
2   <html lang="es">
9   <body>
16   <section class="about">
17     <h2>¿Quienes Somos?</h2>
18     <p>En Laboweb nos especializamos en el diseño y desarrollo de sitios web adaptados a las necesidades de pequeñas empresas, brindando soluci
19
20     <h2>Nuestros Servicios</h2>
21     <ul>
22       <li>Diseño de sitios web personalizados</li>
23       <li>Desarrollo de tiendas en linea</li>
24       <li>Optimización SEO para motores de búsqueda</li>
25       <li>Soporte y mantenimiento continuo</li>
26     </ul>
27     <h2>Conoce a nuestro equipo</h2>
28     <p>Laboweb está formado por un equipo de diseñadores y desarrolladores apasionados que trabajan juntos para ofrecer soluciones tecnológicas
29       de alta calidad. Cada miembro de nuestro equipo aporta experiencia en diseño, programación y soporte técnico, asegurando que nuestros
30       clientes reciben un servicio completo y personalizado.</p>
31     <p>Nuestro objetivo es que cada cliente se sienta acompañado en su aventura digital, con un equipo profesional que entiende sus necesidades
32       y trabaja para hacer crecer su presencia en línea.</p>
33   </section>
34
35   <footer>
36     <p>© 2024 Laboweb - Gijón, Asturias</p>
37     <nav>
38       <a href="index.html">Inicio</a> | <a href="contacto.html">Contacto</a>

```

Subimos los cambios a la **staging area** seleccionando el fichero **index.html** en la parte inferior de la ventana de **SourceTree**, concretamente donde pone **Unstaged files**, y al pulsar en el botón **Stage Selected** el archivo queda listo para hacerle el **commit**:

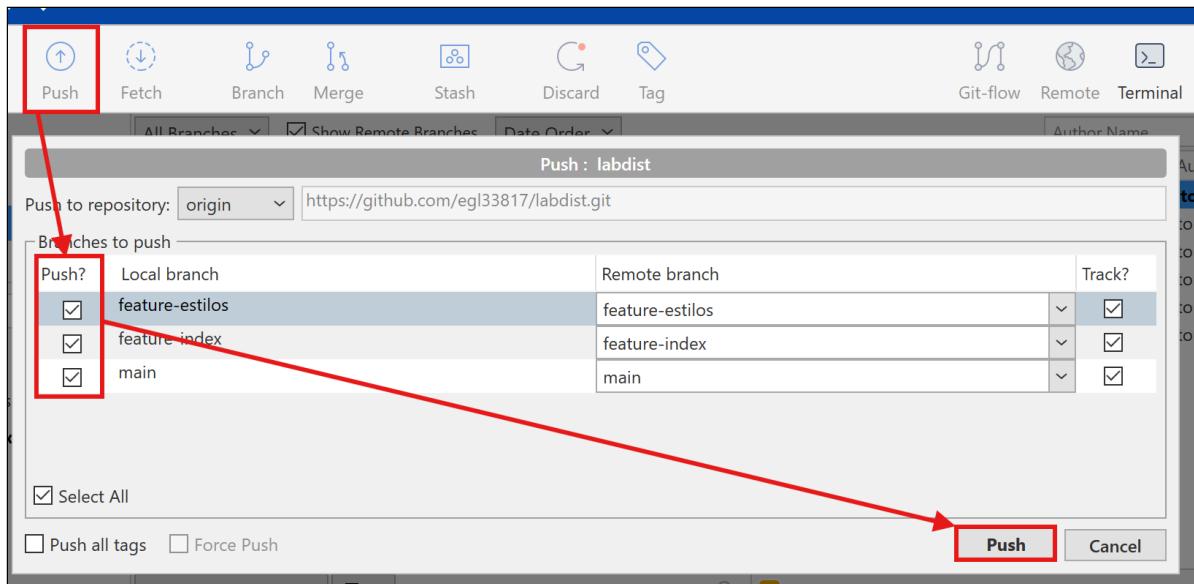


Para hacer el **commit** pulsamos en el botón correspondiente y escribimos el mensaje en el recuadro inferior:



All Branches	Show Remote Branches	Date Order	Author Name	Jump to:	
Graph	Description	Date	Author	Commit	
feature-index	Añadido párrafo equipo en index.html	13 feb. 2025 21:34	Roberto Delgado	d1fbbfb	
origin/main	main	Fusionamos la rama 'feature-estilos' con mai	13 feb. 2025 11:38	Roberto Delgado	80a09d6
		Añadido autor en index	13 feb. 2025 11:30	Roberto Delgado	a6f196e
origin/feature-estilos	feature-estilos	Actualizados estilos a azule	13 feb. 2025 11:18	Roberto Delgado	d069db0
		Agregamos el fichero .gitignore a nuestro repositorio	13 feb. 2025 10:37	Roberto Delgado	bbd746f
		Realizamos el primer commit con los archivos de partida de nuestra web	13 feb. 2025 10:03	Roberto Delgado	eb21217

Para subir los cambios al remoto hacemos clic en el botón **Push**, seleccionamos las ramas que queremos subir y ejecutamos el **Push**:



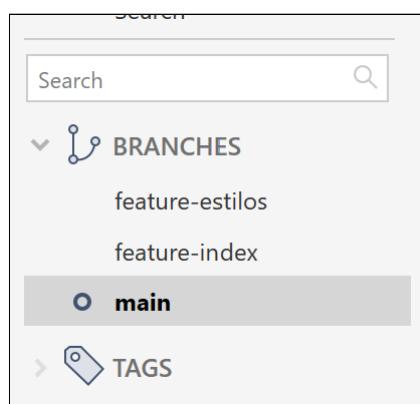
Comprobamos que los cambios quedan reflejados en GitHub:

Author	Commit Message	Date	Commits
egl33817	Añadido párrafo equipo en index.html	5 minutes ago	6 Commits
	Actualizados estilos a azules	10 hours ago	
	Realizamos el primer commit con los archivos de partida de ...	11 hours ago	
	Agregamos el fichero .gitignore a nuestro repositorio	11 hours ago	
	Realizamos el primer commit con los archivos de partida de ...	11 hours ago	
	Añadido párrafo equipo en index.html	5 minutes ago	

### 3.4 Cuestión 4 - Regreso a main y fusión con la rama antes creada

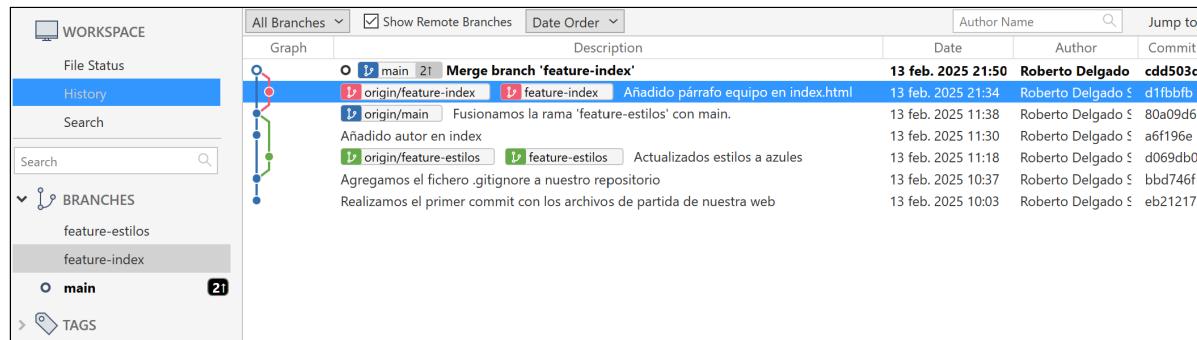
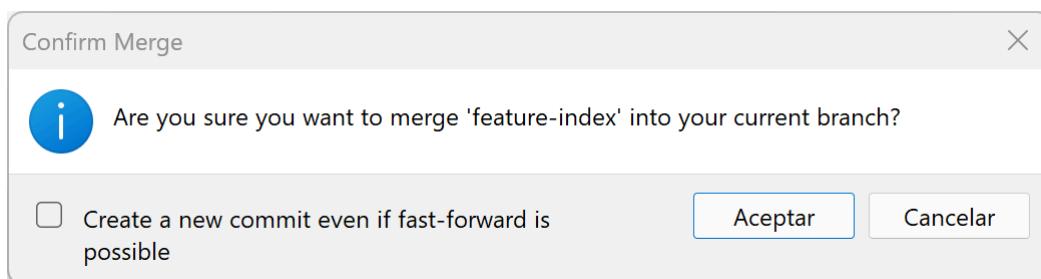
En el repositorio local, fusiona la rama `feature-index` en la rama `main`.

En primer lugar, nos movemos a la rama `main` haciendo doble clic en su nombre:



Luego hago clic con el botón derecho sobre la rama llamada `feature-index` y, haciendo clic con el botón derecho sobre su nombre, selecciono la opción denominada `Merge feature-index into current branch` del menú contextual que aparece, con la que haremos esa fusión de la rama `feature-index` en `main`:

Confirmamos que queremos hacer el `merge` y este queda finiquitado:



### 3.5 Cuestión 5 - Hacemos cambios en otro archivo

Edita el archivo `contacto.html`, borra unas cuantas líneas y muestra los ficheros con cambios pendientes y las diferencias entre ellos. Añade los cambios y haz un `commit`.

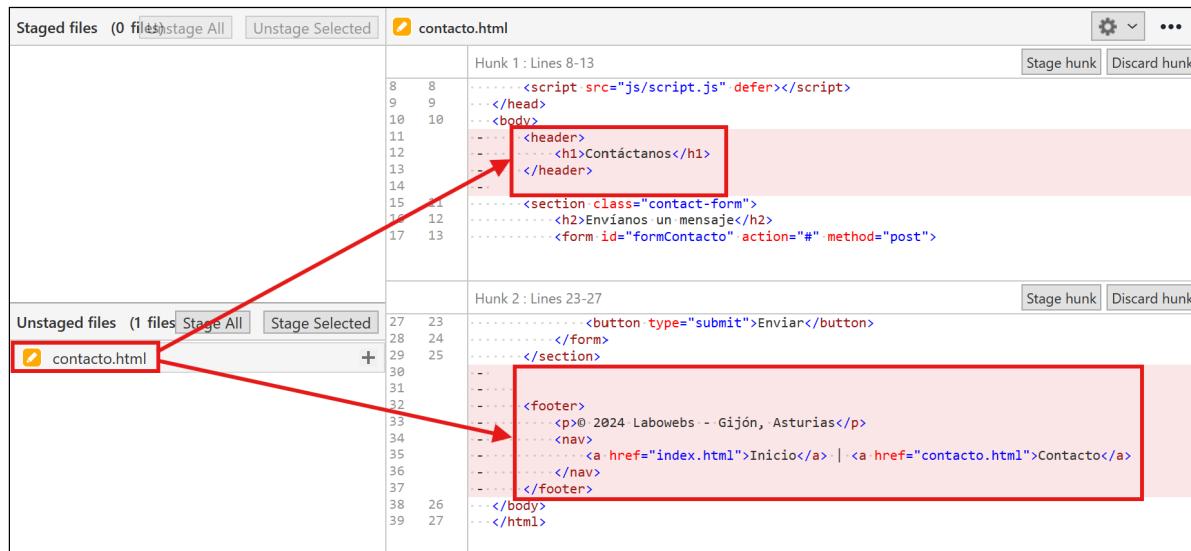
Vamos a borrar las líneas en `contacto.html` (en concreto, borramos el `header` y el `footer`) usando el IDE `Visual Studio Code`:

```

# estilos.css      < index.html      < contacto.html M X
< contacto.html > html > body > section.contact-form
1   <!DOCTYPE html>
2   <html lang="es">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Contacto - Labowebs</title>
7       <link rel="stylesheet" href="css/estilos.css">
8       <script src="js/script.js" defer></script>
9   </head>
10  <body>
11      <section class="contact-form">
12          <h2>Envíanos un mensaje</h2>
13          <form id="formContacto" action="#" method="post">
14              <label for="nombre">Nombre:</label>
15              <input type="text" id="nombre" name="nombre" required>
16
17              <label for="email">Correo Electrónico:</label>
18              <input type="email" id="email" name="email" required>
19
20              <label for="mensaje">Mensaje:</label>
21              <textarea id="mensaje" name="mensaje" rows="4" required></textarea>
22
23              <button type="submit">Enviar</button>
24          </form>
25      </section>
26  </body>
27 </html>
28

```

Para ver los ficheros con cambios pendientes y las diferencias respecto a sus versiones originales sólo hacer falta ir a la parte inferior de **SourceTree**, seleccionar el único fichero que aparece como modificado (`contacto.html`) y ver en la parte derecha las diferencias (en concreto, marca en un color rosáceo las líneas que hemos eliminado, incluidas las que originalmente estaban en blanco):



Añadimos los cambios y hacemos el `commit` correspondiente siguiendo los pasos ya vistos con anterioridad:

The screenshot illustrates the steps to stage and commit changes in GitHub Desktop:

- Unstaged files:** Shows 'contacto.html' with a yellow edit icon. Buttons include 'Stage All' (highlighted with a red box) and 'Stage Selected'.
- Staged files:** Shows 'contacto.html' with a yellow edit icon. Buttons include 'Unstage All' and 'Unstage Selected'.
- Main GitHub Desktop View:** Shows the 'File Status' tab selected. The commit message 'Borramos líneas en el archivo contacto.html' is highlighted with a red box.
- Commit Dialog:** Shows the commit message 'Borramos líneas en el archivo contacto.html' and the 'Commit' button highlighted with a red box.

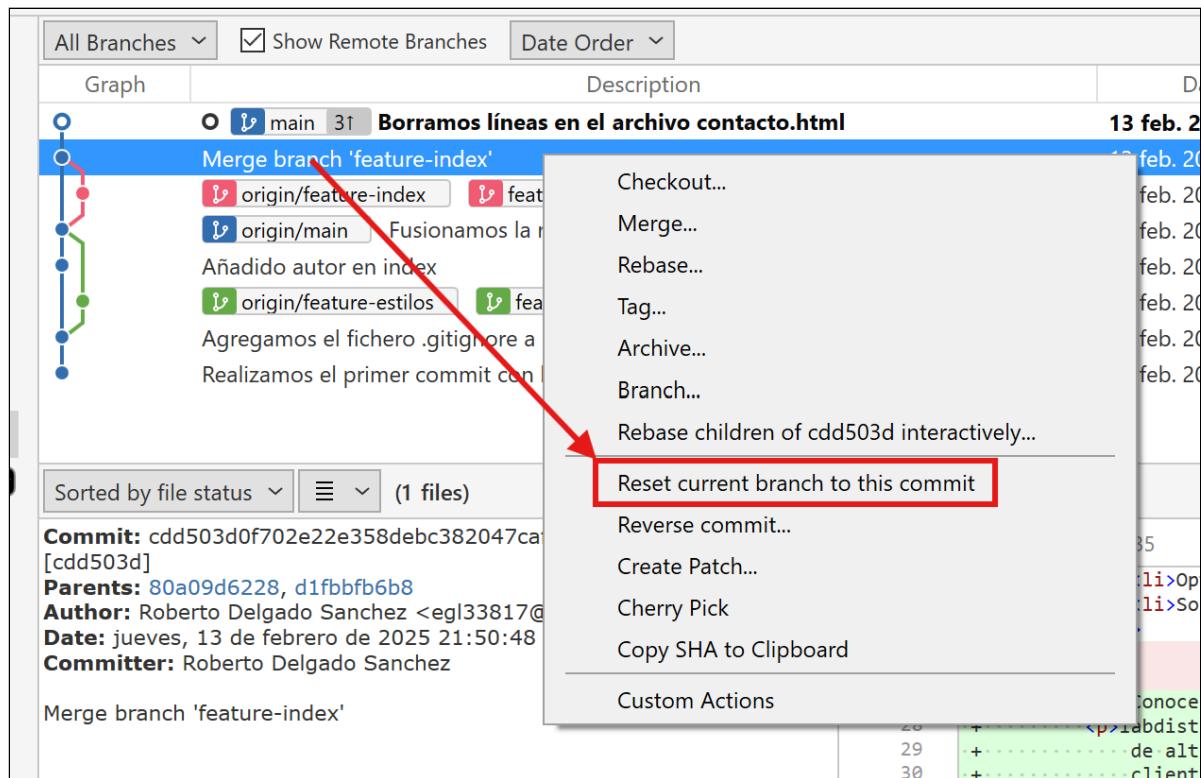
Below the commit dialog is a detailed log of the commit history:

Date	Author	Commit
13 feb. 2025 22:09	Roberto Delgado	2e1c05b
13 feb. 2025 21:50	Roberto Delgado	cdd503d
13 feb. 2025 21:34	Roberto Delgado	d1fbfbfb
13 feb. 2025 11:38	Roberto Delgado	80a09d6
13 feb. 2025 11:30	Roberto Delgado	a6f196e
13 feb. 2025 11:18	Roberto Delgado	d069db0
13 feb. 2025 10:37	Roberto Delgado	bbd746f
13 feb. 2025 10:03	Roberto Delgado	eb21217

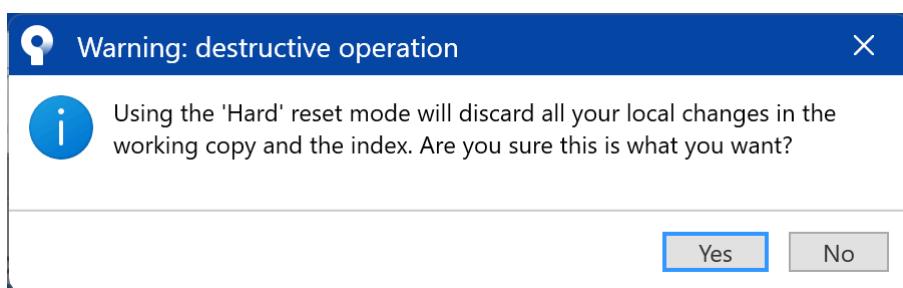
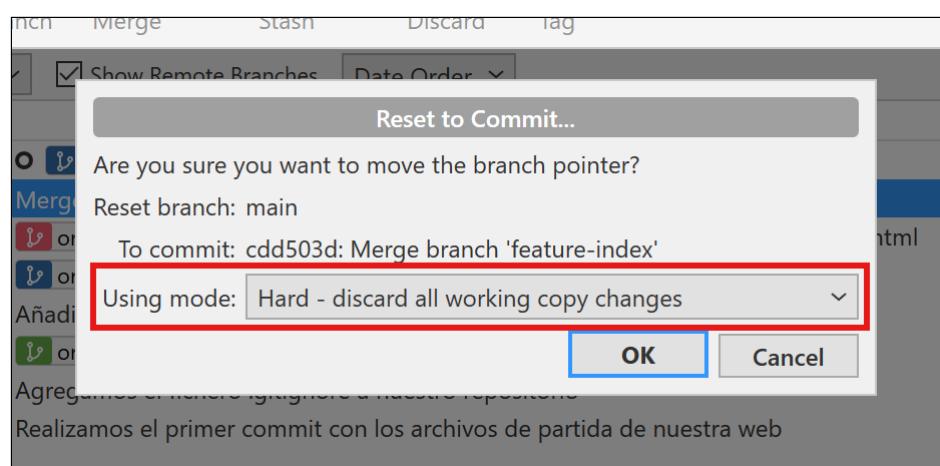
### 3.6 Cuestión 6 - Deshacer cambios antes hechos

Te das cuenta del error, así que deshaz TOTALMENTE el **commit** anterior. Captura el estado actual del repositorio y asegúrate de que el fichero **contacto.html** ha recuperado las líneas borradas y no hay cambios pendientes en el repositorio.

Para deshacer el **commit**, nos situamos en el punto del repositorio al que queremos volver (el **commit** anterior en nuestro caso) y, haciendo clic con el botón derecho sobre el mismo, seleccionamos la opción **Reset current branch to this commit**:



En la ventana emergente que aparece, escogemos la opción **Hard - discard all working copy changes** en el desplegable **Using mode** para que el archivo `contacto.html` recupere su estado original:



Confirmamos el mensaje que nos advierte que se descartarán todos los cambios que había en el directorio de trabajo y verificamos que el archivo `contacto.html` ha vuelto a su estado original y que no hay cambios pendientes en el repositorio:

The screenshot shows the GitHub desktop application interface. The left sidebar includes sections for WORKSPACE, File Status, History (selected), Search, BRANCHES (with feature-estilos and feature-index branches), TAGS, REMOTES, and STASHES. The main area displays a timeline of commits under 'All Branches'. A specific commit is highlighted: 'main 21 Merge branch 'feature-index''. This commit has three parents: 'origin/feature-index', 'feature-index', and 'origin/main'. The commit message details the merge of 'feature-estilos' into 'main', adding a paragraph in 'index.html', adding an author to 'index', updating styles in 'feature-estilos', adding '.gitignore' to the repository, and performing the first commit with website files. Below the commit details, a file list shows 'index.html' with a preview icon. On the right, a calendar view shows days 24 through 31.

```

# estilos.css    < index.html    < contacto.html X
<> contacto.html > < html > < body >
1   <!DOCTYPE html>
2   <html lang="es">
3   <head>
4       <meta charset="UTF-8" />
5       <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6       <title>Contacto - Labowebs</title>
7       <link rel="stylesheet" href="css/estilos.css" />
8       <script src="js/script.js" defer></script>
9   </head>
10  <body>
11      <header>
12          <h1>Contáctanos</h1>
13      </header>
14
15      <section class="contact-form">
16          <h2>Envíanos un mensaje</h2>
17          <form id="formContacto" action="#" method="post">
18              <label for="nombre">Nombre:</label>
19              <input type="text" id="nombre" name="nombre" required>
20
21              <label for="email">Correo Electrónico:</label>
22              <input type="email" id="email" name="email" required>
23
24              <label for="mensaje">Mensaje:</label>
25              <textarea id="mensaje" name="mensaje" rows="4" required></textarea>
26
27              <button type="submit">Enviar</button>
28          </form>
29      </section>
30
31
32      <footer>
33          <p>© 2024 Labowebs - Gijón, Asturias</p>
34          <nav>
35              <a href="index.html">Inicio</a> | <a href="contacto.html">Contacto</a>
36          </nav>
37      </footer>
38  </body>
39 </html>
40

```

### 3.7 Cuestión 7 - Creación de otra rama

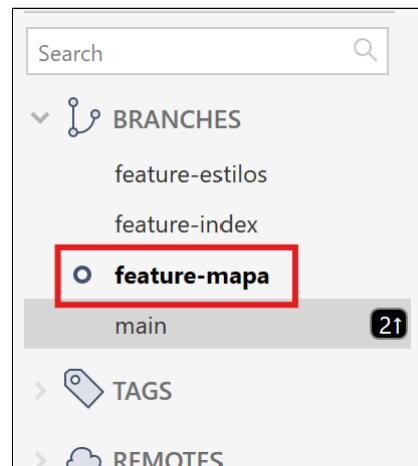
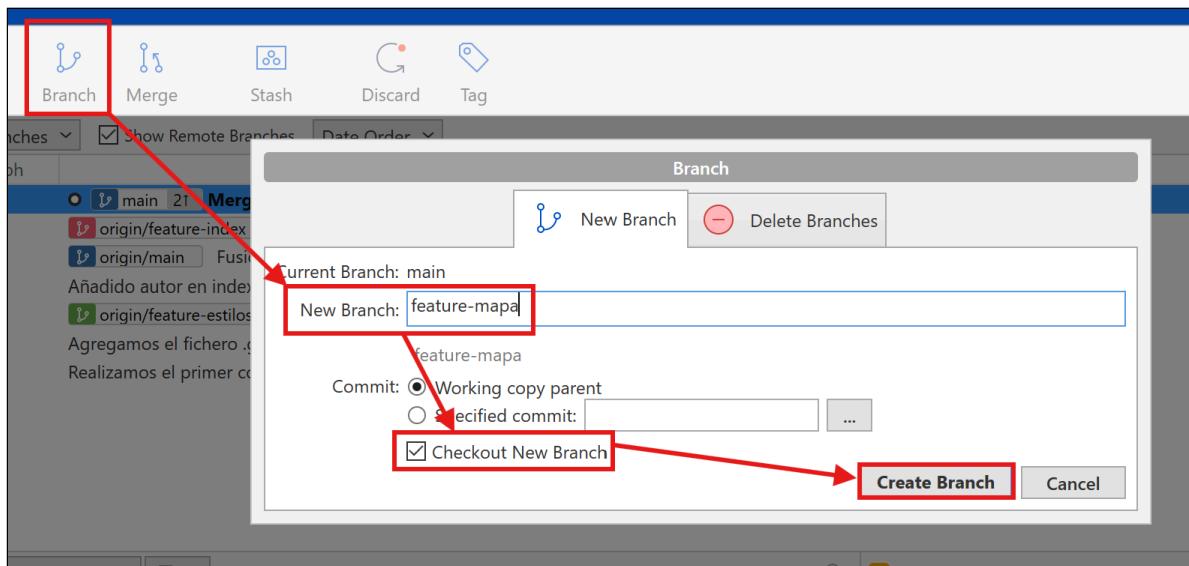
Crea una rama `feature-mapa` y cámbiate a ella. Incluye este código en el archivo `contacto.html`. Añade los cambios y haz un `commit`.

```

<section class="map">
    <h2>Nuestra Ubicación</h2>
    <p>C/Luis Moya 335, Gijón, Asturias</p>
    <iframe src="https://www.google.com/maps?
q=C%2FLuis%20Moya%20335%2C%20Gij%C3%B3n%2C%20Asturias&output=embed"
width="600" height="450" style="border:0;" allowfullscreen=""
loading="lazy"></iframe>
</section>

```

Los pasos para crear la rama son los mismos que ya hemos visto con anterioridad:



Vemos que es la rama activa, así que incluimos el código en el archivo indicado:

```

# estilos.css index.html contacto.html M
contacto.html > html > body > section.map
  2   <html lang="es">
  10  <body>
  15    <section class="contact-form">
  17      <form id="formContacto" action="#" method="post">
  24        <label for="mensaje">Mensaje:</label>
  25        <textarea id="mensaje" name="mensaje" rows="4" required></textarea>
  26
  27        <button type="submit">Enviar</button>
  28      </form>
  29    </section>
  30
  31    <section class="map">
  32      <h2>Nuestra Ubicación</h2>
  33      <p>C/Luis Moya 335, Gijón, Asturias</p>
  34      <iframe src="https://www.google.com/maps?q=C%2FLuis%20Moya%20335%2C%20Gij%C3%B3n%2C%20Asturias&output=embed"
  35          width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy">
  36    </iframe>
  37  </section>
  38
  39  <footer>
  40    <p>© 2024 Laboweb - Gijón, Asturias</p>
  41    <nav>
  42      <a href="index.html">Inicio</a> | <a href="contacto.html">Contacto</a>
  43    </nav>
  44  </footer>
  45 </body>
  46 </html>

```

Añadimos los cambios a la staging area y hacemos el commit correspondiente:

Pending files, sorted by file status

Staged files (0 files)

Unstage All Unstage Selected

contacto.html

Hunk 1 : Lines 28-40

```

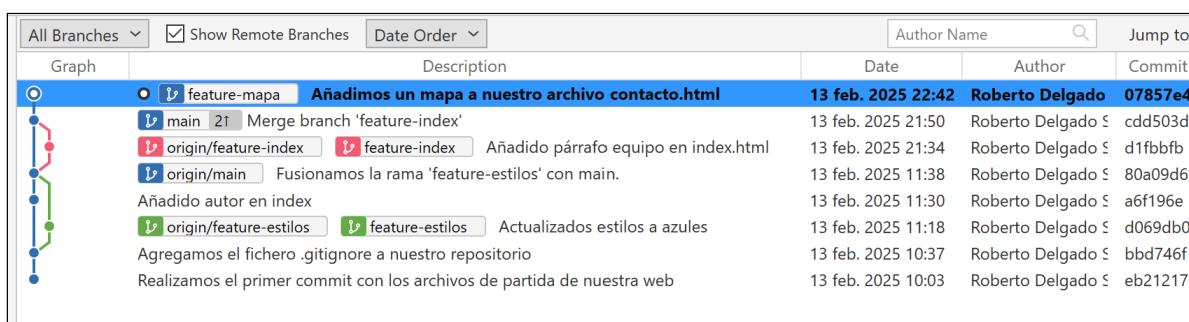
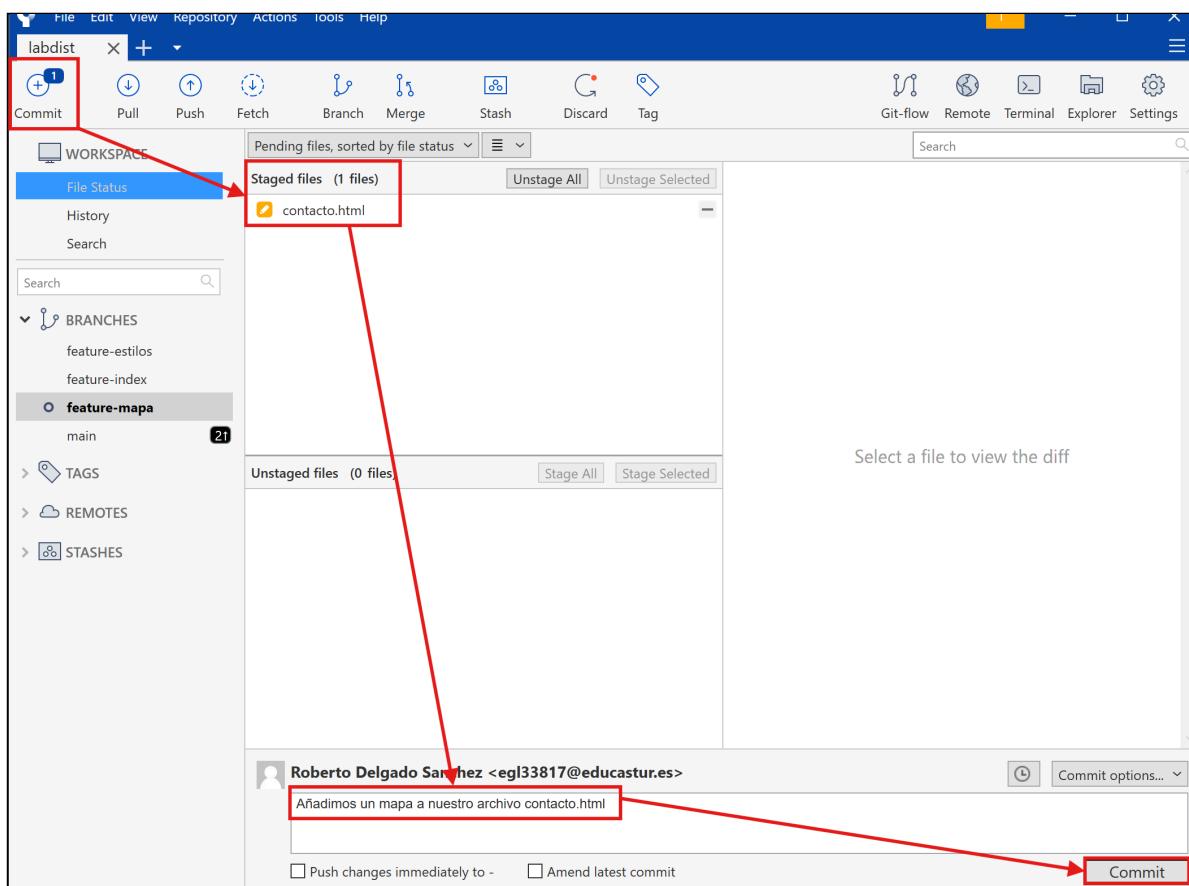
28 28 ..... </form>
29 29 ..... </section>
30 30 + <section class="map">
31 31 + <h2>Nuestra Ubicación</h2>
32 32 + <sect><Luis Moya 335, Gijón, Asturias</p>
33 33 + <iframe src="https://www.google.com/maps?q=CX2FLuis%20Moya%20335,Gij%C3%B3n,Asturias" width="600" height="450" style="border:0;" allowfullscreen></iframe>
34 34 + </section>
35 35 ..... <footer>
36 36 ..... <p>© 2024 Laboweb - Gijón, Asturias</p>
37 37
38 38
39 39
40 40

```

Unstaged files (1 files)

Stage All Stage Selected

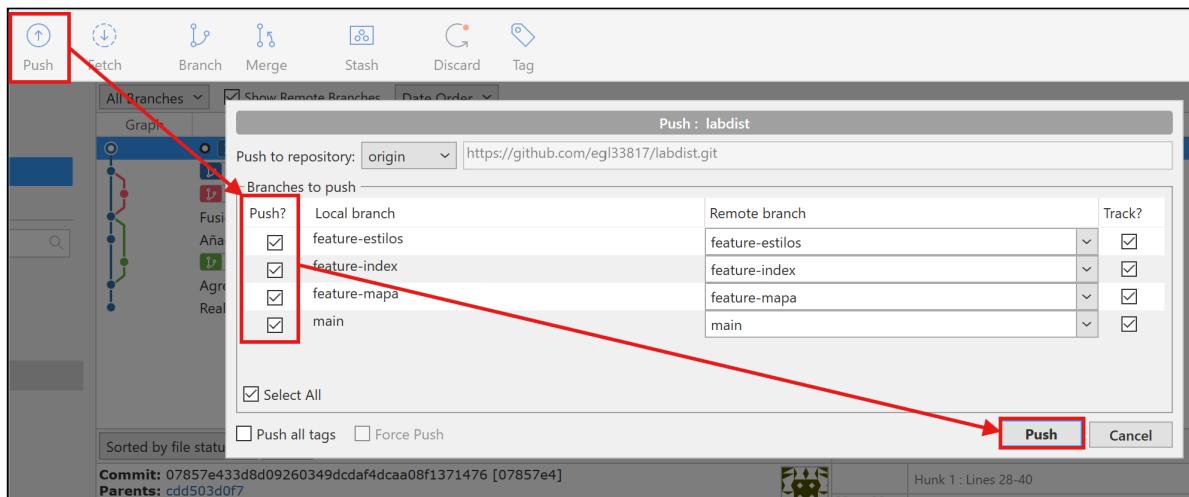
contacto.html



### 3.8 Cuestión 8 - Subir los cambios de todas las ramas al remoto

Sube los cambios de todas las ramas al repositorio remoto y, a continuación, muestra en GitHub los cambios del archivo **contacto.html** en la rama **feature-mapa**.

Como ya hemos hecho con anterioridad, hacemos un **Push** de los cambios subiendo todas las ramas:



Luego comprobamos en **GitHub** los cambios que ha sufrido el archivo `contacto.html` en la rama

`feature-mapa` :

File	Commit Message	Time Ago
contacto.html	Añadimos un mapa a nuestro archivo contacto.html	6 minutes ago
index.html	Añadido párrafo equipo en index.html	1 hour ago
.gitignore	Agregamos el fichero .gitignore a nuestro repositorio	12 hours ago
js	Realizamos el primer commit con los archivos de partida de ...	12 hours ago
css	Actualizados estilos a azules	11 hours ago

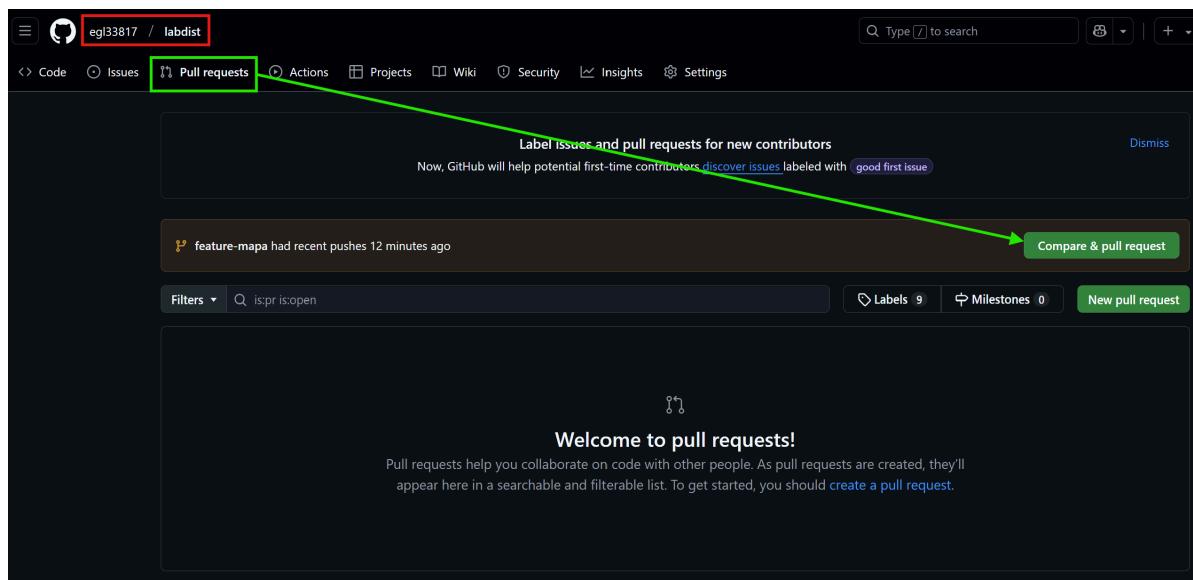
The screenshot shows a GitHub commit page for a repository named 'egl33817 / labdist'. The commit hash is '07857e4'. A red box highlights the commit message: 'Añadimos un mapa a nuestro archivo contacto.html'. Another red box highlights the file 'contacto.html' in the 'Filter files' dropdown. A red arrow points from the commit message to the file name. The commit details show '1 file changed +7 -0 lines changed'. The diff view shows the addition of a map section to the 'contacto.html' file. A red box highlights the added code block:

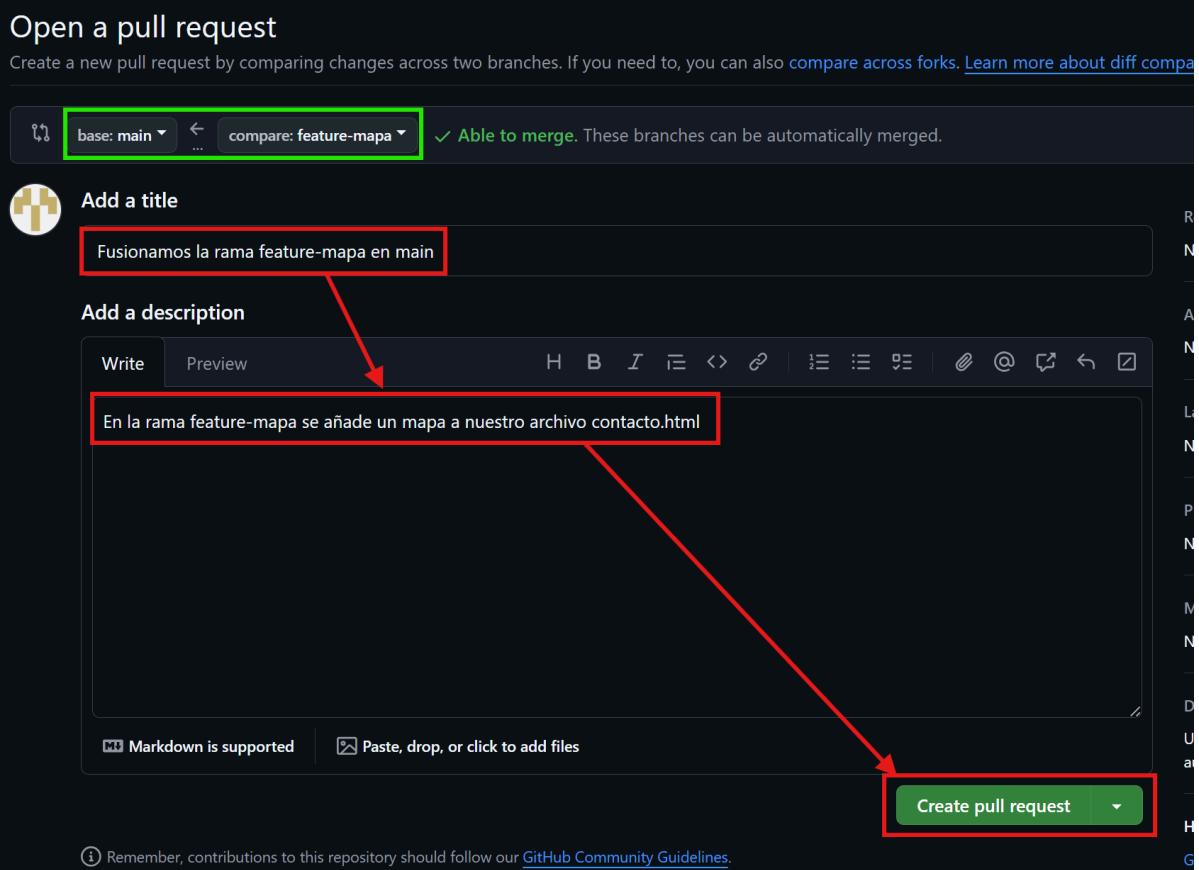
```
@@ -28,6 +28,13 @@ <h2>Envianos un mensaje</h2>
28   28   <form>
29   29   </section>
30   30
31 + <section class="map">
32 +   <h2>Nuestra Ubicación</h2>
33 +   <sect>C/Luis Moya 335, Gijón, Asturias</p>
34 +   <iframe src="https://www.google.com/maps?q=C%2FLuis%20Moya%20335%2C%20Gij%C3%B3n%2C%20Asturias&output=embed" width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy">
35 +   </iframe>
36 + </section>
37 + <footer>
38   39   <p>© 2024 Labwebs - Gijón, Asturias</p>
39   40
40   41
```

### 3.9 Cuestión 9 - Regreso a main y fusión con la última rama creada

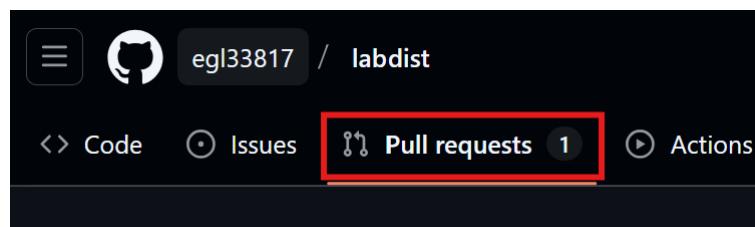
En GitHub, en la rama **main**, fusiona la rama **feature-mapa**. Baja los cambios del remoto al repositorio local. Deja los dos repositorios sincronizados y muestra una captura donde se ve la página principal de tu repositorio remoto.

Para hacer esa fusión, nos vamos a la pestaña **Pull requests** y hacemos un **pull request** de los cambios introducidos por la rama **feature-mapa**, tal y como se puede ver a continuación:

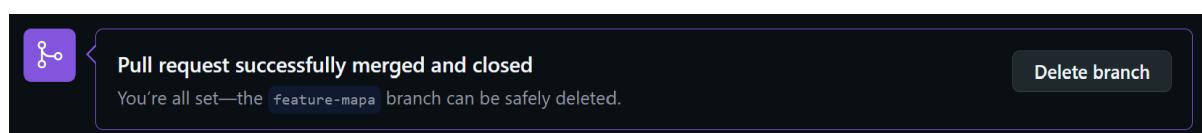
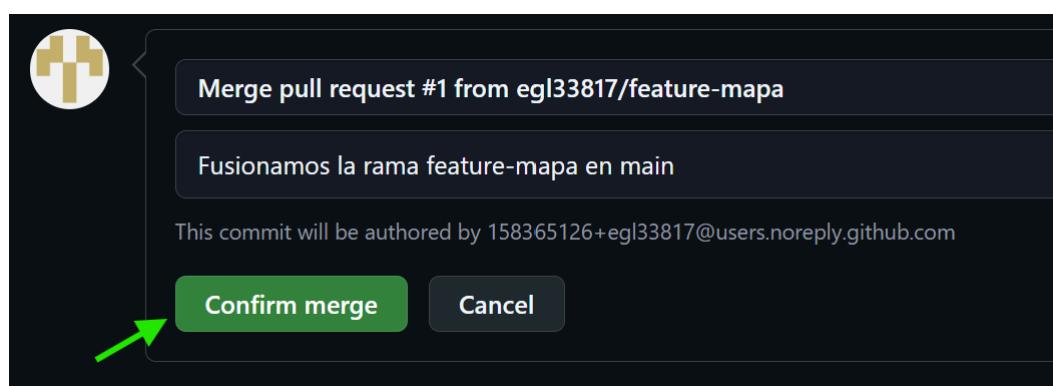




Nos aparece un mensaje indicando que tenemos una **pull request** pendiente de revisión:



Pulsamos en **Merge pull request** y ya estarían fusionadas las ramas **main** y **feature-mapa**:



Actualizados estilos a azules

Realizamos el primer commit con los archivos de partida de ...

Agregamos el fichero .gitignore a nuestro repositorio

Añadimos un mapa a nuestro archivo contacto.html

Añadido párrafo equipo en index.html

Para dejar los dos repositorios sincronizados, me voy a **SourceTree**, cambio la rama activa a **main** y hago un **Pull** para bajar los cambios desde el repositorio remoto alojado en **GitHub**:

**Pull**

**main**

**Pull**

**All Branches**

**Graph**

**main** Merge branch 'main' of https://github.com/egl33817/labdist

**origin/main** Merge pull request #1 from egl33817/feature-mapa

**origin/feature-mapa** feature-mapa Añadimos un mapa a nuestro archivo contacto.html

Merge branch 'feature-index'

**origin/feature-index** **feature-index** Añadido párrafo equipo en index.html

Fusionamos la rama 'feature-estilos' con main.

Añadido autor en index

**origin/feature-estilos** **feature-estilos** Actualizados estilos a azules

Agregamos el fichero .gitignore a nuestro repositorio

Realizamos el primer commit con los archivos de partida de nuestra web

**contacto.html**

## 4. Conflictos

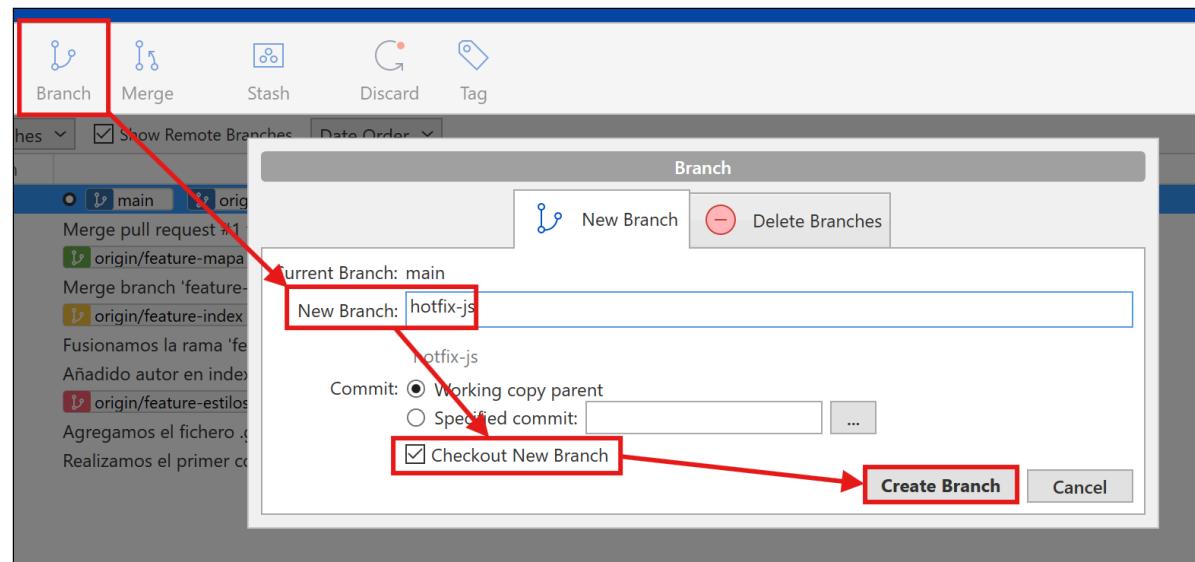
Esta parte se realizará con la herramienta gráfica **SourceTree** y **GitHub**.

#### 4.1 Cuestión 1 - Crear una nueva rama y añadir código a un fichero

Crea una rama `hotfix.js`. Cámbiate a ella. Añade este código en el fichero `script.js`. Confirma el cambio y haz un `commit` con el mensaje "Corregido problema en script.js" (fíjate en los números de línea de tu editor).

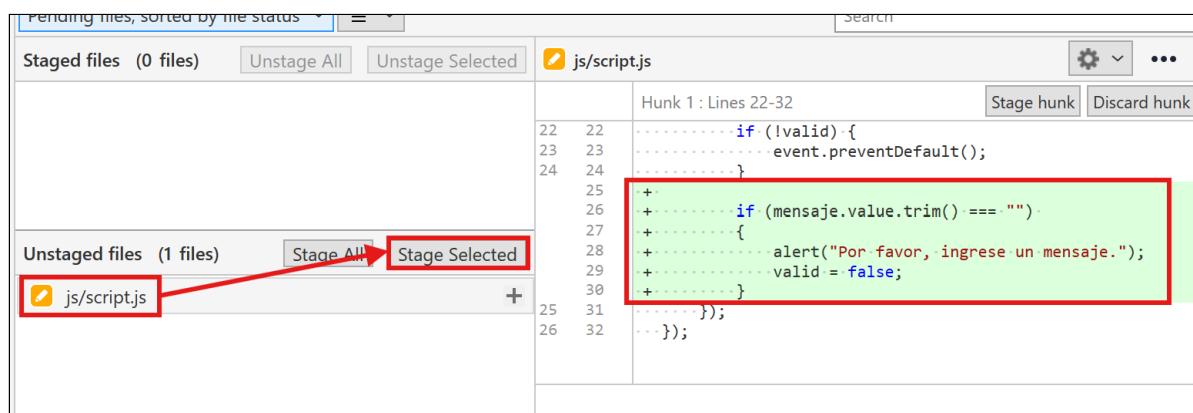
```
if (mensaje.value.trim() === "")  
{  
    alert("Por favor, ingrese un mensaje.");  
    valid = false;  
}
```

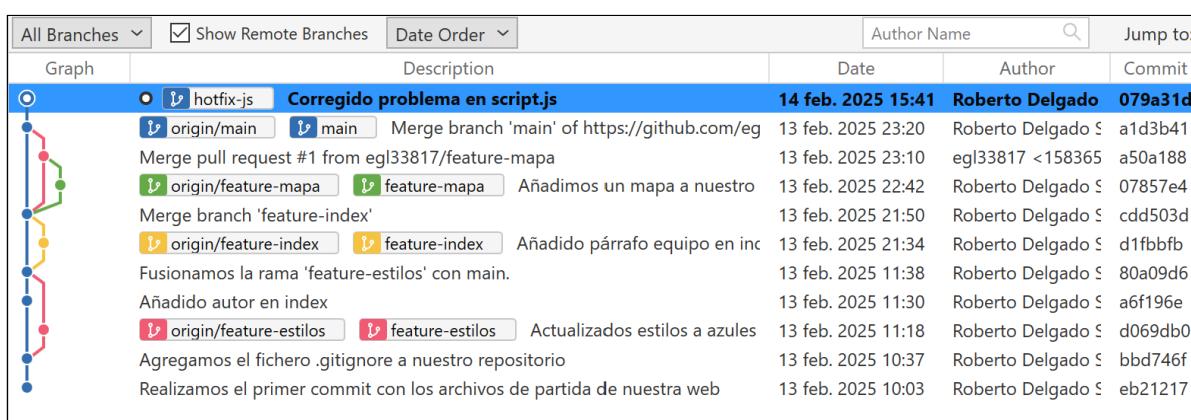
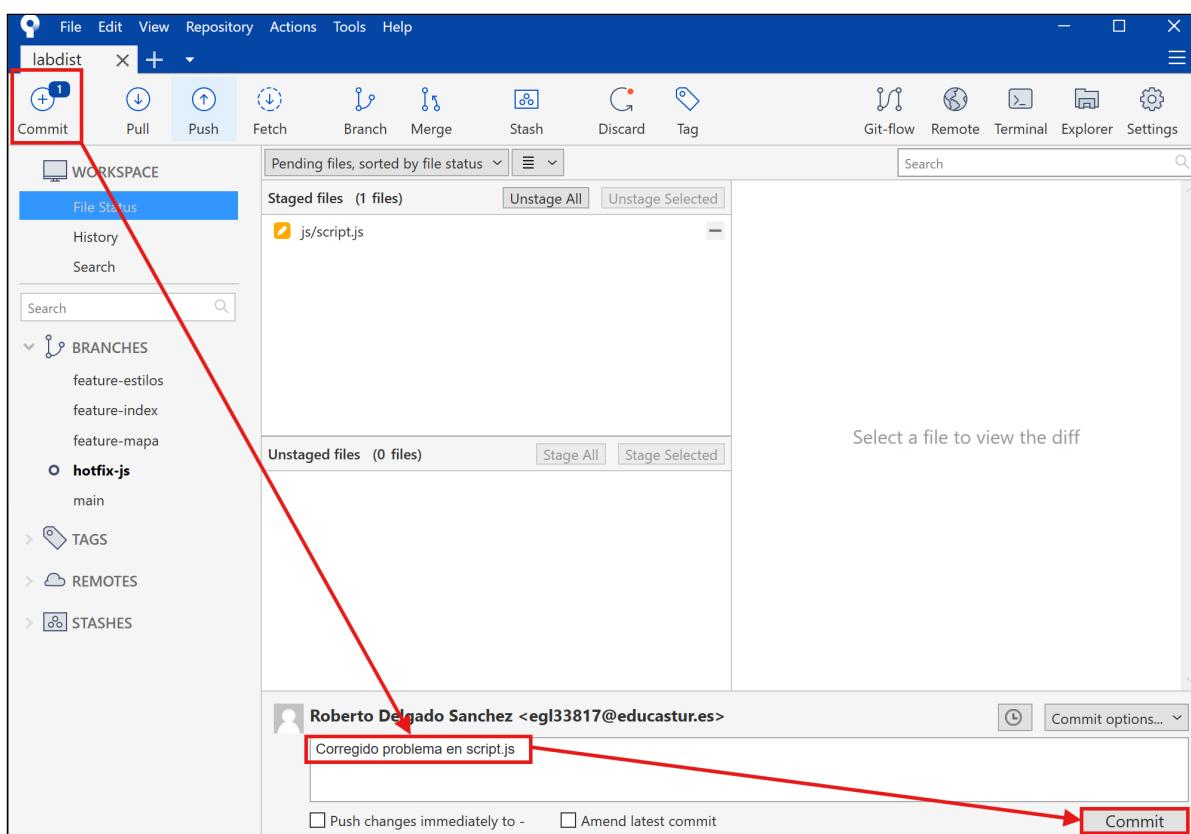
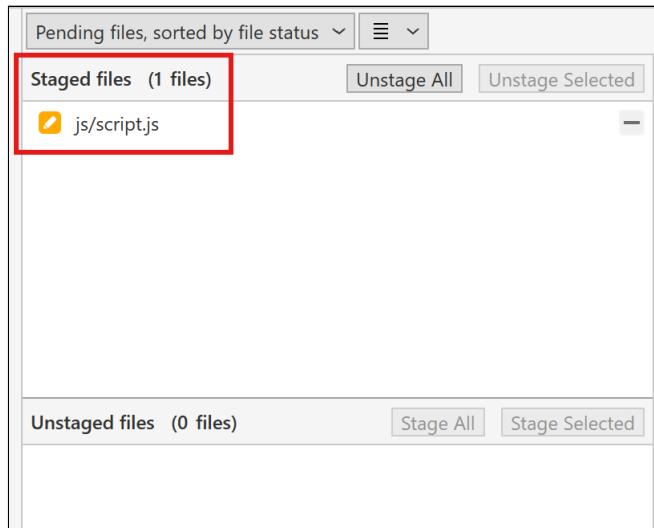
Creamos la rama `hotfix-js`, nos cambiamos a ella y añadimos en `script.js` el código facilitado:



```
JS script.js M X
js > JS script.js > ...
1  document.addEventListener("DOMContentLoaded", function() {
2    const form = document.getElementById("formContacto");
3
4    form.addEventListener("submit", function(event) {
5      let valid = true;
6
7      const nombre = document.getElementById("nombre");
8      const email = document.getElementById("email");
9      const mensaje = document.getElementById("mensaje");
10
11      if (nombre.value.trim() === "") {
12        alert("Por favor, ingrese su nombre.");
13        valid = false;
14      }
15
16      if (!/^\w-[\w-\.]+@[.\w-]+\.\w{2,4}$/.test(email.value)) {
17        alert("Por favor, ingrese un correo electrónico válido.");
18        valid = false;
19      }
20
21      if (!valid) {
22        event.preventDefault();
23      }
24
25      if (mensaje.value.trim() === "") {
26        alert("Por favor, ingrese un mensaje.");
27        valid = false;
28      }
29    });
30  });
31 });
32 });
33 }
```

Confirmamos el cambio (es decir, añadimos el archivo a la `staging area`) y hacemos un `commit` con el mensaje que figura en el enunciado:



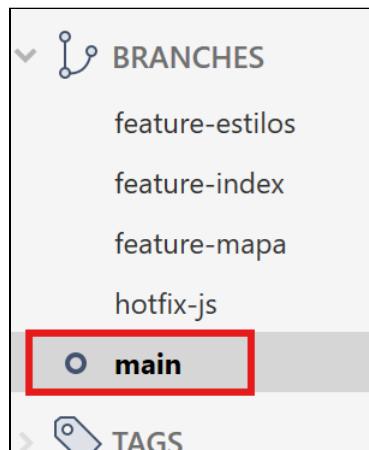


## 4.2 Cuestión 2 - Volver a main y modificar ese mismo fichero

Vuelve a la rama `main`. En el fichero `script.js` en las mismas líneas que en la cuestión anterior, añade el código siguiente. Confirma el cambio y haz un `commit` con el mensaje "corregido problema en `script.js` rama `main`".

```
if (mensaje.value.trim() === "")  
{  
    alert("Ingrese un mensaje, por favor");  
    valid = false;  
}
```

Volvemos a la rama `main`:



Añadimos el código facilitado al fichero `script.js`, haciendo que su ubicación coincida con la de las líneas que añadimos en este mismo fichero cuando estábamos en la rama `hotfix-js`:

JS script.js M X

```

js > JS script.js > ⚡ document.addEventListener("DOMContentLoaded") callback > ⚡ form.addEventListener("submit", function(event) {
  1   document.addEventListener("DOMContentLoaded", function() {
  2     const form = document.getElementById("formContacto");
  3
  4     form.addEventListener("submit", function(event) {
  5       let valid = true;
  6
  7       const nombre = document.getElementById("nombre");
  8       const email = document.getElementById("email");
  9       const mensaje = document.getElementById("mensaje");
 10
 11      if (nombre.value.trim() === "") {
 12        alert("Por favor, ingrese su nombre.");
 13        valid = false;
 14      }
 15
 16      if (!/^\w-\w+\w+@\w+\.\w{2,4}$/.test(email.value)) {
 17        alert("Por favor, ingrese un correo electrónico válido.");
 18        valid = false;
 19      }
 20
 21      if (!valid) {
 22        event.preventDefault();
 23      }
 24    })
 25
 26    if (mensaje.value.trim() === "") {
 27      alert("Ingrese un mensaje, por favor");
 28      valid = false;
 29    }
 30  });
 31 });
 32 });
 33 
```

Confirmamos el cambio subiendo el archivo a la **staging area** y hacemos el **commit** correspondiente:

Pending files, sorted by file status ▾

Staged files (0 files)

Unstage All Unstage Selected

Unstaged files (1 files)

**js/script.js**

Stage All Stage Selected

Search

Hunk 1 : Lines 22-32

22 22 if (!valid) {  
23 23 ..... event.preventDefault();  
24 24 ..... }  
25 +  
26 + if (mensaje.value.trim() === "") {  
27 + ..... alert("Ingrese un mensaje, por favor");  
28 + ..... valid = false;  
29 + ..... }  
30 31 };  
31 32 ...});

Pending files, sorted by file status ▾

**Staged files (1 files)**

Unstage All Unstage Selected

js/script.js

The screenshot shows the GitHub Desktop interface. In the top left, there's a red box around the 'Commit' button in the toolbar. Below the toolbar, the 'WORKSPACE' sidebar has 'main' selected. In the center, the 'Pending files' section shows 'js/script.js' with a yellow icon. The 'Unstaged files' section is empty. At the bottom, the commit message is 'Corregido problema en script.js rama main'. Two checkboxes are at the bottom: 'Push changes immediately to origin/main' and 'Amend latest commit'. A red arrow points from the commit message area to the 'Commit' button. To the right, a large table titled 'Graph' shows the commit history with a red box around the first commit message.

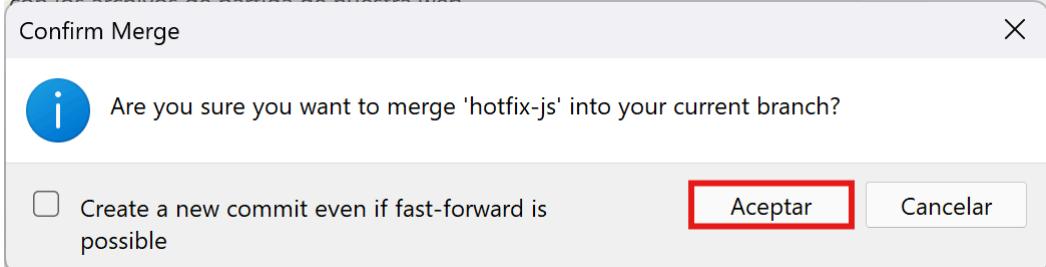
Description	Date	Author	Commit
main 11 Corregido problema en script.js rama main	14 feb. 2025 16:01	Roberto Delgado	49e4bd1
hotfix-js Corregido problema en script.js	14 feb. 2025 15:41	Roberto Delgado	079a31d
origin/main Merge branch 'main' of https://github.com/egl33817/labdist	13 feb. 2025 23:20	Roberto Delgado	a1d3b41
Merge pull request #1 from egl33817/feature-mapa	13 feb. 2025 23:10	egl33817 <158365	a50a188
origin/feature-mapa feature-mapa Añadimos un mapa a nuestro archivo contact	13 feb. 2025 22:42	Roberto Delgado	07857e4
Merge branch 'feature-index'	13 feb. 2025 21:50	Roberto Delgado	cdd503d
origin/feature-index feature-index Añadido párrafo equipo en index.html	13 feb. 2025 21:34	Roberto Delgado	d1fbfbf
Fusionamos la rama 'feature-estilos' con main.	13 feb. 2025 21:38	Roberto Delgado	80a09d6
Añadido autor en index	13 feb. 2025 11:30	Roberto Delgado	a6f196e
origin/feature-estilos feature-estilos Actualizados estilos a azules	13 feb. 2025 11:18	Roberto Delgado	d069db0
Agregamos el fichero .gitignore a nuestro repositorio	13 feb. 2025 10:37	Roberto Delgado	bbd746f
Realizamos el primer commit con los archivos de partida de nuestra web	13 feb. 2025 10:03	Roberto Delgado	eb21217

#### 4.3 Cuestión 3 - Fusionar ambas ramas y resolver el conflicto

Fusiona la rama **hotfix-js** en **main**. Debe producirse un conflicto. Resuélvelo como consideres oportuno. Cuando termines la resolución del conflicto sube los cambios al remoto.

Estando seleccionada como rama activa **main**, hago clic con el botón derecho sobre la rama **hotfix-js** y escojo la opción **Merge hotfix.js into current branch**:

The screenshot shows the GitHub Desktop interface with a context menu open over the 'hotfix-js' branch in the 'BRANCHES' sidebar. The menu items are: 'Checkout hotfix-js...', 'Merge hotfix-js into current branch' (which is highlighted with a red box), 'Rebase current changes onto hotfix-js', 'Fetch hotfix-js', 'Pull (tracked)', 'Push to (tracked)', and 'Push to'. A red arrow points from the 'Merge hotfix-js into current branch' option to the menu itself.

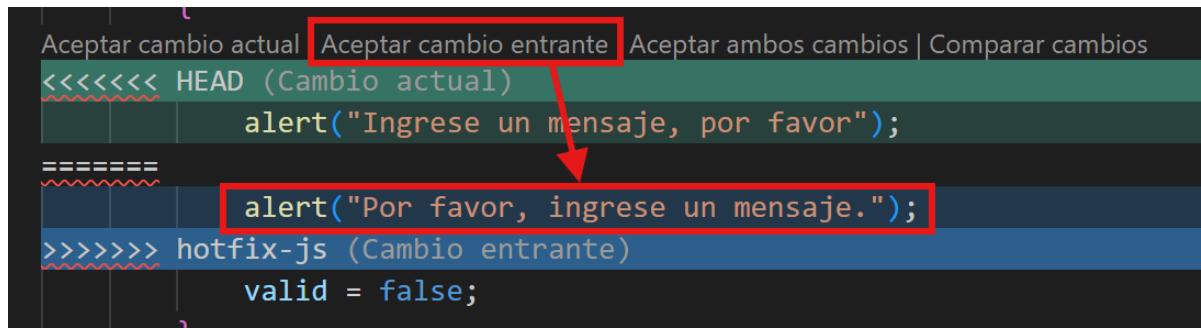


Una vez aceptada la confirmación de que queremos fusionar ambas ramas, nos aparece un mensaje advirtiéndonos de la existencia de un conflicto. Si voy a **Visual Studio Code** puedo ver marcadas las líneas que generan dicho conflicto, el cual debo resolver manualmente, ya que el sistema de control de versiones no puede saber cuál es la mejor opción.

En mi caso, me quedo con la versión que habíamos puesto en la rama **hotfix-js**:

```
JS script.js 3, ! X
js > JS script.js > ...
1  document.addEventListener("DOMContentLoaded", function() {
2      const form = document.getElementById("formContacto");
3
4      form.addEventListener("submit", function(event) {
5          let valid = true;
6
7          const nombre = document.getElementById("nombre");
8          const email = document.getElementById("email");
9          const mensaje = document.getElementById("mensaje");
10
11         if (nombre.value.trim() === "") {
12             alert("Por favor, ingrese su nombre.");
13             valid = false;
14         }
15
16         if (!/^\w-\.\w-@([\w-]+\.)+[\w-]{2,4}\$/.test(email.value)) {
17             alert("Por favor, ingrese un correo electrónico válido.");
18             valid = false;
19         }
20
21         if (!valid) {
22             event.preventDefault();
23         }
24
25         if (mensaje.value.trim() === "")
26         {
27             Aceptar cambio actual | Aceptar cambio entrante | Aceptar ambos cambios | Comparar cambios
28             <<<<< HEAD (Cambio actual)
29             | alert("Ingrese un mensaje, por favor");
30             =====
31             | alert("Por favor, ingrese un mensaje.");
32             >>>>> hotfix-js (Cambio entrante)
33             | valid = false;
34         }
35     });
36 });
37 }
```

Hago clic en **Agregar cambio entrante** para que se quede el código que aporta la rama **hotfix-js**:

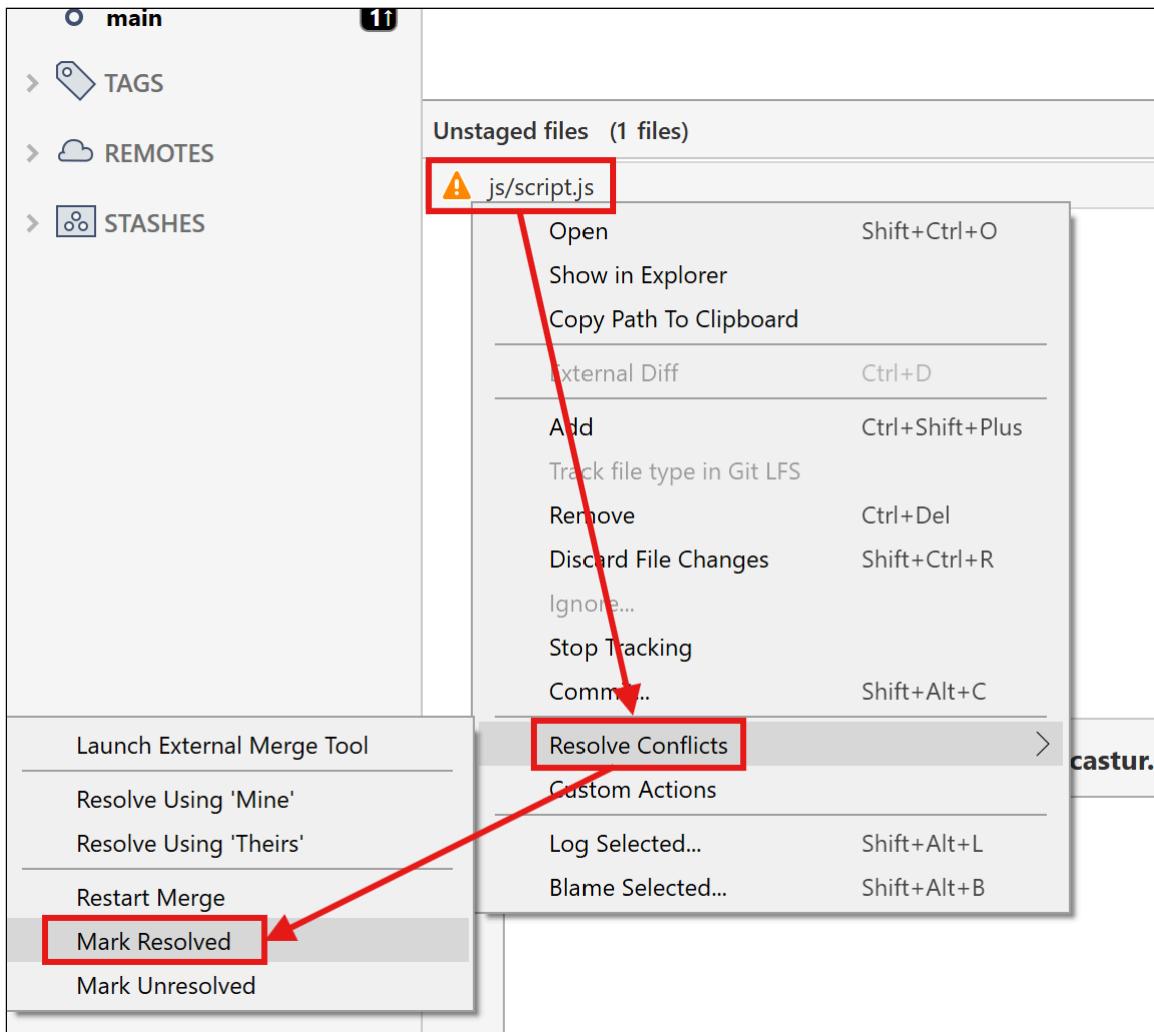


```
Aceptar cambio actual | Aceptar cambio entrante | Aceptar ambos cambios | Comparar cambios
<<<<< HEAD (Cambio actual)
alert("Ingrese un mensaje, por favor");
=====
===== alert("Por favor, ingrese un mensaje.");
>>>>> hotfix-js (Cambio entrante)
valid = false;
```

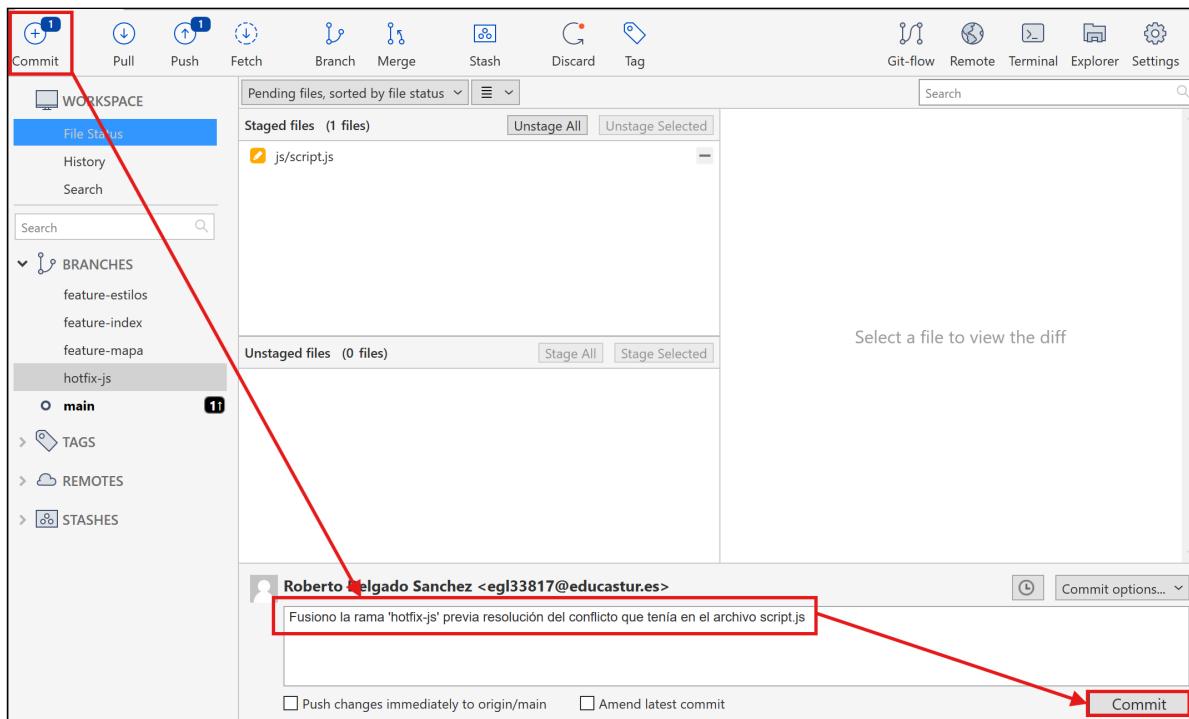


```
JS script.js ! X
js > JS script.js > ...
1  document.addEventListener("DOMContentLoaded", function() {
4    form.addEventListener("submit", function(event) {
21
22      if (!valid) {
23        event.preventDefault();
24      }
25
26      if (mensaje.value.trim() === "") {
27        alert("Por favor, ingrese un mensaje.");
28        valid = false;
29      }
30    },
31  });
32
33
```

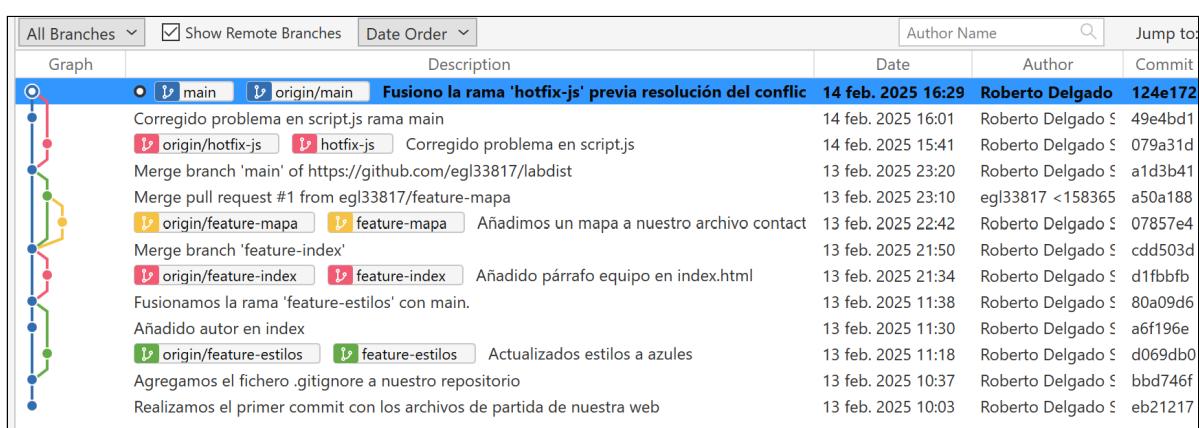
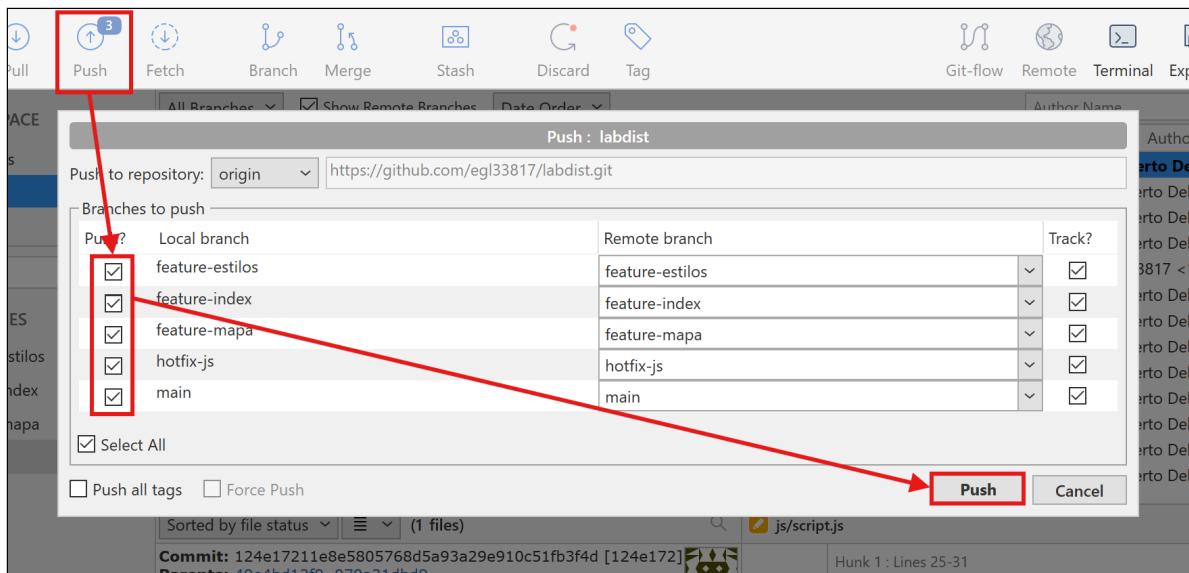
Una vez resuelto el conflicto en el **IDE**, lo marco como tal en **SourceTree** haciendo clic con el botón derecho sobre el nombre del archivo:



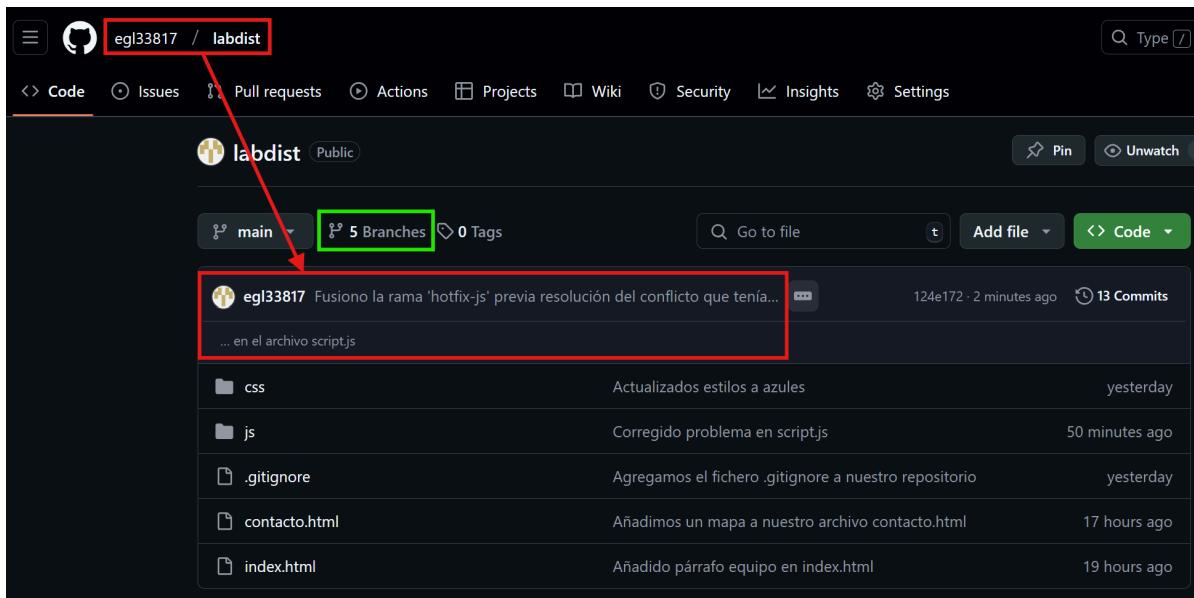
Hacemos el **commit** correspondiente:



Y subo los cambios al repositorio remoto:



Hago la comprobación en GitHub:



## 5. Subida de documentación al repositorio

En la carpeta del proyecto **labdist** añade una carpeta **docs**. Copia en esta carpeta el fichero markdown y la carpeta con las imágenes. Incluye también el PDF. Añade todo al repositorio y súbelo al remoto.

La tarea de crear la carpeta **docs** y copiar en ella el fichero markdown y la carpeta con las imágenes se hace con el explorador de archivos de Windows (el archivo no incluye el enlace al vídeo).

## 6. Enlaces de interés

### 6.1 Repositorio en GitHub

<https://github.com/egl33817/labdist>

### 6.2 Vídeo solicitado para la práctica