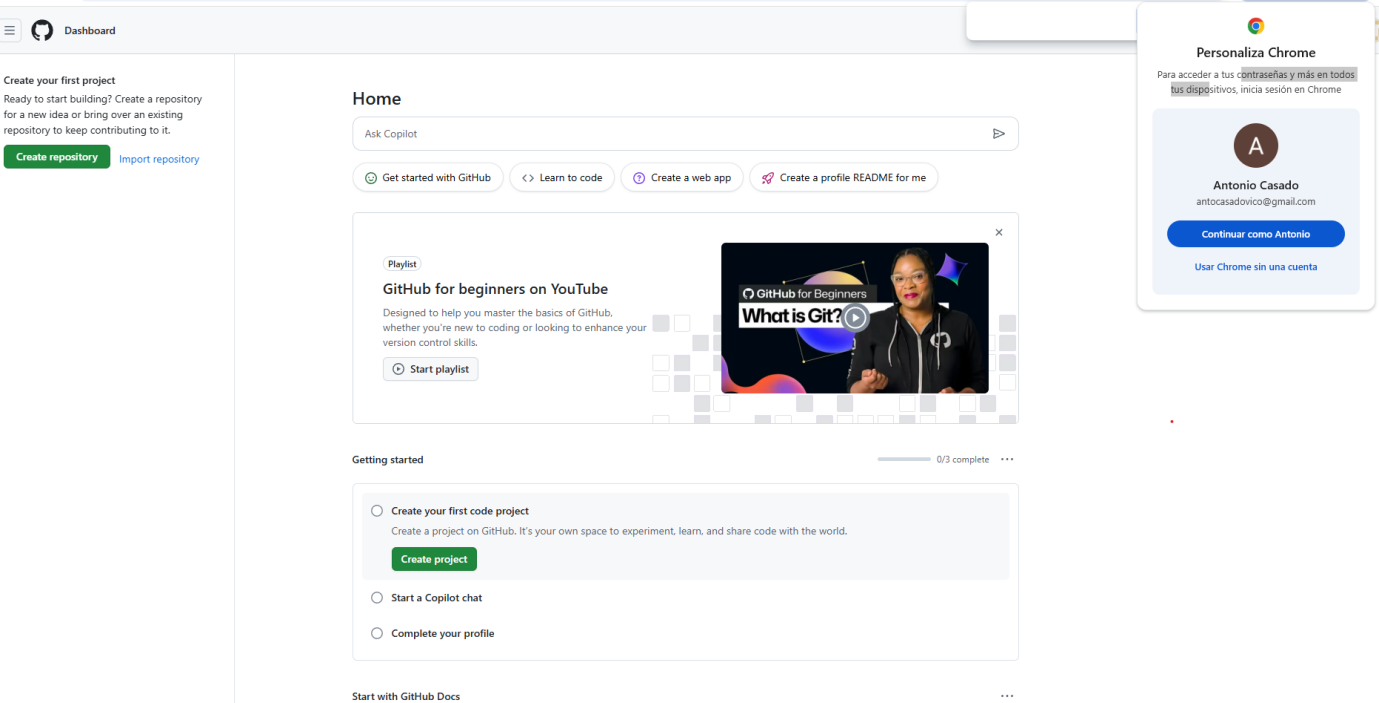


**Ejercicio 1.-**



**Ejercicio 2.-**

CVS es un sistema de control de versiones para gestionar y editar los cambios de un proyecto y que permite a varios usuarios acceder a él y editarlo.

Las diferencias con Git es que CVS es un sistema centralizado que requiere conexión a internet y Git funciona de manera local.

**Ejercicio 3.-**

-SVN, es un sistema de control de versiones de open office de código abierto, capaz de manejar archivos binarios.

-DARCS, es una herramienta como las anteriores aunque con la habilidad de escoger qué cambios aceptar hechos en otros dispositivos.

**Ejercicio 4.-**

GitLab es una plataforma de código abierto basada en Git, que sirve como repositorio online para almacenar proyectos.

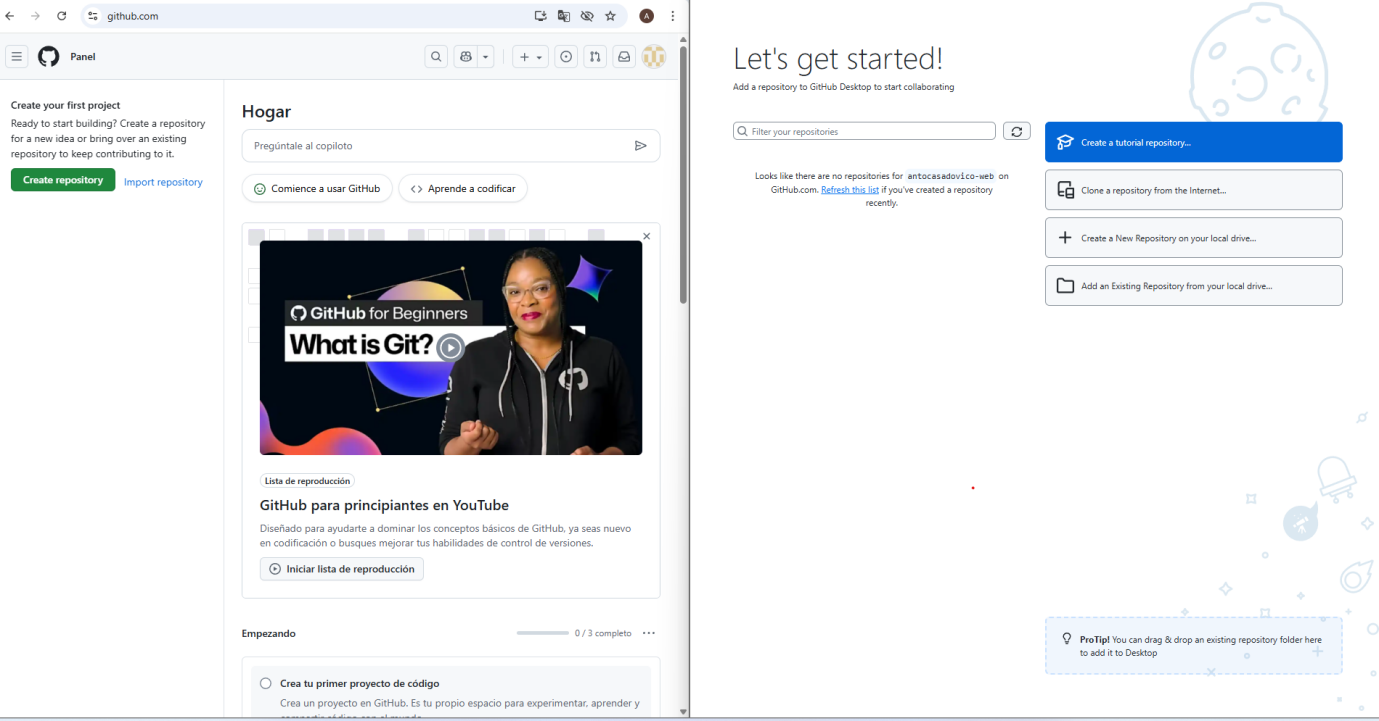
La principal diferencia entre GitLab y Git es que Git está enfocada en el control de versiones local, mientras que GitLab está basada en Git pero para el control en la nube.

**Ejercicio 5.-**

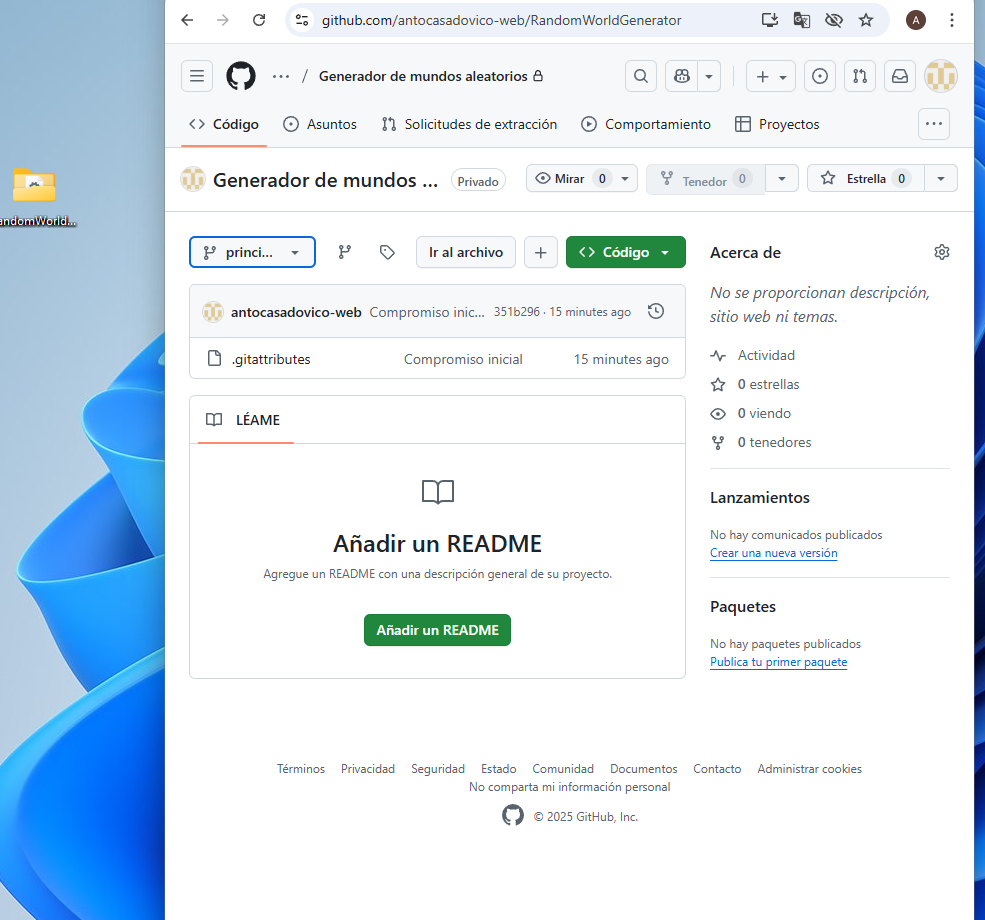
DevOps es un modo de trabajo en el que los desarrolladores y los operadores trabajan juntos para realizar los proyectos de forma más rápida y con mayor calidad.

Pues CVS al ser un sistema de control de versiones que permite modificar archivos de manera concurrente, si un grupo de DevOps trabaja de manera concurrente con los mismos archivos permite hacerlo.

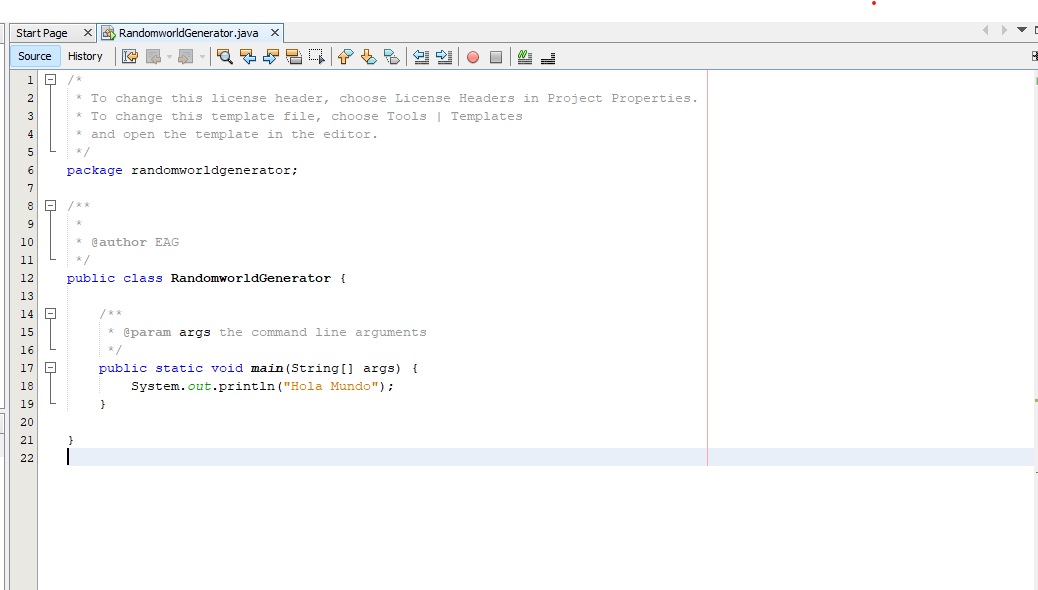
**Ejercicio 6.-**



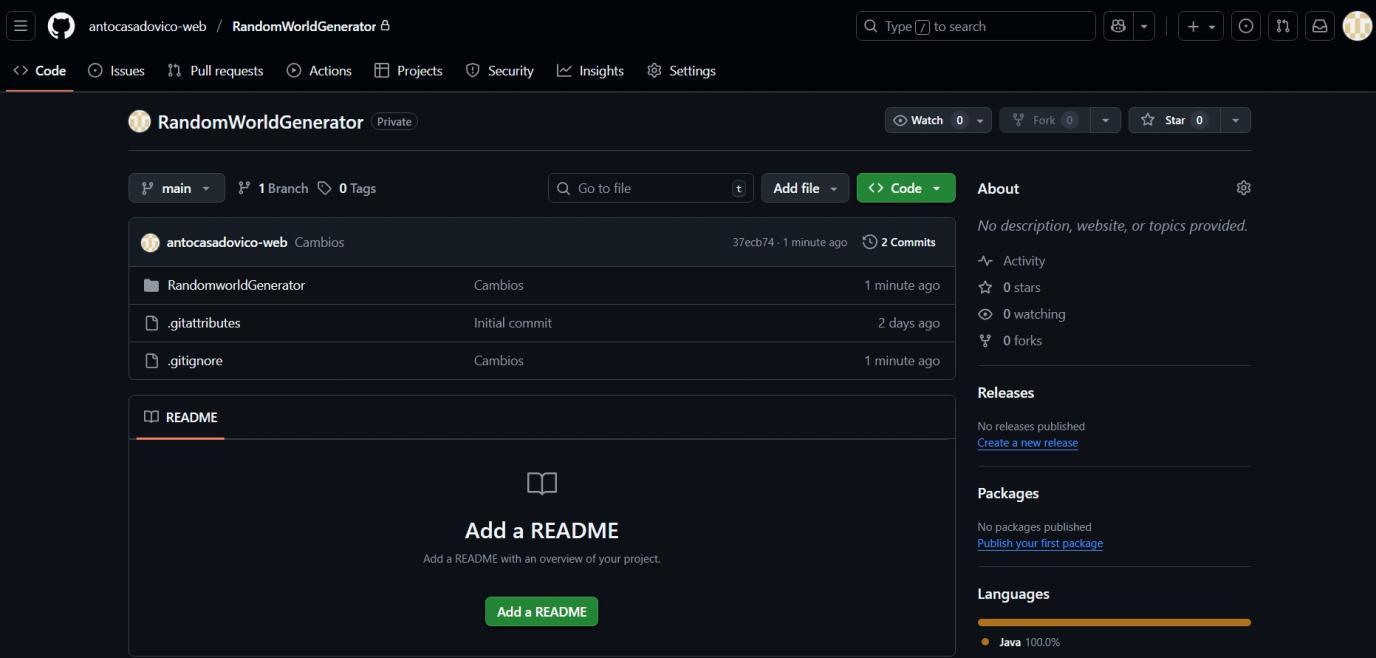
**Ejercicio 7.**

**-**

**Ejercicio 8.**



**Ejercicio 9.-**

****

Al principio no se subieron solamente al añadir los archivos a la carpeta, debido a que no detectaba los cambios hasta que les cambie el nombre, pero una vez cambiado el nombre ya se puede hacer hacer commit y push y ya si aparecen los cambios.

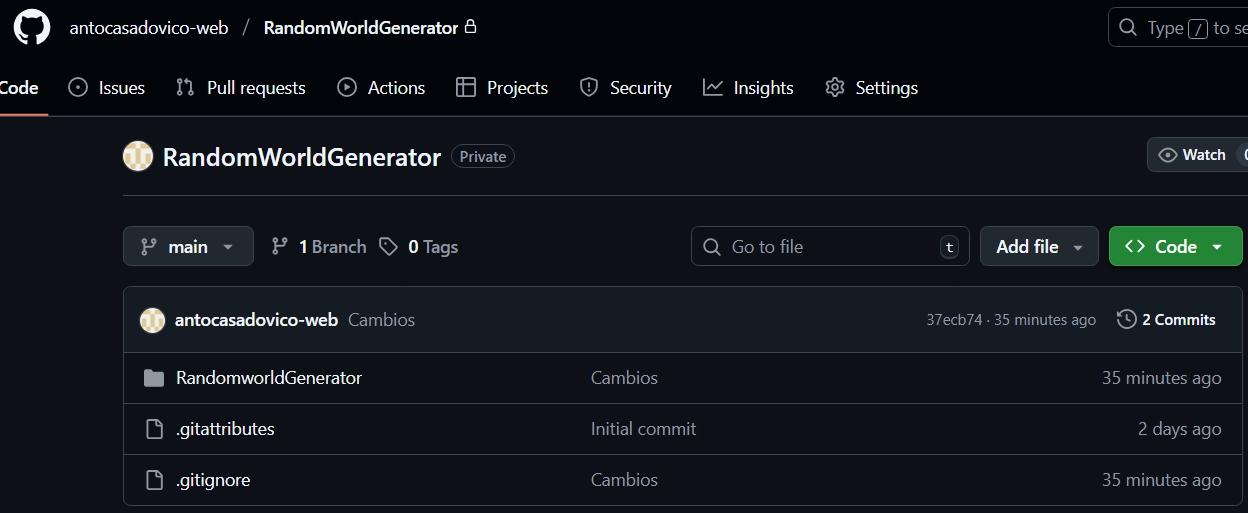
**Ejercicio 10**.-

En Github desktop nada más abrir aparece la lista de repositorios y si se han realizado cambios o no localmente.

**Ejercicio 11**.-

Solamente al hacer commit el archivo no se sube a la web, se debe hacer también push.

**Ejercicio 12**.-



**Ejercicio 13**.-

Al principio se debe usar clone, por que no tengo el archivo, primero debo descargarlo usando clone, pull se usa cuando ya tengo los archivos en local, para actualizar los últimos cambios.

**Ejercicio 14**.-

Una vez terminados los cambios se debe usar commit para confirmar los cambios, también es recomendable utilizar pull para actualizar por si alguien ha hecho cambios antes, y después al finalizar ya se utiliza push para subir los archivos a la nube.

**Ejercicio 15. Define con tus propias palabras los términos: Git, GitHub, repositorio, ‘clone’, ‘commit’, ‘push’ y ‘pull’.**

Git: es un sistema de control de versiones en el que se usan comandos para ir almacenando los archivos y que permite realizar un seguimiento de los cambios para poder volver atrás en cualquier momento.

Github: Es una plataforma web basada en git y permite guardar los repositorios en la nube.

Repositorio: es el lugar donde se almacenan, organizan y administran los archivos y datos de un proyecto.

Clone: es un comando de git y de github que se usa para clonar los archivos que están en la nube y poder descargarlos en local.

Commit: es otro comando de git que se usa para guardar los cambios realizados en un proyecto.

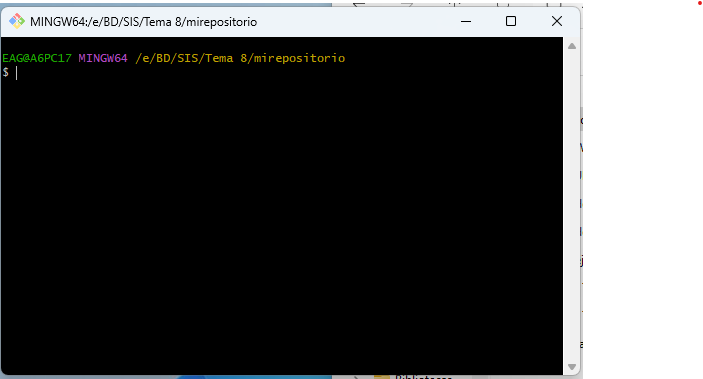
Push: un comando que se usa para subir los cambios realizados a la nube.

Pull: otro comando también de git que se encarga de descargar los archivos actualizados de la nube.

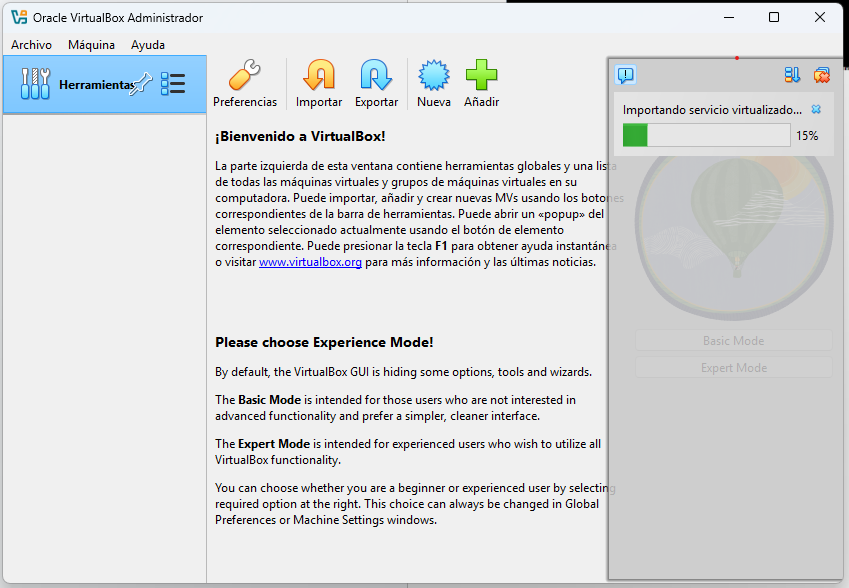
**Ejercicio 16. A partir de lo visto en tu clase, escribe un breve texto en el que intentes convencer a un amigo que está empezando a programar para que use Git. Explícale cuáles son las ventajas de usar Git.**

Pues git permite un historial de cambios completo y colaborar de forma eficiente en distintos equipos, trabajar sin conexión y volver a versiones anteriores.

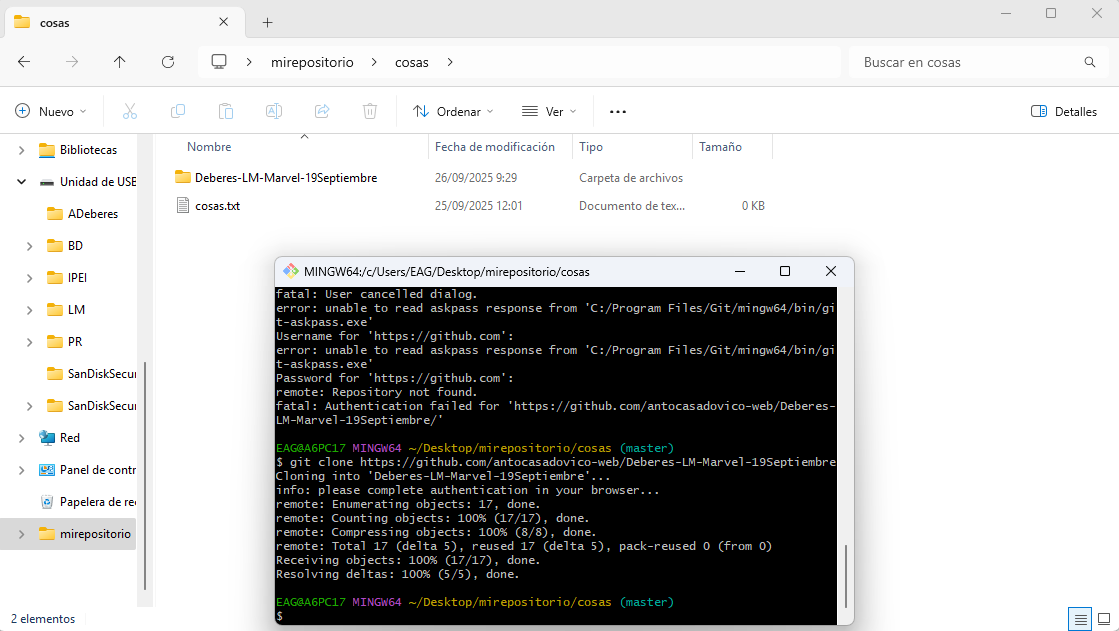
**Ejercicio 17. Comprueba que Git Bash está instalada en el PC de clase**



**Ejercicio 18. Instala Git en la MV del tema anterior.**



**Ejercicio 19. Comprueba que el código del repositorio se ha descargado en tu PC tras hacer el comando clone.**

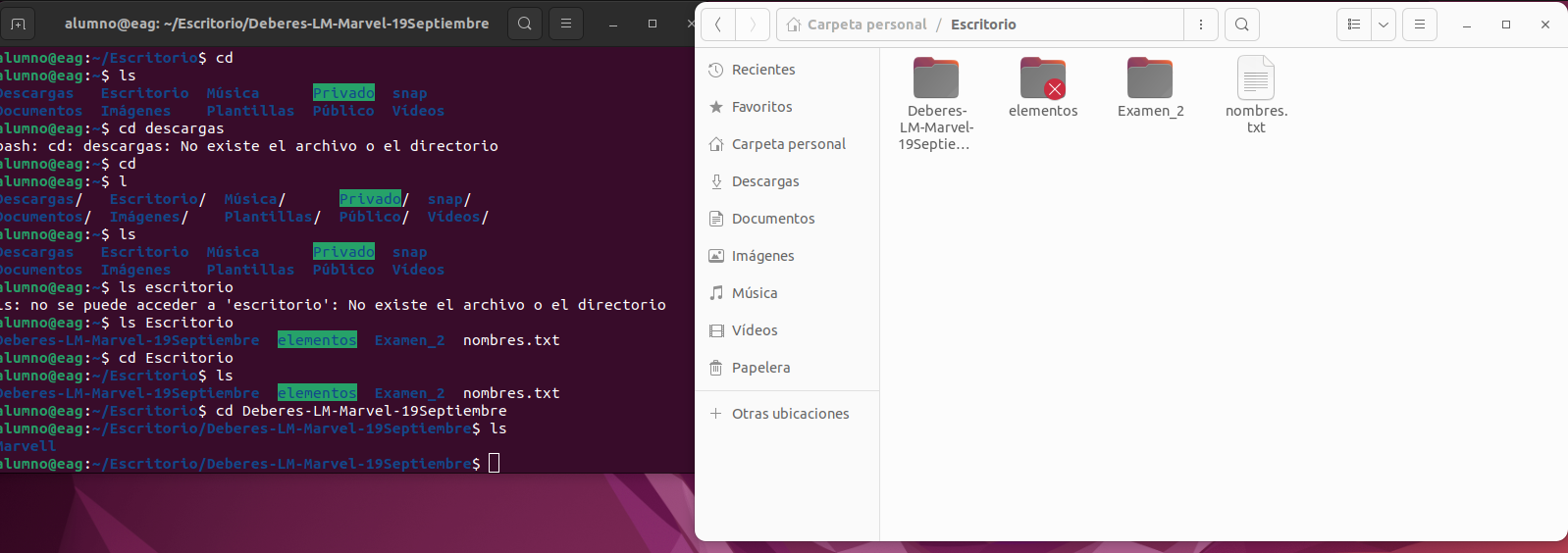


**Ejercicio 20. Prueba algunos de los comandos vistos en el tema anterior en la terminal. ¿Funcionan? ¿por qué si estamos en Windows?**

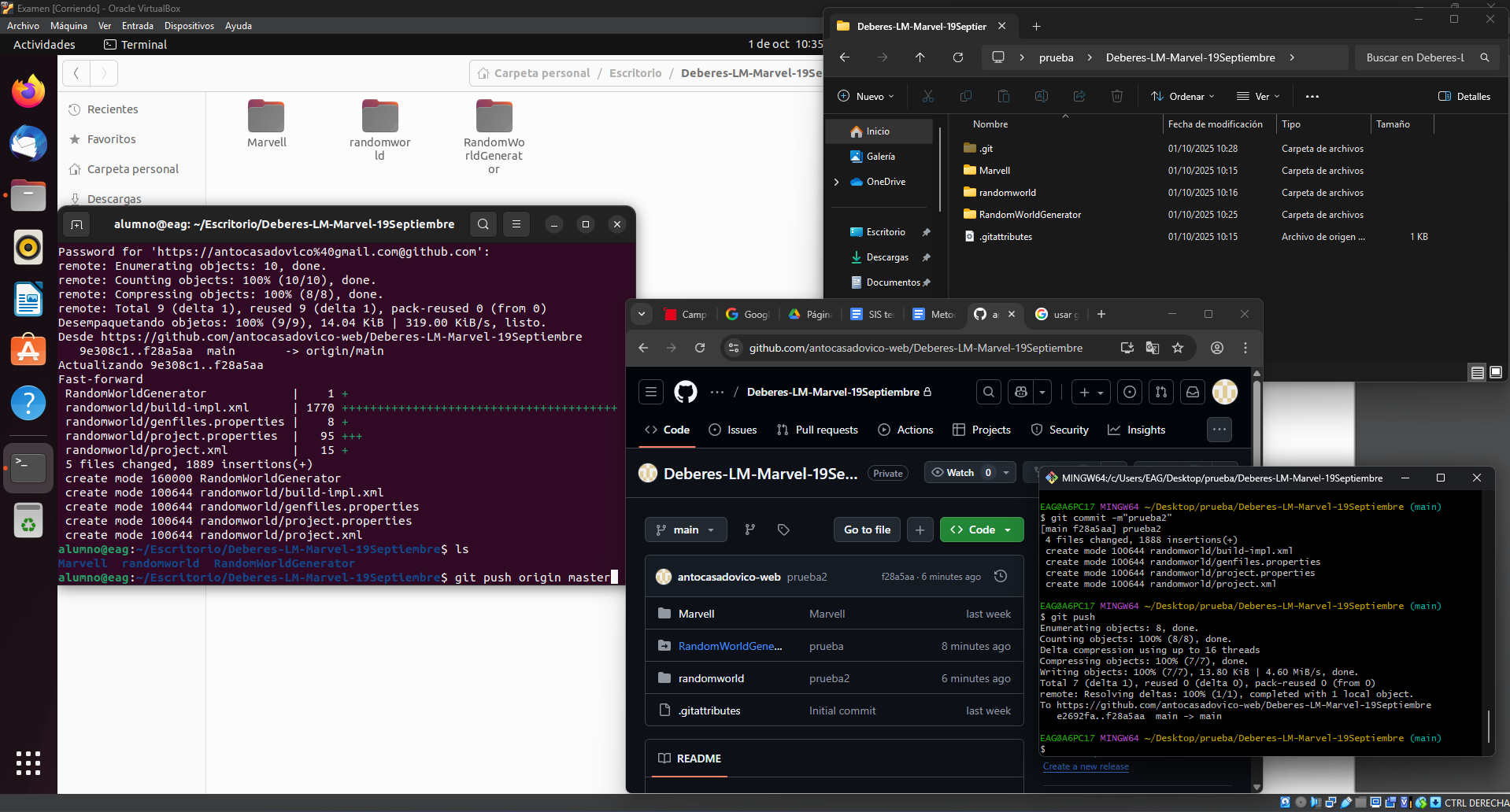
Si funcionan en la terminal de windows con comandos de linux, gracias a que git instala las herramientas que necesita para poder comunicarse con windows con comandos que de otra forma no comprendería.

**Ejercicio 21. Repite el ejercicio 19, ahora en la MV con Linux ¿cambia la forma de realizar el clone por estar en Linux?**

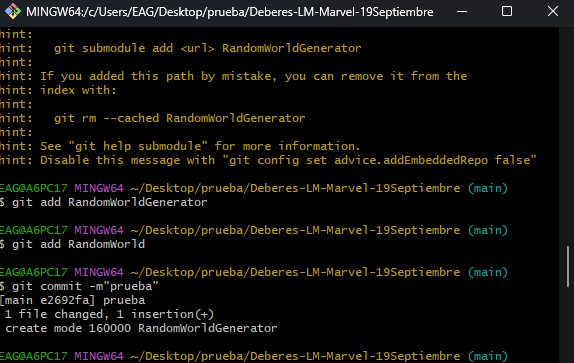
La forma es la misma, aunque cambia que en la máquina virtual debes usar token para poder usarlo, y en windows te redirige a una ventana externa para autorizar el acceso.



**Ejercicio 22. Una vez tengas el repositorios cargando tanto en Windows como en Linux ábrelo con Netbeans. Modifica el código en Windows, por ejemplo añade un nuevo ‘sout’**

****

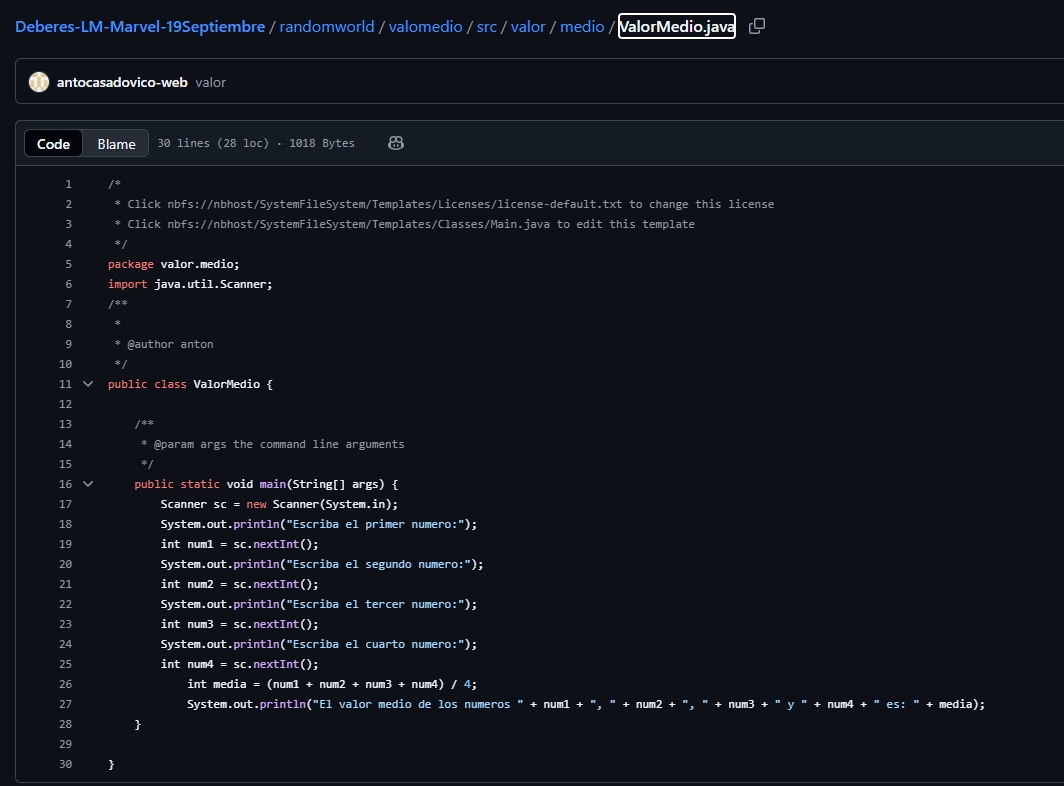
**Ejercicio 23. En Windows, realiza el commit del cambio realizado.**



**Ejercicio 24. Visita la web de tu repositorio, navega hasta el fichero .java en el que hayas realizado el cambio ¿se ve reflejado? ¿por qué?**

No se ve por que solo con commit no se ve reflejado hasta que haces además push.

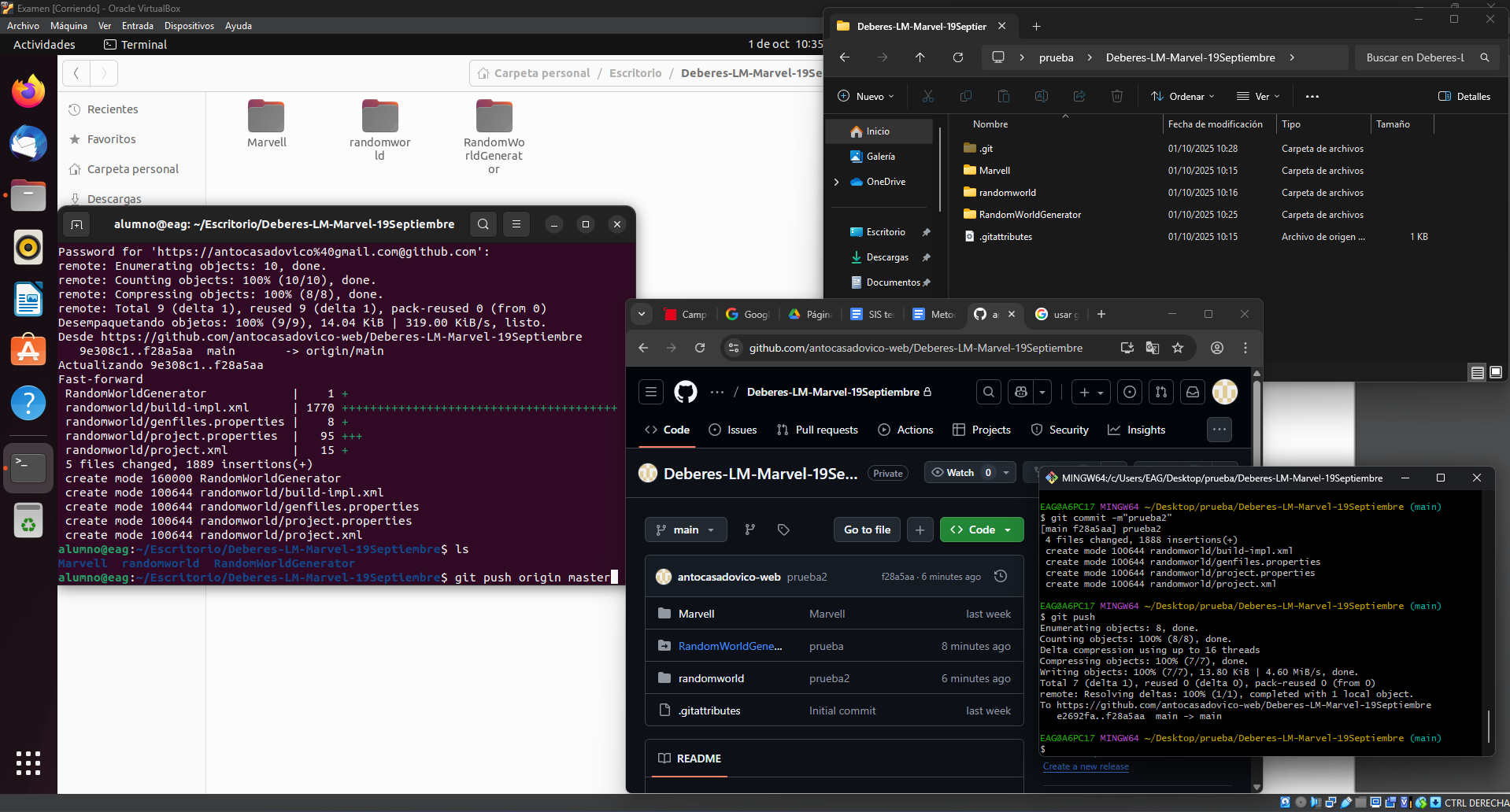
**Ejercicio 25. Haz un push al servidor y comprueba desde la web que los cambios se han realizado.**

****

**Ejercicio 26. ¿El código del proyecto en Linux está actualizado? ¿por qué? ¿Qué comando debemos realizar?**

En principio no está actualizado por que los cambios se han hecho solamente en local y se han subido a github, para que aparezca en el repositorio local de linux se debe usar el comando pull en linux.

**Ejercicio 27. Realiza un pull de proyecto desde Linux. Comprueba que el código pasa a estar actualizado.**

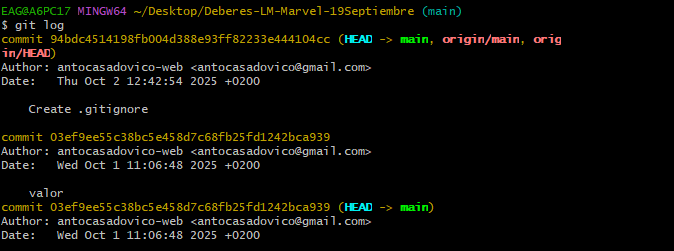
****

**Ejercicio 28. Mira el video https://www.youtube.com/watch?v=PT4otSBaieM y repite todo lo que va haciendo el profe. Comprueba que no surgen errores. Si tienes errores comunícaselo al profesor.**

Al principio me surgieron errores al intentar sincronizar con el repositorio online, pero resultó ser por que el repositorio local lo había almacenado en un pendrive, al moverlo al escritorio todo funciono bien.

**Ejercicio 29. ¿Con qué comando podemos volver a versiones anteriores de Git? ¿Cuál es su sintaxis? Intenta volver a la primera versión de tu proyecto**

Para restaurar una versión anterior se debe usar el comando git log para ver el registro de modificaciones y su código, después se debe usar el comando git - -hard y el código del commit que aparecía en el git log



**Ejercicio 30. En Git, ¿qué es un rama?, ¿cómo crees que pueden beneficiarte a la hora de organizar el código de tu proyecto?**

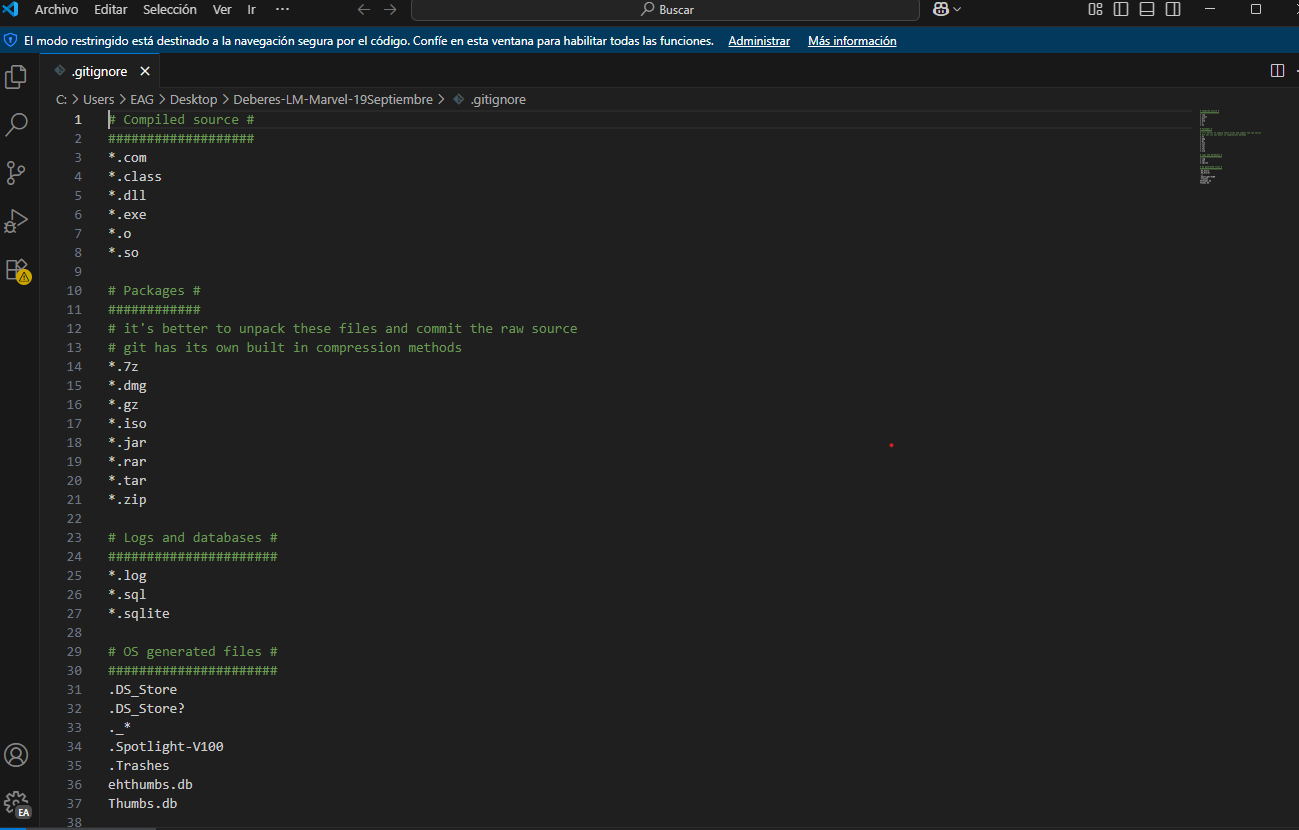
Una rama sirve para que en un proyecto se puedan seguir realizando cambios sin afectar al proyecto principal e ir añadiendo al principal solo las partes que interesen añadir.

**Ejercicio 31. Explica con tus propias palabras qué tipos de ficheros ignora el fichero https://gist.github.com/octocat/9257657 ¿Por qué crees que se ignoran dichos ficheros?**

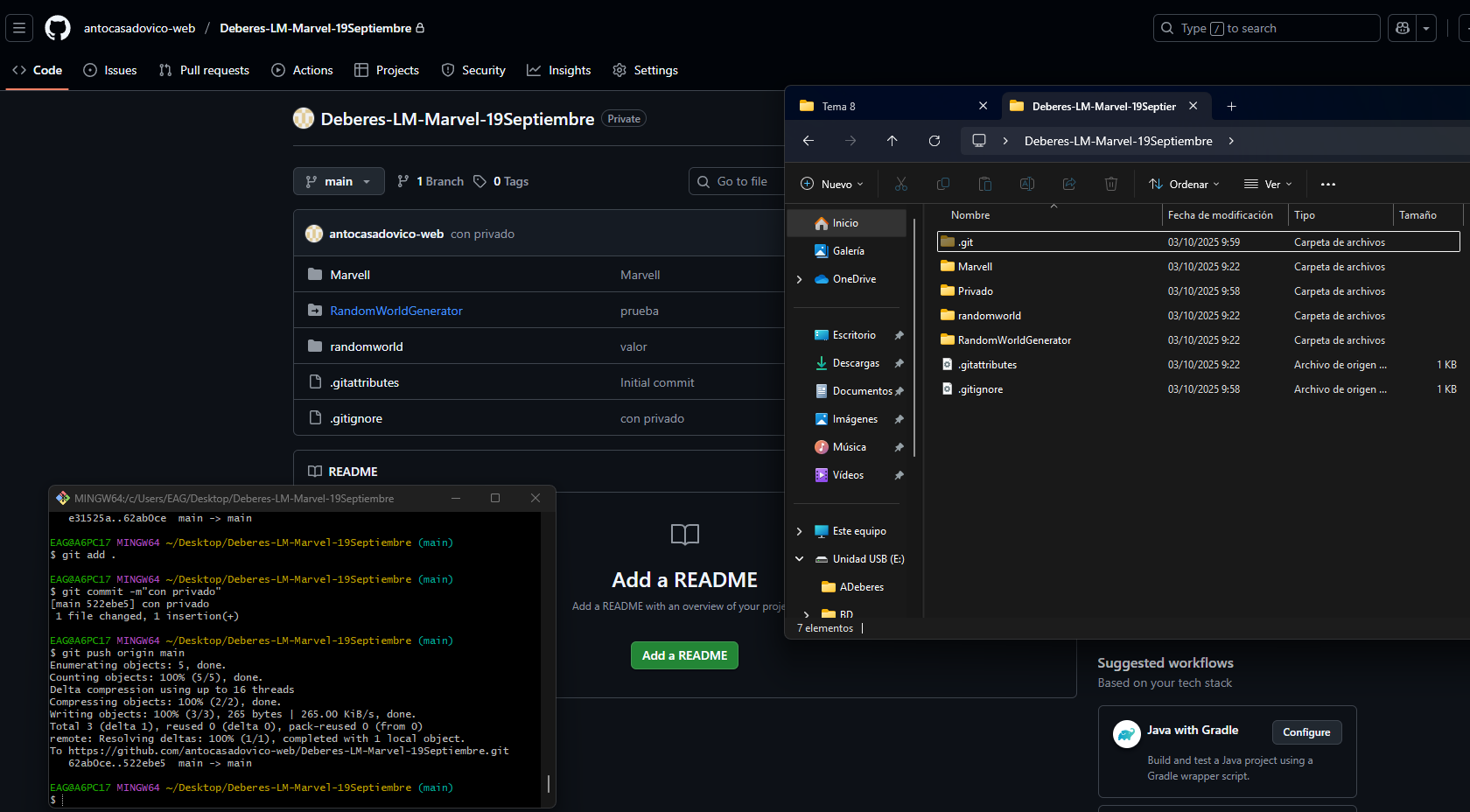
El fichero ignora los archivos con la extensiones .com, .class, .dll, .exe, .o, .so, las extensiones .7z, .dmg, .gz, .iso, .jar, .rar, .tar, .zip, tambien .log, .sql, .sqlite y la carpeta ·\_, a excepción de los archivos .Spotlight-V-100, .Trashes, ethumbs.db, Thumbs.db, que si son visibles.

Se ignoran debido a que se ha usado \* dentro de la carpeta .gitignore, que git detecta que solo debe dejar guardados los cambios en local sin subirlos al repositorio en la nube.

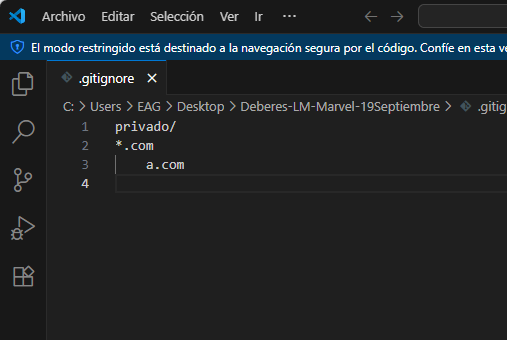
**Ejercicio 32. Crea un fichero .gitignore, o si ya está creado modifícalo, con el contenido de https://gist.github.com/octocat/9257657 dentro de tu proyecto**

****

**Ejercicio 33. Añade una carpeta con el nombre ‘privado’ al proyecto y dentro un fichero con el nombre ‘log.txt’. Haz que todo lo que hay dentro de dicha carpeta no se sincronice con el servidor.**

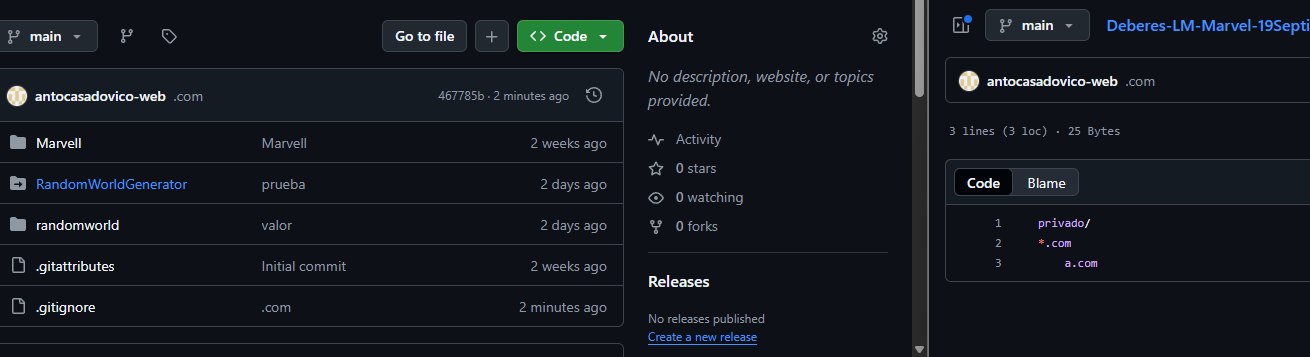
****

**Ejercicio 34. Ignora todos los ficheros .com excepto** [**a.com**](http://a.com)

****

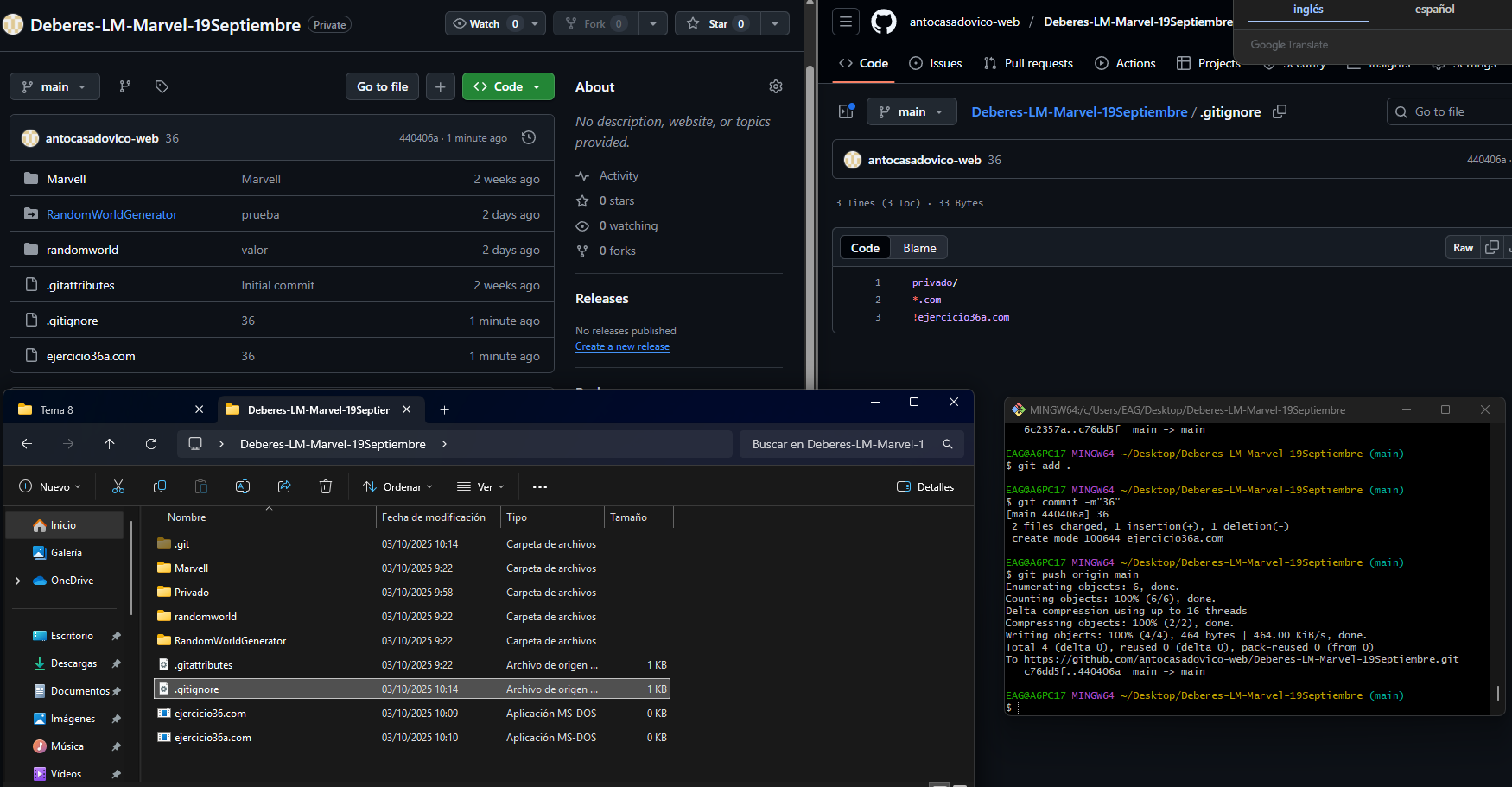
**Ejercicios 35. Haz que los cambios realizados en el fichero .gitignore tengan efecto sobre el repositorio. ¿Se ha subido la carpeta ‘private’? ¿Por qué?**

La carpeta private no se ha subido al estar dentro del .gitignore, pero el archivo .gitignore si se ha actualizado con los datos que debe ignorar.



**Ejercicio 36. Añade los ficheros a.com y b.com en local. Sincroniza la BD remota ¿se han subido los dos ficheros? ¿Por qué?**

Se ha subido solo el fichero con !a.com, que es el que está puesto como excepción, y el archivo .com no se ha subido por que esta como \*.com para que esos archivos no se suban.



**Ejercicio 37. Modifica un fichero del proyecto y guarda los cambios. Cierra el fichero y haz un restore del repositorio. ¿Qué ha pasado?**

Pues depende de si se hace git reset solamente o si se hace git reset –hard, con git reset solamente se vuelve al commit que elijas, pero si haces git reset –hard, se vuelve al commit que elijas y además se borran los commit posteriores.

**Ejercicio 38. Hemos visto cómo volver a versiones concretas en local pero... ¿cómo podríamos volver a dichas versiones en remoto?**

En este caso deberíamos hacer git pull para sincronizar con el servidor los últimos cambios, y hacer después el comando git push para subir a remoto la vuelta a la versión elegida.