Desarrollo Web en Entorno Cliente

UT 0. Navegadores y entorno de trabajo

Actualizado Noviembre 2022

ÍNDICE DE CONTENIDO

UT 0. NAVEGADORES Y ENTORNO DE TRABAJO

1. NAVEGADORES

1.1 Introducción

Para realizar cualquier desarrollo web, es imprescindible comprobar que el resultado que queremos es el adecuado con la mayor cantidad de navegadores posibles, especialmente aquellos más usados.

Además de procesar etiquetas HTML, los navegadores suelen interpretar lenguajes de script, siendo Javascript uno de los más populares.

1.2 Definición de navegador

Un navegador web es un software que permite el acceso a Internet, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser visualizados.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Además, permite visitar páginas web y hacer actividades en ella, es decir, podemos enlazar un sitio con otro, imprimir, enviar y recibir correo, entre otras funcionalidades más.

Los documentos que se muestran en un navegador pueden estar ubicados en la computadora en donde está el usuario, pero también pueden estar en cualquier otro dispositivo que esté conectado en la computadora del usuario o a través de Internet, y que tenga los recursos necesarios para la transmisión de los documentos (un software servidor web).

Tales documentos, comúnmente denominados páginas web, poseen hipervínculos que enlazan una porción de texto o una imagen a otro documento, normalmente relacionado con el texto o la imagen.

El seguimiento de enlaces de una página a otra, ubicada en cualquier computadora conectada a Internet, se llama navegación, de donde se origina el nombre navegador.

Para acceder a estos recursos, se utiliza un identificador único llamado URL (Uniform

Resource Locator).

El formato general de una URL es “protocolo://máquina/directorio/archivo”.

- Si no se especifica el directorio, toma como directorio el raíz.
- Si no se especifica el fichero, toma alguno de los nombres por defecto. Usualmente estos nombres por defecto son similares a “index.html” o “index.php”.

Un ejemplo muy típico es https://www.google.es donde se accede al recurso www.google.es usando el protocolo https.

1.3 Funcionamiento de los navegadores

La comunicación entre el servidor web y el navegador se realiza mediante el protocolo HTTP, aunque la mayoría de los navegadores soportan otros protocolos como FTP y HTTPS (una versión cifrada de HTTP basada en Secure Socket Layer o Capa de Conexión Segura (SSL)).

La función principal del navegador es obtener documentos HTML e interpretarlos para mostrarlos en pantalla. En la actualidad, no solamente descargan este tipo de documentos, sino que muestran con el documento sus imágenes, sonidos e incluso vídeos streaming en diferentes formatos y protocolos. Además, permiten almacenar la información en el disco o crear marcadores (bookmarks) de las páginas más visitadas.

Algunos de los navegadores web más populares se incluyen en lo que se denomina una Suite. Estas Suite disponen de varios programas integrados para leer noticias de Usenet y correo electrónico mediante los protocolos NNTP, IMAP y POP.

Los primeros navegadores web sólo soportaban una versión muy simple de HTML. El rápido desarrollo de los navegadores web propietarios condujo al desarrollo de dialectos no estándares de HTML y a problemas de interoperabilidad en la web. Los más modernos (como Google Chrome, Mozilla, Netscape, Opera e Internet Explorer / Microsoft Edge) soportan los estándares HTML y XHTML (comenzando con HTML 4.01, los cuales deberían visualizarse de la misma manera en todos ellos).

Los estándares web son un conjunto de recomendaciones dadas por el World Wide Web consortium (W3C) y otras organizaciones internacionales acerca de cómo crear e interpretar documentos basados en la web. Su objetivo es crear una web que trabaje mejor para todos, con sitios accesibles a más personas y que funcionen en cualquier dispositivo de acceso a Internet.

Se puede comprobar de manera online si un documento Web cumple el estándar definido por W3C mediante https://validator.w3.org/

Actualmente la mayoría de navegadores aceptan páginas no estándar, pero cuanto más estándar se la aplicación web desarrollada, mayor probabilidad que funcione

correctamente en todos los navegadores.

Es una práctica imprescindible el comprobar que cualquier desarrollo Web funcione correctamente en los principales navegadores (y también en diferentes dispositivos).

2. PRINCIPALES NAVEGADORES

- **Microsoft Edge (Antiguo Internet Explorer)**
  - URL Oficial: https://www.microsoft.com/es-es/windows/microsoft-edge
  - Sustituye al antiguo "Internet Explorer".
  - Microsoft Edge está diseñado para ser un navegador web ligero con un motor de renderizado de código abierto construido en torno a los estándares web.
- **Mozilla Firefox**
  - URL Oficial: https://www.mozilla.org/es-ES/firefox/new/
  - Mozilla Firefox es un navegador web libre y de código abierto desarrollado por la Fundación Mozilla. Usa el motor Gecko para renderizar páginas webs, el cual implementa actuales y futuros estándares web.
    - Posee una versión para desarrolladores: "Firefox Developer Edition" https://www.mozilla.org/es-ES/firefox/developer/
- **Google Chrome**
  - URL Oficial: https://www.google.com/chrome/
  - Google Chrome es un navegador web desarrollado por Google y compilado con base en varios componentes e infraestructuras de desarrollo de aplicaciones (frameworks) de código abierto, como el motor de renderizado Blink (fork de WebKit). Está disponible gratuitamente bajo condiciones específicas del software privativo.
- **Safari**
  - URL oficial: http://www.apple.com/es/safari/
  - Safari es un navegador web de código cerrado desarrollado por Apple Inc. Está disponible para Mac OS, iOS (el sistema usado por el iPhone, el iPod touch y iPad) y Windows (sin soporte desde el 2012).
- **Opera**
  - URL oficial: http://www.opera.com/es
  - Opera es un navegador web creado por la empresa noruega Opera Software. Usa el motor de renderizado Blink. Tiene versiones para escritorio, teléfono y tablet.

2.1 ¿Qué navegador se recomienda?

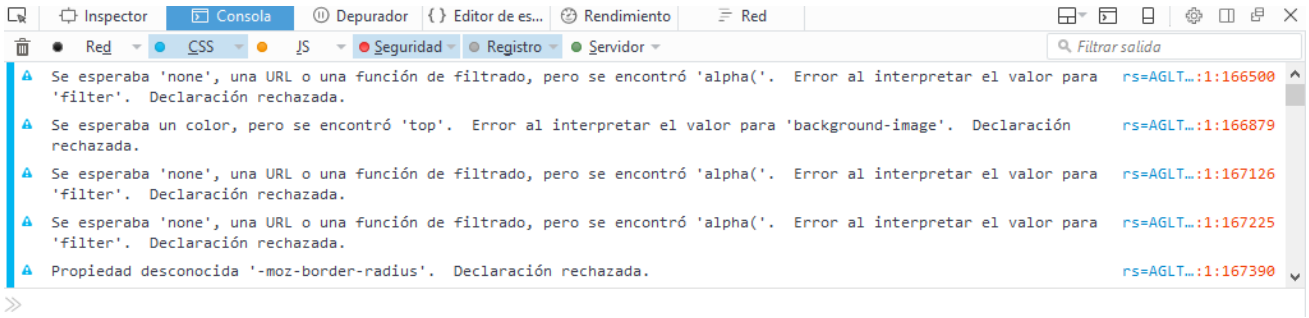
Por cuota de mercado y sencillez de uso recomiendo utilizar **Mozilla Firefox** o **Google Chrome**.

El motivo de usar estos es la gran cantidad de herramientas para depuración que poseen incluso en su versión estándar. Para la mayoría de acciones con este será suficiente. En el caso de Firefox está disponible una versión que amplía las herramientas de desarrollo "Firefox Developer Edition".

3. HERRAMIENTAS ÚTILES PARA EL DESARROLLO: LA CONSOLA WEB

La mayoría de los navegadores incorporan de manera nativa herramientas para facilitar el desarrollo, entra la que destacamos la "Consola Web". Asimismo, también mediante ampliaciones (extensiones, plugins, etc.) se amplían características para facilitar el desarrollo y la depuración de código.

3.1 Consola Web



En Firefox y Google Chrome, se puede acceder a la consola web pulsando la tecla F12.

Esta consola incluye varias pestañas:

- Red: registro de Peticiones HTTP.
- CSS: análisis y errores CSS.
- JS: análisis y errores Javascript
- Seguridad: registra advertencias o fallos de seguridad.
- Registro: registra mensajes enviados al objeto "window.console"
- Servidor: registrar mensajes recibidos del servidor Web.

El resultado de las peticiones HTTP se muestra de color negro, CSS de color azul, JavaScript amarillo y los errores o advertencias de seguridad de color rojo, registro objeto "window.console" en gris y Servidor en verde.

Mas información del uso de la "Web console" de Mozilla Firefox en la web https://developer.mozilla.org/en-US/docs/Tools/Web_Console

4. ENTORNO DE DESARROLLO

Existen diversos entornos de desarrollo, desde los más sencillos (Brackets, Notepad++, Sublime, Visual Studio Code, etc...) a interfaces más complejas (Aptana, Eclipse, etc...) o incluso online como JSFiddle, Codepen, JS Bin, LiveWeave, …

4.1 Visual Studio Code

Recomendamos el uso de Visual Studio Code. Es muy potente y posee un gran ecosistema de plugins para ampliar su funcionalidad https://code.visualstudio.com/

Aquí algunos manuales libres de uso de Visual Studio Code en castellano:
http://www.mclibre.org/consultar/informatica/lecciones/vsc-instalacion.html
http://www.mclibre.org/consultar/informatica/lecciones/vsc-personalizacion.html

4.2. Instalación y configuración

- Para **instalar** en Linux si no aparece en el Centro de Software: https://ubunlog.com/visual-studio-code-editor-codigo-abierto-ubuntu-20-04/ o descargándolo desde https://code.visualstudio.com/Download
- Cambiar **idioma**: View > **Command Palette** o **CTRL+MAY+P** o F1> Configure display language > Install Additional Languages
- **Atajos**:
  - CTRL+ p: Permite buscar un archivo por nombre dentro de la carpeta que tengamos abierta.
  - ALT +z: Ajuste de palabra, para ajustar textos largos al ancho de la pantalla y no tener que hacer scroll horizontal.
  - ALT+MAY y hacer clic en diferentes zonas de nuestro fichero: activar el *multicursor*.
  - ALT+SHIFT+ A comenta/descomenta la selección.
  - CTRL+k+s guarda todos los ficheros modificados de tu proyecto.
- **Emmet** viene integrado. Por ejemplo se puede crear un archivo .html y solo con escribir ! y TAB o ENTER te crea un esqueleto de un documento HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
    <title>Document</title>
</head>
<body>

</body>
</html>
```

También puedes escribir div.contenedor y TAB y te lo completa de esta forma:

Puedes ver más en su página web y en esta chuleta.

- **Terminal** integrada en Terminal > Nueva Terminal. Se puede cambiar por otra que tengas en tu sistema (por ejemplo, en Windows puede ser que quieras usar la terminal de GIT), desde la parte superior derecha de la terminal.
- La **configuración** de VSCode se puede encontrar en:
  - ~/.config/Code/User, en Linux (en mi caso, un derivado de Ubuntu)
  - C:\Users\username\AppData\Roaming\Code\User, en Windows 10

  Los archivos son settings.jsony keybindings.json.

- Tus **extensiones** están en:
  - ~/.vscode/extensions, en Linux
  - C:\Users\username\.vscode\extensions, en Windows 10

- Con la opción de ***Activar sincronización de configuración*** y una cuenta de Microsoft o GitHub puedes tener la misma configuración en todos los Visual Studio Code en los que inicies sesión.

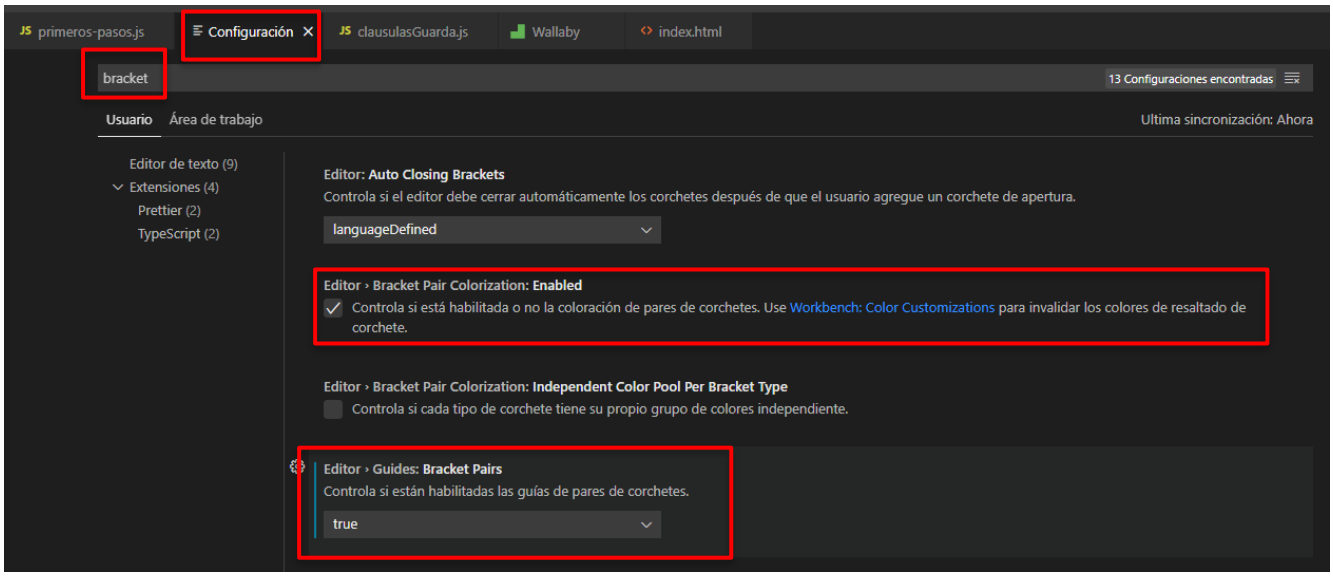
- **Plugins**:
  - Auto Close Tag. Automático, crea la de cierre.
  - Auto Rename Tag. Automático, cuando cambias una etiqueta se cambia la otra.
  - Live Server. Pulsar con el botón derecho en el fichero .html > Open with Live Server. No hace falta refrescar con F5 para ver los cambios, se hace de forma automática.
  - HTML CSS Support. Completa atributos HTML y CSS y valida CSS. Pulsa CTRL+ESPACIO para ver todas las sugerencias.
  - Intellisense for CSS Class Names. Proporciona automáticamente el nombre de las clases CSS en HTML.
  - Prettier o Beautify (prefiero Prettier, arregla, por ejemplo, los saltos de línea). Puedes pulsar botón derecho y “Dar formato al documento” o ir a Archivo >Preferencias>Configuración>Format on sabe y activar la opción para que le de formato al archivo cada vez que guardes.
  - Javascript (ES6) code snippets.
    Permite codificar más rápido con snippets de JS.

- Quokka.js. Para mostrar errores o console.log directamente en el fichero con javascript. Ver: https://www.youtube.com/watch?v=ZqW8JT1gt4U
- Wallaby.js. Similar al anterior, pero a nivel de proyecto (Angular, React) y además incluye funcionalidades de testing.
- Rest client. Para hacer peticiones a una API. Ver: https://www.youtube.com/watch?v=ZqW8JT1gt4U
- Git Lens. Para trabajar con GIT y GitHub.
    1. Instalar GIT.
    2. En el terminal ir al directorio de trabajo donde queremos utilizarlo y escribir *git init* para inicializar ese directorio como un repositorio local (ver directorio *.git*)

Otras extensiones:

- Can I use. Indica si el elemento se puede usar en todos los navegadores o no (se ve mejor desde la web: https://caniuse.com)
- Color Picker. Muestra una paleta de colores
- CSS Peek. Permite mostrar el CSS correspondiente a un selector o ir al CSS del selector correspondiente.
- Polacode. Realiza capturas de pantalla del código.
- VS icons. Para utilizar iconos según el tipo de fichero, es meramente visual.

Para colorear los pares de paréntesis ve a Archivo>Preferencias>Configuración y modifica las siguientes opciones:



1

20 Tips para PROGRAMAR MÁS RÁPIDO en Visual Studio:

[1] En la imagen de la página 9 (por orden de arriba a abajo): Configuración, bracket, Editor·Bracket pair Colorization Enabled se activa y Editor· Guides: Bracket Pairs a true

https://www.youtube.com/watch?v=orblf3Ueld4

Ver más trucos y atajos: https://code.visualstudio.com/docs/getstarted/tips-and-tricks

Puedes trabajar de forma colaborativa y en tiempo real con la extensión VS Live Share (https://ed.team/blog/comparte-tu-codigo-y-colabora-en-tiempo-real-con-visual-studio-code) con una cuenta de GitHub.

4.3. Control de versiones usando Git

Git es un software de control de versiones (lo que permite a los desarrolladores, entre otras cosas a hacer un seguimiento de lo que se ha hecho y volver a una fase anterior si deciden revertir algunos de los cambios realizados).

Esto es útil por varias razones. Por ejemplo, facilita la resolución de errores y la corrección de otros errores que puedan ocurrir durante el desarrollo. También puedes anotar los cambios en cada versión, para ayudar a cualquier miembro del equipo a mantenerse al día sobre lo que se ha completado y lo que aún queda por hacer.

Git también permite «empujar» y «tirar» de los cambios hacia y desde las instalaciones de otros ordenadores. Esto lo convierte en lo que se conoce como 'Sistema de control de versiones distribuido', y permite que varios desarrolladores trabajen en el mismo proyecto.

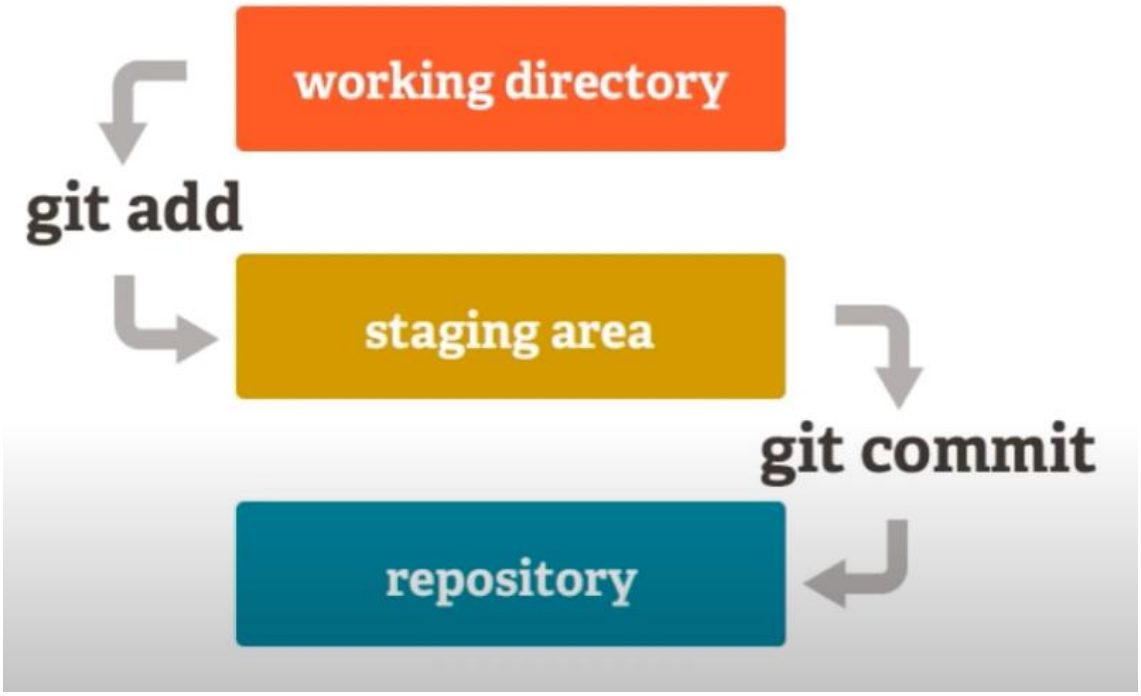
Sin embargo, hay algunos inconvenientes para manejar el desarrollo de esta manera. Como el software local está instalado en su máquina individual, git no puede leer las ediciones que otros desarrolladores puedan estar haciendo en tiempo real. Esto significa que si tú y un compañero de equipo estáis trabajando en un proyecto simultáneamente, no podrán ver el trabajo del otro.

GitHub facilita la colaboración con git. Es una plataforma que puede mantener repositorios de código en almacenamiento basado en la nube para que varios desarrolladores puedan trabajar en un solo proyecto y ver las ediciones de cada uno en tiempo real.

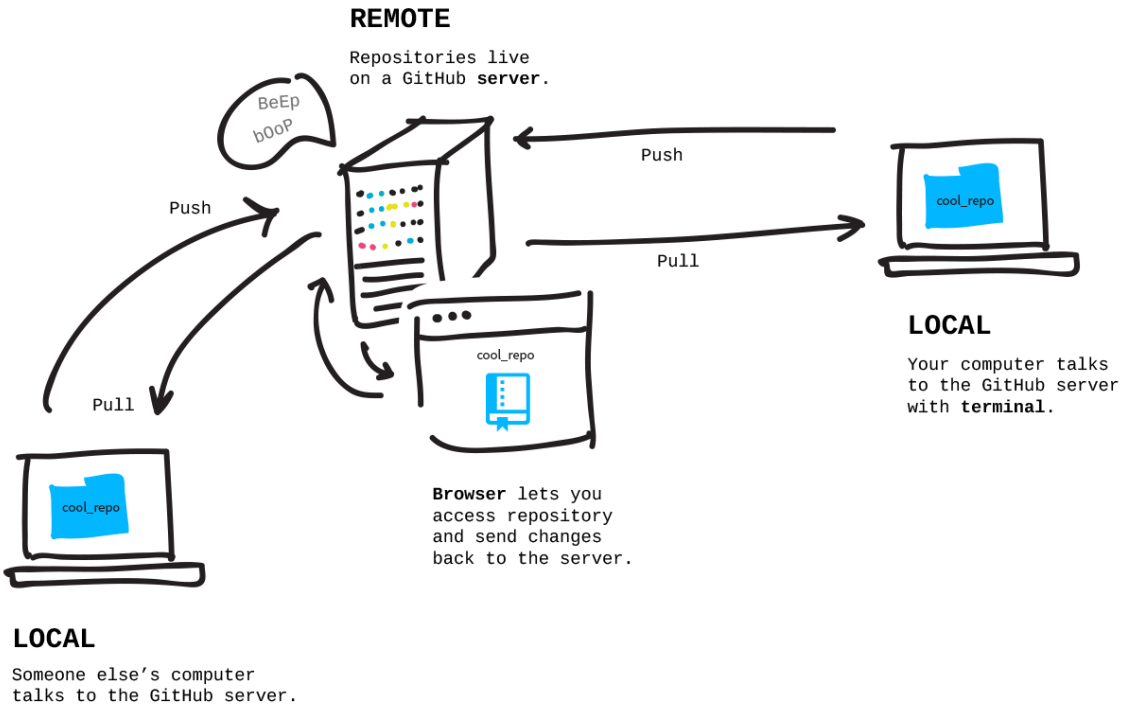
Además, también incluye funciones de organización y gestión de proyectos. Puede asignar tareas a individuos o grupos, establecer permisos y roles para los colaboradores y usar la moderación de comentarios para mantener a todos en la tarea.

GitHub no es el único repositorio git que puede considerar si está buscando colaborar en un proyecto de desarrollo. **GitLab** es otra plataforma muy similar.

Este es el flujo de trabajo de Git:



Y este es un flujo de trabajo usando GitHub (con GitHub, lo que se hace es subir lo que esté en tu repositorio local)



Durante el curso, se utilizarán repositorios Git tanto para la entrega de prácticas como para

facilitaros el disponer de un repositorio con control de versiones a través de GitHub.

IMPORTANTE: La rama principal en GitHub ya no se llama *master* sino *main* (master tenía connotaciones "racistas": https://platzi.com/blog/cambios-en-GitHub-master-main/). En Git si sigue llamándose *master*.

Para instalar **Git**:

- Ubuntu:
  - sudo apt-get update
  - sudo apt-get install git

- Windows: https://git-for-windows.GitHub.io/

Al instalarlo, seguir el asistente. En el caso de seleccionar el editor, puedes elegir otro diferente a Vim. También puedes seleccionar "main" como nombre de la rama principal.

Comandos básicos:

Configurar Git

Necesitamos proporcionar nuestro nombre y dirección de correo electrónico porque Git incorpora esta información en cada confirmación (commit) que hacemos.

1. git config --global user.email "you@example.com"
2. git config --global user.name "Tu Nombre"

Sólo es necesario hacerlo una vez.

Crear proyecto Git local

1. git init
   Crea una carpeta .git dentro del directorio de trabajo, donde se guardan todos los archivos necesario con los que trabaja Git.
2. git add <file> o git add . (para todos los ficheros del directorio)
   Lleva los ficheros al área de Staging
3. git status
   Revisa el estado de los ficheros del proyecto (staging, commit, …)
4. git commit -m "mensaje del commit"
   Lleva los ficheros al área de commit
5. git checkout -- . (hay que poner también el punto final)
   Reconstruye el proyecto desde el último commit realizado.
6. git log o git log --oneline
   Para ver el log de git.

Subir proyecto Git local a GitHub

1. git remote add origin https://GitHub.com/usuarioGitHub/repositorioGitHub.git
   IMPORTANTE: Al crear el repositorio en GitHub, hazlo sin crear ningún archivo, será más sencillo hacer luego el push.
   Indica a Git la dirección de GitHub. Esta URL la puedes copiar desde GitHub > vas al repositorio > botón verde de Code y copias la URL.
2. git branch -M main (opcional)

   Esto es debido a que es posible que, por defecto, Git siga usando master y no main para nombrar a la rama principal.
3. git push -u origin main (pedirá usuario y contraseña de GitHub)

   Si el repositorio en GitHub tiene algo (p.e. un README) que no tengas en tu local es posible que no te deje, para ello forzamos el push para eliminar lo que haya en el remoto: git push -f origin main
4. Una vez realizado todos los pasos anteriores podemos ir ejecutando los comandos:
   a. git add .
   b. git acommit -m “mensaje”
   c. git push

   Para ir subiendo a GitHub los sucesivos cambios que vamos realizando

Descargar un proyecto desde GitHub:

1. git clone https://GitHub.com/usuarioGitHub/repositorioGitHub.git
   Esto crea un directorio llamado repositorioGitHub dentro de tu carpeta de trabajo. Si quieres que el contenido de repositorioGitHub pase a tu carpeta de trabajo tu directorio debe estar vacío (ni .git ni nada) y luego ejecutar git clone https://GitHub.com/usuarioGitHub/repositorioGitHub.git . (con el punto al final para indicar el directorio actual)
2. git pull
   Se trae las actualizaciones desde el servidor GitHub a tu local.

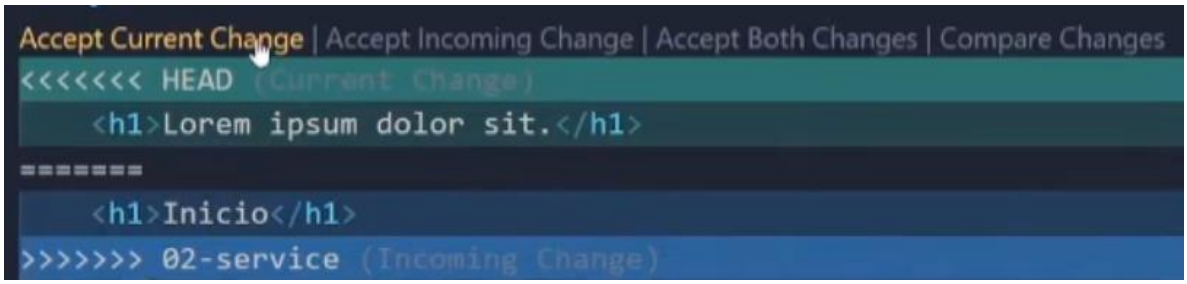
Hacer una modificación y crear una rama nueva con dicha modificación y subirla a GitHub:

1. git branch
   Para ver las ramas de tu repositorio (tienes que haber hecho al menos un commit)
2. git branch experimental.
   Crea una rama llamada experimental
3. git checkout experimental.
   Para moverse a la rama expermiental y ponerla como activa.
4. Con git log –oneline –graph se ven más fácilmente las ramas.
5. git add .
6. git commit -m “mi primer commit en experimental”
7. git push -u origin experimental

Fusionar una rama con la principal:

Para hacer un merge nos situamos en una rama, en este caso la main, y decimos con qué otra rama se debe fusionar el código.

1. git merge experimental -m “Esto es un mensaje para una fusión”
   Se sobrescribe lo que había en main con lo de experimental. Es a nivel local, en GitHub siguen las dos ramas.
2. Si hay conflicto se puede resolver de forma manual cambiando el código o usando las herramientas que nos proporciona VsCode:



Después hacemos el git add y el git commit -m “mensaje”

Eliminar una rama en Git y GitHub:

1. git branch -d experimental
   Borra una rama local. Se puede usar -D para forzar el borrado
2. git push origin --delete experimental
   Barra una rama en GitHub

Eliminar todo lo referente a git (por ejemplo, para empezar de nuevo):

1. rm -r .git desde una terminal y dentro del directorio de trabajo donde se encuentra el .git (también se puede usar rm -rf .git pero hay que tener mucho cuidado, porque no solicita ninguna confirmación)

Usar GitHub para publicar un sitio web:

1. Ir al repositorio que contenga el código fuente de tu web y acceder a la opción de Settings.
2. En Source escoger la rama que se desee y seleccionar /(root)
3. IMPORTANTE: Hay que tener mucho cuidado con las rutas relativas. P.e.:

```
<script src="./js/parImpar.js"></script>
<script src="/js/parImpar.js"></script>
```

Estas dos líneas son totalmente válidas en VSCode, pero sólo la primera funciona en GitHub Pages[2]

4. No usar Themes (ver https://programminghistorian.org/es/lecciones/sitios-estaticos-con-jekyll-y-GitHub-pages)

Crear versiones

Con los tags podemos hacer versiones de nuestro proyecto.

```
// Crear un tag
git tag versionAlpha -m "versión alpha"

// Listar tags
git tag

// Borrar tags
git tag -d nombreTags

// Hacer una versión en un commit anterior ej: f52f3da
git tag -a nombreTag f52f3da -m "version alpha"

// Mostrar información del tag
git show nombreTag

//Subir el commit con el tag
git push --tags
```

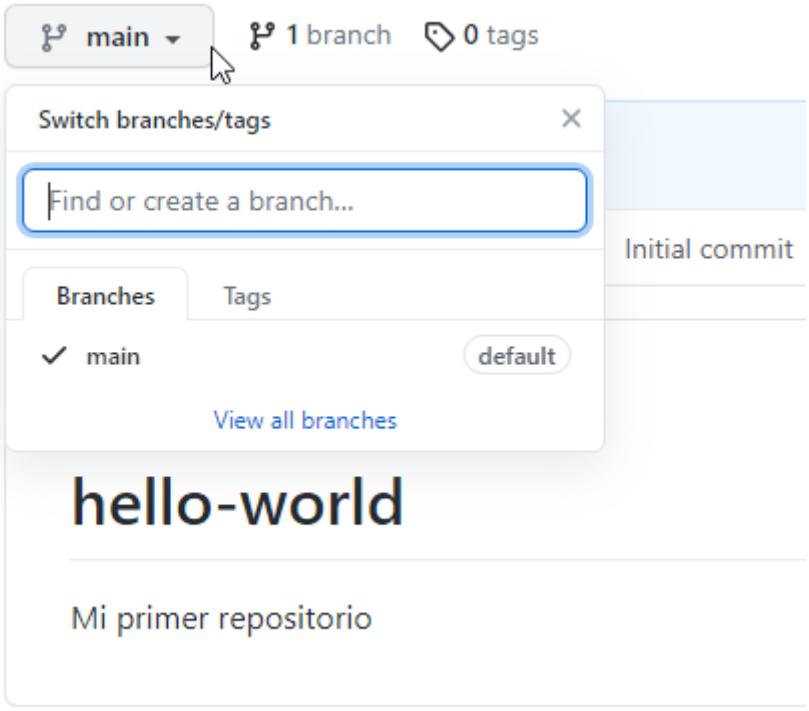
Además desde GitHub si accedes a tags, puedes descargarte el .zip con el código de esa versión.

Lee esto para saber 5 cosas que no debes hacer con Git: https://somostechies.com/05-cosas-que-no-debes-hacer-con-git/

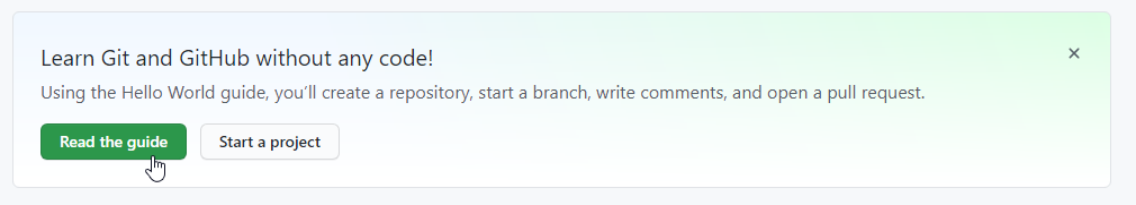
Al crear un repositorio en **GitHub** tienes la opción de crear un fichero **README.md** para incluir instrucciones sobre tu proyecto. Este fichero sigue la codificación markdown para darle formato al texto. Puedes usar alguna herramienta online como https://stackedit.io/app# para crearlo o visuallizarlo.

También puedes crear ramas o branches del desarrollo principal desde la interfaz, para por ejemplo, probar una nueva funcionalidad o diseño.

[2] Parece que esto solo pasaba en versiones anteriores



Puedes usar la guía de GitHub dentro de tu cuenta para ver toda esta información



Para facilitar la tarea del uso de Git es recomendable instalar alguna extensión o entorno que os facilite su uso. Para usar Git en Visual Studio Code recomendamos:

- https://code.visualstudio.com/docs/editor/versioncontrol
- http://www.mclibre.org/consultar/informatica/lecciones/vsc-git-repositorio.html

Aquí un ejemplo del uso de Git dentro de Visual Studio Code:
https://code.visualstudio.com/docs/introvideos/versioncontrol

5. MATERIAL ADICIONAL

[1] Curso de Git en Udacity
https://www.udacity.com/course/how-to-use-git-and-github--ud775

[2] Taller de Git

https://aulasoftwarelibre.GitHub.io/taller-de-git/

[3] Videotutorial sobre Git y GitHub

https://www.youtube.com/watch?v=HiXLkL42tMU

[4] Manual de Git y GitHub

https://desarrolloweb.com/manuales/manual-de-git.html

[5] Uso de la "Consola Web"

https://developer.mozilla.org/en-US/docs/Tools/Web_Console

[6] **BlackBox** Extensión para Google Chrome o Mozilla Firefoz (permite copiar texto de imágenes/vídeos y pegarlo como texto editable.

[7] Imprescindibles bibliografía HTML, CSS y Javacript: https://developer.mozilla.org/es/ y https://devdocs.io