

Módulo: Entornos de Desarrollo

Unidad 6 – Optimización documentación y control de versiones

Control de Versiones

Descripción:

Para practicar como se usa un control de versiones vamos a trabajar con Git, realizaremos una serie de ejercicios para completar las siguientes tareas:

- Creación y actualización de Repositorios.
- Trabajar con el historial de cambios.
- Deshacer cambios.
- Trabajar con ramas
- Trabajar con repositorios Remotos

Objetivos:

- Utilizar herramientas de control de versiones.

Recursos:

- Git
- Cuenta en Github.

Resolución:

Creación y actualización de Repositorios

Ejercicio 1

Configurar Git definiendo tu nombre de usuario, el correo electrónico y activar el coloreado de la salida. Mostrar la configuración final. Obviamente en los comandos cuando aparezca mis datos o mi nombre deberán ir los vuestros.

```
> git config --global user.name Alfredo Valero
> git config --global user.email alfredo.valero@foc.es
> git config --global color.ui auto
> git config --list
```

Ejercicio 2

Ahora vamos a crear un repositorio nuevo. Primero crea una carpeta con tu nombre y apellido donde quieras, dentro abre una terminal, ve hasta la carpeta y crea el repositorio. Una vez hayas terminado muestra el contenido de la carpeta.

Crea la carpeta, en la terminal escribe:

```
> cd C:\Users\alfredo.valero\Documents\alfredoalero
> git init
> dir
```

Ejercicio 3

Comprobar el estado del repositorio.

```
> git status
```

Crear un fichero *sobremi.txt* con el siguiente contenido, puedes hacerlo con el bloc de Nota si quieres:

Nombre: Alfredo Valero

Profesión: Ingeniero informático

Aficiones: La montaña, la música, el Deporte,

Comprobar de nuevo el estado del repositorio.

```
> git status
```

Añadir el fichero al área de staged.

Volver a comprobar una vez más el estado del repositorio.

```
> git add sobremini.txt
```

```
> git status
```

Ejercicio 4

Realizar un commit de los últimos cambios con el mensaje “Añadido información personal” y ver el estado del repositorio.

```
> git commit -m " Añadido información personal "
```

```
> git status
```

Ejercicio 5

Cambiar el fichero sobremini.txt para que contenga lo siguiente:

Nombre: Alfredo Valero

Profesión: Ingeniero informático

Aficiones: La montaña, la música, el Deporte,

Película Favorita: Star Wars

Mostrar los cambios con respecto a la última versión guardada en el repositorio.

```
> git diff
```

Hacer un commit de los cambios con el mensaje “Añadido Película Favorita”.

```
> git add sobremini.txt
```

```
> git commit -m "Añadida Película Favorita"
```

Ejercicio 6

Mostrar los cambios de la última versión del repositorio con respecto a la anterior.

```
> git show
```

Cambiar el mensaje del último commit por "Añadida mi película favorita de Ciencia ficción."

```
> git commit --amend -m " Añadida mi película favorita de Ciencia  
ficción."
```

Volver a mostrar los últimos cambios del repositorio.

```
> git show
```

Para ver en qué estado quedan los commit en el repositorio local, ejecutar ver el historial de cambios:

```
> git log
```

Historial de Cambios

Ejercicio 7

Crear la carpeta Ciudades y crear dentro de ella el fichero CiudadesDondeVivo.txt con el siguiente texto.

Granada: es la ciudad donde vivo y estudio.

Añadir los cambios a la zona de intercambio temporal.

```
> git add .
```

Hacer un commit de los cambios con el mensaje "Añadido Ciudades donde vivo."

```
> git commit -m " Añadido Ciudades donde vivo."
```

Volver a mostrar el historial de cambios del repositorio.

```
> git log
```

Ejercicio 8

Crear el fichero CiudadesVisitadas.txt en la carpeta Ciudades con el siguiente texto.

Españolas: Almería, Málaga, Jaén, Córdoba, Sevilla, Madrid, Valencia, Barcelona, Cáceres, Vigo, Coruña, Santa Cruz de Tenerife, San Cristóbal de la Laguna, Santa Cruz de la Palma.

Extranjeras: Lisboa, Ponta Delgada, Gibraltar, Londres, Edimburgo, París, Ámsterdam, Bruselas, Brujas, Bérgamo, Venecia, Liubliana.

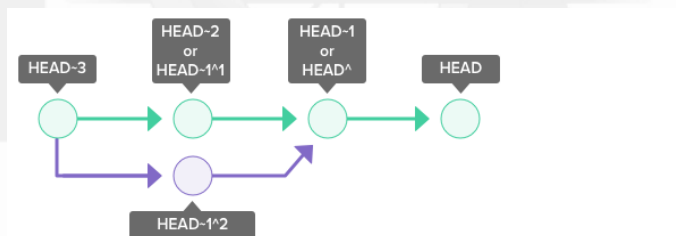
Añadir los cambios a la zona de intercambio temporal.

Hacer un commit de los cambios con el mensaje "Añadido ciudades visitadas."

Mostrar las diferencias entre la última versión y dos versiones anteriores.

```
> git add .
> git commit -m "Añadido ciudades visitadas"
> git diff HEAD~2..HEAD
```

Nota: HEAD es un puntero que apunta al último commit. HEAD~1 es otro puntero que apunta al commit anterior al que apunta HEAD, es decir al penúltimo commit. HEAD~2 puntero al antepenúltimo commit. HEAD~3 ...



Nota: La virgulilla (~) se añade con alt +126 del Teclado numérico.

Ejercicio 9

Crear el fichero CiudadesPorVisitar.txt en la carpeta Ciudades con el siguiente texto.

Españolas: Cádiz, Huelva, San Sebastián, Palma de Mallorca, Gijón, Toledo, Cuenca.

Extranjeras: New York, Viena, Roma, Florencia, Casa Blanca, Ifrán, Pekin, Kioto

Añadir los cambios a la zona de intercambio temporal.

Hacer un commit de los cambios con el mensaje “Añadido Ciudades por visitar”

Mostrar las diferencias entre la primera y la última versión del repositorio.

```
> git add .
> git commit -m "Añadido Ciudades por visitar"
> git log
```

En git log podemos ver el código hash de la última versión.

```

C:\Users\alfredo.valero\Documents>git log
commit 18f514084680d31e0e1fc929e0b544275c88e712 (HEAD -> master)
Author: alfredovalero <alfredo.valero@civica-soft.com>
Date: Tue Apr 20 14:27:16 2021 +0200

    Añadido Ciudades por visitar

commit b7024a66343ae2202a98025a596db12241010981
Author: alfredovalero <alfredo.valero@civica-soft.com>
Date: Tue Apr 20 13:51:09 2021 +0200

    Añadido ciudades visitadas

commit 0d5b23f7fd35a3b84800b1d6c552d903473d4c4d
Author: alfredovalero <alfredo.valero@civica-soft.com>
Date: Tue Apr 20 13:41:39 2021 +0200

    Añadido Ciudades donde vivo.

commit 8fe0e370258a4285d479525edddf2e26e8ba627a (origin/master)
Author: alfredovalero <alfredo.valero@civica-soft.com>
Date: Sat Apr 17 18:05:50 2021 +0200

    Añadida mi película favorita de Ciencia ficción.

commit 6a6cdfbda5ebaeb00c616f4c7d1043592007bfff1
Author: alfredovalero <alfredo.valero@civica-soft.com>
Date: Sat Apr 17 18:05:50 2021 +0200

    Añadido información personal
  
```

```
> git diff 6a6cd..HEAD
```

Nota: El código hash es como la matrícula del commit, el identificador único de ese commit. Cuando referenciamos un código lo hacemos con su código hash y no es necesario escribirlo entero basta con los primeros 5 caracteres.

Ejercicio 10

Añadir al final del fichero *sobremi.txt* la siguiente línea:

Ciudad Favorita: Granada

Añadir los cambios a la zona de intercambio temporal.

Hacer un commit de los cambios con el mensaje "Añadido ciudad favorita a sobre mí"

Mostrar quién ha hecho cambios sobre el fichero *sobremi.txt*.

```
> git add .
```

```
> git commit -m " Añadido ciudad favorita a sobre mi "
```

```
> git annotate sobremi.txt
```

Deshacer Cambios

En los siguientes ejercicios se recomienda ir viendo en el directorio Ciudades y en el archivo sobremi.txt que pasa antes y después de la ejecución de los comandos checkout o reset.

Ejercicio 11

Eliminar la última línea del fichero sobremi.txt y guardarlo.

Comprobar el estado del repositorio.

Deshacer los cambios realizados en el fichero sobremi.txt para volver a la versión anterior del fichero.

Volver a comprobar el estado del repositorio.

Eliminar la última línea y guardar el fichero.

```
> git status
```

```
> git checkout -- sobremi.txt
```

```
> git status
```

Ejercicio 12

Eliminar la última línea del fichero sobremi.txt y guardarlo.

Añadir los cambios a la zona de intercambio temporal.

Comprobar de nuevo el estado del repositorio.

Eliminar la última línea y guardar el fichero.

```
> git add .
```

```
> git status
```

Quitar los cambios de la zona de intercambio temporal, pero mantenerlos en el directorio de trabajo.

Comprobar de nuevo el estado del repositorio.

```
> git reset sobremi.txt  
> git status
```

Deshacer los cambios realizados en el fichero sobremi.txt para volver a la versión anterior del fichero.

Volver a comprobar el estado del repositorio.

```
> git checkout -- sobremi.txt  
> git status
```

Ejercicio 13

Eliminar la última línea del fichero sobremi.txt y guardarlo.

Eliminar el fichero ciudades/ CiudadesPorVisitar.txt

Añadir un fichero nuevo ciudades /CiudadesPatrimonio.txt vacío.

Añadir los cambios a la zona de intercambio temporal.

Comprobar de nuevo el estado del repositorio.

Eliminar la última línea y guardar el fichero.

Elimina el fichero CiudadesPorVisitar.txt de la carpeta Ciudades

Añadir un fichero nuevo ciudades /CiudadesPatrimonio.txt vacío

```
> git add .  
> git status
```

Quitar los cambios de la zona de intercambio temporal, pero mantenerlos en el directorio de trabajo.

Comprobar de nuevo el estado del repositorio.


```
> git reset  
> git status
```

Deshacer los cambios realizados para volver a la versión del repositorio.

Volver a comprobar el estado del repositorio.

```
> git checkout -- .  
> git status  
> git clean -f  
> git status
```

Ejercicio 14

Eliminar la última línea del fichero sobremini.txt y guardarlo.

Eliminar el fichero ciudades/ CiudadesPorVisitar.txt.

Añadir los cambios a la zona de intercambio temporal y hacer un commit con el mensaje "Borrado accidental." en un solo comando.

Comprobar el historial del repositorio.

Eliminar la última línea y guardar el fichero.

Elimina el fichero CiudadesPorVisitar.txt de la carpeta Ciudades

```
> git commit -a -m "Borrado accidental."  
> git status  
> git log
```

Deshacer el último commit pero mantener los cambios anteriores en el directorio de trabajo y la zona de intercambio temporal.

Comprobar el historial y el estado del repositorio.

```
> git reset --soft HEAD~1  
> git log
```

```
> git status
```

Volver a hacer el commit con el mismo mensaje de antes.

Deshacer el último commit y los cambios anteriores del directorio de trabajo volviendo a la versión anterior del repositorio.

Comprobar de nuevo el historial y el estado del repositorio.

```
> git commit -m "Borrado accidental."  
> git status  
> git log  
> git reset --hard HEAD~1  
> git log  
> git status
```

Ramas

Ejercicio 15

Crear una nueva rama curriculum y mostrar las ramas del repositorio.

```
> git branch curriculum  
> git branch -av
```

Ejercicio 16

Crear el fichero ciudades/ *CiudadesPatrimonio.txt* y añadir el texto siguiente

Ciudades Patrimonio de la humanidad visitadas: Cáceres, San Cristóbal de la Laguna, Bérgamo

Añadir los cambios a la zona de intercambio temporal.

Hacer un commit con el mensaje “*Añadido Ciudades Patrimonio de la Humanidad*”

Mostrar la historia del repositorio incluyendo todas las ramas.

Añadir un fichero nuevo ciudades /CiudadesPatrimonio.txt con el texto.

```
> git add .  
> git commit -m " Añadido Ciudades Patrimonio de la Humanidad"  
> git log --graph --all --oneline
```

Ejercicio 17

Cambiar a la rama curriculum y comprueba el cambio.

Crear el fichero curriculum.txt (en la carpeta raíz que contiene sobremini.txt) y añadir la siguiente referencia

2024 / Técnico Superior en Aplicaciones Web / Instituto FOC

Añadir los cambios a la zona de intercambio temporal.

Hacer un commit con el mensaje "Añadida primera referencia a curriculum."

Mostrar la historia del repositorio incluyendo todas las ramas.

```
> git checkout curriculum  
> git branch
```

Añadir el fichero curriculum.txt con el contenido (nunca olvidar darle a guardar)

```
> git add .  
> git commit -m "Añadida primera referencia a curriculum."  
> git log --graph --all --oneline
```

Ejercicio 18

Fusionar la rama curriculum con la rama master.

Mostrar la historia del repositorio incluyendo todas las ramas.

```
> git checkout master
```

```
> git branch  
> git merge curriculum  
> git log --graph --all --oneline
```

Eliminar la rama bibliografia.

Mostrar de nuevo la historia del repositorio incluyendo todas las ramas.

```
> git branch -d curriculum  
> git log --graph --all --oneline
```

Ejercicio 19

Crear la rama curriculum.

Cambiar a la rama curriculum.

Cambiar el fichero *curriculum.txt* para que contenga las siguientes referencias:

2024 / Técnico Superior en Aplicaciones Web / Instituto FOC

2024 / Junior Developer / Microsoft

Añadir los cambios a la zona de intercambio temporal y hacer un commit con el mensaje "Añadida nueva referencia a curriculum."

```
> git branch curriculum  
> git checkout curriculum
```

Añadir el fichero *curriculum.txt* con el contenido (nunca olvidar darle a guardar)

```
> git commit -a -m "Añadida nueva referencia curriculum."
```

Cambiar a la rama master.

Cambiar el fichero *curriculum.txt* para que contenga las siguientes referencias:

2024 / Técnico Superior en DAW / Instituto FOC

2024 / Junior Developer / Google

Añadir los cambios a la zona de intercambio temporal y hacer un commit con el mensaje "Añadida nueva referencia curriculum."

Ver el historial incluyendo las ramas

```
> git checkout master
```

Añadir el fichero curriculum.txt con el contenido.

```
> git commit -a -m "Añadida nueva referencia curriculum."
> git log --graph --all --oneline
```

Fusionar la rama curriculum con la rama master.

Resolver el conflicto dejando el fichero curriculum.txt con las referencias:

2024 / Técnico Superior en Aplicaciones Web / Instituto FOC

2024 / Junior Developer / Microsoft

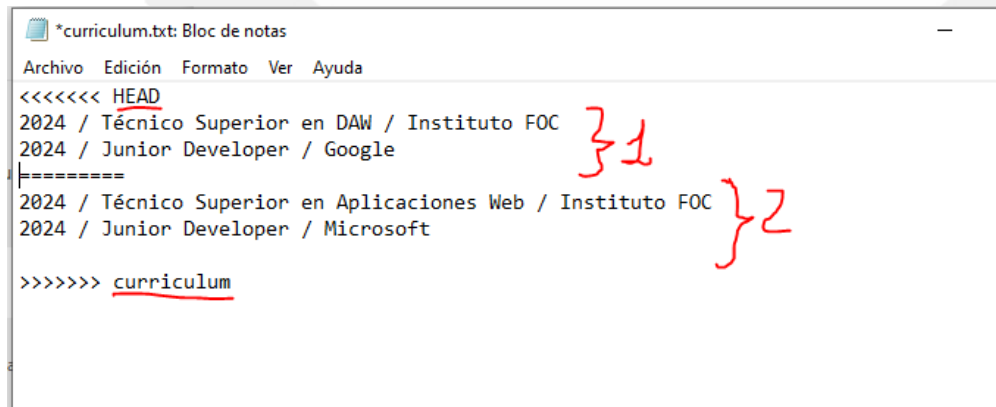
2025 / Senior Developer / Google

Añadir los cambios a la zona de intercambio temporal y hacer un commit con el mensaje "Resuelto conflicto de curriculum."

Mostrar la historia del repositorio incluyendo todas las ramas.

```
> git merge curriculum
```

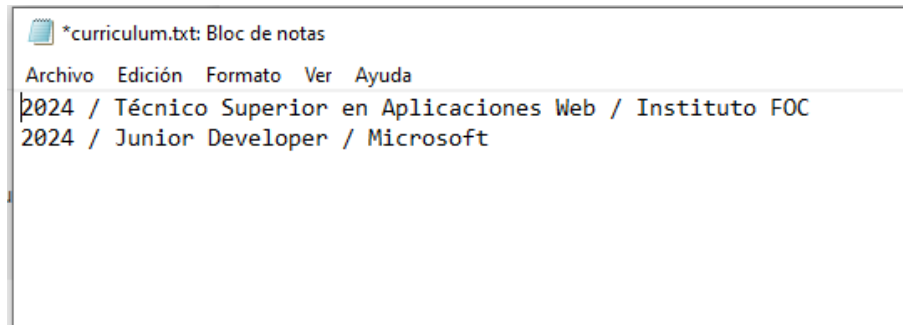
Editar el fichero curriculum.txt para solucionar el conflicto



```
*curriculum.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
<<<<<< HEAD
2024 / Técnico Superior en DAW / Instituto FOC
2024 / Junior Developer / Google
|=====
2024 / Técnico Superior en Aplicaciones Web / Instituto FOC
2024 / Junior Developer / Microsoft
>>>>>> curriculum
```

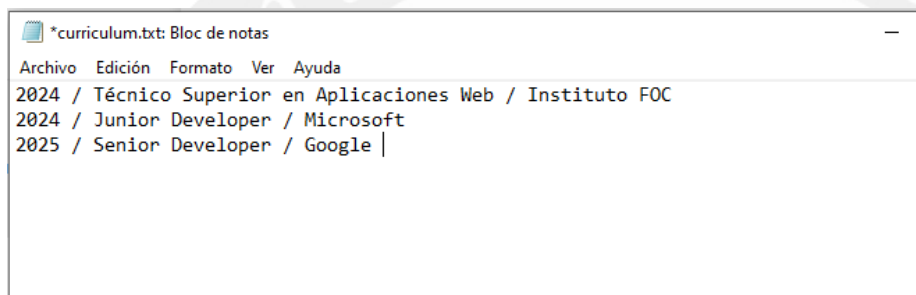
Ahora el fichero indica las líneas de conflicto dentro de <<<<<< ===== >>>>>>, como se puede apreciar muestra dos bloques. El primer Bloque es el código que viene del HEAD, el puntero actual, que si recordamos es el commit que hemos hecho en la rama máster que es donde estamos ahora mismo. El segundo Bloque

pertenece a las líneas de código con conflicto en la rama curriculum. Para solucionar el conflicto elegimos una de las dos opciones, por ejemplo:



```
*curriculum.txt: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
2024 / Técnico Superior en Aplicaciones Web / Instituto FOC
2024 / Junior Developer / Microsoft
```

Como el enunciado nos dice que además añadamos una línea tendríamos:



```
*curriculum.txt: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
2024 / Técnico Superior en Aplicaciones Web / Instituto FOC
2024 / Junior Developer / Microsoft
2025 / Senior Developer / Google |
```

```
> git commit -a -m "Solucionado conflicto curriculum."
> git log --graph --all --oneline
```

Repositorios Remotos

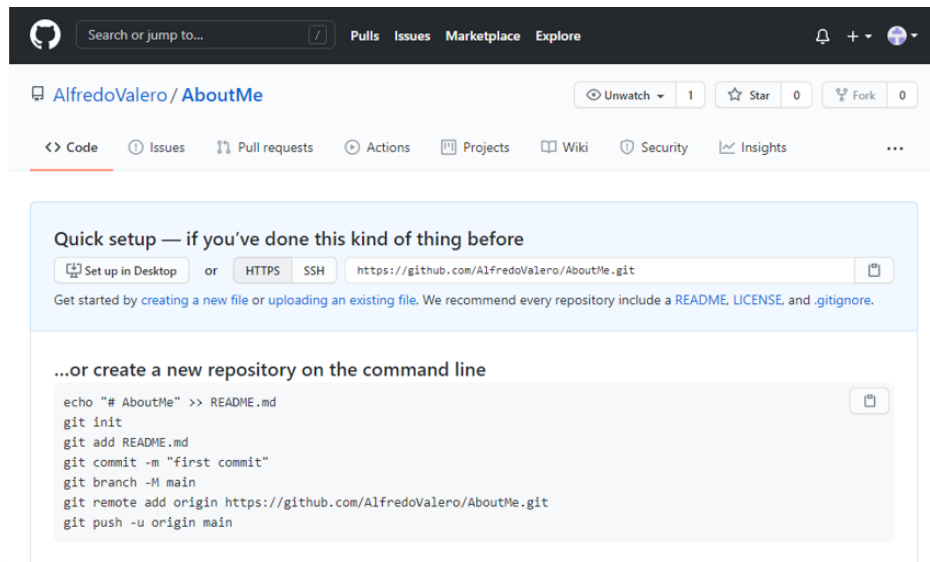
Ejercicio 20

Crear un nuevo repositorio público en GitHub con el nombre AboutMe.

Añadirlo al repositorio local.

Mostrar todos los repositorios remotos configurados.

Crear el repositorio en GitHub y copiar su url con protocolo https.



```
> git remote add origin https://github.com/AlfredoValero/AboutMe.git
```

```
> git remote -v
```

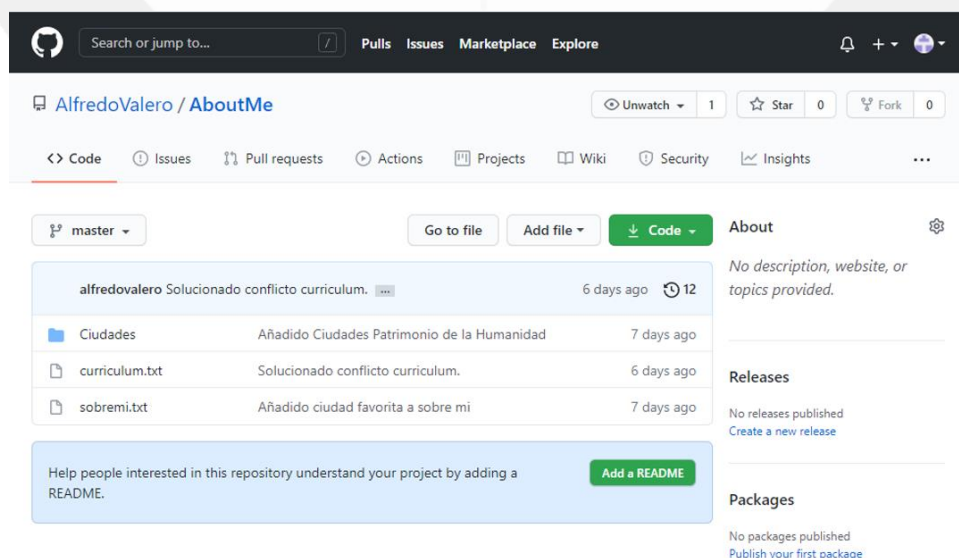
Nota: 'origin' es un Alias que se le asigna a la url del repositorio introducido, de tal forma que a partir de ese momento se puede utilizar el alias en vez de la dirección. Podemos poner el alias que queramos, pero normalmente, por buenas prácticas, se nombra al repositorio principal que utilicemos como origin.

Ejercicio 21

Añadir los cambios del repositorio local al repositorio remoto de GitHub.

Acceder a GitHub y comprobar que se han subido los cambios mostrando el historial de versiones.

```
>git push origin master
```



Nota: Al ser la primera vez que utilizéis git con vuestro repositorio github, es posible que tengáis que dar autorización a git para poder utilizar vuestra cuenta. Para ellos, debéis introducir el comando:

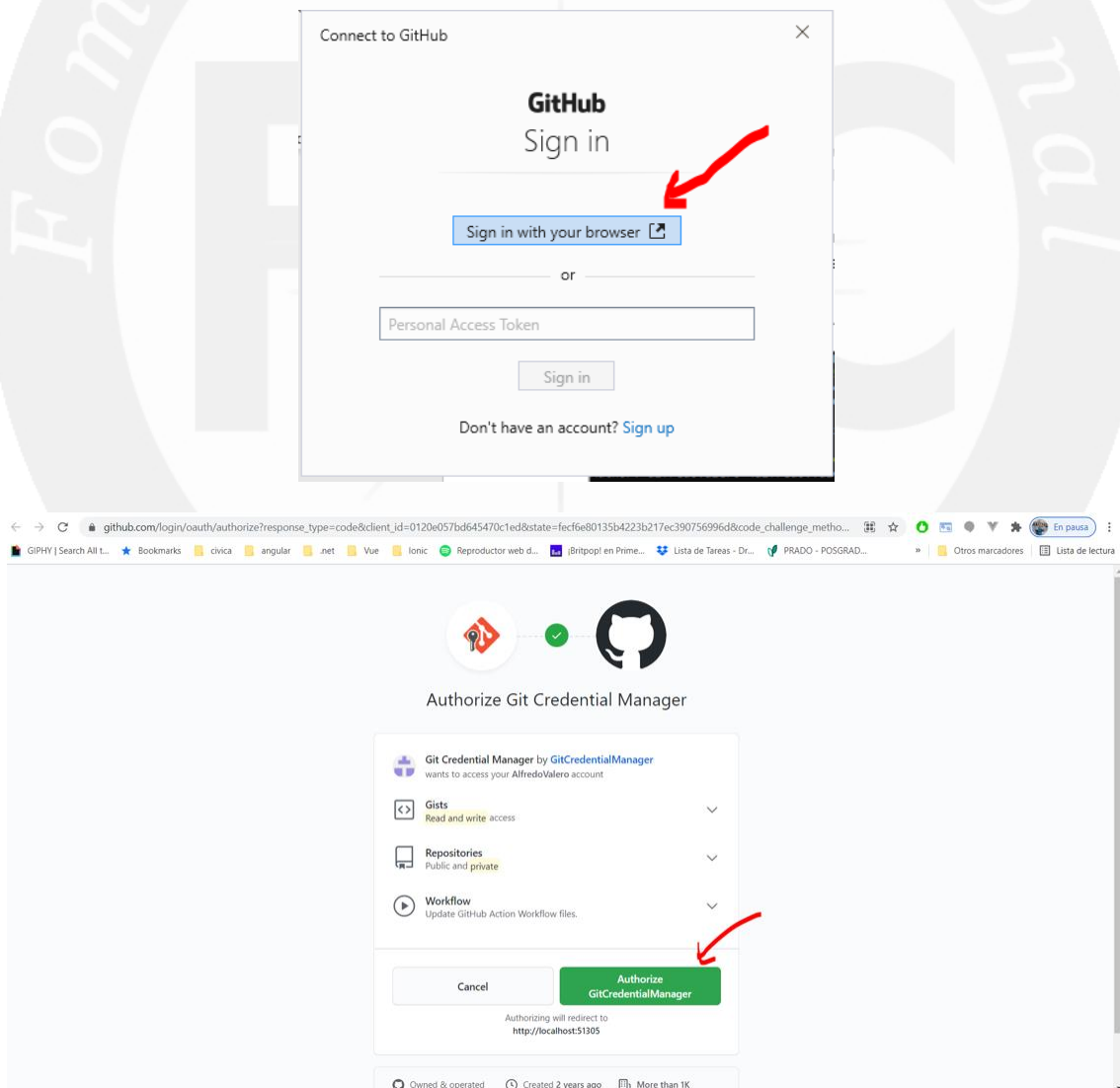
git push --set-upstream origin master

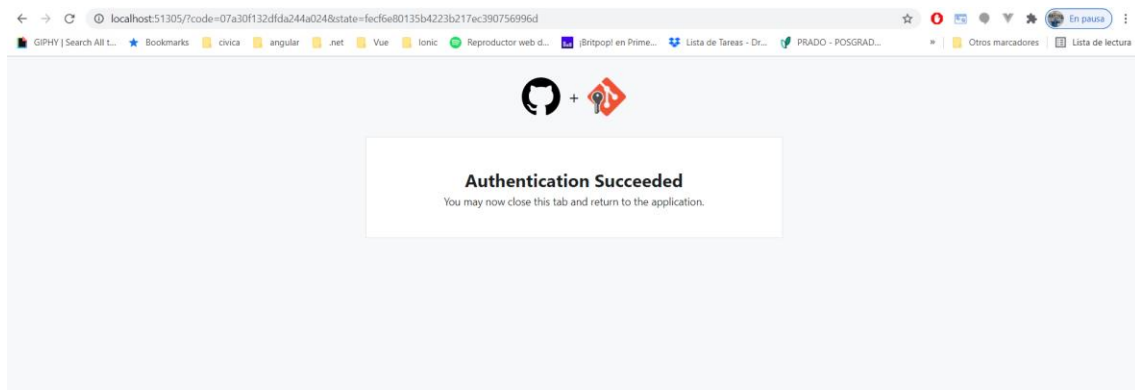
```
C:\Users\alfredo.valero\Documents\alfredoalero>git push
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

  git push --set-upstream origin master

C:\Users\alfredo.valero\Documents\alfredoalero>git push --set-upstream origin master
info: please complete authentication in your browser...
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 627 bytes | 627.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/AlfredoValero/InformacionPersonal
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Esperar que aparezca el cuadro de navegador





Ejercicio 22

Colaborar en el repositorio remoto de otro usuario.

Clonar su repositorio AboutMe.

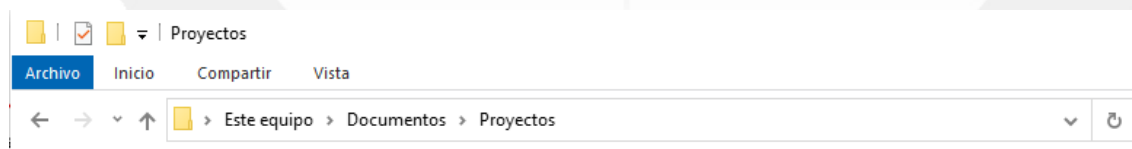
Añadir el fichero colaborador **MiNombre.txt** que contenga mi nombre, mi nombre de usuario github y mi correo electrónico.

Añadir los cambios a la zona de intercambio temporal.

Hacer un commit con el mensaje “Añadido autor **MiNombre.**”

Subir los cambios al repositorio remoto.

Para realizar todo este proceso en primer lugar vamos a nuestra carpeta de proyectos y ejecutamos el terminal, en caso de no tener una carpeta de proyectos la creamos.



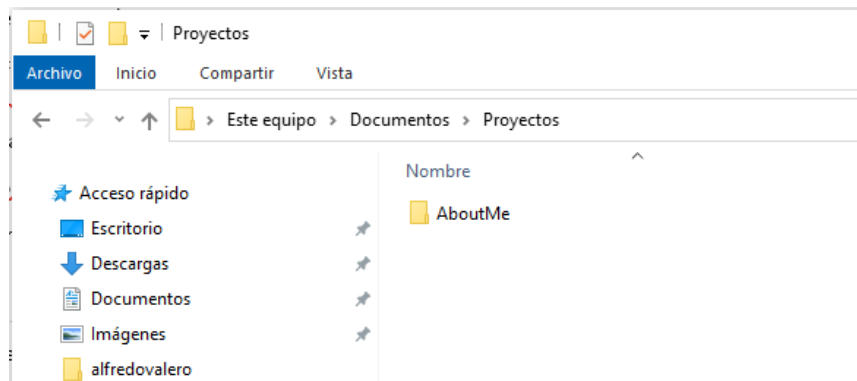
Crea la carpeta, en la terminal escribe:

```
> cd C:\Users\alfredo.valero\Documents\Proyectos
```

Entrar en GitHub, en el proyecto AboutMe del profesor y copiar la url. En nuestro caso.

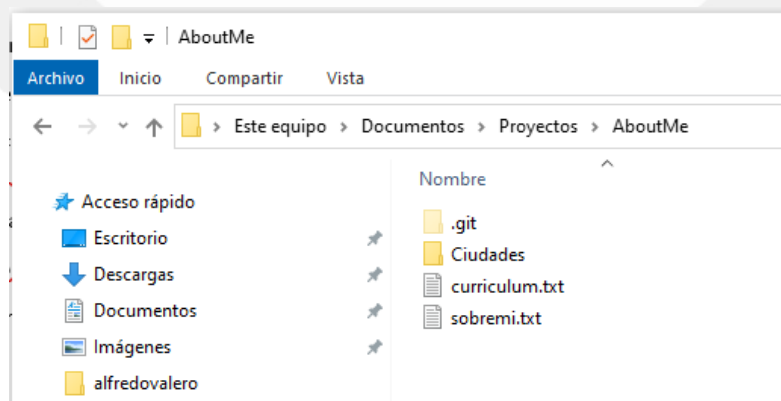
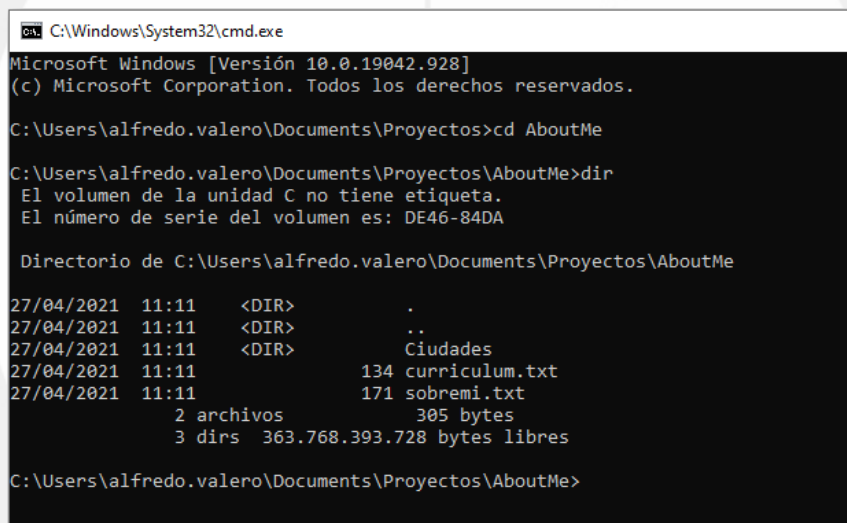
<https://github.com/AlfredoValero/AboutMe>

```
> git clone https://github.com/AlfredoValero/AboutMe
```



Ahora deberemos entrar en la carpeta del proyecto para trabajar con él.

```
> cd C:\Users\alfredo.valero\Documents\Proyectos\AboutMe
```



Crear el fichero colaborador **MiNombre**.txt (Ejemplo: colaboradorAlfredoValero.txt) y añadir el texto siguiente

Nombre: Alfredo Valero

Usuario Github: AlfredoValero

Email: alfredo.valero@foc.es

```
> git add .  
> git commit -m "Añadido colaborador."  
> git push origin master.
```

Podéis comprobarlo entrando en la dirección del repositorio y viendo que vuestro archivo esta <https://github.com/AlfredoValero/AboutMe>

Ejercicio 23

Crear una nueva rama dev**MiNombre** y activarla.

Añadir la nota que merecemos en esta práctica al fichero colaborador **MiNombre**.txt.

Añadir los cambios a la zona de intercambio temporal.

Hacer un commit con el mensaje "Añadido nota merecida"

Subir los cambios de la rama dev**MiNombre** al repositorio remoto en GitHub.

Hacer un "Pull Request" de los cambios en la rama dev**MiNombre**.

```
> git checkout -b devAlfredoValero
```

Editar el fichero colaborador **MiNombre**.txt (Ejemplo: colaboradorAlfredoValero.txt) y añadir la nota que nos merecemos.

Nombre: Alfredo Valero

Usuario Github: AlfredoValero

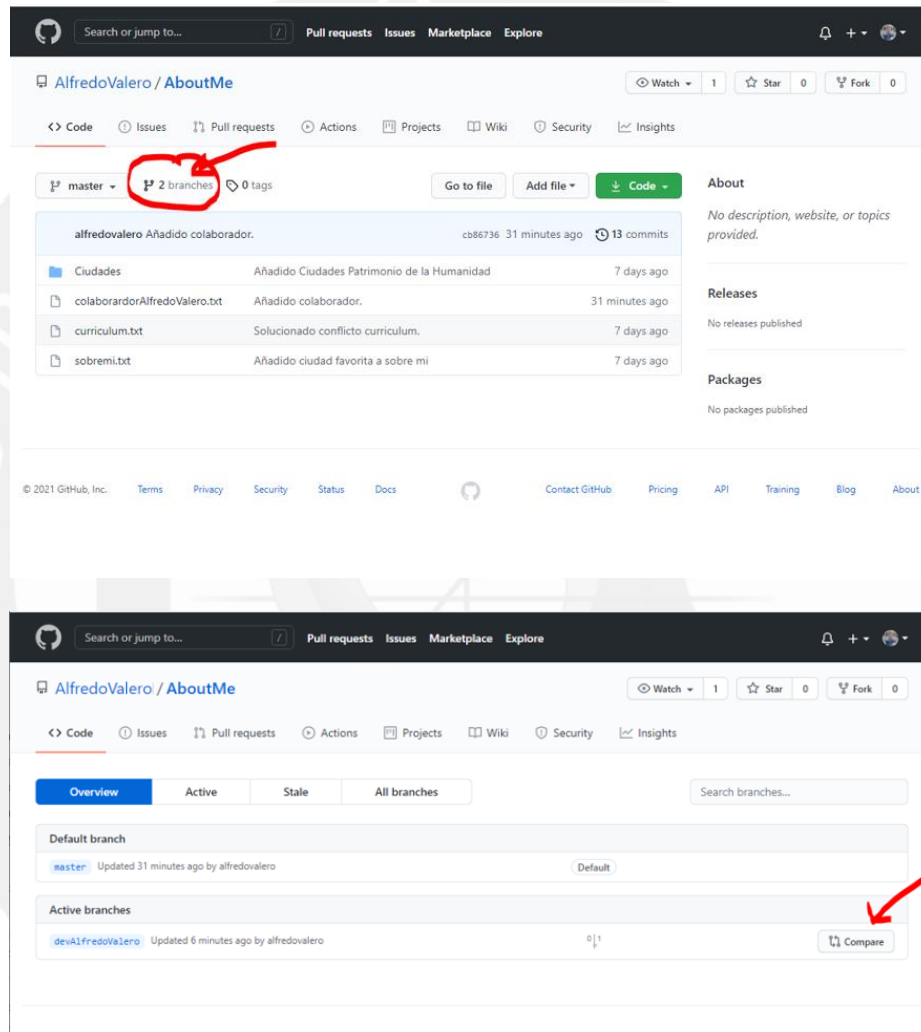
Email: alfredo.valero@foc.es

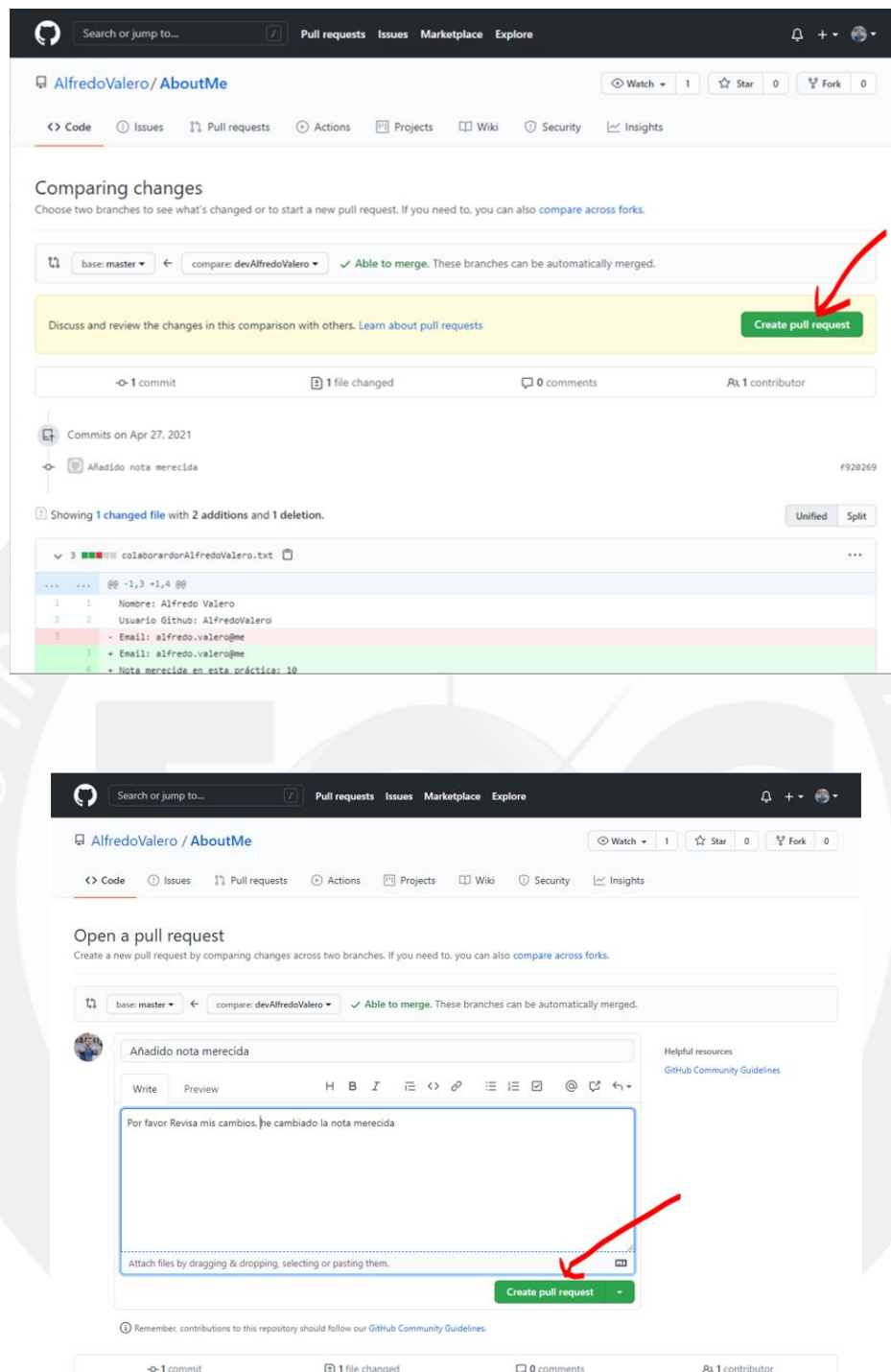
Nota merecida en esta práctica: 10

```
> git commit -am "Añadido nota merecida"
```

```
> git push origin devAlfredoValero
```

Ir al repositorio remoto en GitHub y hacer clic en la rama, en el botón Compare & Pull Request y después completar la solicitud haciendo clic en el botón Create Pull Request.





Te preguntará ¿Qué acabo de realizar? ¿Qué es un Pull Request?, pues bien, un Pull Request es una petición de validación que se hace al revisor del repositorio, por ejemplo, el jefe del proyecto (En este caso, al profesor), el cual inspeccionará los cambios realizados y que dará su aprobación o rechazará la fusión de la rama.

Este proceso es muy común en las empresas y se considera una buena práctica en cuestiones de calidad del código, ya que hay alguien que revisa algunos aspectos del código que se acaba

de realizar y da la aprobación para fusionar ese código con la rama correspondiente, dev, master, ...

