

Tema 5 : Trabajo con Git y GitHub



Ciclo Superior DAW

Asignatura: Despliegue de aplicaciones web

Curso 20/21



Introducción

- En este capítulo veremos aspectos como:
 - ¿Git o GitHub?
 - Instalar Git
 - Cómo se crea un repositorio
 - Cómo se actualizan los archivos del repositorio
 - Cómo volver a una versión anterior
 - Asociar Git con Github



¿Git o GitHub?

- ¿Es lo mismo Git y GitHub? **La respuesta es NO.**
- A continuación veremos una descripción breve de cada uno de ellos que nos ayude a comprender las diferencias entre uno y otro.



¿Git o GitHub?. Git

- Git es un **software local que está destinado al control de versiones**
- Permite a los desarrolladores guardar instantáneas de sus proyectos a lo largo del tiempo.
- Generalmente es mejor para uso individual.



¿Git o GitHub?. GitHub

- GitHub es una **plataforma basada en la web.**
- **Incorpora las características de control de versiones de Git para que puedan ser utilizadas de forma colaborativa**
- También incluye características de gestión de proyectos y equipos, así como oportunidades para la creación de redes y la codificación social.



Actividad 1

Entraremos en la página de GitHub y accederemos a la cuenta

Si no tenemos una, la creamos



Instalando Git

- Si instalamos Git en un ordenador en local, podremos disponer de la terminal donde lanzar los comandos de Git.
- Es muy fácil de instalar, obteniendo la descarga desde su página oficial:

<http://git-scm.com/>



Instalando Git

- **En Windows**, la descarga es un ejecutable que puedes instalar normalmente
- **En Mac**, también puedes instalarlo a través del archivo descargable
- **En Linux**, lo encuentras en todos los repositorios de las principales distribuciones



Instalando Git

- Trabajaremos con la consola de comandos.
- Pero si lo deseamos podemos usar herramientas visuales, así como también se puede integrar Git en la mayoría de los IDE e incluso por medio de plugins en editores de código.



Actividad 2

Comprobamos que el Git bash se ha instalado correctamente



Configurando Git

- Una vez instalado Git, lo primero que debemos hacer es lanzar un par de comandos de configuración.

```
git config global user.name "Tu nombre aquí"  
git config global user.email "tu_email_aquí@example.com"
```



Configurando Git

- En estos dos comandos estamos indicando:
 - Nuestro nombre de usuario
 - Nuestro email.
- Esta configuración nos sirve para que cuando hagas cambios en el repositorio local, tener la información de que fuiste tú el que los realizó



Formas de comenzar a trabajar con Git

- Para trabajar con Git o Github tenemos dos formas de trabajar:
 - **En local:** Creando un repositorio en mi máquina.
 - **En remoto:** Clonando un repositorio de Github para traernos a local el repositorio completo y empezar a trabajar con ese proyecto.
- Empezaremos con la opción de trabajo en local



Creando un repositorio en local

- Accedemos a un directorio vacío de nuestra máquina desde la línea de comandos o desde Git Bash.
- Lanzamos el siguiente comando:

```
git init
```



Creando un repositorio en local

- A partir de ese momento tendremos un repositorio de software completo en la carpeta que acabamos de crear
- Podemos crear los archivos que queramos en esa carpeta y luego enviarlo al repositorio



Actividad 3

Elegimos una carpeta vacía y creamos un repositorio



Añadir ficheros al repositorio local

- Una vez tenemos archivos en la carpeta, debemos enviar los cambios al repositorio, lo que vamos a conocer como hacer commit.
- Sin embargo, antes de eso, hay que enviar los ficheros a un área intermedia llamada "**staging area**" en donde se almacenan los archivos que serían actualizados en el próximo commit.



Añadir ficheros al repositorio local

- La “**staging area**” es uno de los conceptos esenciales para aprender Git y para mandar los ficheros a este área tenemos que añadirlos al repositorio.
- Lanzamos el siguiente comando:

```
git add .
```



Añadir ficheros al repositorio local

- Este comando permite enviar todos los cambios en el working directory, incluyendo archivos nuevos y archivos que hayan sido modificados.
- **Una vez añadidos los archivos al staging area, ahora podremos hacer el `commit` sobre esos ficheros.**



Actividad 4

Añadimos varios ficheros nuevos a la carpeta y los enviamos a la staging area



Añadir ficheros al repositorio local

- Lanzamos el siguiente comando:

```
git commit -m "este es mi primer commit"
```

- Enviamos los archivos a una especie de base de datos donde guardamos los archivos cuyo estado queremos siempre memorizar



Actividad 5

Realizamos nuestro primer commit, con la descripción “Mi primer commit”



Subir ficheros al repositorio

- A continuación, veremos cómo subir los ficheros que tenemos en nuestro repositorio local a un repositorio en GitHub.
- Para eso, deberemos:
 - Crear un repositorio en Github
 - Subir el proyecto a Github con push



Creando un repositorio

- Para crear un repositorio debemos registrarnos en Github.
- El registro es gratuito
- Creamos un repositorio con el botón "+" de arriba a la derecha:





Creando un repositorio

- Saldrá la siguiente ventana, donde elegiremos:
 - Nombre repositorio
 - Descripción
 - Ámbito

The screenshot shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository'. Below this, there are two main input fields: 'Owner *' and 'Repository name *'. The 'Owner' field has a dropdown menu showing 'martingarciafigueira' with a small profile icon to its left. The 'Repository name' field is empty. Below these fields, a hint text says: 'Great repository names are short and memorable. Need inspiration? How about [glowing-enigma?](#)'. Underneath is a 'Description (optional)' text area. Further down, there are two radio button options for visibility: 'Public' (selected) and 'Private'. The 'Public' option has a lock icon and says 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has an unlocked lock icon and says 'You choose who can see and commit to this repository.' Below these is a section titled 'Initialize this repository with:' with the instruction 'Skip this step if you're importing an existing repository.' There are three checkboxes: 'Add a README file' (with a note 'This is where you can write a long description for your project. [Learn more.](#)'), 'Add .gitignore' (with a note 'Choose which files not to track from a list of templates. [Learn more.](#)'), and 'Choose a license' (with a note 'A license tells others what they can and can't do with your code. [Learn more.](#)'). At the bottom, there is a green 'Create repository' button.



Actividad 6

Crearemos un repositorio en nuestra cuenta Github



Subir ficheros al repositorio

- ¿Cómo haremos para subir los archivos al Github?
- Utilizaremos el propio Git, el sistema de control de versiones.
- La operación que debemos realizar es "**push**".



Subir ficheros al repositorio

- Debemos utilizar dos comandos:

```
git remote add origin https://github.com/aqui-tu-repo.git
```

```
git push -u origin master
```



Actividad 7

Subiremos el contenido de nuestro repositorio local al Github



Archivo .gitignore

- El archivo "gitignore" sirve para decirle a Git qué archivos o directorios completos debe ignorar y no subir al repositorio de código.
- En el gitignore se especificarán todas las rutas y archivos que no se requieren y con ello, el proceso de control de versiones simplemente ignorará esos archivos.



Archivo .gitignore

- Dentro del archivo .gitignore tendremos texto plano, con todas las carpetas que queremos que Git simplemente ignore, así como los archivos.
- Por ejemplo:

```
carpeta_archivos
```

```
*.jpg
```



Archivo .gitignore

- Podemos generar el código del .gitignore de forma automática.
- La herramienta online <http://gitignore.io> permite seleccionar las extensiones que quieras ignorar y genera el fichero de forma automática



Actividad 8

Crearemos un fichero .gitignore para que no nos suba ninguna imagen GIF al repositorio



Eliminar archivos

- A veces podemos necesitar eliminar archivos o carpetas que se han subido sin querer a nuestro repositorio.
- Para ello utilizamos el comando:

```
git rm -r NOMBRE_RECURSO
```



Eliminar archivos

- Una vez hayamos eliminado el recurso habrá que hacer un commit para que esos cambios se apliquen al sistema de control de versiones
- Por ejemplo:

```
git commit -m 'Eliminada carpeta imagenes del repositorio'
```



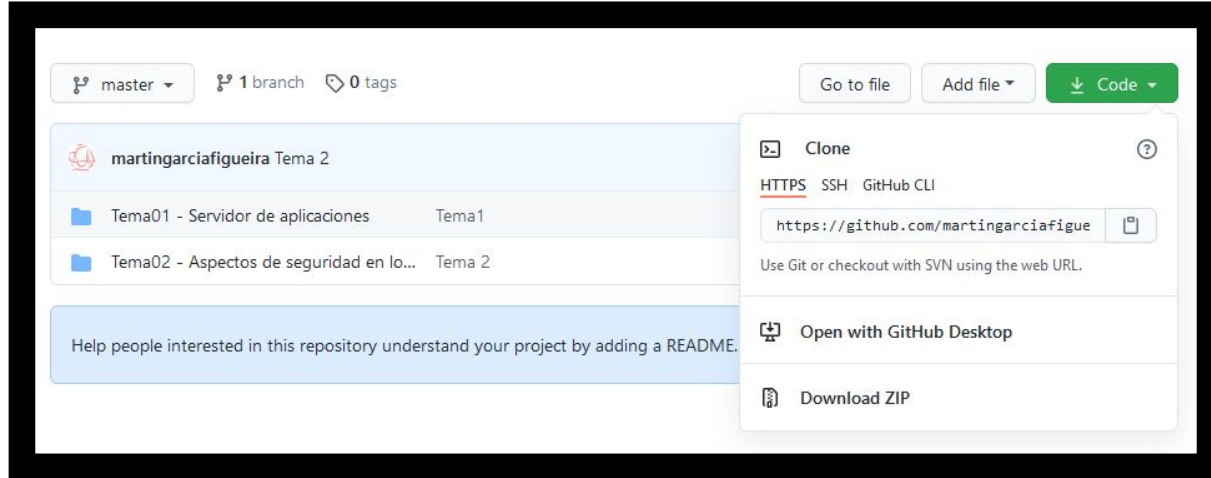
Actividad 9

Sube una carpeta “PRUEBA_BORRADO” al repositorio y posteriormente bórrala



Descargar o Clonar un repositorio

- Cuando trabajemos con un repositorio, podemos elegir entre clonarlo o descargar la información:





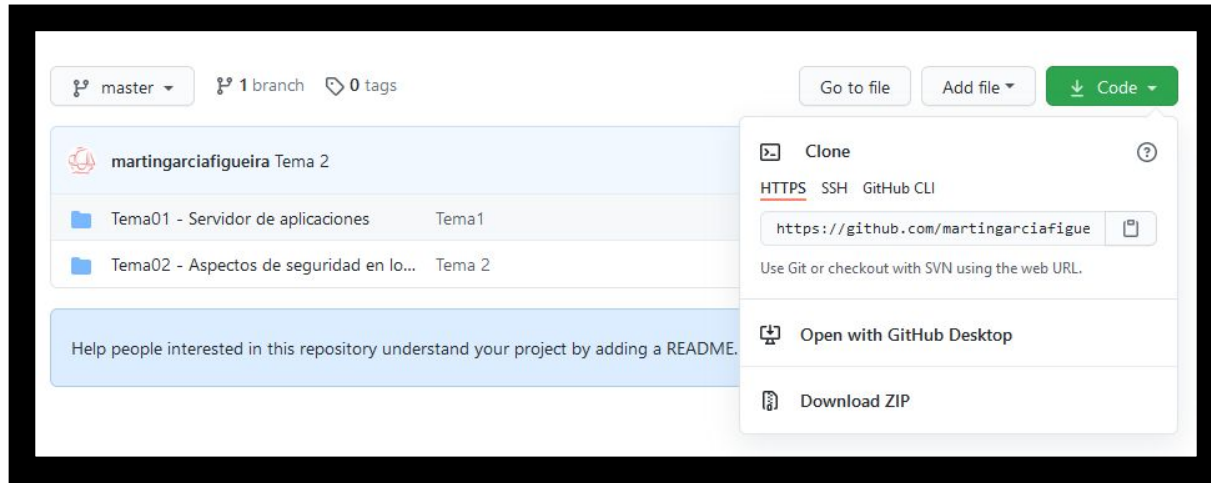
Descargar o Clonar un repositorio

- Descargar un repositorio tiene las desventajas con respecto a clonarlo que:
 - No nos crea un repositorio Git en local con los cambios del repositorio remoto
 - No podremos enviar cambios al repositorio remoto, una vez los hayamos realizado en local
- Por lo tanto, lo ideal siempre es clonar el repositorio



Clonando un repositorio

- Para clonar un repositorio, debemos obtener su URL:





Clonando un repositorio

- Una vez tenemos la URL, debemos lanzar el siguiente comando:

```
git clone https://github.com/aqui-tu-repo.git .
```

- El último punto le indica que el repositorio clonado lo pondremos en la carpeta donde estamos situados



Actividad 10

Clonaremos el repositorio que hemos creado previamente



Trabajando con ramas

- El trabajo con ramas (o branches) se utiliza en diversas situaciones:
 - Cuando hay varias personas trabajando en el mismo proyecto
 - Cuando se van a realizar funcionalidades experimentales
 - Cuando se necesite tener diversas versiones del proyecto
 - Etc



Trabajando con ramas

- Para trabajar con la creación de ramas, borrado de ramas y demás utilizaremos el siguiente comando:

```
git branch
```

- Veremos otros subcomandos de Git para trabajar con ramas, como **checkout** para moverse entre ramas o **merge** para fusionar ramas.



Trabajando con ramas. Rama master

- Cuando inicializamos un proyecto con Git automáticamente nos encontramos en una rama a la que se denomina "master"
- Esta rama es la principal del proyecto y a partir de la que podrás crear nuevas ramas cuando lo necesites.



Trabajando con ramas. Rama master

- Para saber en qué rama estamos usamos el comando:

```
git branch
```

- **¡OJO!:** Si aún no hemos hecho un commit veremos que no se ha creado todavía ninguna rama y que el comando branch no produce ninguna salida.



Creando una rama nueva

- Para crear una rama nueva:

```
git branch montecastelo
```

- No veremos ninguna salida, pero con los comandos **git branch** o **git show-branch**



Pasando de una rama a otra

- Para movernos entre ramas:

```
git checkout montecastelo
```

- Esta operación hará que todos los archivos de nuestro proyecto cambien por los de la otra rama



Actividad 11

- *Creamos una rama nueva “MontecasteloDAW2021”*
 - *Nos moveremos a ella*
 - *Realizaremos un commit sobre esta rama*
 - *Mostraremos todas las ramas existentes*



Subir una rama al repositorio remoto

- Como vimos anteriormente, por mucho que trabajemos con ramas en local, si no las subimos al repositorio no veremos los cambios actualizados en Github
- Para publicar una rama en remoto usamos el comando:

```
git push -u origin montecastelo
```



Actividad 12

Publicaremos la rama “MontecasteloDAW2021” en Github



Fusionar ramas

- Una vez controlemos el trabajo con ramas, llegará un momento que nos interesará pasar los cambios realizados en una rama a la rama master.
- Este proceso se conoce como “**merge**”, y debemos seguir los pasos:
 - Situarnos en la rama master
 - Decidir con qué rama se debe fusionar el código



Fusionar ramas

- Para fusionar dos ramas utilizaremos el comando:

```
git merge montecastelo -m 'Mergeamos la rama montecastelo con master'
```

- Para acabar, deberemos comprobar que se han fusionado los cambios de la rama **montecastelo** en la rama **master** correctamente



Actividad 13

Fusionaremos la rama “MontecasteloDAW2021” con la rama master



Borrar ramas

- Para borrar una rama, tenemos que contemplar dos situaciones:
 - Borrar una rama en local
 - Borrar una rama en remoto
- Veremos a continuación los pasos a seguir para borrar ramas en las dos situaciones



Borrar una rama en local

- Para borrar una rama en local utilizaremos el comando:

```
git branch -d rama_a_borrar
```

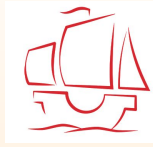
- Es posible que esta acción no nos funcione porque hayamos hecho cambios que no se hayan salvado en el repositorio remoto, o no se hayan fusionado con otras ramas. Podemos forzarlo con la opción **-D**, pero es irreversible



Borrar una rama en remoto

- Para borrar una rama en remoto utilizaremos el comando:

```
git push origin --delete rama_a_borrar
```

Actividad 14

Fusionaremos la rama “MontecasteloDAW2021” con la rama master



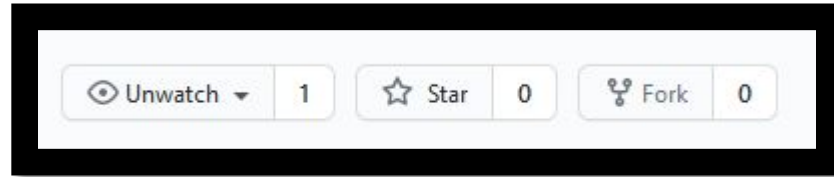
Fork de un proyecto

- Un **fork** es un repositorio que es copia de otro que ya está publicado en GitHub
- Solemos utilizar fork cuando queremos trabajar con un repositorio que no es nuestro



Fork de un proyecto

- Para hacer un fork en Github accederemos a la parte de arriba de la página del repositorio, a la pestaña Fork:



- Una vez hecho el Fork, ya podremos trabajar como si el repositorio fuera nuestro. Podremos clonarlo y realizar los cambios que queramos.



Fork de un proyecto

- Una vez realizados los cambios, tendremos que enviarlos a nuestro propio repositorio, el fork, publicado en GitHub.
- Para eso, usamos el comando:

```
git push origin master
```



Actividad 14

Realiza un fork de una rama

Clónala para poder trabajar con ella en local



Pull request

- Un **pull request** es una serie de cambios que realizamos en un repositorio al que hayamos hecho **fork** y que le proponemos al dueño del repositorio para que los aplique
- Debemos acceder a:






Pull request

- En esta página veremos la comparación contra el repositorio original
- Nos aparecerá un detalle de todos los archivos que se han cambiado, junto con un informe del número de commit realizados y los colaboradores que los han enviado



Pull request

Comparing changes



base repository: profesorDAW/pruebaPullRequest ▾

base: master ▾

←


head repository: martingarciafigueira/blog-ejem... ▾


compare: master ▾


✓ **Able to merge.** These branches can be automatically merged.


Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

 1 commit

 1 file changed

 0 comments

 1 contributor



Pull request

- Clicaremos sobre el botón **“Create pull request”** para lanzar la solicitud de fusión de nuestro fork con el repositorio original:



- Nos abrirá una página donde especificar nuestro **pull request**



Pull request

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: profesorDAW/pruebaPullRequest ▾


base: master ▾

←

head repository: martingarciafigueira/blog- Ejem... ▾










compare: master ▾

✓ **Able to merge.** These branches can be automatically merged.





Cambiar la distribución de las páginas

Write Preview


H B I         


Cambiar la distribución de las páginas


Attach files by dragging & dropping, selecting or pasting them. 


☒ Allow edits by maintainers 


Create pull request ▾

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

 1 commit

 1 file changed

 0 comments

 1 contributor



Pull request

- De manera automática, al dueño del repositorio le llegará un correo y una notificación a través del sitio de GitHub para que revisen el pull request y acepten o no los cambios al código.



Actividad 15

Realiza cambios en tu fork y crea un pull request

Comprueba si ha sido aceptado por el dueño del repositorio

Tema 5 : Trabajo con Git



Ciclo Superior DAW

Asignatura: Despliegue de aplicaciones web

Curso 20/21