

Documentación y Control de Versiones

Francisco Javier Sueza Rodríguez

9 de mayo de 2024

Centro:	IES Aguadulce
Ciclo Formativo:	Desarrollo Aplicaciones Web (Distancia)
Asignatura:	Despliegue de Aplicaciones Web
Tema:	Tema 5 - Documentación y Control de Versiones

Índice

1	Actividad 1: phpDocumentor	3
1.1	Actividad 1.1	3
1.1.1	Solución	3
1.2	Actividad 1.2	4
1.2.1	Solución	4
2	Ejercicio 2: Git y GitHub	6
2.1	Ejercicio 2.1	6
2.1.1	Solución	6
2.2	Ejercicio 2.2	7
2.2.1	Solución	7
2.3	Ejercicio 2.3	8
2.3.1	Solución	8

1. Actividad 1: phpDocumentor

Este ejercicio relacionado con generación de documentación consistirá en los siguientes apartados :

1.1. Actividad 1.1

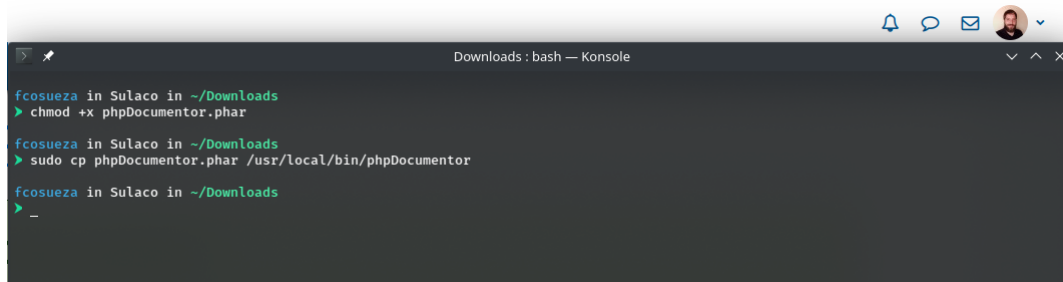
Instala la herramienta phpdocumentor en tu servidor Linux y comenta los aspectos más importantes de tal herramienta, así como las etiquetas principales que se usan.

1.1.1. Solución

En este ejercicio vamos a realizar la instalación de **phpDocumentor** en la distribución **Kubuntu 22.04** de Linux. Hay varios métodos para realizar la instalación, se puede hacer mediante **Docker**, usando **Phive** o **manualmente**. La instalación mediante composer no es recomendada por los desarrolladores de phpDocumentor, ya que hay una alta probabilidad de que se generen conflictos con otras librerías.

Nosotros hemos optado por la **opción de instalación manual**. Para ello, solo hemos tenido que descargar el **paquete de phpDocumentor** del repositorio de **repositorio de Github** de la aplicación, desde donde hemos descargado la **versión 3.4.3**.

Una vez descargada la versión simplemente le hemos **agregado permisos de ejecución** y la hemos movido al directorio `/usr/local/bin`, para que se puede ejecutar desde cualquier parte del sistema, como podemos ver en la siguiente captura.



```
Downloads : bash — Konsole
fcosueza in Sulaco in ~/Downloads
> chmod +x phpDocumentor.phar
fcosueza in Sulaco in ~/Downloads
> sudo cp phpDocumentor.phar /usr/local/bin/phpDocumentor
fcosueza in Sulaco in ~/Downloads
>
```

Actualmente, **phpDocumentor** es la aplicación más empleada para la **generación de documentación** en aplicaciones desarrolladas con PHP. Es el equivalente para PHP de **JavaDoc** para Java. Su instalación, como hemos visto, se hace de forma muy sencilla y su uso es igualmente sencillo, pudiendo ejecutarse desde la **línea de comandos**, la **interfaz web** o desde el **propio código PHP**.

Esta aplicación está desarrollada en PHP y usa el lenguaje para la generación de la documentación, empleando para ello los **DocBlocks**, que son trozos de código incluidos en el código PHP que sirven para describir la funcionalidad, parámetros, valores devueltos, etc..., de una función, constante, propiedad, etc.

Además, phpDocumentator nos permite generar la documentación en una **variedad de formatos**, entre los se incluyen **HTML**, **PDF** y **XML**. Esta última opción es muy interesante, ya que mediante el uso de XSL podemos realizar transformaciones sobre el documento, extraer información, etc.

Dentro de los DocBlocks podemos usar un **conjunto de etiquetas** para describir diferentes elementos de la función que estamos documentando, siendo las más empleadas las siguientes:

- **@package**: se emplea para añadir documentación general en un fichero a nivel de clase.

- **@access:** se usa para especificar si se va a generar o no la documentación de un elemento concreto. En caso de que no queramos que esta genere, por ejemplo, porque es un método privado, podemos indicarle el valor *private*, con lo que la documentación de ese elemento no se generará.
- **@author:** se emplea para indicar el autor de código.
- **@copyright:** indicar información sobre los derechos de uso del elemento.
- **@deprecated:** se usa para indicar que un elemento ha sido deprecado y que no debería usarse.
- **@internal:** se usa para incluir documentación que solo estará disponible para los desarrolladores.
- **@link:** incluye un enlace a un recurso.
- **@see:** se usa para crear un enlace interno a un elemento, por norma general relacionado con el elemento que se esta documentando.
- **@version:** indica la versión del elemento.
- **@global:** se usa para especificar variables globales dentro de una función.
- **@param:** describe los parámetros que recibe una función.
- **@return:** describe el valor devuelve por una función.

Como podemos ver, para aquel que este familiarizado con JavaDoc todo esto ya le sonará, ya que las similitudes son muchas e incluso la sintaxis que se emplea es prácticamente la misma.

1.2. Actividad 1.2

En esta actividad debes crear en tu servidor un script PHP con el nombre practica-XXXXXX.php, donde XXXXXX será tu apellido. A continuación escribe dentro de este script bloques de código y DocBlocks para que luego se pueda generar la documentación correspondiente. El script debe contener al menos dos funciones documentadas, indicando mediante las etiquetas vistas en la unidad los siguientes elementos:

- Parámetros de entrada de la función.
- Parámetros de devuelve la función.
- Autor y versión del script.
- Una anotación que solo sea visible en la documentación para desarrolladores.

1.2.1. Solución

En este ejercicio se ha creado un script, *practica-SUEZA.php*, que genera un mensaje sencillo y procesa una fecha indicando la fecha actual. El script se adjunta con el trabajo por lo que no vamos a entrar en detalles sobre éste.

Una vez creado el script, se ha usado **phpDocumentor** para generar la documentación empleando para ello el comando **phpDocumentor -run -d . -t docs/**, donde se le indica que el directorio donde tiene que el actual y que la documentación generada la almacene en el directorio *docs*.

En la siguiente captura, vemos la ejecución de phpDocumentor y la estructura de directorios que ha generado en el proceso.

```
DAW-Tema5 : bash — Konsole

fcosueza in Sulaco in ~/NetBeansProjects/DAW-Tema5 via  v8.1.2
> ls
practica-SUEZA.php

fcosueza in Sulaco in ~/NetBeansProjects/DAW-Tema5 via  v8.1.2
> phpDocumentor run -d . -t docs/
phpDocumentor v3.4.3

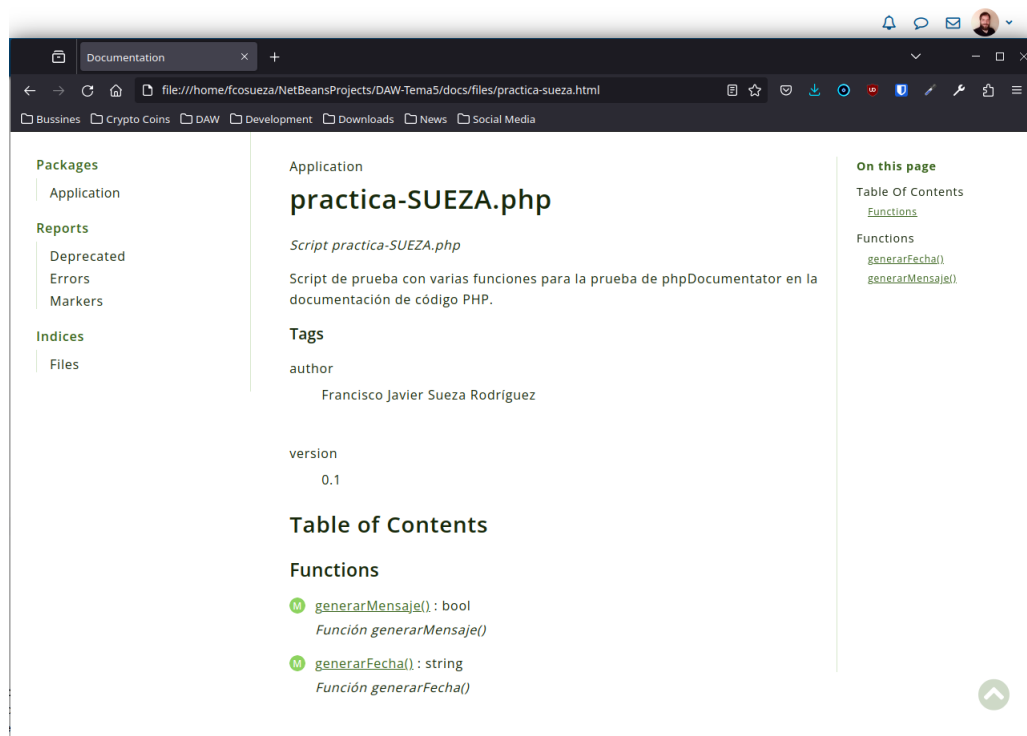
Parsing files
1/1 [=====] 100%
Applying transformations (can take a while)

All done in 0 seconds!

fcosueza in Sulaco in ~/NetBeansProjects/DAW-Tema5 via  v8.1.2
> ls -las docs/
total 52K
4,0K drwxr-xr-x 10 fcosueza fcosueza 4,0K may  8 12:50 .
4,0K drwxrwxr-x  4 fcosueza fcosueza 4,0K may  8 12:50 ..
4,0K drwxr-xr-x  2 fcosueza fcosueza 4,0K may  8 12:50 css
4,0K drwxr-xr-x  2 fcosueza fcosueza 4,0K may  8 12:50 files
4,0K drwxr-xr-x  2 fcosueza fcosueza 4,0K may  8 12:50 graphs
12K -rw-rw-r--  1 fcosueza fcosueza 9,2K may  8 12:50 index.html
4,0K drwxr-xr-x  2 fcosueza fcosueza 4,0K may  8 12:50 indices
4,0K drwxr-xr-x  2 fcosueza fcosueza 4,0K may  8 12:50 js
4,0K drwxr-xr-x  2 fcosueza fcosueza 4,0K may  8 12:50 namespaces
4,0K drwxr-xr-x  2 fcosueza fcosueza 4,0K may  8 12:50 packages
4,0K drwxr-xr-x  2 fcosueza fcosueza 4,0K may  8 12:50 reports
```

Como podemos ver, se ha generado una estructura de directorios donde se almacenan diferentes elementos, en este caso que la salida ha sido en **HTML**, hay archivos de estilo CSS, archivos Javascript, etc...

En la siguiente captura, podemos ver la visualización del archivo principal de la documentación en una navegador.



2. Ejercicio 2: Git y GitHub

Este ejercicio está relacionado con **GitHub**, se puede realizar en Windows, que consistirá en los siguientes apartados.

2.1. Ejercicio 2.1

Elabora un pequeño tutorial donde será necesario instalar git en Windows y configurarlo en tu computadora, donde se detallarán las explicaciones teóricas (¿Qué es Git? ¿Para qué sirve? ¿Última versión? etc..)y las capturas de pantallas que sean necesarias.

2.1.1. Solución

Git es un software de **control de versiones** de versiones desarrollado por **Linus Torvalds** en 2005 como sustituto para Bitkeeper para el control del código del kernel de Linux. Desde entonces, se ha popularizado y es sistema de control de versiones más extendido en el desarrollo de aplicaciones.

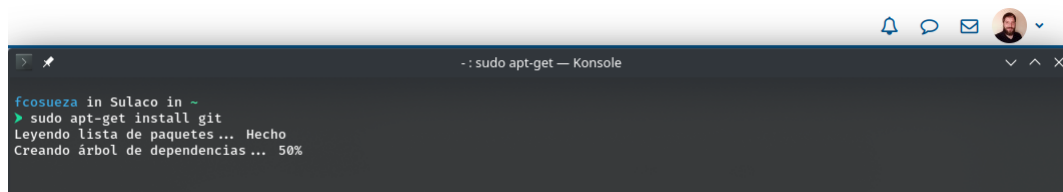
Algunas de las principales características de Git son las siguientes:

- Buen **soporte** para el **desarrollo no lineal**, permitiendo la creación de ramas y su mergeado rápidamente, así como herramientas para la navegación sobre el historial de cambios.
- Es un **sistema distribuido**. Cada desarrollador tiene una copia local del repositorio y los cambios se van propagando entre estos repositorios locales, agrupando los cambios en ramas y mergeándolos a la rama principal.
- Los **repositorios** pueden ser **publicados** empleando protocolos existentes como **HTTP**, **HTTPS**, **FTP** o **SSH**.
- Es **muy eficiente** en el **manejo de proyectos grandes**, siendo muy rápido y escalable.
- Permite **autenticación criptográfica del historial** de cambios.

Actualmente, la **última versión** de Git es la **2.45.0** que fue lanzada el **29 de Abril de 2024**, y este puede ser instalados la mayoría de sistemas operativos, entre los que nos encontramos **Linux**, **Windows**, **MacOS** o **Solaris**.

En nuestro caso, vamos a realizar su instalación en una **Kubuntu 22.04**, ya que Linux es uno de los sistemas operativos más empleado para el desarrollo de aplicaciones y uno de los que mejor integra Git.

Para realizar la **instalación y configuración** de Git en Linux vamos a usar el gestor de paquetes **APT** para instalar la aplicación, empleando para ello el comando ***apt-get install git***, lo cual no solo nos realizará la instalación sino que nos dejará la aplicación con una configuración inicial lista para ser usada. En la siguiente captura podemos ver la ejecución de este comando en la consola de Linux.



```
--: sudo apt-get -- Konsole
fcosueza in Sulaco in ~
> sudo apt-get install git
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... 50%
```

2.2. Ejercicio 2.2

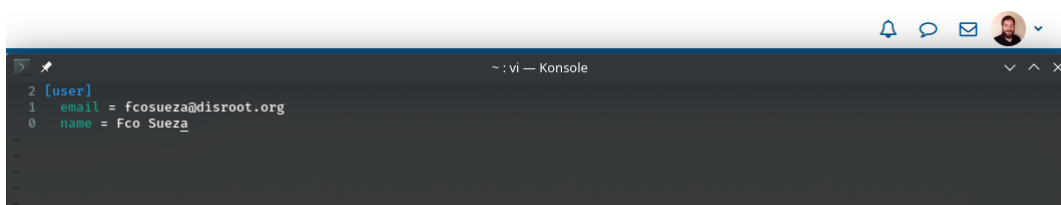
Realiza la siguiente configuración:

- Configura tu nombre, apellidos y cuenta de correo
- Muestra la versión instalada.
- Indica cuál es tu directorio de trabajo.

2.2.1. Solución

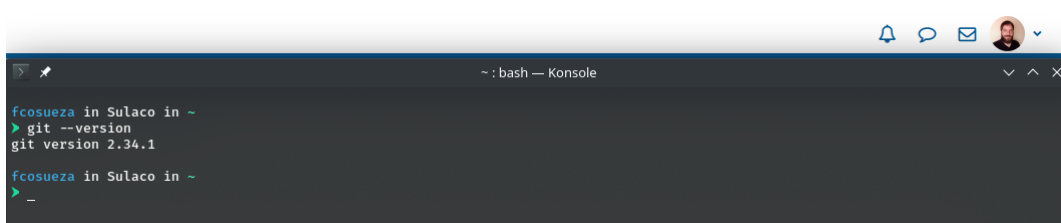
Una vez realizada la instalación de Git, vamos a realizar una configuración básica y a mostrar información sobre la versión instalada y el directorio de trabajo.

1. Primero vamos a configurar **nuestro usuario y correo electrónico**. Aunque se puede hacer de forma global, es recomendable hacerlo de forma local al usuario con el que vamos a usar Git. Para ello, debemos crear el fichero **.gitconfig** en nuestro **directorio Home** y añadir la información del usuario y el correo como vemos en la siguiente captura.



```
~ : vi — Konsole
2 [user]
1   email = fcosueza@disroot.org
0   name = Fco Sueza
```

2. A continuación, podemos emplear el comando **git --version** para comprobar que la instalación se ha realizado correctamente y que versión es la que se instaló. En nuestro caso ha sido la **versión 2.34.1**, que no es la última versión de la aplicación ya que se ha descargado de los repositorios de Ubuntu y a veces tardan un poco en incluir las nuevas versiones en las ramas estables de la distribución.



```
~ : bash — Konsole
fcosueza in Sulaco in ~
> git --version
git version 2.34.1
fcosueza in Sulaco in ~
> _
```

3. Ya tenemos git instalado y configurado. Ahora vamos a **crear un repositorio** en el proyecto PHP del apartado anterior. En nuestro caso el proyecto solo tiene un archivo PHP y la documentación generada con phpDocumentor.

Para crear el repositorio, usamos el comando **git init**, que creará el repositorio git generando el directorio **.git** con toda la información del repositorio, como el índice, los objetos, las ramas que tiene, etc.

En la siguiente captura, podemos ver la creación del repositorio, donde además se nos muestra un **mensaje de warning**, ya que por defecto el nombre de la rama master está configurado para que sea **master**, aunque esto no nos causara ningún problema ahora. Además podemos ver el **directorio de trabajo** donde se ha iniciado el repositorio.

```
DAW-Tema5: bash — Konsole

fcosueza in Sulaco in ~/NetBeansProjects/DAW-Tema5 via v8.1.2
> git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:   git config --global init.defaultBranch <name>
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:   git branch -m <name>
Initialized empty Git repository in /home/fcosueza/NetBeansProjects/DAW-Tema5/.git/

fcosueza in Sulaco in DAW-Tema5 on master [?] via v8.1.2
> ls -las
total 24K
4,0K drwxrwxr-x  5 fcosueza fcosueza 4,0K may  9 12:53 .
4,0K drwxrwxr-x  3 fcosueza fcosueza 4,0K may  8 12:06 ..
4,0K drwxr-xr-x 10 fcosueza fcosueza 4,0K may  8 12:50 docs
4,0K drwxrwxr-x  7 fcosueza fcosueza 4,0K may  9 12:53 .git
4,0K drwxrwxr-x  3 fcosueza fcosueza 4,0K may  8 12:49 .phpdoc
4,0K -rw-rw-r--  1 fcosueza fcosueza 1,6K may  8 12:58 practica-SUEZA.php

fcosueza in Sulaco in DAW-Tema5 on master [?] via v8.1.2
>
```

2.3. Ejercicio 2.3

En este apartado que se escogerá el fichero que hemos realizado en el apartado 1.2 y demostrar como funciona git en tal proyecto, por ejemplo subiendo a git todo el proyecto completo y posteriormente es necesario modificar algo para que se detecten los cambios. Es necesario realizarlo con NetBeans. Hay que darse de alta en la url <https://github.com/> y crear un repositorio llamado distanciadaw2324 para llevar los ficheros del proyecto php. Demostrarlo con capturas de pantalla.

2.3.1. Solución

En este apartado vamos a crear un repositorio en **GitHub**, y linkearlo con el que hemos creado en el punto anterior, para posteriormente subir todos los archivos del proyecto y realizar alguna modificación.

1. En primer lugar vamos a **crear el repositorio en GitHub**. La creación de la cuenta no se va a incluir en este punto, ya que tenemos una cuenta ya creada y funcional en la plataforma. Para crear el repositorio, desde la pantalla de **Your Repositories** en Github pulsamos en el botón **new** y nos aparecerá un formulario donde podremos introducir los datos del repositorio, como podemos ver en la siguiente captura.

New repository

https://github.com/new

Required fields are marked with an asterisk (*).

Owner * Repository name *

fcosueza / distanciadaw2324

distanciadaw2324 is available

Great repository names are short and memorable. Need inspiration? How about *probable-waffle*?

Description (optional)

Repositorio para el ejercicio 2.3 del Tema5 de DAW

☒ 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

.gitignore template: None

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

Choose a license

License: None

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.

Create repository

2. Una vez que hemos creado el repositorio en Github, vamos a **añadir** los cambios iniciales del repositorio que creamos en de forma local, mediante los comandos **git add -A**, que añade todos los cambios realizados a la cabecera, en este caso, los archivos que ya estaban en el directorio que hemos creado, y a continuación realizar el commit con **git commit -m**, para añadir finalmente los cambios, como vemos en la siguiente captura.

```
DAW-Tema5 : bash — Konsole

fcosueza in Sulaco in DAW-Tema5 on  master [?] via  v8.1.2
> git add -A

fcosueza in Sulaco in DAW-Tema5 on  master [?] via  v8.1.2
> git commit -m "Commit inicial con los ficheros originales"
[master (root-commit) edc75f6] Commit inicial con los ficheros originales
21 files changed, 4409 insertions(+)
create mode 100644 .phpdoc/cache/acle01bd81d25a49725bc38f4603e687-descriptors/0/G/3mBqtlQ+DmdYz5W1RASQ
create mode 100644 .phpdoc/cache/acle01bd81d25a49725bc38f4603e687-descriptors/0/I/2cP3aci7IxcMgk90xA2A
create mode 100644 .phpdoc/cache/acle01bd81d25a49725bc38f4603e687-descriptors/0/K/XbHAB56WQQA5AVKFIig
create mode 100644 .phpdoc/cache/acle01bd81d25a49725bc38f4603e687-files/H/9/30wC1x5f0L3ZTUEQWg
create mode 100644 docs/css/base.css
create mode 100644 docs/css/normalize.css
create mode 100644 docs/css/template.css
create mode 100644 docs/files/practica-sueza.html
create mode 100644 docs/graphs/classes.html
create mode 100644 docs/index.html
create mode 100644 docs/indices/files.html
create mode 100644 docs/js/search.js
create mode 100644 docs/js/searchindex.js
create mode 100644 docs/js/template.js
create mode 100644 docs/namespaces/default.html
create mode 100644 docs/packages/Application.html
create mode 100644 docs/packages/default.html
create mode 100644 docs/reports/deprecated.html
create mode 100644 docs/reports/errors.html
create mode 100644 docs/reports/markers.html
create mode 100644 practica-SUEZA.php

fcosueza in Sulaco in DAW-Tema5 on  master via  v8.1.2
```

3. Una vez que tenemos nuestro repositorio local al día, vamos a enlazar nuestro repositorio local con el repositorio creado en GitHub. Para ello, vamos a usar el comando **git remote add origin** **git@github.com:fcosueza/distanciadaw2324.git**, y después los comandos **git branch -M main**, para establecer la rama principal y **git push -u origin main** para subir los cambios al servidor de GitHub.

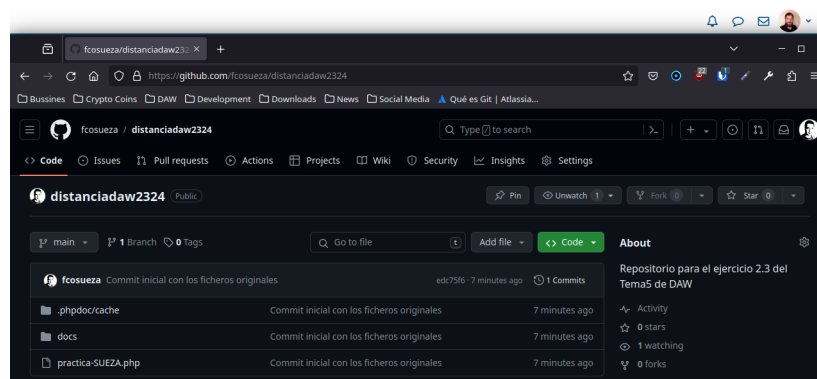
```
DAW-Tema5 : bash — Konsole

fcosueza in Sulaco in DAW-Tema5 on  master via  v8.1.2
> git remote add origin git@github.com:fcosueza/distanciadaw2324.git

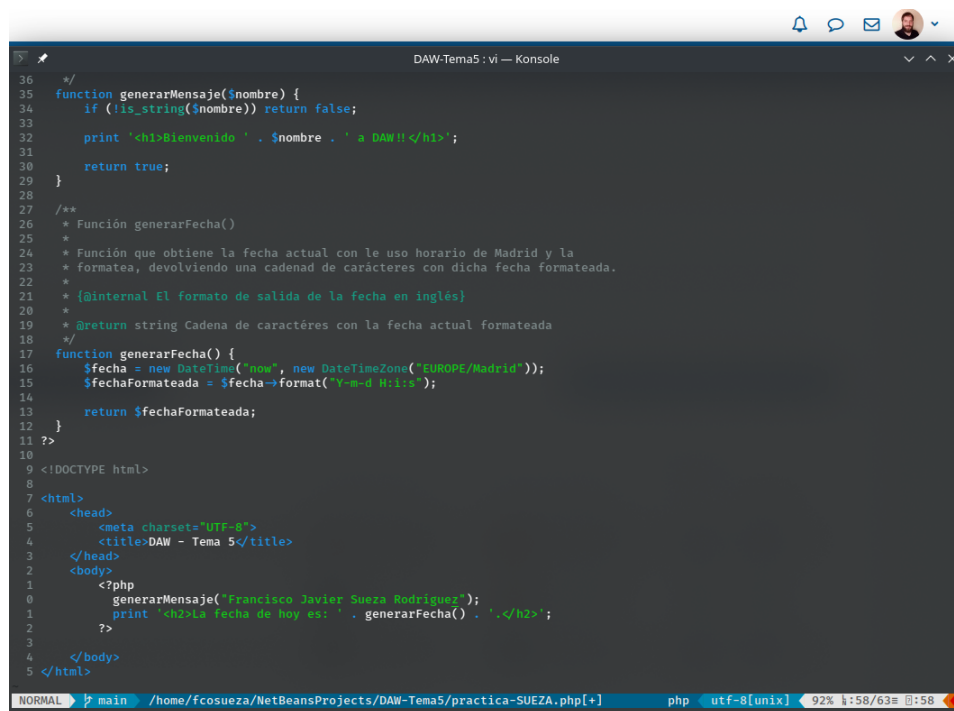
fcosueza in Sulaco in DAW-Tema5 on  master via  v8.1.2
> git branch -M main

fcosueza in Sulaco in DAW-Tema5 on  main via  v8.1.2
> git push -u origin main
Enumerating objects: 44, done.
Counting objects: 100% (44/44), done.
Delta compression using up to 8 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (44/44), 26.27 KiB | 2.63 MiB/s, done.
Total 44 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), done.
To github.com:fcosueza/distanciadaw2324.git
 * [new branch] main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Como podemos ver en la siguiente captura, nuestro repositorio de Github se ha actualizado con los archivos que teníamos en nuestro repositorio local.



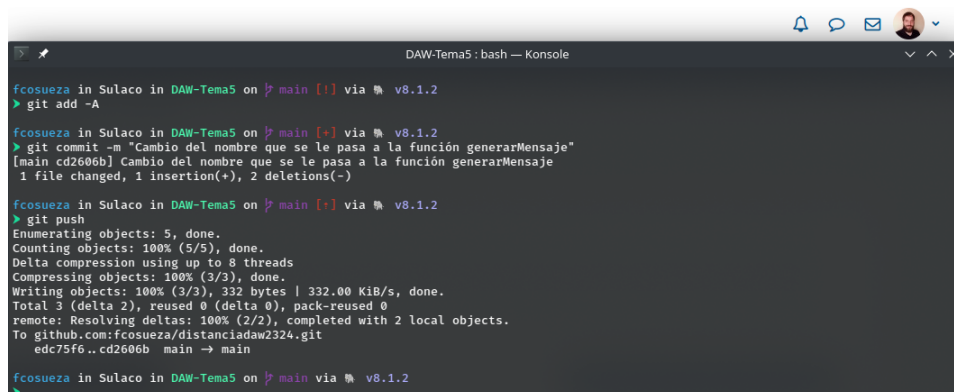
- En el siguiente paso, vamos a **modificar el código** del archivo PHP que tenemos en el repositorio, cambiando el nombre que le pasamos como parámetro a la función **generarMensaje**, como podemos ver en la siguiente captura.



```
36 */
37 function generarMensaje($nombre) {
38     if (!is_string($nombre)) return false;
39
40     print '<h1>Bienvenido ' . $nombre . ' a DAW!! </h1>';
41
42     return true;
43 }
44
45 /**
46  * Función generarFecha()
47  *
48  * Función que obtiene la fecha actual con el uso horario de Madrid y la
49  * formatea, devolviendo una cadena de caracteres con dicha fecha formateada.
50  *
51  * @internal El formato de salida de la fecha en inglés
52  *
53  * @return string Cadena de caracteres con la fecha actual formateada
54  */
55 function generarFecha() {
56     $fecha = new DateTime("now", new DateTimeZone("EUROPE/Madrid"));
57     $fechaFormateada = $fecha->format("Y-m-d H:i:s");
58
59     return $fechaFormateada;
60 }
61
62 ?>
63
64 <!DOCTYPE html>
65
66 <html>
67 <head>
68     <meta charset="UTF-8">
69     <title>DAW - Tema 5</title>
70 </head>
71 <body>
72     <?php
73         generarMensaje("Francisco Javier Suez Rodríguez");
74         print "<h2>La fecha de hoy es: " . generarFecha() . "</h2>";
75     ?>
76 </body>
77 </html>
```

- Ahora debemos actualizar el repositorio, para ellos, usaremos los comandos **git add -A**, para añadir los cambios al stage, el comando **git commit -m** para añadir los cambios al repositorio local, y por último **git push**, para enviar los cambios al repositorio remoto. En la siguiente captura, vemos la sucesión de comandos.

Como vemos, no hace falta indicar en el comando push el origen ni la rama ya que hemos establecido por defecto que se suba a la rama master **main**. Si no fuera el caso, deberíamos especificarlo en cada ejecución del comando push, o en el caso de que tuviéramos varias ramas y quisiéramos actualizar una concreta.



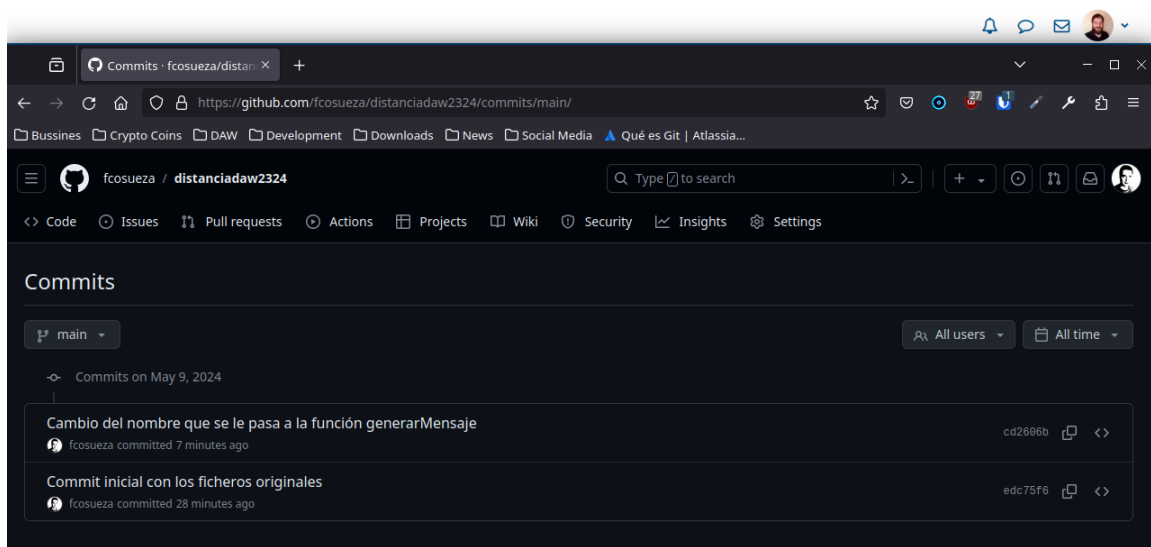
```
fcosueza in Sulaco in DAW-Tema5 on main [!] via v8.1.2
> git add -A

fcosueza in Sulaco in DAW-Tema5 on main [!] via v8.1.2
> git commit -m "Cambio del nombre que se le pasa a la función generarMensaje"
[main cd2606b] Cambio del nombre que se le pasa a la función generarMensaje
1 file changed, 1 insertion(+), 2 deletions(-)

fcosueza in Sulaco in DAW-Tema5 on main [!] via v8.1.2
> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 332 bytes | 332.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:fcosueza/distancia daw2324.git
   edc75f6..cd2606b  main -> main

fcosueza in Sulaco in DAW-Tema5 on main via v8.1.2
>
```

- Una vez realizado esto, nuestro repositorio de GitHub estará actualizado y todos nuestros archivos se habrán modificado. En la siguiente captura podemos ver el historial de modificaciones y los commits que se han realizado en el repositorio.



7. Por último, si queremos, podemos visitar el repositorio creado de GitHub en el siguiente enlace:
<https://github.com/fcosueza/distanciadaw2324>.