



**CICLO: D.A.W.
MÓDULO DE DESARROLLO WEB
ENTORNO SERVIDOR**

Github

**Alumno:
Armando Herrero Silva
76442135W**

Los documentos, elementos gráficos, vídeos, transparencias y otros recursos didácticos incluidos en este contenido pueden contener imprecisiones técnicas o errores tipográficos. Periódicamente se realizan cambios en el contenido. Fomento Ocupacional FOC SL puede realizar en cualquier momento, sin previo aviso, mejoras y/o cambios en el contenido.

Es responsabilidad del usuario el cumplimiento de todas las leyes de derechos de autor aplicables. Ningún elemento de este contenido (documentos, elementos gráficos, vídeos, transparencias y otros recursos didácticos asociados), ni parte de este contenido puede ser reproducida, almacenada o introducida en un sistema de recuperación, ni transmitida de ninguna forma ni por ningún medio (ya sea electrónico, mecánico, por fotocopia, grabación o de otra manera), ni con ningún propósito, sin la previa autorización por escrito de Fomento Ocupacional FOC SL.

Este contenido está protegido por la ley de propiedad intelectual e industrial. Pertenecen a Fomento Ocupacional FOC SL los derechos de autor y los demás derechos de propiedad intelectual e industrial sobre este contenido.

Sin perjuicio de los casos en que la ley aplicable prohíbe la exclusión de la responsabilidad por daños, Fomento Ocupacional FOC SL no se responsabiliza en ningún caso de daños indirectos, sean cuales fueren su naturaleza u origen, que se deriven o de otro modo estén relacionados con el uso de este contenido.

© 2022 Fomento Ocupacional FOC SL todos los derechos reservados.

Contenido

1. ¿Qué es Git?.....	2
2. ¿Qué es GitHub?	2
3. Creación de una cuenta en GitHub.	3
4. Creación de un repositorio en GitHub.	3
5. Instalación y configuración de Git en el equipo.....	4
6. Vinculación de Git con GitHub.	4
7. Comandos básicos de Git.....	5
8. Trabajo en equipo con GitHub.	5
9. Consejos y buenas prácticas.....	6
10. Enlace a nuestro Github.....	6

1. ¿Qué es Git?

- *Explica qué es Git y cuál es su propósito.*
- *¿Por qué es importante en el desarrollo de software?*
- *Diferencia entre Git y otros sistemas de control de versiones.*

Git es un sistema de control de versiones que permite a los desarrolladores gestionar el historial de cambios en su código fuente. Su propósito es facilitar la colaboración y el seguimiento de modificaciones en proyectos de software.

Permite trabajar en equipo sin sobrescribir cambios, facilita la recuperación de versiones anteriores, agiliza la integración de nuevas funcionalidades y mejora la organización y documentación del proyecto.

Vamos a comparar Git con SVN y Mercurial.

- **Git vs. SVN:** Git es distribuido, mientras que SVN es centralizado.
- **Git vs. Mercurial:** Ambos son distribuidos, pero Git ofrece más flexibilidad en la gestión de ramas.

2. ¿Qué es GitHub?

- *Describe qué es GitHub y para qué se usa.*
- *¿En qué se diferencia de Git?*
- *Menciona al menos dos alternativas a GitHub.*

GitHub es una plataforma basada en la nube que permite alojar repositorios Git y facilitar la colaboración entre desarrolladores.

Git es el sistema de control de versiones, mientras que GitHub es un servicio que aloja repositorios Git y proporciona herramientas de colaboración.

Como alternativas tenemos las siguientes plataformas:

- **GitLab**
- **Bitbucket**

3. Creación de una cuenta en GitHub.

- *Explica los pasos para registrarse en GitHub.*
- *¿Qué configuraciones básicas se pueden realizar en el perfil?*

Primero hacemos clic en "Sign up" tras meternos en la página. Luego ingresamos un nombre de usuario, correo y contraseña. Tras confirmar el correo electrónico solo nos quedaría configurar nuestras preferencias.

Algunas configuraciones básicas en el perfil son agregar una foto de perfil, configurar claves SSH para autenticación y habilitar la autenticación en dos pasos.

4. Creación de un repositorio en GitHub.

- *¿Qué es un repositorio y para qué sirve?*
- *Explica cómo se crea un nuevo repositorio en GitHub.*
- *Diferencia entre repositorios públicos y privados.*
- *¿Cuáles son los archivos esenciales en un repositorio y cuál es su función?*

Un repositorio es un espacio donde se almacenan archivos y su historial de versiones.

Para crear uno, debemos ir a "Repositories" y hacer clic en "New", luego asignar un nombre y descripción al repositorio, seleccionar si queremos hacerlo público o privado y por último creamos el repositorio.

Mientras que un repositorio público es visible para todos, uno privado es solo accesible para colaboradores autorizados.

Los archivos esenciales de un repositorio son:

- **README.md:** Documentación del proyecto.
- **.gitignore:** Archivos a excluir del repositorio.
- **LICENSE:** Indica los derechos de uso del código.

5. Instalación y configuración de Git en el equipo.

- *¿Cómo se instala Git en los diferentes sistemas operativos?*
- *¿Qué comandos se utilizan para configurar Git con el nombre y el correo del usuario?*
- *¿Cómo se verifica que Git está correctamente instalado?*

-
- **Windows:** Descargar e instalar desde git-scm.com.
 - **Linux:** `sudo apt install git` (Debian) o `sudo dnf install git` (Fedora).
 - **macOS:** `brew install git`.

`git config --global user.name "Tu Nombre"`

`git config --global user.email tuemail@example.com`

`git --version`

6. Vinculación de Git con GitHub.

- *¿Qué métodos existen para autenticar Git con GitHub?*
- *Explica cómo clonar un repositorio desde GitHub a tu equipo.*
- *¿Cómo se enlaza un repositorio local con GitHub?*

-
- Claves SSH
 - Tokens de acceso personal

`git clone https://github.com/usuario/repositorio.git`

`git remote add origin https://github.com/usuario/repositorio.git`

`git push -u origin main`

7. Comandos básicos de Git.

- *¿Cómo se agregan archivos al área de preparación en Git?*
- *¿Cómo se realiza un commit y qué significa este proceso?*
- *¿Cómo se envían los cambios de un repositorio local a GitHub?*
- *¿Cómo se descargan actualizaciones del repositorio remoto?*
- *¿Cómo se puede ver el historial de commits?*

<code>git add archivo.txt</code>	# Agregar archivos al área de preparación
<code>git commit -m "Mensaje"</code>	# Guardar cambios en el historial
<code>git push origin main</code>	# Subir cambios al repositorio remoto
<code>git pull origin main</code>	# Descargar actualizaciones
<code>git log</code>	# Ver historial de commits

8. Trabajo en equipo con GitHub.

- *¿Qué es un fork y para qué se utiliza?*
- *Explica qué es un pull request y cómo se hace.*
- *¿Qué son las ramas (branches) en Git y cómo se crean?*
- *¿Cómo se pueden fusionar ramas en un repositorio?*

Un "fork" es una copia de un repositorio que permite hacer cambios sin afectar el original.

Pull request nos permite proponer cambios a un repositorio original desde un "fork".

Las ramas permiten trabajar en nuevas funciones sin afectar la versión principal.

```
git branch nueva-rama    # Crear una nueva rama

git checkout nueva-rama  # Cambiar a la nueva rama

git merge nueva-rama     # Fusionar una rama con la principal
```

9. Consejos y buenas prácticas.

- *¿Qué características debe tener un buen mensaje de commit?*
 - *¿Qué archivos no deberían subirse a un repositorio y cómo se pueden excluir?*
 - *¿Por qué es importante mantener actualizado el repositorio y cómo se logra?*
-

Un buen commit debe ser conciso pero descriptivo y usar un formato claro.

Los documentos que debemos excluir son:

- Credenciales y claves.
- Archivos generados por el sistema.
- Se definen en .gitignore.

Para mantener el repositorio actualizado debemos hacer git pull regularmente y resolver conflictos antes de hacer git push.

10. Enlace a nuestro Github

<https://github.com/Hesiar?tab=repositories>