Control de versiones con Git

UD1: Tecnologías para el desarrollo de interfaces

Objetivos de aprendizaje

- Comprender características básicas de Git
- Utilizar Git en un repositorio local y cambiar los archivos entre las distintas carpetas de trabajo.
- Conectarme y trabajar con repositorios remotos
- Crear una organización de trabajo para las distintas asignaturas

Al acabar la lección…

- Tendrás creada una cuenta de GitHub Education.
- Habrás creado un repositorio local y otro remoto, ambos sincronizados.
- Habrás clonado el repositorio con material de la asignatura.

Indice

Características de Git

Indice

Git

- Sistema que permite manejo de repositorios software, incluyendo:
  - Control de versiones.
  - Revisión de código.
  - Corrección de código.

Creado por Linus Trovards, creador de Linux!

¿Por qué?

- Facilitan la administración y gestión del código llevado a cabo en un equipo de trabajo.
  - Nos podemos centrar más en tareas de diseño e implementación.

Principales sistemas de gestión de versiones

- CVS
- Apache Subversion
- Git
- Mercurial

Características de Git

- Sistema **distribuido**
  - Frente a otras herramientas centralizadas como Subversion
- Es muy potente
- No depende de un repositorio central
  - Si pierdo conectividad puedo seguir creando hitos del estado de mi código
- **Software libre**
- Trabajar con ramas (*branches*) y fusiones (*merges*) de ramas de código es un proceso ágil.

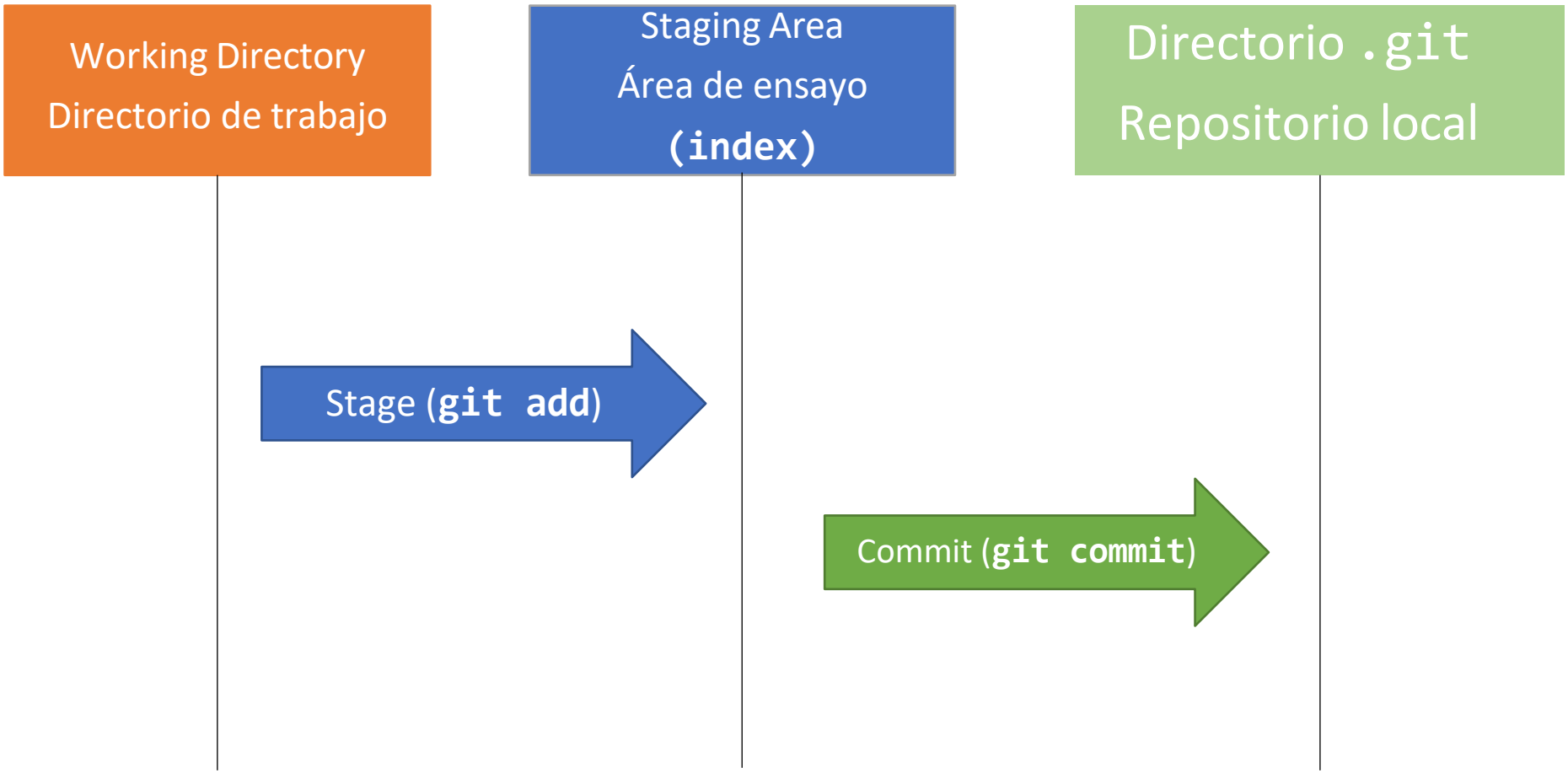
Estados de un fichero en Git

Modificado Untracked	Preparado Stagged	Confirmado **Commited**
• Has modificado el archivo pero aún no lo has confirmado a tu base de datos.	• Has preparado un archivo modificado en su versión actual para que vaya en tu próxima confirmación	• Los datos del archivo están almacenados de manera segura en tu base de datos local.
Estoy trabajando en el archivo		Cambios seguros

Estados de un fichero en Git

Modificado Untracked	Preparado Stagged	Confirmado **Commited**
• Has modificado el archivo pero aún no lo has confirmado a tu base de datos.	• Has preparado un archivo modificado en su versión actual para que vaya en tu próxima confirmación	• Los datos del archivo están almacenados de manera segura en tu base de datos local.
NO RECUPERABLE	RECUPERABLES	

Secciones de un proyecto Git



Trabajando con un repositorio local

Indice

Instalación de Git en equipo local

- Instalamos Git para Windows desde la página de Git
- Instalamos con las opciones por defecto salvo el editor por defecto que será VSCode

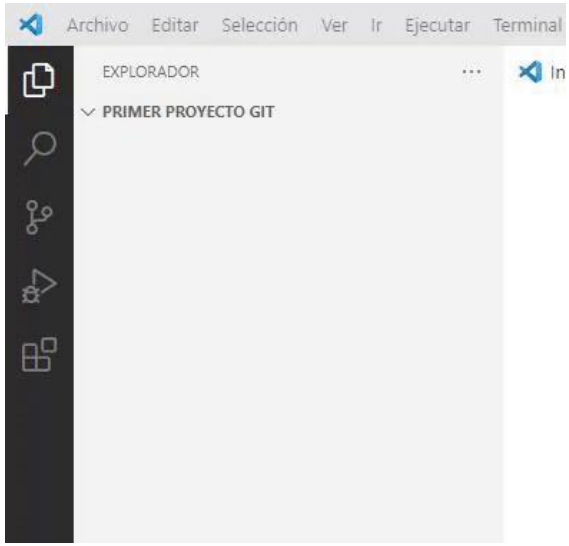
Trabajando con Visual Studio

- VScode tiene soporte para Git integrado, solo necesita que lo tengamos instalado en el equipo.
- Podemos instalar el plugin GitLens para añadir características adicionales

Creación de un repositorio local

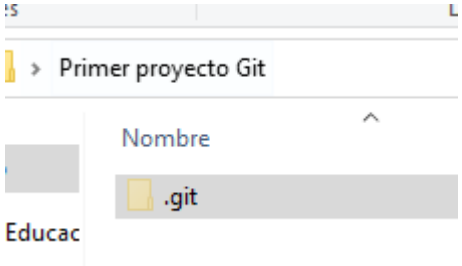
`git init` inicializa el repositorio local

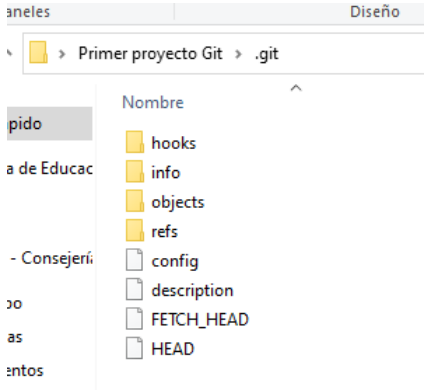
- Ejecutar desde la raíz del proyecto
- Desde VSCode podemos ir a Control de Código Fuente > Inicializar repositorio



Carpeta `.git`

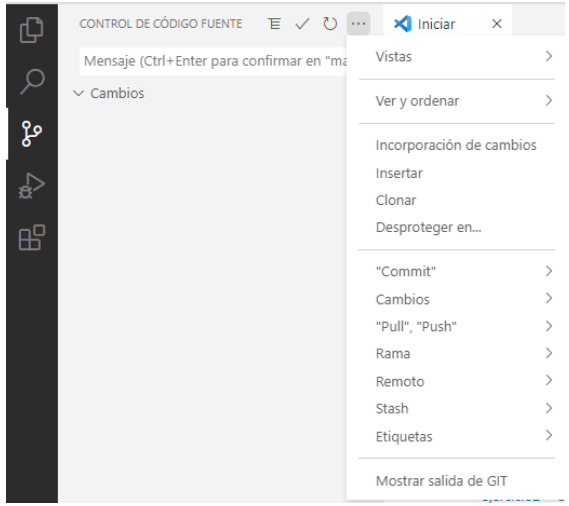
- Al iniciar un repositorio se crea una carpeta oculta `.git` en la raíz de mi proyecto.
- Contiene toda la información acerca de como manejar el repositorio local.
- Además es el Working Directory (a donde se pasan los cambios confirmados)





Trabajando en VSCode

- Desde VSCode podemos trabajar en Git con dos alternativas:
  - La pestaña “Control de código fuente” ofrece una vista del estado de mis ficheros y una lista de comandos Git
  - El Terminal nos permite introducir los comandos Git de manera textual



Creamos ficheros

Working Directory

- Creamos un proyecto `readme.md` y añadimos algo de contenido
- Veremos que se muestra con una U (Untracked) a su lado

`git status` permite comprobar el estado actual de los ficheros de mi proyecto [-s] → Formato corto

Incorporación de cambios

Staging Area

`git add fichero` incorpora cambios al stagging area

- `git add readme.m`
- `git add .`

- VSCode: Cambios > Almacenar todos los cambios
  - También podemos pulsar en el "+" al lado del nombre del archivo

- El fichero pasa a tener una A (Añadido)

Registrando cambios en el proyecto

`git commit` registra los cambios realizados en el proyecto

- Representa un hito de una cierta importancia
- Cada `commit` va acompañado de un comentario descriptivo que nos debería permitir identificarlo.
- Al resultado de cada commit lo denominamos **revisión**

Ejemplo: Hacer un **`commit`** tras cada ejercicio de una actividad

Consejos para escribir commits

Consejos para escribir commits

- Usar verbos imperativos
  - Que indique la funcioanlidad que se añade, no la que se ha hecho.
- Max 50 caracteres
- No usar puntos suspensivos
- Usar prefijos:

`feat` (característica)	**`fix`** (bug corregido)
`refactor` (cambio en la estructura del código sin funcionalidad	**`test`** (pruebas)

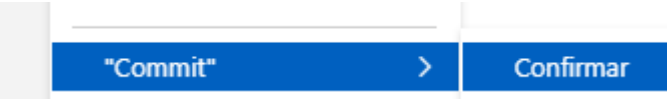
Aprobando cambios

Directorio `.git` (repositorio)

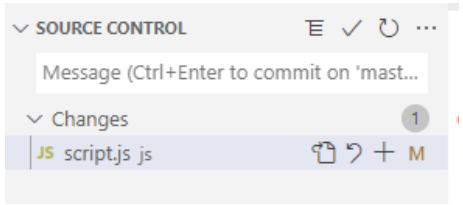
```
git commit -m 'Mensaje'
```

- `git commit -m 'Creamos el proyecto'`

- VSCode: “Comit” > Confirmar



- Podemos hacer un commit con el atajo en “Source Control”



Podemos comprobar que el archivo desaparece del área de cambios (o no aparece si hago `git status`)

Aprobando cambios

Directorio `.git` (repositorio)

```
git commit -am 'Mensaje'
```

- Agrega los cambios antes de hacer el commit
  - Evita tener que hacer `git add .` previamente

Aprobando cambios

Directorio `.git` (repositorio)

- En nuestro primer commit se nos pedirá, si no lo tenemos ya, que configuremos email y nombre en la configuración global de Git.

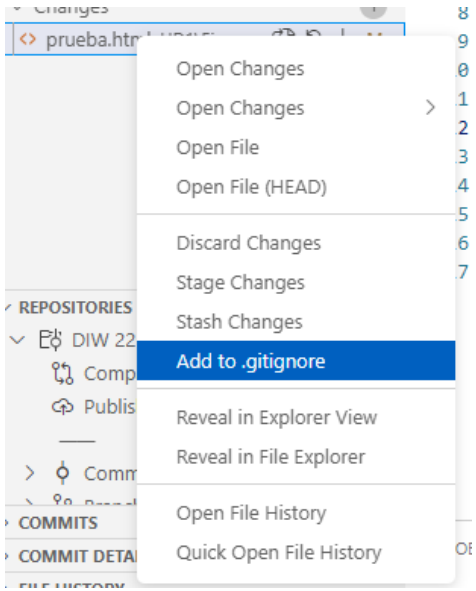
```
git config –-global user.name ‘Fulanito Perez'
git config –-global user.email 'fulanito@educastur.com'
```

Usa tu cuenta de Educastur

Fichero `.gitignore`

- Permite añadir una lista de los ficheros que no quiero incluir en mis revisiones
- Los ficheros (o la máscara) indicada no se va a agregar nunca al área de stagging.
- Ejemplo:

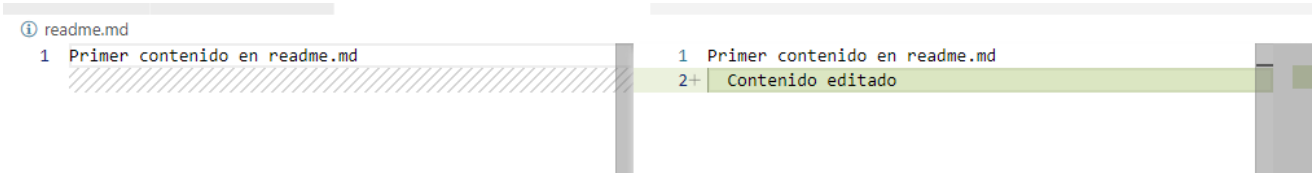
```
.gitignore
1   *.tmp
```



Comprobando cambios en VSCode

- Si modificamos el fichero podemos ver que pasa a tener una M (modificado)

- Si pulsamos sobre el fichero y escogemos "Abrir cambios" podemos ver la diferencia entre la última incorporación y el archivo modificado

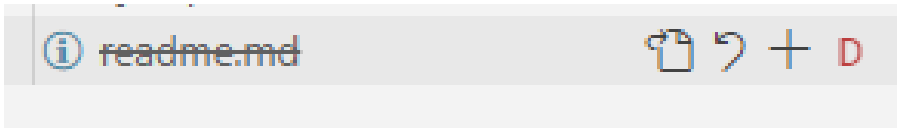


- Comando `git diff`
  - `git diff` → Compara área de trabajo y área de aprobación
  - `get diff --staged` → Compara área de preparación (stagged) y área de confirmación

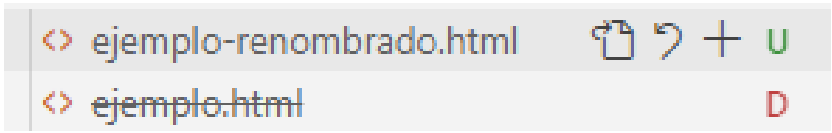
Nombrando y renombrando ficheros

- Podemos borrar o renombrar ficheros en nuestro proyecto
- Existen comandos para notificar a Git que lo hemos hecho.
  - `git rm fichero`
  - `git mv fichero`

- VSCode se da cuenta de los cambios que hago y marca los ficheros como borrados (D) o modificados (borra el anterior y añade el nuevo)



BORRADO



RENOMBRADO

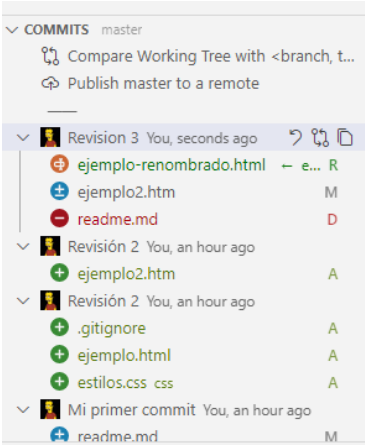
Comprobando revisiones

`git log [--oneline]` muestra los commits que hemos ido realizando

```
Javi@JGPISANO MINGW64 ~/Desktop/Primer Proyecto Git (master)
$ git log --oneline
3e6cf8c (HEAD -> master) Revisión 2
02b206e Revisión 2
92068bd Mi primer commit
9755684 Mi primer Commit
```

HEAD Es la versión más reciente que tenemos en el histórico de cambios

- Gracias al plugin GitLens podemos ver los cambios en la barra COMMITS



Trabajando con un repositorio remoto

Indice

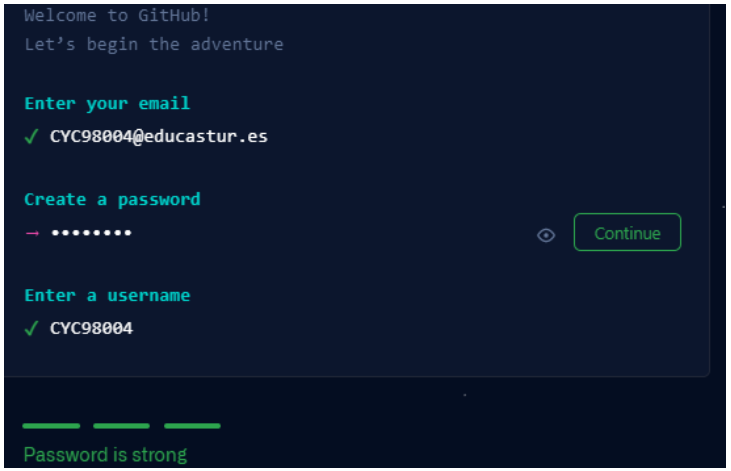
GitHub y GitLab

- Servicios de hosting remoto para Git
  - GitLab es además un servidor que podemos descargar e instalar de manera local
    - Más ventajoso si queremos guardar los datos de manera privada.
- Ambos propocionan una interfaz Web sobre Git.
- Ambos proveen características de control de acceso y herramientas como gestión de tareas

GitHub proporciona acceso a herramientas muy interesantes para cuentas de estudiante

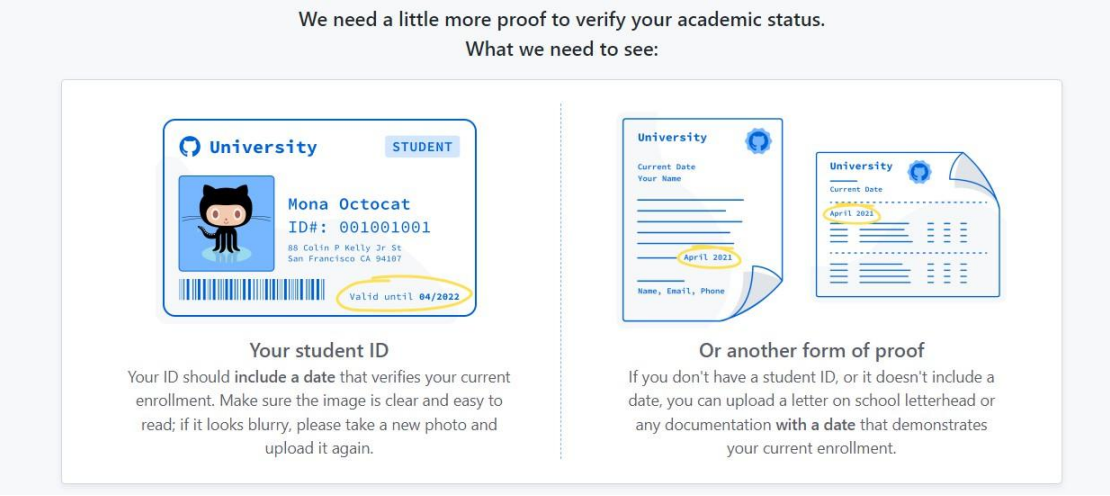
Creación de cuenta en GitHub Education

- Entramos en https://education.github.com/
  - Join Global Campus > Sign Up for Student Developer Pack > Yes, _I am a student
- Crea una cuenta **con tus credenciales de Educastur** (Username=login educastur).



Creación de cuenta en GitHub Education

- Verificamos según el código que nos envían al email.
- Al acabar el proceso tendremos que verificar nuestra identidad académica subiendo un **justificante de matrícula**.



GitHub
Student Developer Pack

Cuenta de GitHub

- Una vez que hayáis creado la cuenta de GitHub, anotad vuestro login junto con vuestro nombre en la lista de Teams
- Os voy a invitar al grupo **DESIN 25-26** dentro de la organización **DAM2A 25-26**
- Tendréis que aceptar la invitación desde vuestra pestaña "*Organizations*"

Creando un repositorio remoto

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Required fields are marked with an asterisk (*).

Owner * DAM2A-DESIN-24-25 / Repository name * DESIN-Tony-Stark

DESIN-Tony-Stark is available.

Great repository names are short and memorable. Need inspiration? How about upgraded-carnival ?

Owner: [DAM2A-DESIN-25-26]

Description (optional)

Nombre: DESIN-Nombre Apellidos

- Public
  Anyone on the internet can see this repository. You choose who can commit.
- Internal
  IES Monte Naranco enterprise members can see this repository. You can choose who can commit.
- Private
  You choose who can see and commit to this repository.

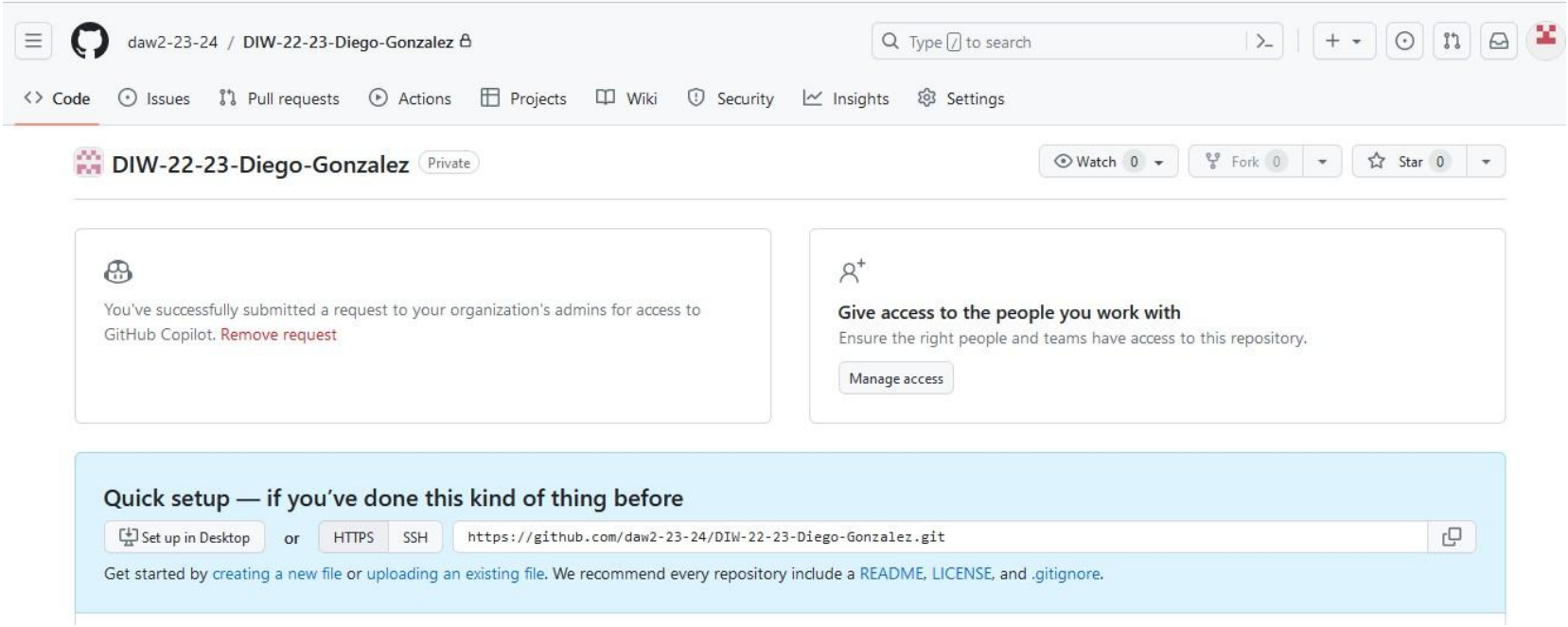
Privado (im portante!)

Initialize this repository with:

- Add a README file
  This is where you can write a long description for your project. Learn more about READMEs.

Creando un repositorio remoto

- Una vez creado se nos informa acerca de cómo podemos comenzar a usar el repositorio



Nos importa la URL del repositorio, del tipo `https://github.com/username/DEMO.git`

Conectando a un repositorio remoto (Opción A)

`git clone path_remoto` **clona** el repositorio remoto en nuestro equipo

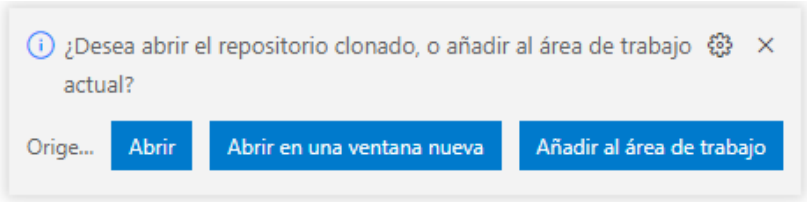
```
git clone https://github.com/username/DEMO
```

`Control + Shift + p` → `Clone`

No es necesario tener el repositorio iniciado en VSCode

Conectando a un repositorio remoto

- Si estamos en un proyecto de VSCode, cuando clonamos se nos da la opción de abrir un nuevo Worskspace o integrarlo en el propio
- En todos los casos se creará **una subcarpeta con el nombre del repositorio que estamos clonando**.



Importante: ESCOGER LA CARPETA
PADRE DONDE QUEREMOS QUE SE CREE EL REPO

Renombrando la rama local

- La rama por defecto en VSCode se llama `master` mientras que en GitHub es `main`
  - Pulsa aquí para ver por qué
- Un paso previo consiste en renombrar mi rama `master` a `main`

```
git branch –M master main
```

Más adelante ampliaremos conceptos sobre ramas

Conectando a un repositorio remoto (Opción B1)

ESCENARIO B1: Tengo contenido remoto y mi repositorio local está vacío

```
git init
```

Iniciamos nuestro propio repositorio Git

```
git remote add origin url_remota
```

Añadimos repositorio remoto

`origin` es el nombre por convención del repositorio remoto (evita usar el path)

```
git fetch
```

Permite comprobar qué cambios ha habido en el repositorio remoto

```
git pull origin main
```

Extrae y descarga contenido desde la rama `main` del repositorio remoto, actualizando el repositorio local

Conectando a un repositorio remoto (Opción B2)

ESCENARIO B2: Tengo contenido local y mi repositorio remoto está vacío

```
git init
```

Iniciamos nuestro propio repositorio Git

```
git remote add origin url_remota
```

Añadimos repositorio remoto

```
git fetch
```

Permite comprobar qué cambios ha habido en el repositorio remoto

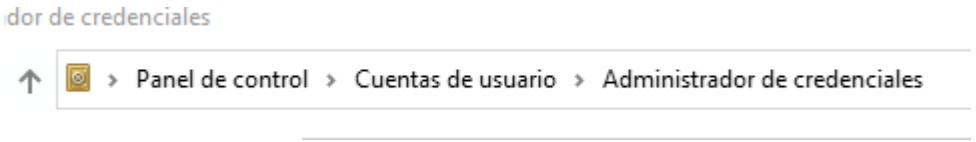
`origin` es el nombre por convención del repositorio remoto (evita usar el path)

```
git push origin main
```

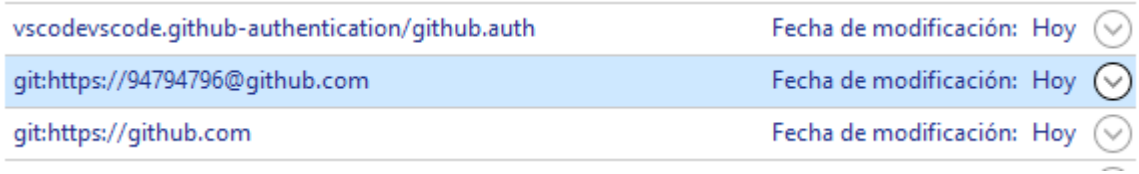
Añade contenido desde la rama `main` del repositorio local, actualizando el repositorio remoto

Credenciales GitHub

- La primera vez que conectamos VSCode a GitHub, en caso de usar un repositorio privado, se nos pedirán las credenciales del usuario en GitHub
- Estas se almacenan en el administrador de credenciales de Windows

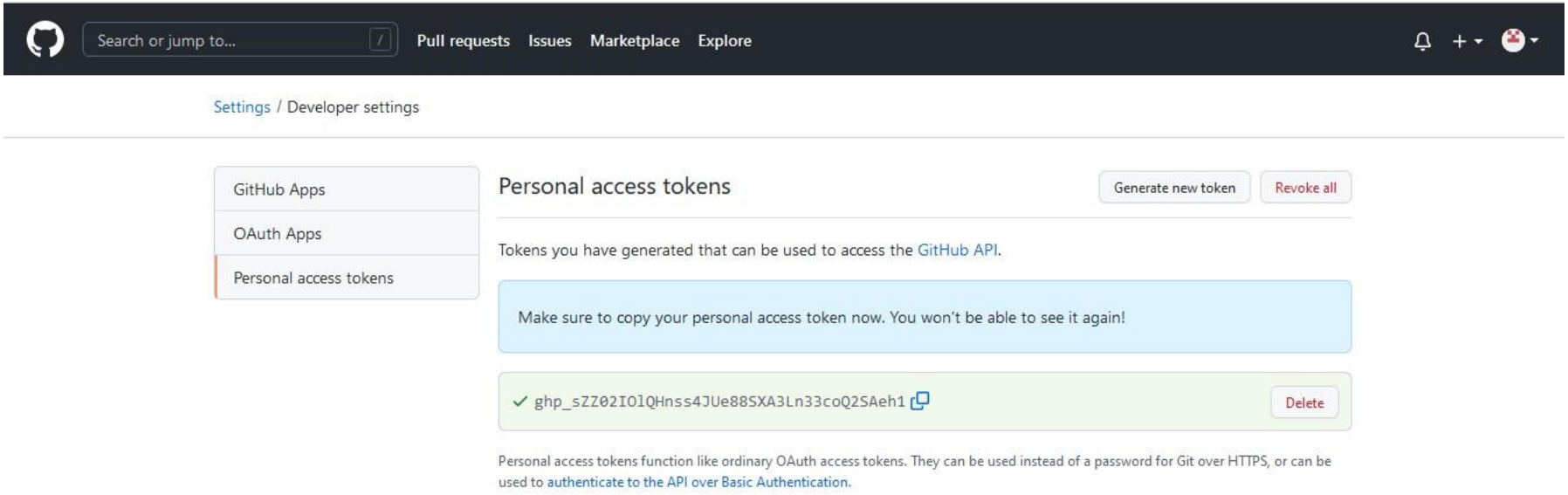


- Si queremos conectarnos con otro usuario deberemos eliminar las claves relacionadas con GitHub



Credenciales GitHub (II)

- Otra opción es conectarnos con un Personal Access Token (PAT)
  - Se generan en GitHub desde Settings > Developer Settings > Personal Access Tokens > Generate New Token
  - Nos aseguramos de dar los permisos necesarios



Resumen Opción A (Clonado)

Guardamos nuestro trabajo en una carpeta temporal		EN CLASE 
Creamos un repositorio en Github (vacío)		
Lo clonamos sobre nuestra carpeta padre (en local)	`git clone url_remota`	
Copiamos manualmente nuestro trabajo desde la carpeta temporal		
Añadimos cambios y hacemos commit	`git commit –am 'Prueba Git'`	
Hacemos un push a Github	`git push origin main`	
Clonamos nuestro repositorio desde GitHub a local (ya tendrá contenido)	`git clone url_remota`	EN CASA (EQUIPO DISTINTO) 

Resumen opción B2 (Sincronización manual)

Inicializamos repositorio local sobre nuestra carpeta de trabajo	`git init`	EN CLASE 
Renombramos la rama master	`git branch -M master main`	
Añadimos cambios y hacemos commit	`git commit -am 'Prueba Git'`	
Creamos un repositorio en Github (vacío)		
Añadimos repo remoto	`git remote add origin url_remota`	
Sincronizamos repos	`git fetch`	
Subimos contenido a GitHub	`git push origin main`	
Clonamos nuestro repositorio desde GitHub a local (ya tendrá contenido)		EN CASA (EQUIPO DISTINTO) 

Sincronizando con repositorio remoto

`git push` envía al repositorio remoto las revisiones (commits)locales


```
git push origin main
```

Escogemos la rama por defecto [**main**] tanto en local como en remoto

`git pull <remote> <branch>` extrae los commits desde la rama remota (sincroniza)

push en GitHub

- Una vez que hemos hecho el **push** podemos ver que los archivos se han actualizado en el repositorio remoto

`pull` y `push`

Se recomienda:

- Finalizar cada sesión de trabajo con **`commit`** + **`push`**
- Comenzar cada sesión de trabajo con **`pull`**

Así minimizamos el número de comparaciones “manuales” a realizar (poco deseable)

Workflow

Se recomienda:

- Finalizar cada sesión de trabajo con `commit` + `push`
- Comenzar cada sesión de trabajo con `pull`

Así minimizamos el número de comparaciones “manuales” a realizar (poco deseable)

Workflow

`git pull` no es necesario
si trabajo en el mismo equipo en casa y clase

CLASE

CASA

`commit`
`(N veces)`

`git push`

`git pull`

`commit`
`(N veces)`

`git push`

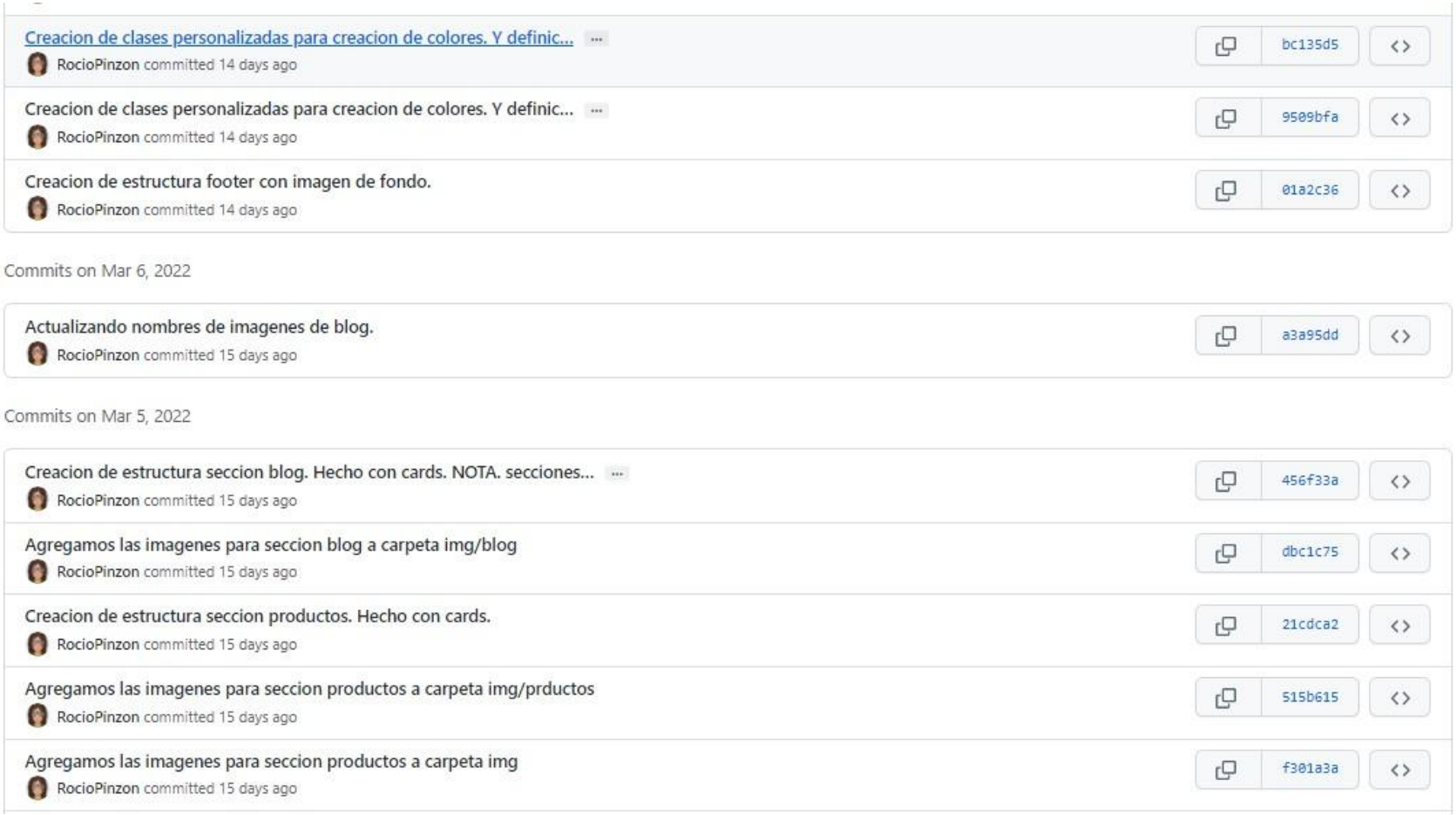
`git pull`

Expectativa sobre vuestro trabajo

- Cada sesión de trabajo debería llevar al menos un commit
- El mensaje de dicho commit será el indicativo de la funcionalidad que vayáis añadiendo trabajo realizado CLASE ó CASA

```
commit –m 'UD1 Ejemplos FlexBox CLASE'
commit –m 'UD1 ACT1 Ejercicio2 CLASE'
commit –m 'UD1 ACT1 Ejercicio3 CASA'
commit –m 'UD1 ACT4 Menú funcional CASA'
```

Mensajes descriptivos....



GitHub

Repositorio de la asignatura

- Clona el repositorio de la asignatura:

```
https://github.com/DAM2A-DESIN-25-26/DESIN-MATERIAL.git
```

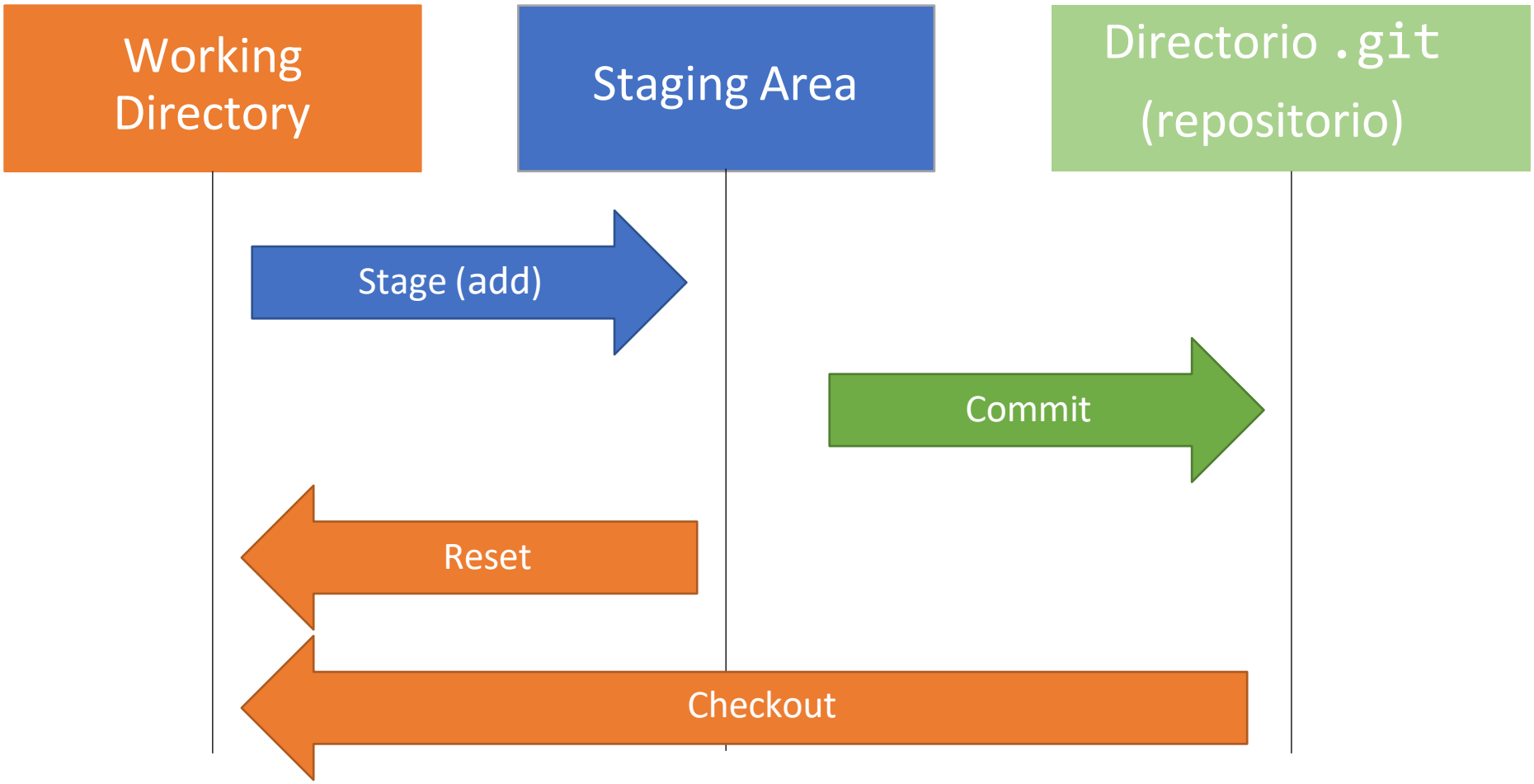
- Es el método que se utilizará para proporcionar código de ejercicios y ejemplos de clase.

De este repositorio podéis leer (clonar y hacer pull) pero no podéis escribir (hacer push)

Control de cambios

Indice

Deshaciendo cambios



Deshaciendo cambios

`git reset` deshace los cambios añadidos

- Pasa de stage area a working directory
- Opción `git reset --hard`
  - Hace checkout (pasa del área commit al working directory)
  - Elimina el contenido del área de stagging.

Utilizar con cuidado!

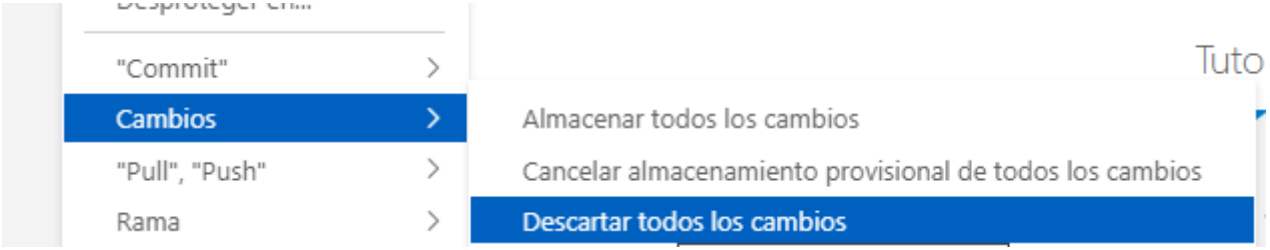
Deshaciendo cambios (I)

`git checkout` deshace los cambios de una versión (commit)

- Pasamos de área de commit a working directory
- Permite especificar:
  - Nombre de un archivo (modificador `--`)
    - `git checkout HEAD -- .`
    - Recupera todos los archivos de la revisión anterior.
  - Nombre de una rama.
    - El nombre por defecto de la rama principal es `master`
  - De qué revisión quiero coger algo especificando su hash
  - Modificador `~N` para referirme a N revisiones atrás

Checkout en VSCode

- Cambios > Descartar todos los cambios



Git y GitHub CheatSheet

Cheatsheet Git básico + Github

Completa - Autor: Sergi García Barea

Crear y compartir repositorios privados en Github

- Enlace visibilidad repositorio: https://docs.github.com/es/github/administering-a-repository/managing-repository-settings/setting-repository-visibility
- Enlace compartir repositorio: https://docs.github.com/es/github/setting-up-and-managing-your-github-user-account/managing-access-to-your-personal-repositories/inviting-collaborators-to-a-personal-repository

Token acceso para trabajar con repositorios

- Crear token de acceso Github (**Obligatorio: Actúa de contraseña para trabajar con repositorios**): https://docs.github.com/es/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token

Establecer usuario y email Github (solo primera vez)

```
git config --global user.name "NOMBREUSUARIO GITHUB"
git config user.email "EMAILCUENTAGITHUB@SERVIDOR.COM"
```

- La primera vez que usamos git con Github, deberemos configurar estos parámetros. Esto indica las credenciales al conectarnos a cuentas Github para manipular repositorios.

Usando "git clone" para clonar un repositorio

```
git clone (dirección del repositorio)
```

- Esta orden clona un repositorio de Github en tu máquina local.
  - Si el repositorio es privado te pedirá tu cuenta de usuario y una contraseña. Esa contraseña **NO ES LA CONTRASEÑA DE TU CUENTA**, sino el token personal

GitHub

HOJA DE REFERENCIA PARA GITHUB GIT

Git es el sistema de control de versiones distribuido de fuente abierta que facilita las actividades de GitHub en su computadora portátil o de escritorio. Esta hoja de referencia rápida resume las instrucciones de las líneas de comando de Git más comúnmente usadas.

INSTALAR GIT

GitHub le ofrece a los clientes de computadoras de escritorio que incluye una interfaz gráfica de usuario para las acciones de repositorio más comunes y una edición de línea de comando de actualización automática de Git para escenarios avanzados.

GitHub para Windows
https://windows.github.com

GitHub para Mac
https://mac.github.com

EFECTUAR CAMBIOS

Revisa las ediciones y elabora una transacción de commit

```
$ git status
```

Enumera todos los archivos nuevos o modificados que se deben confirmar

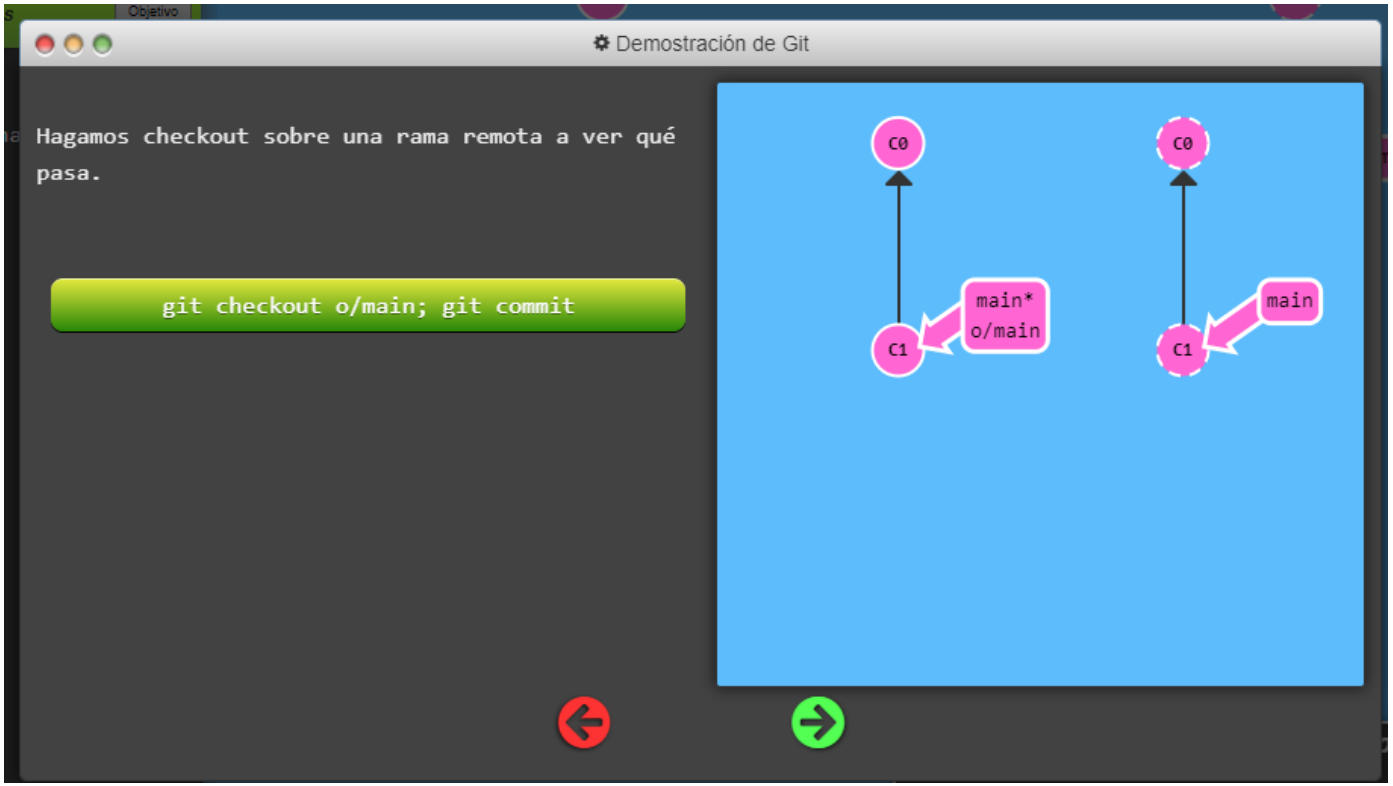
```
$ git diff
```

Muestra las diferencias de archivos que no se han enviado aún al área de espera

[CheatSheet #1]

[CheatSheet #2]

Aprendiendo Git



Otros enlaces

- Git- La guía sencilla
- Aprende Git en una hora
- Curso de Github con VSCode [Video]
- Documentación completa de Git