Práctica 8: Repositorios remotos con GitHub

Repositorios remotos

En esta práctica vamos a seguir utilizando Git, pero en este caso lo haremos teniendo el repositorio en remoto, en un hosting en la nube. Para esto existen varias plataformas gratuitas, las más comunes son GitHub y GitLab.

En nuestro caso vamos a optar por GitHub. El uso de esta plataforma está tan extendido que a menudo se equipara Git con GitHub, tratándolos como lo mismo. Es importante dejar claro que Git es el SCV, el programa que permite hacer el seguimiento de los cambios, mientras que GitHub es una plataforma que utiliza Git y facilita el hosting gratuito de repositorios de este tipo.

Como veremos, los comandos y operaciones empleados para gestionar el repositorio son los mismo que utilizábamos en local, al emplear también Git. Sin embargo, al trabajar en remoto, se añaden dos opciones nuevas al ciclo de uso: principalmente push y pull.

Con **push** subimos cambios al repositorio remoto. Básicamente, una vez hemos hecho commits en nuestro repositorio local, tenemos que hacer push para trasladarlos al remoto.

Pull, por el contrario, sirve para actualizar nuestra copia local a la última versión que haya en remoto.

Como es de esperar, al hacer estas operaciones podemos encontrarnos con conflictos, especialmente al trabajar de forma colaborativa. Estos habrá que solucionarlos para poder hacer merge de las distintas versiones.

Por esto, el ciclo de trabajo recomendable es hacer pull antes de empezar a trabajar, para partir de la última versión, y hacer push una vez acabemos, asegurándonos siempre de que el proyecto sigue funcionando correctamente antes de subir nuestros cambios.

Parte 1 – Acceder a GitHub

Ahora vamos a trabajar de manera colaborativa utilizando un repositorio central alojado en GitHub. En este caso, la práctica tendrá que hacerse en pareja. En el resto del guion se hablará de EstudianteA y EstudianteB para referirse a cada uno de los dos miembros del grupo. Cada uno necesitaréis una cuenta en GitHub. Para ello, vamos a crear un repositorio nuevo distinto del de la práctica anterior.

A lo largo de esta práctica, iremos trabajando sobre el repositorio con comandos de Git desde Git Bash y utilizando aplicaciones de escritorio. Cuando se indique, adjunta el comando introducido y su salida con una captura de pantalla. Para ello, estructura el documento que entregues en los apartados del enunciado, señalando las partes de EstudianteA y EstudianteB.

Parte 2 – Git como repositorio remoto

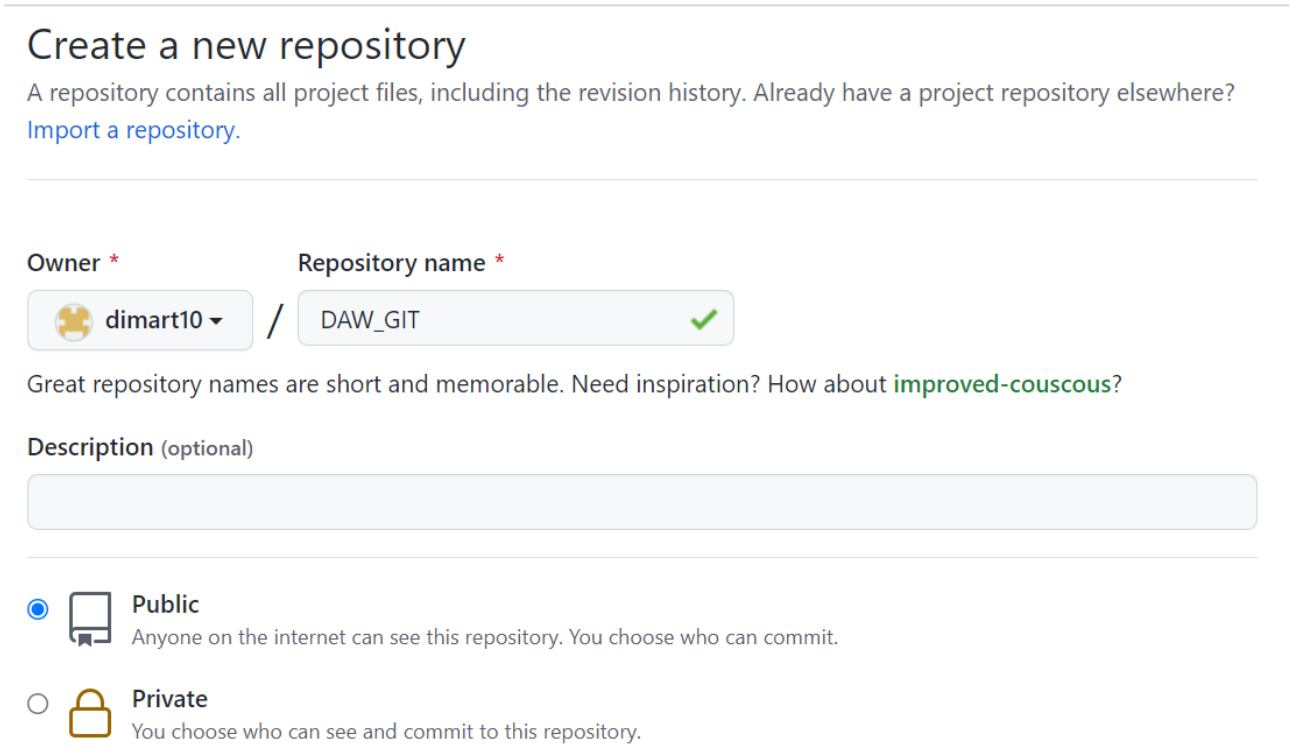
Inicialización del repositorio

Estudiante A

Primero hay que crear el repositorio, hacemos lo mismo que en la práctica anterior: desde Git Bash configuramos nombre y email, inicializamos el repositorio con init y hacemos un primer commit.

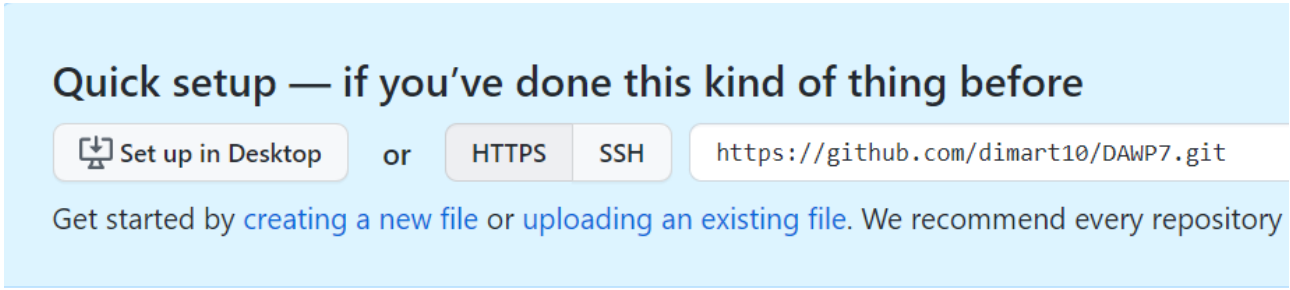
Una vez creado, tenemos que enlazar el repositorio con GitHub, para esto se utiliza el comando git remote add [url]. Sin embargo, para poder enlazarlo a GitHub, primero tenemos que crear el repositorio remoto también en GitHub. Para ello, estando logeados en github.com elegimos la opción de añadir un nuevo repositorio. Al repositorio le tendremos que dar un nombre, que será el que aparezca en la url de GitHub de la siguiente forma github.com/[nombre_de_usuario]/[nombre_de_repositorio].

A la hora de crearlo, disponemos de varias opciones, en primer lugar, podemos añadir una descripción que aparecerá al acceder al repositorio en la web. Lo más importante, es hacer el repositorio público (todo el mundo puede verlo y descargarlo) o privado (solo los usuarios con permisos pueden verlo o descargarlo), independientemente de si es público o privado, para subir cambios al repositorio los usuarios tendrán que estar autorizados.



Además GitHub nos facilita la inicialización con la opción de crear un README.md, un .gitignore y asignar una licencia a nuestro repositorio. De momento creamos el repositorio como público y no marcamos las otras opciones.

Una vez hecho esto, se nos mostrará información sobre qué hacer a continuación para enlazar un repositorio local ya existente con GitHub o comenzar un repositorio nuevo.



En nuestro caso, ya hemos inicializado el repositorio local, así que solo nos quedaría asociar el repositorio local con github.com. Para esto primero hacemos git remote add:

```
git remote add origin https://github.com/[usuario]/[repo].git
```

Con esto, asignamos al repositorio local esta url como origen del repositorio remoto, es decir, el lugar del que descargamos cambios y subimos los nuestros propios. Hacemos:

```
git branch -M main
```

Esto sirve para asignar la rama main para utilizarla junto con github.com. Después, quedará hacer push, es decir, subir los cambios que hemos hecho.

```
git push -u origin main
```

En este caso, con los parámetros -u origin main vamos a inicializar el repositorio remoto tomando como base el nuestro local, es decir, subir todos los cambios que hemos hecho en local hasta ahora, que se limitan a hacer el init y un primer commit.

Copia el contenido de lo que aparezca por la consola. Ejecuta también git branch -a y copia lo que aparece por consola. ¿Cuántas ramas hay ahora en tu repositorio?

EstudianteB

Ahora, vamos a descargar otra versión del repositorio remoto, otra copia local. Para esto, en otro ordenador, vamos a descargar el repositorio que creamos anteriormente con git clone:

```
git clone https://github.com/[usuario]/[repo].git
```

Copia el contenido de lo que aparezca por la consola. Una vez hecho esto, configura el nombre de usuario y el email tal cual se hizo en la práctica individual. Ejecuta también git branch -a y copia lo que aparece por consola. ¿Cuántas ramas hay ahora en tu repositorio?

Esto lo podremos hacer aunque no tengamos permisos en el repositorio, ya que lo hemos creado como público. Por el contrario, cuando hagamos algún commit desde esta copia y queramos subirlo con push, el usuario tendrá que estar autorizado en el repositorio. Para esto, el creador del repositorio tiene que añadir al otro desde la página del repositorio en github.com → settings → manage access.

Crea un archivo FrontPage.html en la carpeta raíz. Sube los cambios al índice y haz un commit (recuerda, con un mensaje significativo). Crea un archivo llamado .gitignore como el que usaste en la práctica individual. De nuevo, añádelo al índice y haz un comité. Por último, súbelo al repositorio con git push.

Ejecuta un git log como los que has hecho en la práctica individual y copia el contenido.

Antes de seguir trabajando en el repositorio, vamos a instalar en cada equipo un programa con interfaz para la gestión de repositorios git. Cada estudiante puede instalar el que prefiera, algunos de los que comentamos en clase son: GithubDesktop, de los propietarios de GitHub y de código abierto, GitKraken, con una interfaz muy buena o SourceTree. Existen muchos más, como los listados en esta página: https://git-scm.com/downloads/guis

En este punto cada uno debe instalar un programa para gestionar repositorios, para ello indica cuál has elegido y adjunta un par de capturas del proceso de instalación.

Una vez instalados, los primeros pasos serán vincular la cuenta de GitHub y el repositorio, en el caso de EstudianteA, localizando el que ya tenemos en el equipo, y en el de EstudianteB, localizando el que clonamos.

De ahora en adelante, podremos gestionar las modificaciones al repositorio desde estos programas (añadir cambios, crear commits, hacer pull y push, etc). Aunque para hacer los logs y otras comprobaciones, seguiremos utilizando Git Bash.

EstudianteA

En el otro ordenador, vamos a descargar los cambios que ha hecho EstudianteB. Para ello, actualiza tu repositorio local a la última versión del remoto con git pull (recuerda, esto descarga y mezcla los cambios del repositorio remoto con tu repositorio local). Comprueba que los archivos que añadió aparecen.

Ejecuta un git log como los que has hecho en la práctica individual y copia el contenido

Trabajando con ramas

EstudianteA

Crea una rama Login y haz un push de dicha rama a GitHub:

```
git checkout -b Login
git push -u origin Login
```

Ejecuta git branch -a y copia el contenido de la consola. ¿Cuántas ramas hay en tu copia local? ¿Cuántas ramas de seguimiento tienes?

Tal y como hicimos en la práctica individual crea, haciendo 2 commit diferentes, un archivo llamado login.html y otro llamado login.php. Termina haciendo un push de estos cambios al repositorio central.

EstudianteB

Crea una rama Register y haz un push de dicha rama a GitHub:

```
git checkout -b Register
git push -u origin Register
```

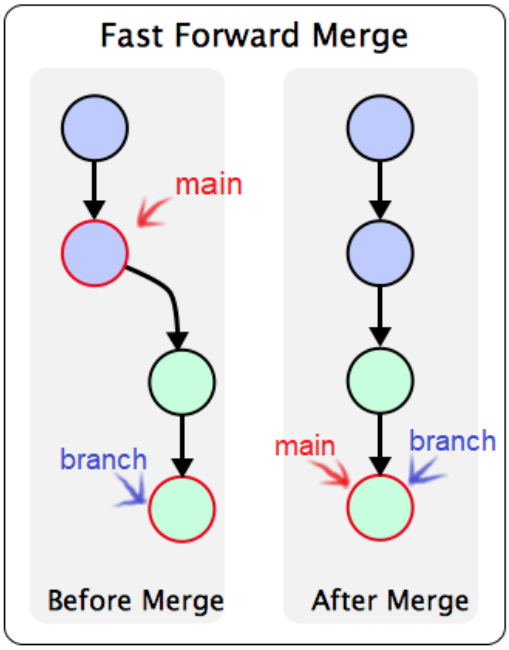
Ejecuta git branch -a y copia el contenido de la consola. ¿Cuántas ramas hay en tu copia local? ¿Cuántas ramas de seguimiento tienes?

Tal y como hicimos en la práctica individual crea, haciendo 2 commits diferentes, un archivo register.html y otro register.php..Haz un push de todos estos cambios al servidor remoto.

EstudianteA

Mezcla la rama Login en main:

```
git checkout main
git merge Login -m "Mezcla de la rama Login en main"
```



Se debería de haber producido una mezcla *fastforward*, este tipo de mezcla "adelanta" la rama en la que estamos (main) a la que hacemos el merge (Login), es decir, la actualiza. Debido a esto, no genera commits, simplemente causa que nuestra rama (main) apunte al último commit de la rama mezclada (Login).

A continuación crea un archivo README.md con el siguiente contenido:

```
# Práctica 3: Git
## Realizado por TuGrupo
- EstudianteA
- EstudianteB
## Cambios realizados
- Hemos incorporado la página de registro
```

Añade el archivo al índice, haz un commit indicando que hemos añadido el archivo README y sube todo al servidor remoto con git push. Ejecuta un log y copia el contenido de la consola.

EstudianteB

Comprueba qué cambios hay en el repositorio remoto:

```
git remote show origin
```

Copia el contenido de la consola. ¿Qué es lo que te está indicando? ¿Qué diferencias hay entre tu repositorio y el remoto?

Vamos a descargar el contenido del repositorio remoto para luego mezclarlo nosotros:

```
git fetch
```

// fetch comprueba los cambios en remoto, sin aplicarlos a la copia local como pull

```
git branch -a
git log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short --all
```

Copia el contenido de la consola. Identifica dónde están cada una de las ramas (tanto las de tu copia local como las del repositorio remoto).

Continuamos mezclando nuestro main (el del repositorio local) con el origin/main (el del repositorio remoto):

```
git checkout main
```

Observa y copia el mensaje que aparece. ¿De qué te está avisando?

```
git merge origin/main
```

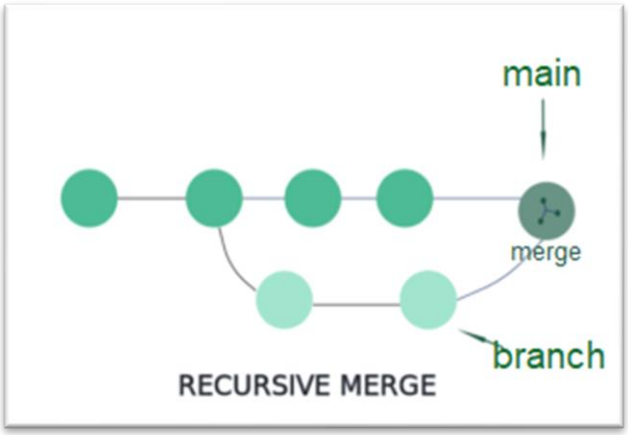
Debería haberse producido una mezcla *fastforward*. Ejecuta un log y copia lo que aparece por consola.

Ahora vamos a mezclar lo que tenemos en la rama Register con main:

```
git merge Register -m "Mezcla de la Rama Register con main"
```

Debería de haber ocurrido una mezcla recursiva, este tipo de mezcla deja un “commit de tipo merge” que indica dónde y cuándo se ha realizado, con nombre -m .

Observa los archivos para comprobar que tienes una versión en la que se han mezclado los cambios realizados en ambas ramas. Si todo es correcto, termina haciendo un push de todos estos cambios al repositorio remoto.



EstudianteA

Haz un pull del repositorio completo con git pull. Resulta que EstudianteB se ha olvidado de actualizar el archivo README así que añade al final de este la siguiente línea:

- Añadida página de registro

Añade el cambio al índice, haz un commit y súbelo al repositorio remoto.

EstudianteB

Te has dado cuenta de que se te olvidó actualizar el archivo README así que añade al final de este la siguiente línea:

- Próximamente se implementará la página de recuperación de credenciales

Añádelo al índice y haz un commit.

Haz un push y copia lo que aparece por consola. ¿Qué ha ocurrido? ¿Has podido completar el push?

Ejecuta git pull y copia el contenido de la consola. ¿Qué ha ocurrido?

Ya que ha habido un conflicto en el archivo README, hay que modificarlo para solucionarlo, para esto se puede editar directamente el archivo o utilizar las aplicaciones de gestión que hemos instalado.

Una vez el archivo esté como queremos, basta con marcar el conflicto como resuelto. Para ello, hay que añadir las modificaciones al archivo al índice y hacer un commit.

Añádelo al índice y ejecuta un git status para comprobar que todos los conflictos han sido resueltos. Copia el contenido de la consola.

Finalmente, haz un commit y un push para subir todo al repositorio remoto.

Ejecuta el log siguiente y copia el contenido de la consola:

```
git log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short –all
```

EstudianteA

Haz un pull de lo que hay en el repositorio remoto y comprueba que todo está correcto. Una vez hecho esto, crea un tag:

```
git tag –a v1.0 –m "Versión completa con los cambios de ambos"
```

Ejecuta el log siguiente y copia el contenido de la consola:

```
git log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short --all
```

Finalmente, súbelo al repositorio remoto:

```
git push origin v1.0
```

Crea un zip con el contenido de la copia local (asegúrate de incluir el directorio .git).

EstudianteB

Ejecuta un git pull. Ejecuta un log para comprobar que se ha descargado el tag del repositorio y copia el contenido de la consola.

Crea un zip con el contenido de la copia local.

Conclusión

Con esta práctica hemos trasladado los conceptos y comandos de Git que trabajamos en la anterior práctica de forma local al entorno de un repositorio remoto. Este repositorio los hemos desplegado desde GitHub, aunque existen otras alternativas también gratuitas como GitLab. Además, se ha probado el uso de aplicaciones de escritorio para la gestión de repositorios de Git.

Esta forma de uso de un SCV es de las más habituales en proyectos de software, siendo Git el sistema de referencia para el control de versiones y el portal GitHub el más extendido tanto a la hora de llevar a cabo proyectos colaborativos como de publicar y compartir estos desarrollos. El uso de aplicaciones de escritorio para la gestión también es muy habitual, para facilitar su uso, especialmente a la hora de gestionar cambios y conflictos.

Requisitos para corrección

Adjuntad en un documento .pdf la memoria de la práctica. A lo largo de esta, se han ido indicando **en negrita las capturas de pantalla que hay que añadir** y **algunas preguntas que responder**. Organizad el **documento de forma clara dividiéndolo en los apartados** que tiene la práctica, **señalando las partes de EstudianteA y las de EstudianteB** e **identificad las capturas** con un título o el comando utilizado**.**

Además, añadid en dos archivos .zip las carpetas de los repositorio Git (aseguraos de que incluyes el directorio .git).

Los miembros de cada grupo tendrán que entregar la memoria que han creado de forma conjunta y los .zip con los contenido de las copias locales del repositorio..