

## **UNIDAD 5: Tarea**

### **Documentación y control de versiones**

#### Ejercicio 1.- phpDocumentor

[Actividad 1.1. \(1,50 Puntos\)](#)

[Actividad 1.2. \(2,5 Puntos\)](#)

[Actividad 1.3. \(1,5 Puntos\)](#)

#### Ejercicio 2.- GitHub

[Actividad 2.1. \(1,50 Puntos\)](#)

[Actividad 2.2. \(1 Punto\)](#)

[Actividad 2.3. \(1,5 Puntos\)](#)

## ¿Qué te pedimos que hagas?

Este tipo de actividades se realizarán en tu servidor Linux, ya sea Ubuntu o Debian. Habrán dos tipos de actividades: una relacionada con phpDocumentor y otra con GitHub.

---

### Ejercicio 1.- phpDocumentor

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

- **Actividad 1.1. (1,50 Puntos)**

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.

---

phpDocumentor es una herramienta que es capaz de analizar el código PHP y comentarios DocBlock para generar un la documentación de la API.

Está inspirado en la primera versión de phpDocumentor y en JavaDoc (herramienta de Java destinada a la generación de documentación).

Sus etiquetas/marcas más usadas son las siguientes:

- **@access:** Puede tener dos valores:
  - 'private' no genera documentación para el elemento (a menos que se indique explícitamente).
  - 'public' genera documentación para el elemento.

Útil para generar documentación sobre la interfaz (métodos públicos) pero no sobre la implementación (métodos privados).

- **@author:** Especifica el autor del código.
- **@copyright:** Establece los derechos del código.
- **@deprecated:** Indica que una parte dada del código ha quedado obsoleta debido a su modificación en versiones más recientes de la API.
- **@example:** Permite especificar la ruta hasta un fichero con código PHP. phpDocumentor se encarga de mostrar el código resaltado (syntax-highlighted).
- **@ignore:** Evita que phpDocumentor documente una parte determinada.
- **@internal:** Para incluir información que no debería aparecer en la documentación pública, pero sí puede estar disponible como documentación interna para desarrolladores. Similar a **@access private**
- **@link:** Incluye un enlace.
- **@see:** Crea enlaces internos a la documentación de otro elemento.
- **@since:** Indica que el elemento está disponible desde una versión determinada.

Para instalar esta herramienta, tendremos que hacer lo siguiente:

- Descargamos el archivo `.phar` de [este enlace](#).
- Ejecutamos `sudo chmod +x phpDocumentor.phar` para añadir el permiso de ejecución para todos los usuarios.
- Movemos el archivo **phpDocumentor.phar** al directorio `/usr/local/bin/phpDocumentor` con el comando `mv phpDocumentor.phar` para poder ejecutar el comando **phpDocumentor** desde cualquier parte.

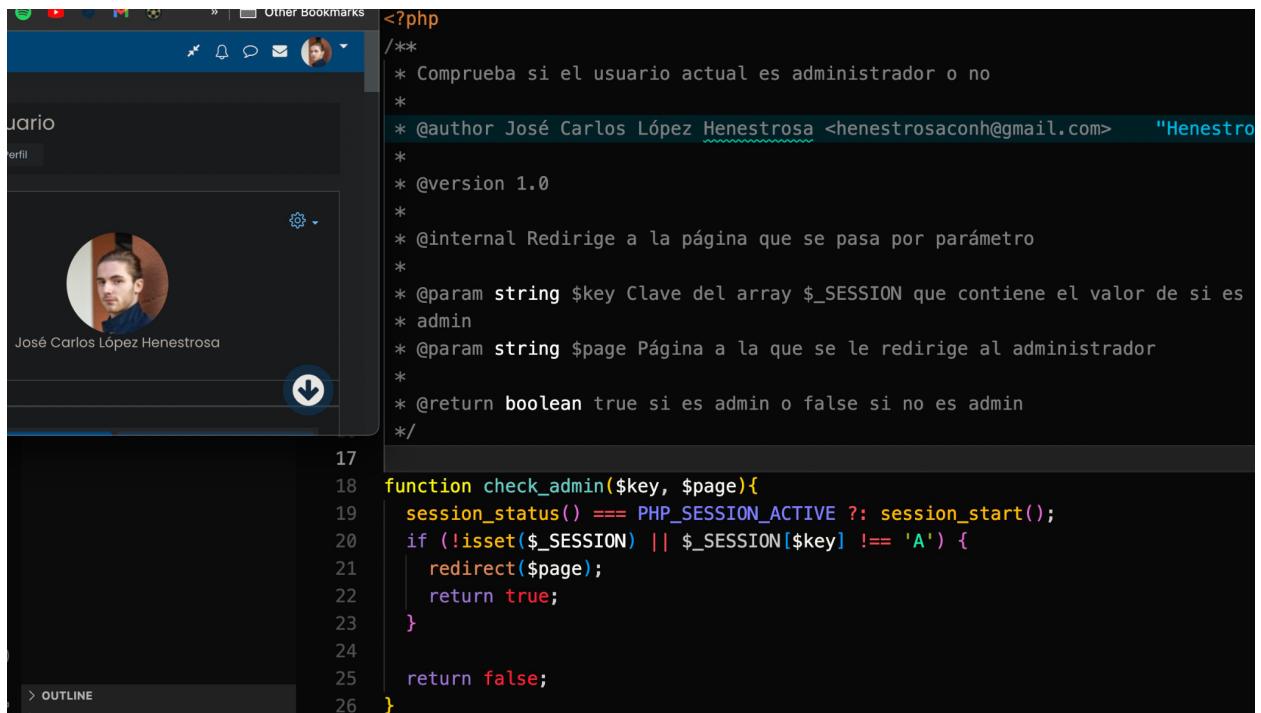
- **Actividad 1.2. (2,5 Puntos)**

En esta actividad debes crear en tu servidor un script PHP con el nombre *práctica-XXXXXX.php*, donde XXXXXX serán tus apellidos. 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.

---

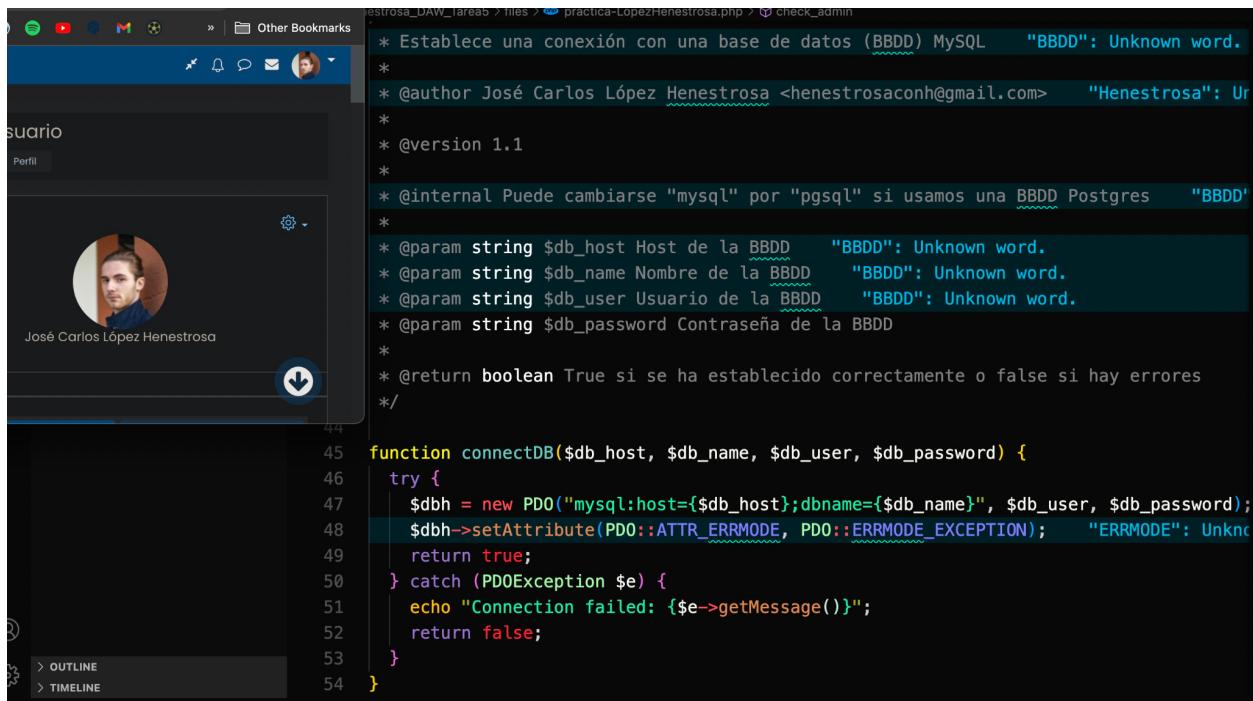
Función 1:



The screenshot shows a web browser window with a dark theme. On the left, there's a sidebar with a profile picture of a man and the text "José Carlos López Henestrosa". The main content area displays a snippet of PHP code in a code editor. The code includes a DocBlock at the top and a function definition below it. The code editor has line numbers from 17 to 26.

```
<?php
/**
 * Comprueba si el usuario actual es administrador o no
 *
 * @author José Carlos López Henestrosa <henestrosaconh@gmail.com>      "Henestro
 *
 * @version 1.0
 *
 * @internal Redirige a la página que se pasa por parámetro
 *
 * @param string $key Clave del array $_SESSION que contiene el valor de si es
 * admin
 * @param string $page Página a la que se le redirige al administrador
 *
 * @return boolean true si es admin o false si no es admin
*/
17
18 function check_admin($key, $page){
19     session_status() === PHP_SESSION_ACTIVE ?: session_start();
20     if (!isset($_SESSION) || $_SESSION[$key] !== 'A') {
21         redirect($page);
22         return true;
23     }
24
25     return false;
26 }
```

## Función 2:



The screenshot shows a web browser window with a user profile on the left and a code editor on the right.

User Profile (Left):

- Header: Other Bookmarks
- Profile picture: José Carlos López Henestrosa
- Name: José Carlos López Henestrosa
- Download button

Code Editor (Right):

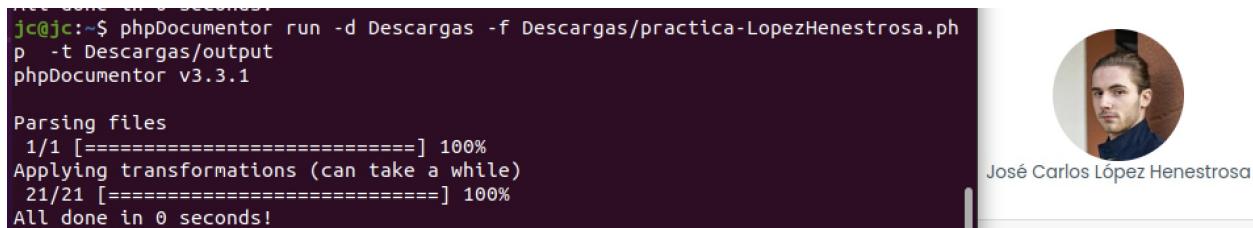
```
* Establece una conexión con una base de datos (BBDD) MySQL      "BBDD": Unknown word.
*
* @author José Carlos López Henestrosa <henestrosaconh@gmail.com>      "Henestrosa": Ur
*
* @version 1.1
*
* @internal Puede cambiarse "mysql" por "pgsql" si usamos una BBDD Postgres      "BBDD"
*
* @param string $db_host Host de la BBDD      "BBDD": Unknown word.
* @param string $db_name Nombre de la BBDD      "BBDD": Unknown word.
* @param string $db_user Usuario de la BBDD      "BBDD": Unknown word.
* @param string $db_password Contraseña de la BBDD
*
* @return boolean True si se ha establecido correctamente o false si hay errores
*/
44
45 function connectDB($db_host, $db_name, $db_user, $db_password) {
46     try {
47         $dbh = new PDO("mysql:host={$db_host};dbname={$db_name}", $db_user, $db_password);
48         $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);      "ERRMODE": Unkn
49         return true;
50     } catch (PDOException $e) {
51         echo "Connection failed: {$e->getMessage()}";
52         return false;
53     }
54 }
```

- **Actividad 1.3. (1,5 Puntos)**

Después de revisar la documentación, especialmente el apartado 2.2, genera la documentación del script PHP que hemos creado en la actividad 1.2. A continuación muestra el árbol de carpetas y archivos que genera como salida.

---

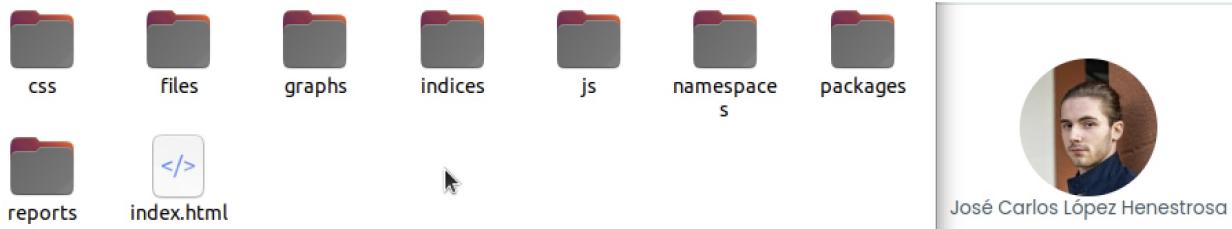
Para poder generar la documentación del archivo del anterior punto, tenemos que ejecutar el comando `phpDocumentor -d . -f ./practica-LopezHenestrosa.php -t ./output` donde el flag `-d` indica el directorio que contiene los archivos a documentar, el flag `-f` indica el archivo a documentar y el flag `-t` indica el directorio de salida de la documentación generada.



```
jc@jc:~$ phpDocumentor run -d Descargas -f Descargas/practica-LopezHenestrosa.php -t Descargas/output
phpDocumentor v3.3.1

Parsing files
 1/1 [=====] 100%
Applying transformations (can take a while)
 21/21 [=====] 100%
All done in 0 seconds!
```

Tras ejecutarlo, nos dirigimos a la carpeta **output** y nos encontraremos los siguientes archivos:



Si abrimos el archivo **index.html**, encontraremos lo siguiente:



En la pestaña **Files**, encontraremos el índice con los archivos documentados.

## Ejercicio 2.- GitHub

Este ejercicio está relacionado con GitHub, que consistirá en los siguientes apartados :

- **Actividad 2.1. (1,50 Puntos)**

Elabora un pequeño tutorial donde será necesario instalar git en Linux y configurarlo en tu computadora, donde se detallarán las explicaciones teóricas (qué es Git, para qué sirve, última versión...) y las capturas de pantallas que sean necesarias.

---

Git es un sistema de control de versiones gratuito y de código abierto escrito en C. Se utiliza para coordinar el trabajo entre los programadores que elaboran el código fuente en colaboración durante el desarrollo del software.

Su creador es Linus Torvalds, conocido por ser el creador del kernel de Linux. La última versión de Git es la 2.36.1 (2022-05-05).

Para proceder con su instalación, tenemos que ejecutar el comando `sudo apt install git-all`

```
jc@jc:~$ sudo apt install git-all
[sudo] password for jc:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 apache2 apache2-bin apache2-data apache2-utils cvs cvspc dh-elpa-helper
 elpa-dash elpa-ghub elpa-git-commit elpa-let-alist elpa-magit
 elpa-magit-popup elpa-treepy elpa-with-editor emacs emacs-bin-common
 emacs-common emacs-el emacs-gtk git-cvs git-daemon-run git-doc git-el
```



José Carlos López Henestrosa

Para comprobar su correcta instalación, ejecutamos el comando `git --version`

```
jc@jc:~$ git --version
git version 2.25.1
jc@jc:~$ █
```



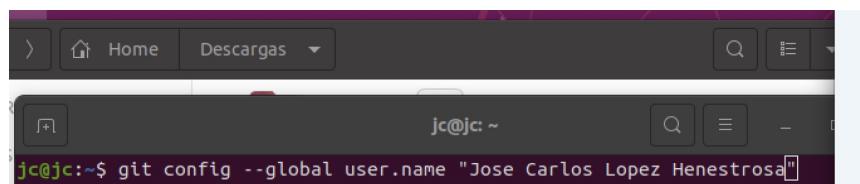
José Carlos López Henestrosa

- **Actividad 2.2. (1 Punto)**

Realiza la siguiente configuración:

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

Configuración de nombre y apellidos:

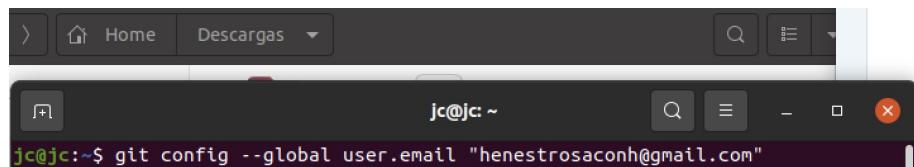


```
jc@jc:~$ git config --global user.name "Jose Carlos Lopez Henestrosa"
```



José Carlos López Henestrosa

Configuración de cuenta de correo:



```
jc@jc:~$ git config --global user.email "henestrosaonh@gmail.com"
```



José Carlos López Henestrosa

Versión instalada:



```
jc@jc:~$ git --version  
git version 2.25.1  
jc@jc:~$
```



José Carlos López Henestrosa

Para indicar el directorio de trabajo:

1. Instalamos **gitweb** con el comando `sudo apt install gitweb`
2. Creamos los directorios **git** y **www\_git** con los comandos `mkdir` `/home/usuario/git` y `mkdir /home/usuario/www_git` para estructurar nuestro modo de trabajo con Git.



José Carlos López Henestrosa

3. Editamos el archivo de configuración de **gitweb** en el directorio de configuración de Apache con el comando `nano /etc/apache2/conf.d/gitweb` y añadimos las siguientes líneas:

```
Alias /git /home/usuario/www_git
<Directory /home/usuario/www_git >
  Allow from all
  AllowOverride all
  Order allow,deny
  Options +ExecCGI
  DirectoryIndex gitweb.cgi
<files gitweb.cgi >
  SetHandler cgi-script
</files>
</directory>
SetEnv GITWEB_CONFIG /etc/gitweb.conf
```



José Carlos López Henestrosa

```
Alias /git /home/usuario/www_git
<Directory /home/usuario/www_git >
  Allow from all
  AllowOverride all
  Order allow,deny
  Options +ExecCGI
  DirectoryIndex gitweb.cgi
<files gitweb.cgi >
  SetHandler cgi-script
</files>
</directory>
SetEnv GITWEB_CONFIG /etc/gitweb.conf
```

4. Movemos los archivos básicos de **gitweb** al directorio de trabajo Apache creado anteriormente con los comandos `mv -v /usr/share/gitweb/* /home/usuario/www_git` y `mv -v /usr/lib/cgi-bin/gitweb.cgi /home/usuario/www_git`.

5. Volvemos a abrir el archivo de configuración de **gitweb** con el comando `nano /etc/apache2/conf.d/gitweb` y realizamos los siguientes cambios:

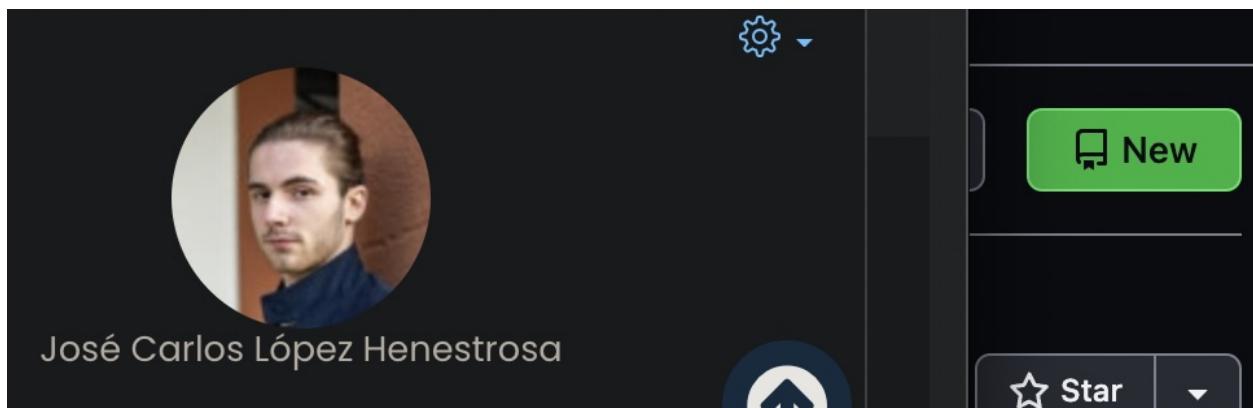
```
#nano /etc/gitweb.conf
$projectroot = '/home/usuario/git/';
$git_temp = "/tmp";
#$home_link = $my_uri || "/";
$home_text = "indextext.html";
$projects_list = $projectroot;
$stylesheet = "/git/gitweb.css";
$logo = "/git/git-logo.png";
$favicon = "/git/git-favicon.png";
```

6. Recargamos Apache para que los cambios surtan efecto con el comando `/etc/init.d/apache2 reload`

- **Actividad 2.3. (1,5 Puntos)**

En este apartado se escogerá el fichero que hemos realizado en el apartado 1.2 y mostraremos cómo funciona git en tal proyecto, por ejemplo subiendo a git todo el proyecto completo. 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 **distanciadaw2122** para llevar los ficheros del proyecto php. Demostrarlo con pantallas.

- 
1. Vamos a la pestaña **Repositories** y pulsamos en **New** para crear un nuevo repositorio.



2. Introducimos el nombre de **distanciadaw2122** como nombre de repositorio y le damos a **Create repository**.

**Owner \*** **Repository name \***

 HenestrosaConH  / distanciadaw2122 

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

**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:**  
Skip this step if you're importing an existing repository.

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

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more](#).

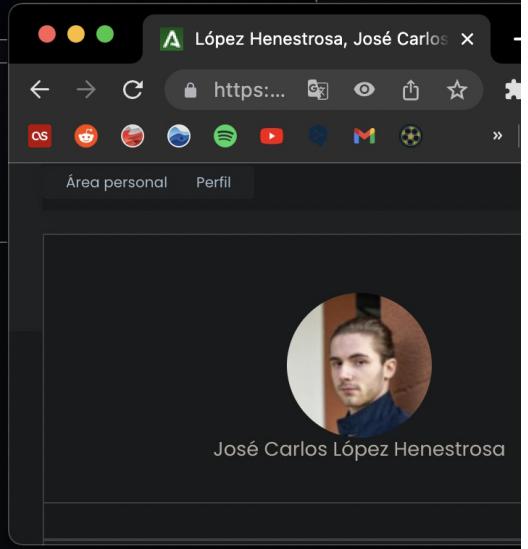
.gitignore template:

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

---

 You are creating a public repository in your personal account.

---



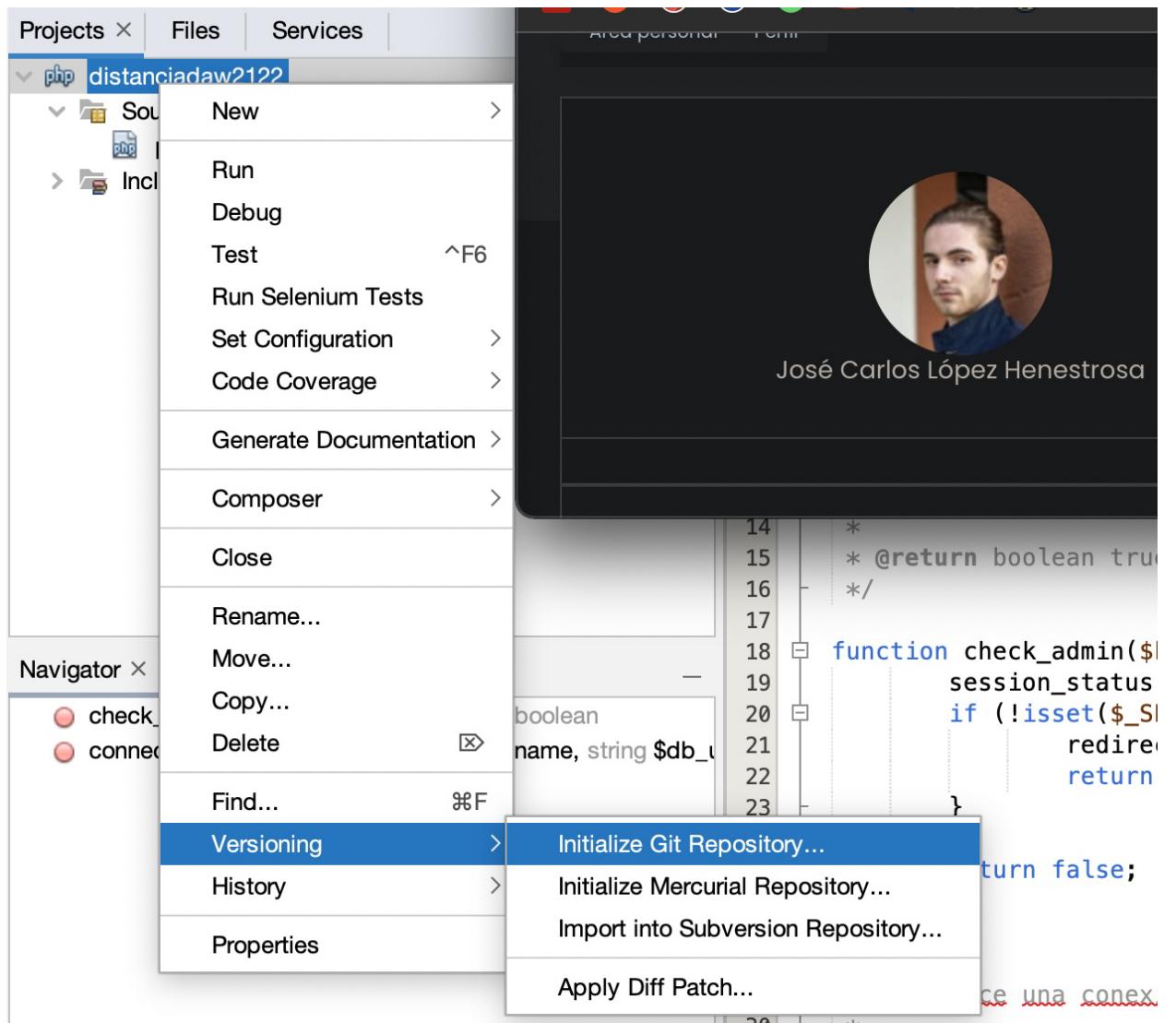
López Henestrosa, José Carlos X

https://... 🔒

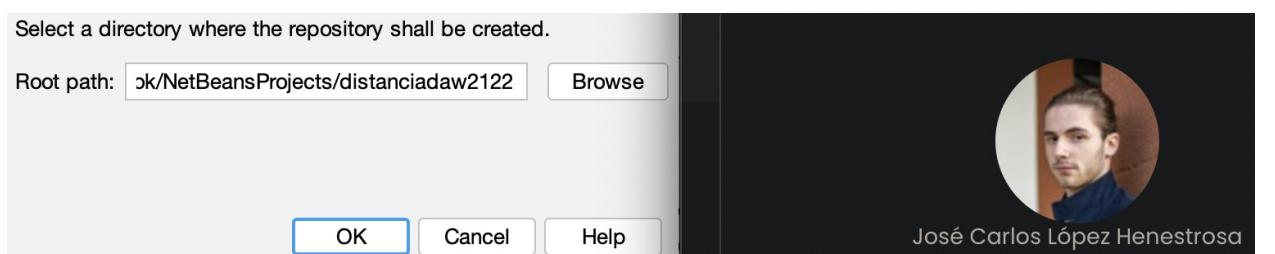
Área personal Perfil

José Carlos López Henestrosa

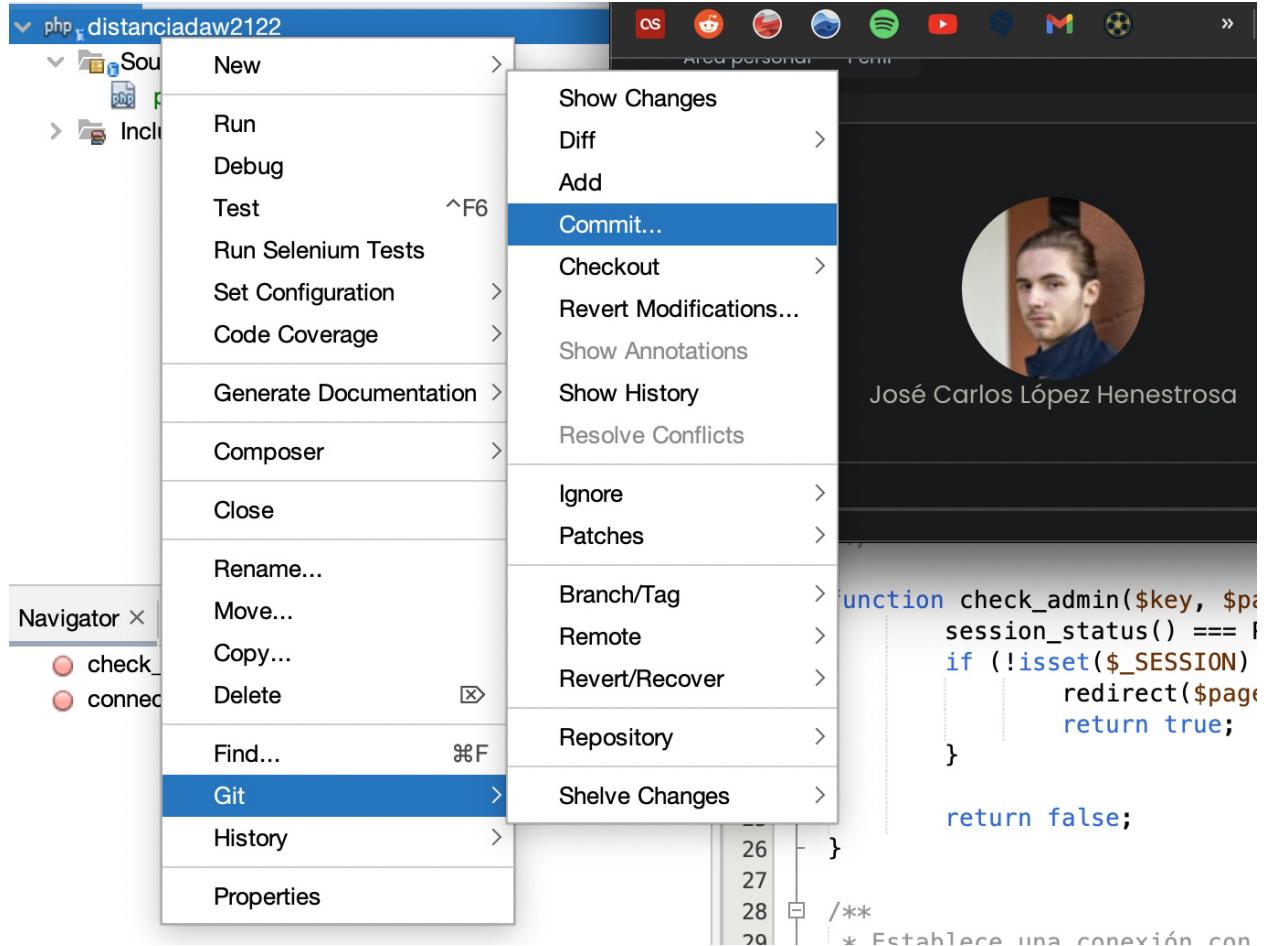
3. Con esto, ya tendríamos el repositorio creado. Ahora, vamos a Netbeans e iniciamos el repositorio Git local.



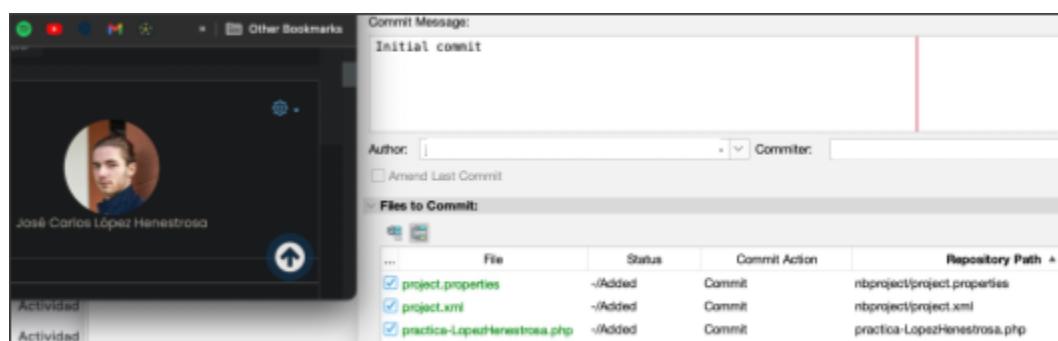
4. Indicamos la carpeta en la que queremos crear el repositorio. Por defecto, la ruta del proyecto es la que está seleccionada.



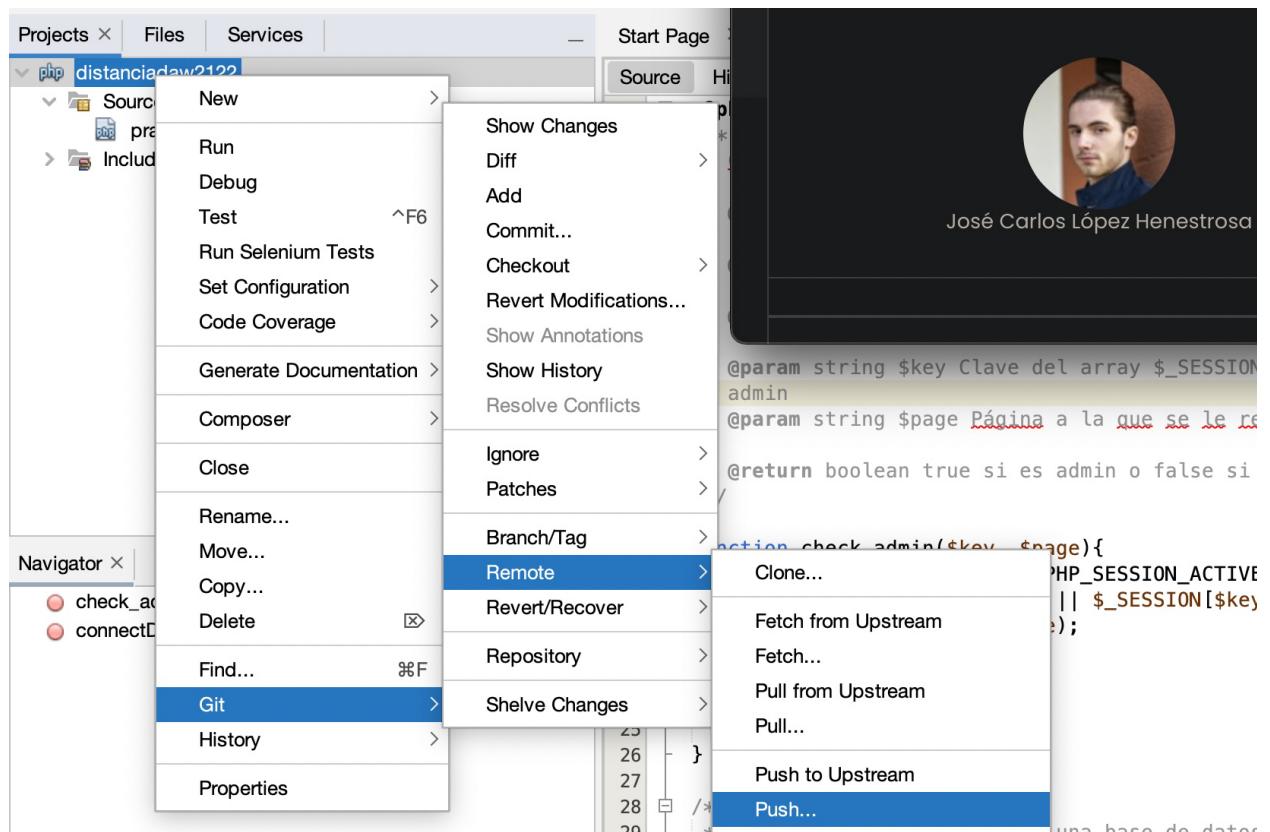
5. Para crear una nueva versión del proyecto se deberá hacer un volcado de los archivos del proyecto al repositorio local Git. Esto se denomina *commit*, y se puede realizar desde el menú contextual del proyecto seleccionando la opción **Git > Commit**



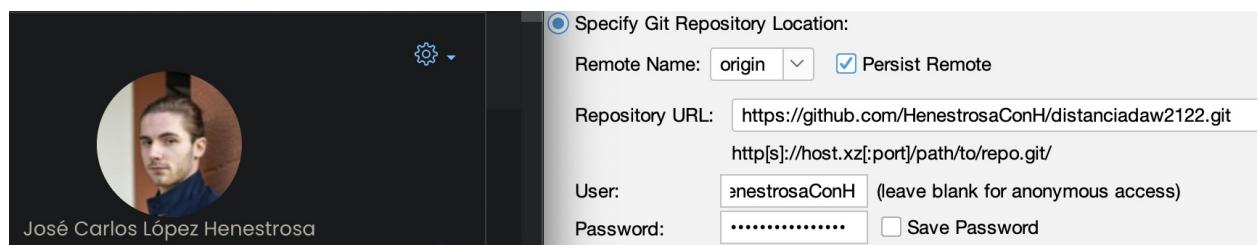
6. Durante el commit se debe introducir un mensaje que describa los detalles de la versión, así como los archivos que se van a almacenar en él. Por defecto aparecerán marcados todos los archivos, así que podemos pulsar el botón **Commit** para almacenar todos ellos, o revisar previamente la selección de archivos.



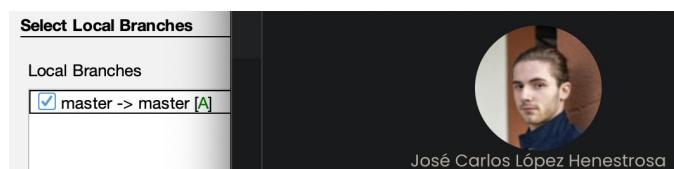
7. Hecho el commit, podemos enviar nuestra versión local al repositorio remoto seleccionando en el menú contextual la opción **Git > Remote > Push**.



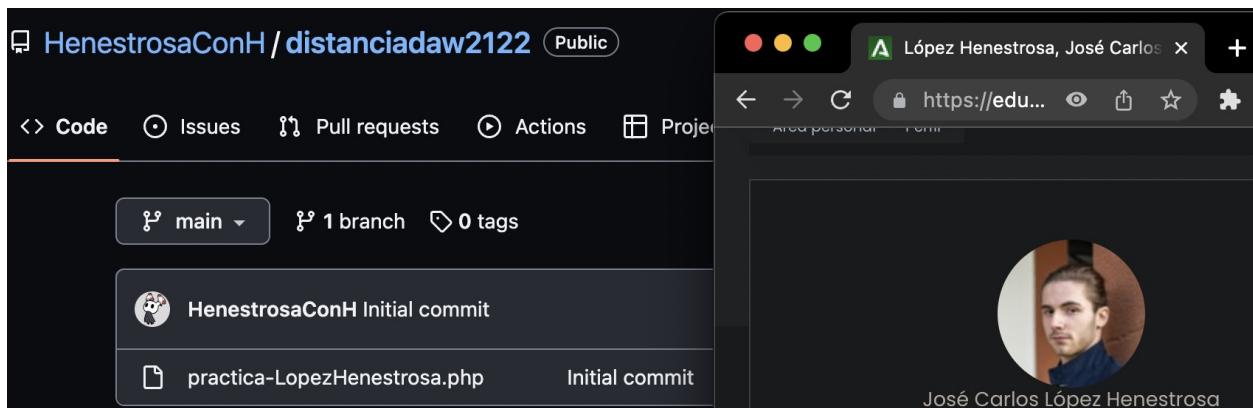
8. Pegamos el enlace que GitHub proporciona al crear el repositorio para el proyecto, dentro del cuadro de texto **Repository URL**. Indica también el nombre de usuario y la contraseña de acceso a GitHub para que se pueda realizar la conexión remota



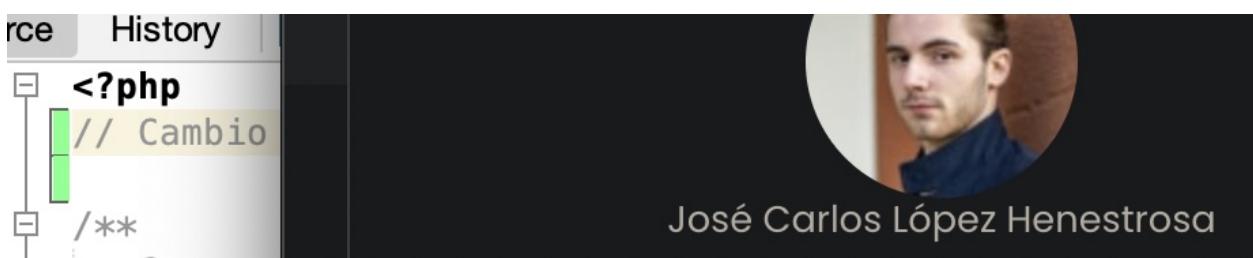
9. Indicamos la rama (*branch*) que deseamos enviar al repositorio remoto. Si no has creado nuevas ramas en el proyecto, simplemente debes seleccionar las opciones ofrecidas (*master*).



10. Nos vamos al repositorio de GitHub y verificamos que el push se ha ejecutado correctamente.

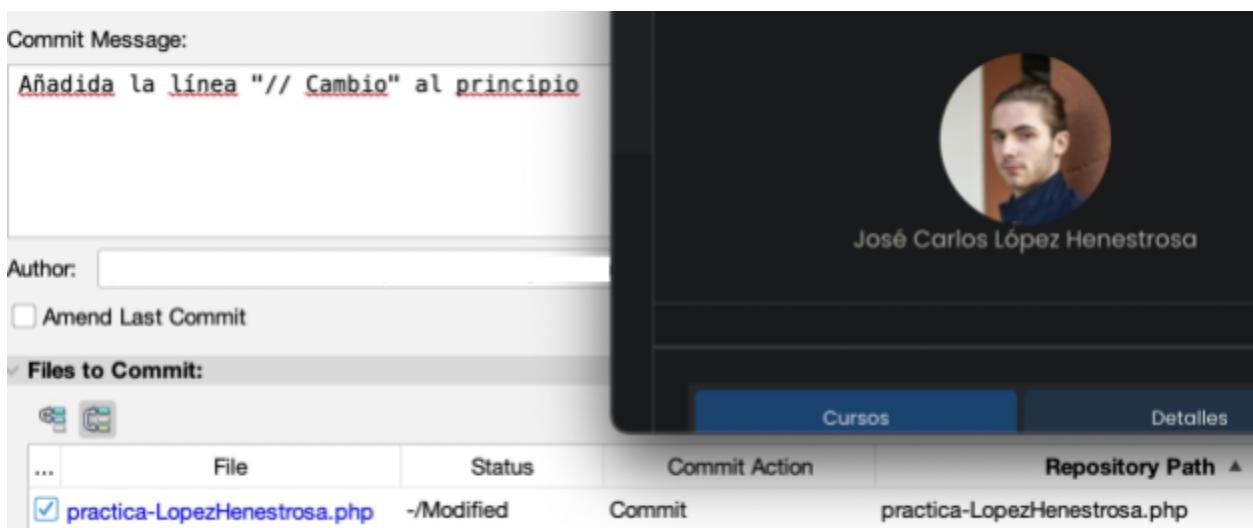


11. Realizamos un pequeño cambio en el archivo. En mi caso, he añadido la línea `// Cambio` al comienzo del documento.

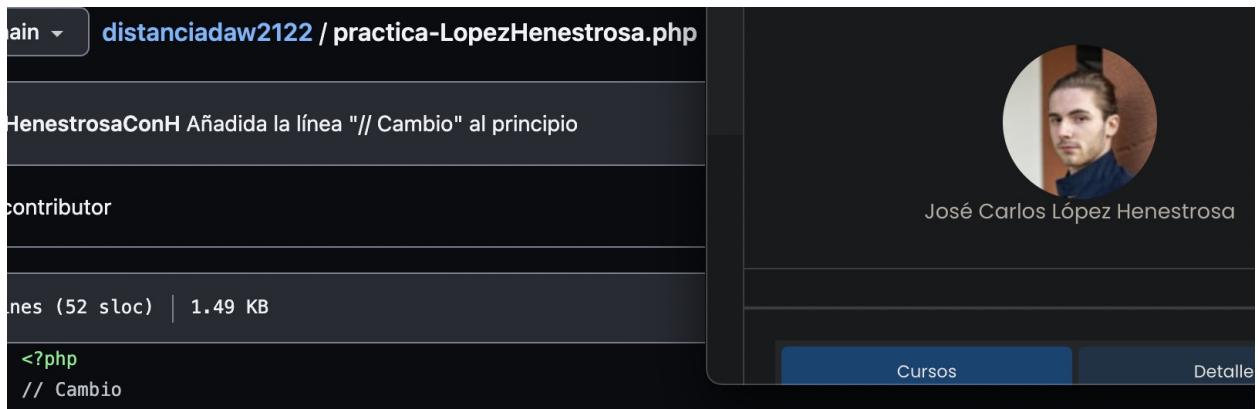


Como podemos ver, Netbeans detecta que ha habido un cambio respecto al commit anterior ya que se marcan en verde las líneas añadidas.

12. Le damos a la opción **Git > Commit** y veremos que NetBeans detecta los cambios del archivo para crear una nueva versión del proyecto. Realizamos el commit.



13. Realizamos el push de la misma forma vista en los pasos 7 y 8.
14. Vamos al repositorio de GitHub y comprobamos que se han efectuado los cambios.



The screenshot shows a GitHub commit history for a repository named 'distanciadaw2122 / practica-LopezHenestrosa.php'. A single commit is visible, made by the user 'HenestrosaConH'. The commit message is 'Añadida la línea "// Cambio" al principio'. The commit details show the file 'index.php' with 52 lines of code and a size of 1.49 KB. The commit includes the PHP code: '<?php // Cambio'. On the right side of the commit card, there is a circular profile picture of a man and the name 'José Carlos López Henestrosa'. Below the commit card, there are two buttons: 'Cursos' and 'Detalle'.

Efectivamente, aparece el mensaje del commit junto a la línea **// Cambio**.