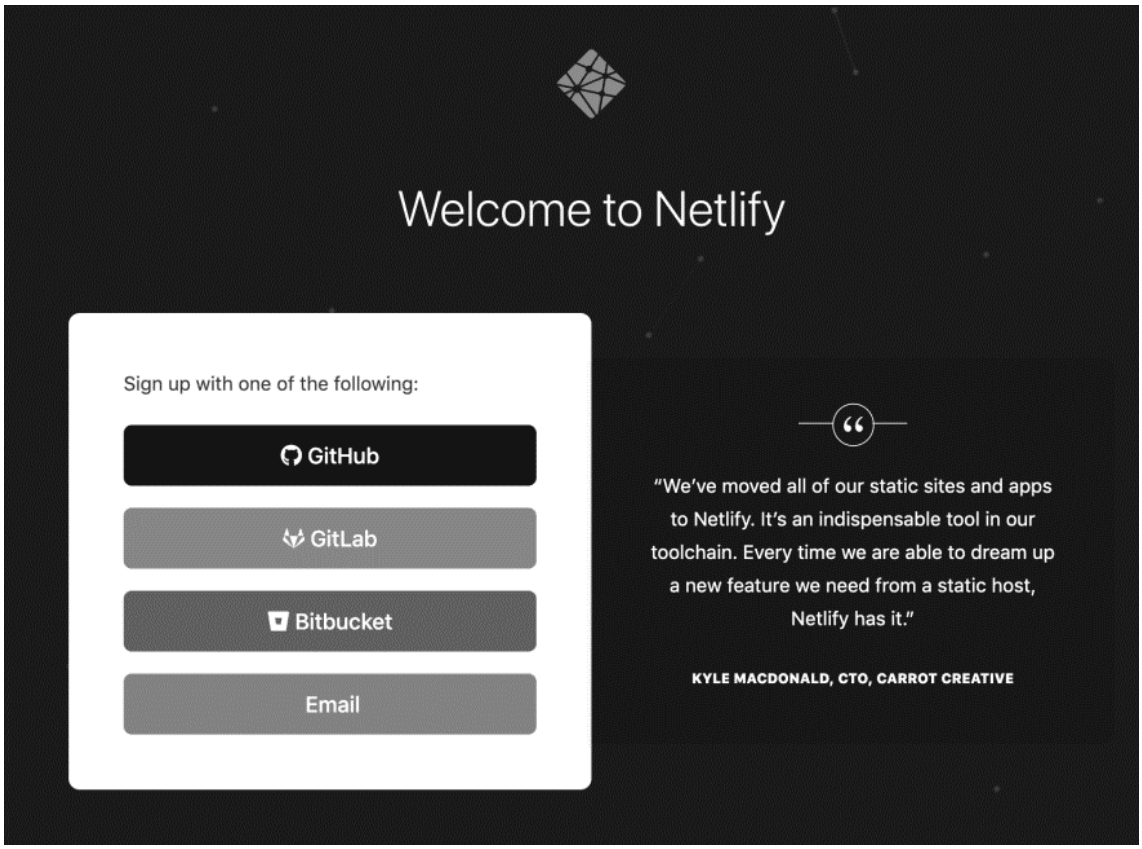
CONTROL DE VERSIONES

¿De qué sirven las emociones si no se pueden compartir?

(Anna Gavalda)

DEPLOY DE UN SITIO WEB ESTÁTICO CON NETLIFY

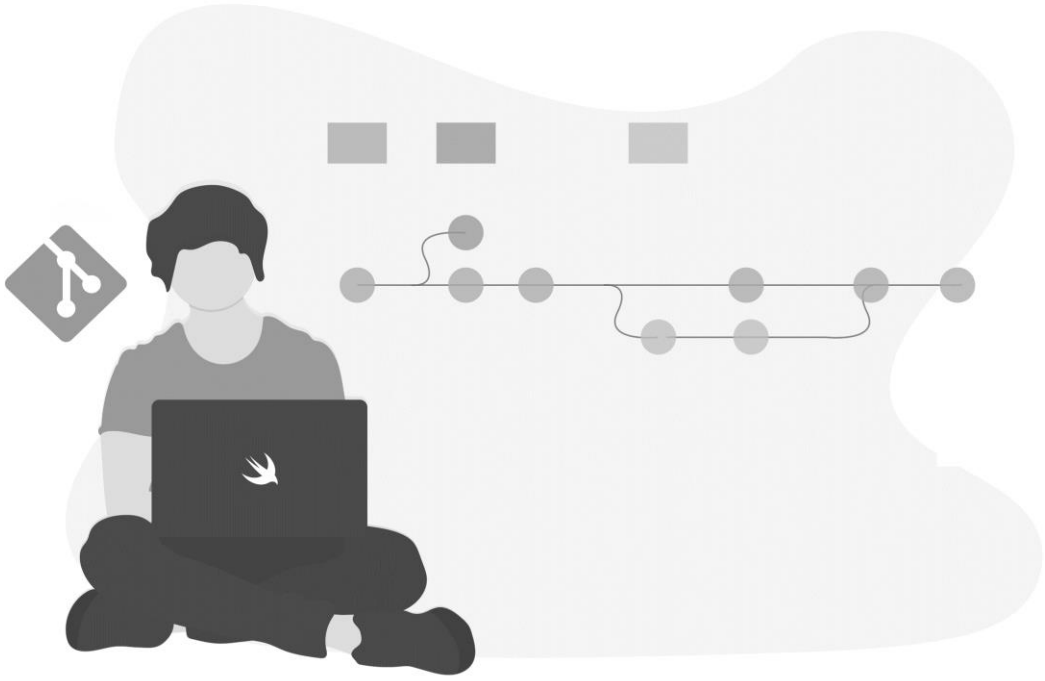
Netlify (https://www.netlify.com/) es un servicio de hosting en la nube que permite subir un sitio estático (deploy) en forma simple y gratuita. Para usarlo simplemente basta con registrarse en Netlify a través de una cuenta de Github, Gitlab, Bitbucket o creando una en base a un correo electrónico, si no cuenta con las anteriores.



Luego tenemos dos opciones para subir el sitio web: Arrastrando la carpeta local de nuestro equipo o hacerlo desde un repositorio de Github. Mostraremos de la primera forma ya que todavía no tenemos cuenta en Github. Una vez logueados con nuestro usuario y contraseña debemos seleccionar la carpeta con nuestro proyecto (debemos asegurarnos que exista dentro de la carpeta un archivo index.html), luego lo que resta hacer en este punto es simplemente arrastrar dicha carpeta.
Realizado este paso, ya tenemos un sitio alojado en Netlify con un subdominio generado por la empresa con https.

¿QUE ES UN CONTROL DE VERSIONES?

Un sistema de control de versiones (VCS por sus siglas en inglés) es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que pueda recuperar en cualquier momento versiones específicas. Si, por ejemplo, usted es un diseñador gráfico o un desarrollador web, y quiere mantener cada versión de una imagen o de un archivo HTML de un sitio web, una de las decisiones más importantes para su proyecto podría ser la de tener un sistema de control de versiones ya que permitiría revertir proyectos o parte de ellos a un estado anterior, comparar cambios a lo largo del tiempo, y ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, entre otras cosas más. Usar un control de versiones, si está en la nube, permite también tener una copia de los archivos en cualquier momento.



En un comienzo (y aun en la actualidad) el control de versiones se realizaba de forma local mediante la generación de directorios o carpetas renombradas con la fecha en la cual se realizaban cambios en los archivos involucrados en el desarrollo, método que generalmente lleva a la confusión y a la generación de errores, debido a que podemos tener una multitud de carpetas y archivos acumulados a lo largo del tiempo.
Todos estos inconvenientes a la hora de desarrollar software le dan vida a los sistemas de control de versiones actuales, primero a sistemas de control de versiones centralizados CVCS (Centralized Version Control Systems) como Subversion y Perforce en los que todos los archivos son almacenados en un servidor central desde donde los desarrolladores pueden descargar cada uno de los archivos a editar.
Con el tiempo, comienzan a surgir sistemas de control de versiones distribuidos DVCS (Distributed Version Control Systems) como Git, Mercurial, Bazaar o Darcs donde los desarrolladores clonan todo el repositorio directamente en su equipo, convirtiéndose en un respaldo completo del proyecto original y de tener la posibilidad de restaurarlo en caso de fallos o inconvenientes con el servidor.

Nota

Podemos hacer una analogía cuando dentro de un video juego, cada cierto tiempo, vamos grabando partidas del mismo. Cada vez que grabamos, guardamos una versión de nuestro juego, para no iniciar nuevamente desde el principio, y poder donde nos quedamos. En algunos juegos podemos "gestionar" y regresar a cualquier punto o "versión" del pasado donde hayamos grabado nuestro juego. Eso es un control de versiones.

GIT

Git es uno de los temas que es necesario dominar en la vida profesional de un desarrollador, es muy probable que la mayoría de empresas y proyectos con los que se vea involucrado usen Git en su día a día. GIT, un sistema de control de versiones gratuito, de código abierto, desarrollado por Linus Torvalds, uno de los principales desarrolladores del kernel de Linux, para manejar todo tipo de proyectos con rapidez y eficiencia. Actualmente es el control de versiones usado para los siguientes proyectos: el kernel de Linux, el Sistema Operativo Android, Rails, Fedora, Twitter, PostgreSQL, VLC y muchos más.

GIT se presenta como un sistema distribuido, en el que todos los nodos manejan la información en su totalidad y por lo tanto pueden actuar de cliente o servidor en cualquier momento, es decir, se elimina el concepto de "centralizado". Esto se logra gracias a que cada vez que se sincroniza los cambios con el repositorio remoto, GIT, guarda una copia entera de los datos con toda la estructura y los archivos necesarios.

Así ya no es necesario salir a Internet para consultar los cambios históricos sobre un archivo o para ver quién fue la última persona que lo editó, todo se hace directamente sobre la copia local y luego, cuando lo considere oportuno, puede enviar esos cambios hacia el repositorio remoto.

Otra gran diferencia con otros scvs que almacenan los archivos originales, conservando una lista de los cambios realizados a dichos archivos en cada versión, Git guarda una "foto" (snapshot) del estado de cada archivo en un momento concreto. Si uno de los archivos no ha cambiado no crea una nueva copia del mismo, simplemente crea una referencia al archivo original.

La segunda es la eficiencia. Dada su naturaleza distribuida, una vez que tengamos nuestro repositorio en forma local en nuestro working directory, podemos trabajar sin necesidad de conexión permitiendo que la velocidad de proceso dependa únicamente en los recursos locales. Todos los cambios se irán almacenando en forma local y cuando lo consideremos necesarios publicaremos los cambios sobre el repositorio en el que se sincronizan los cambios.

GITHUB

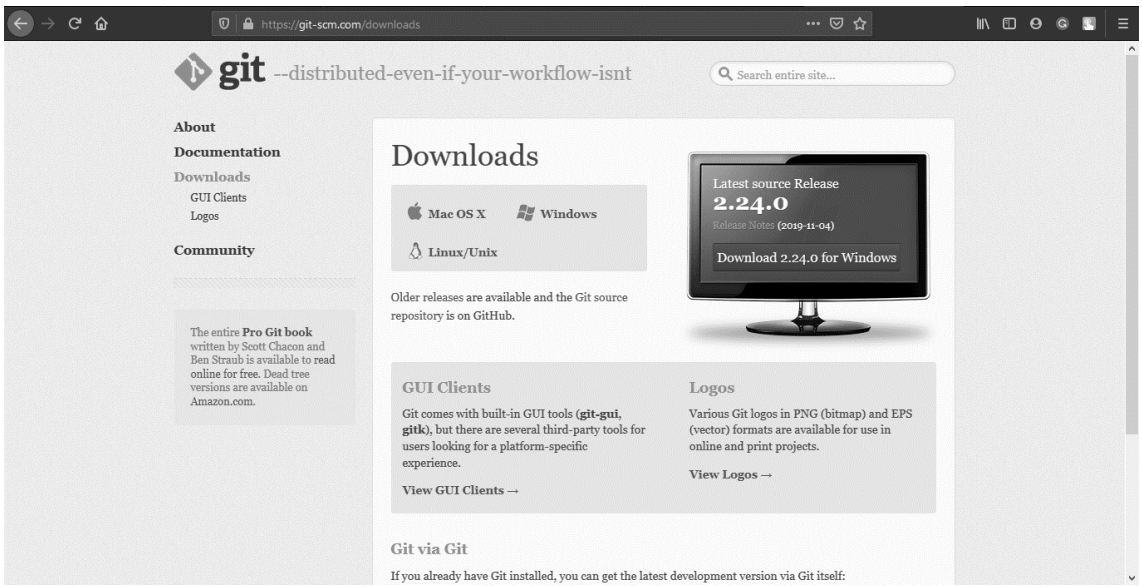
¿Y qué es GitHub? Un hosting online para los repositorios que utiliza Git para el mantenimiento y versionado del código fuente, añadiendo una serie de servicios extras para la gestión del proyecto y el código fuente. La versión gratuita de este hosting permite alojar nuestro código en repositorios públicos.

Nota

Tras la compra de GitHub por Microsoft en el año 2018. La empresa anuncio que se puede alojar en forma gratuita repositorios privados, pero con un número máximo de tres colaboradores por proyecto. En el caso de quiera tener más colaboradores deberá monetizar el servicio.

¿COMO TRABAJAR CON GIT?

Una forma es descargarse la versión para su sistema operativo, por ejemplo, la versión para Sistemas Operativos Windows puede descargarse de la siguiente WEB (https://git-scm.com/). Esta versión tiene tanto la versión de línea de comando como la interfaz gráfica de usuario estándar. Otra solución es trabajarlo directamente desde un IDE, esto tiene como ventaja que puede trabajarlo todo desde el propio entorno sin tener que salir del mismo.



REPOSITORIOS

Existen dos repositorios principales en un proyecto Git, el repositorio local y el remoto. El repositorio local cuenta con tres zonas, el primero es el directorio de trabajo (working directory) que contiene solo los archivos y carpetas del proyecto (los que estamos editando), el segundo es el INDEX (staging area), un área intermedia que va almacenando los archivos y carpetas que serán controlados en sus distintas versiones y finalmente con un commit lo enviamos al directorio de git.

Por lo tanto, el flujo de trabajo básico en Git es así:

- Se modifica una serie de archivos en el directorio de trabajo.
- Se prepara los archivos, añadiéndolos al área de preparación.
- Se confirma los cambios al mover los archivos tal y como están del área de preparación al directorio de Git.

Para guardar una versión de los cambios sobre el repositorio se realiza un commit, junto a ello se suele escribir un comentario indicando los cambios que se han realizado en el código. Una vez realizado un commit Git creara una nueva versión y generara una snapshot, de esta forma si en un futuro queremos revertir un cambio podremos volver a la versión anterior entre los commits realizados.
Imaginemos una línea recta donde cada cambio es representado por un circulo, está línea contiene todos los registros de cambios. Pero además se pueden generar "ramas" (branches) que son líneas alternas que se generan a la par de la línea principal (master). Con esta característica podemos generar un branch para agregar una nueva funcionalidad a nuestro proyecto sin afectar el contenido principal, por ejemplo, se podría tener dos ramas adicionales más la principal es la rama master (producción), la segunda rama para desarrollo y una tercera para pre-producción.
Para crear un repositorio remoto debemos ir a github (https://github.com/), y crearemos una nueva cuenta. Una vez creada la cuenta, debemos de terminar el proceso de registración yendo a nuestro correo electrónico y realizando los que nos piden. Luego iniciamos sesión y pedimos crear un nuevo repositorio yendo a la esquina superior derecha de la página, y luego realizando un click sobre la opción New Repository.

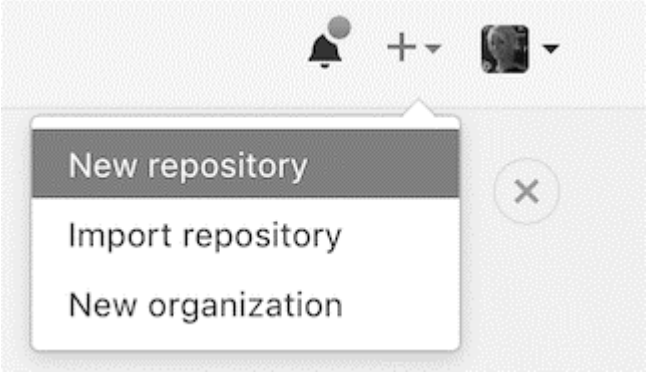


Figura 11.1. Creando un nuevo repositorio en GitHub.

Luego colocamos un nuevo nombre a nuestro proyecto y una descripción y entonces hacemos "click" en "Create repository". Otra opción interesante en este momento es crear un README.md, esto es un archivo de texto que mostrara a otras personas de que se trata específicamente el proyecto.

GIT EN VSCODE

Si aún no tiene instalado el software GIT en su computadora deberá descargarlo de https://gitforwindows.org/. Después de la instalación, abra una ventana con la terminal de Windows y ejecute estos dos comandos.

```
git config --global user.email "micorreoelectronico@dondesea.com"
git config --global user.name "Mi nombre real"
```

Estos dos comandos configuran las credenciales (usuario y correo) para trabajar con los repositorios, es necesario ejecutarlo la primera vez que trabaja con GIT y son necesarias para etiquetar los commits con sus datos.

CLONAR UN REPOSITORIO REMOTO

Clonar un repositorio extrae una copia integral de todo el proyecto alojado en GitHub que tiene en ese momento, incluyendo un registro de todo su historial para cada archivo y carpeta del proyecto. Para realizar esto debemos tener la dirección del repositorio que queremos clonar (generalmente es del tipo https://github.com/<username>/ <repo-name> .git):

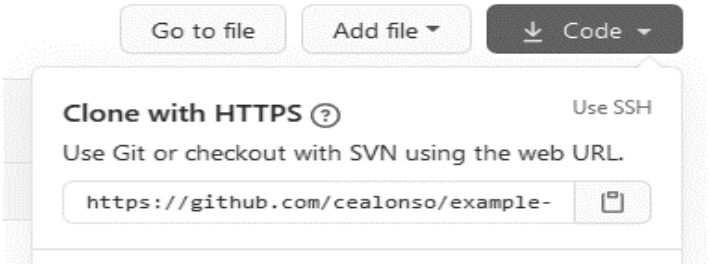


Figura 11.2. Nombre del repositorio de GitHub.

En VSCode a través de la paleta de comando (Ctrl+Shift+P) debemos tipear git clone (que es el comando de Git para clonar repositorios), después de esto nos pedirá la URL del directorio a clonar, la tipeamos y presionamos Enter. Luego nos pedirá la carpeta destino local donde se guardará el repositorio remoto, seleccionada la carpeta al momento tendremos una copia local de todo el repositorio remoto.

REPOSITORIOS LOCALES

Hay dos formas de crear un repositorio local, una es presionando el botón + del Control de Código en la Barra de Actividades y la otra yendo a la paleta de comandos (CTRL+SHIFT+P) y escribiendo:

```
git init
```

y luego seleccionando la opción Git: Inicializar repositorio.

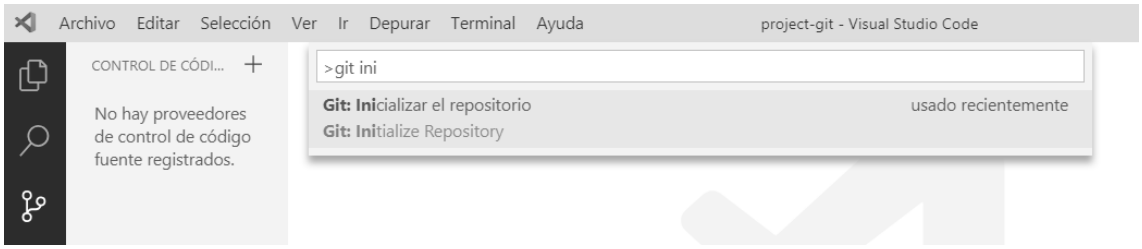


Figura 11.3. Inicializando un repositorio en GIT.

En cualquiera de los casos pedirá la ruta del repositorio local puede ser la misma ruta u otra que usted determine. También podrá observar el icono de Control de Fuente en la barra de actividad donde vera un circulo de color azul con el número de archivos que aún no tienen commit.
A partir de estos momentos VSCode creará una carpeta con el nombre .git dentro de su área de trabajo (no podrá ver esto de su Explorador de archivos de VSCode, ya que es un directorio oculto, pero puede encontrarlo en su administrador de archivos en la carpeta raíz de su proyecto).
Ahora vera que al costado de cada archivo aparece una U (Untracked) esto significa que hay cambios pendientes.
El próximo paso es llevarlo a la staging área para luego hacer un commit. Para ello de la Barra de Actividad seleccione el botón de Control de Código. Sobre cada archivo tendremos la posibilidad de enviarlo a dicha área presionando el símbolo + que aparece al costado del archivo.

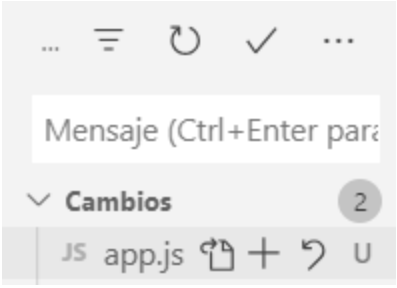
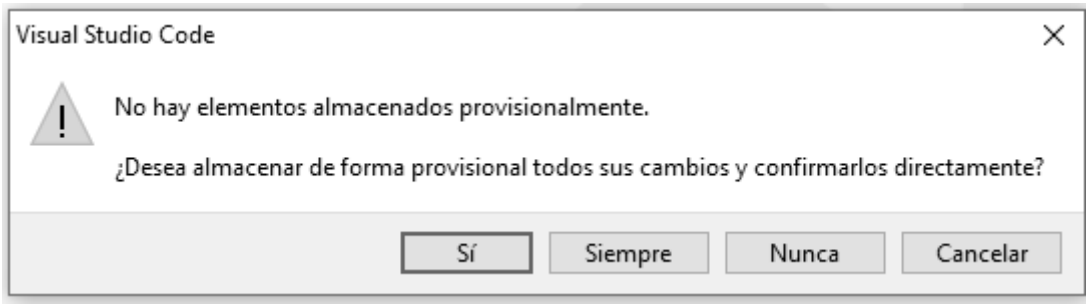
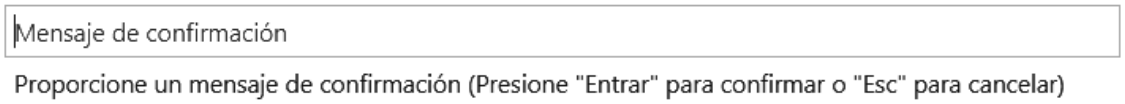


Figura 11.4. Commiteando en GIT.

También podemos directamente ponerlos a todos en el staging área y que haga un commit aplicando la tilde que se visualiza en la parte superior de la ventana. Luego de aplicar la tilde se visualiza la siguiente ventana:



Aplique el botón SI. A continuación, aparecerá una nueva ventana pidiendo el mensaje del commit, por ejemplo, coloque el mensaje: First Commit y presione ENTER.



Nota

¿Cuándo conviene hacer commit? Es una pregunta que no tiene una única respuesta correcta. Cada desarrollador tiene sus criterios. En general se realiza cuando el proyecto responde a nueva funcionalidad o un gran cambio.

Nota

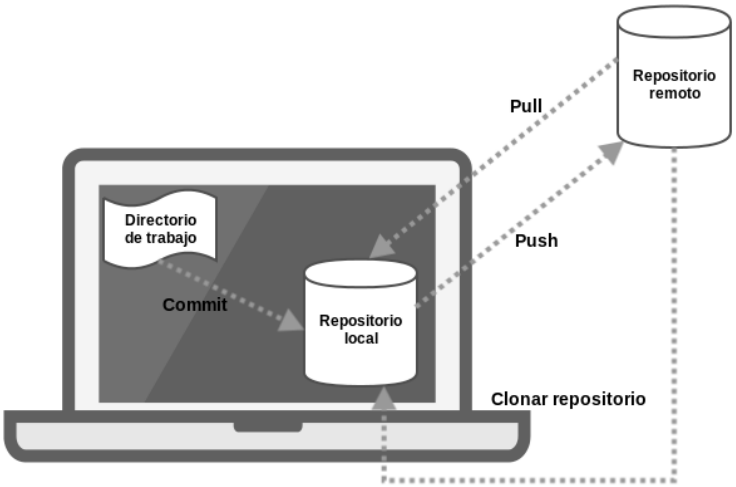
Cuando continua el trabajo después de hacer un commit es posible que quiera ver la diferencia entre el archivo actual y el commiteado, sobre el Control de Codigo realice un doble click sobre el archivo que desea analizar y VSCode mostrara la versión actual y la anterior en dos ventanas distintas para poder compararlas rápidamente.

Nota

El comando git status que es lanzado sobre la terminal permite saber el estado del repositorio en cualquier momento.

NUESTRO PROYECTO EN REMOTO

Todos los procesos de Git ocurren en forma local, sobre nuestra computadora. No obstante, podemos usar servicios como GitHub o BitBucket para alojar una copia de nuestros proyectos en remoto. Esto es una buena idea para tener una copia de seguridad en remoto y es imprescindible para trabajar en equipo. Cada persona del equipo puede subir sus commits al mismo repositorio en remoto. Colaborar con otros implica gestionar estos repositorios remotos, y mandar (push) y recibir (pull) datos de ellos cuando necesite compartir cosas.



Para realizar un PUSH es necesario indicar el repositorio remoto, es decir, la dirección del servidor remoto con el que deseamos sincronizar nuestros archivos. Para ello debe desde la terminal escribir el siguiente comando:

```
git remote add origin url_del_repositorio
```

En la mayoría de los casos el nombre del remote se define por defecto en la creación del repositorio con el nombre de origin (este es un alias para no recordar la dirección completa pero puede ser cualquier otro nombre). Para ver los remotos que tenemos configurados, debemos ejecutar:

```
git remote -v
```

Luego de esto puede ir a la parte derecha del Control de Fuente y seleccionar el icono de tres puntos donde aparecerá un menú. Desde el menú que aparece a continuación debe elegir la opción INSERTAR.

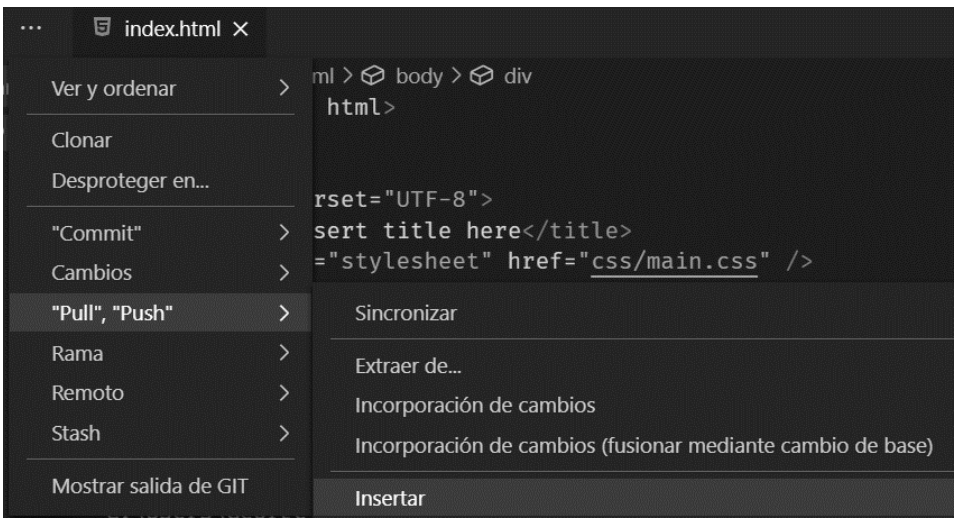


Figura 11.5. Realizar un push a un repositorio remoto.

Ahora vaya a su navegador y visite la página de su repositorio, en este momento deberá ver todos los archivos subidos.
Como consecuencia de lo expuesto nuestro flow (esquema) de trabajo será el siguiente:

1. Hacemos cambios en los archivos de trabajo del proyecto (workspace)
2. Añadimos al stage los cambios que luego se harán en el commit.
3. Realizo el commit.
4. Subo los cambios al remoto.

AÑADIENDO COLABORADORES AL PROYECTO

Para colaborar con un proyecto es necesario estar registrado para ese proyecto como colaborador en GitHub, para ello es necesario primero poseer una cuenta en GitHub. Luego el jefe de proyecto desde su cuenta dará acceso de escritura al repositorio a los miembros del proyecto que podrán modificar el código, de esta forma podrán funcionar los push. Desde su cuenta de GitHub deberá ir a la opción Settings y luego seleccionar su repositorio y agregar los colaboradores del proyecto.

COMO HACER UN PULL

Supongamos que alguien más ha realizado cambios en el código y los ha llevado al repositorio remoto. A través de GIT podemos obtener esos cambios y combinarlos en nuestro código. Esto es lo que se denomina hacer un PULL.

Nota

Puede realizar un PULL en cualquier momento, pero generalmente es mejor commitear su código antes de traer los cambios del repositorio remoto.

Para hacer un PULL deber ir a la parte derecha del Control de Fuente y seleccionar el icono de tres puntos, allí aparecerá un menú y desde allí debe elegir la opción EXTRAER. Si los cambios desde el repositorio externo difieren de lo que usted posee, los nuevos archivos, las actualizaciones y elecciones se aplicaran sobre su proyecto. Si las modificaciones se superponen con sus archivos GIT declarara un conflicto. VSCODE mostrara cuales son los conflictos y dará opciones de cómo resolverlo. Cuando haya corregido los conflictos deberá confirmar los cambios con un COMMIT.

SINCRONIZACION (PUSH y PULL)

Si queremos traernos los últimos cambios que se han realizado en nuestra rama y subir nuestros últimos podemos sincronizar mediante la opción de menú SINCRONIZACION. Esta opción realiza un PUSH y un PULL simultáneamente. Esto puede ser una buena opción si nadie más está trabajando en su código ya que debemos evitar los PUSH antes de hacer un PULL. En el caso de que haya cambios entrantes, debe volver a verificar el código del repositorio local antes de hacer un PULL.