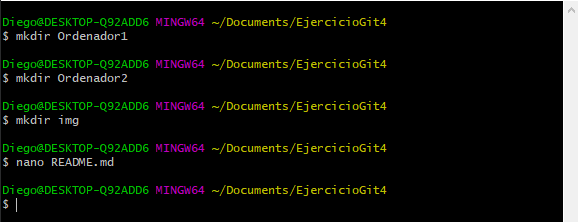
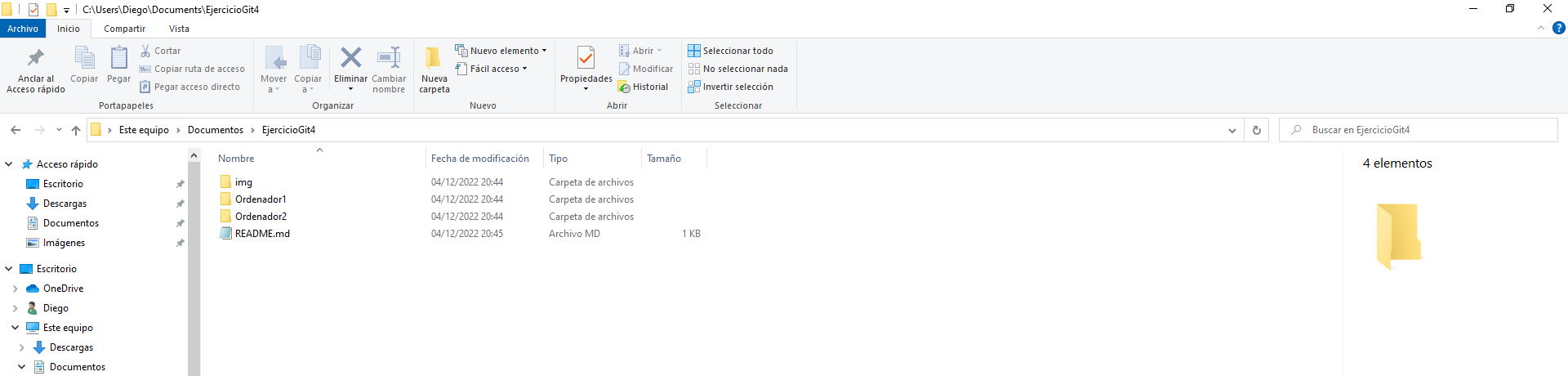
**1- Creamos una carpeta de trabajo con:**

1.1 → Dos carpetas (Ordenador1 y Ordenador2), que simularán dos equipos remotos.

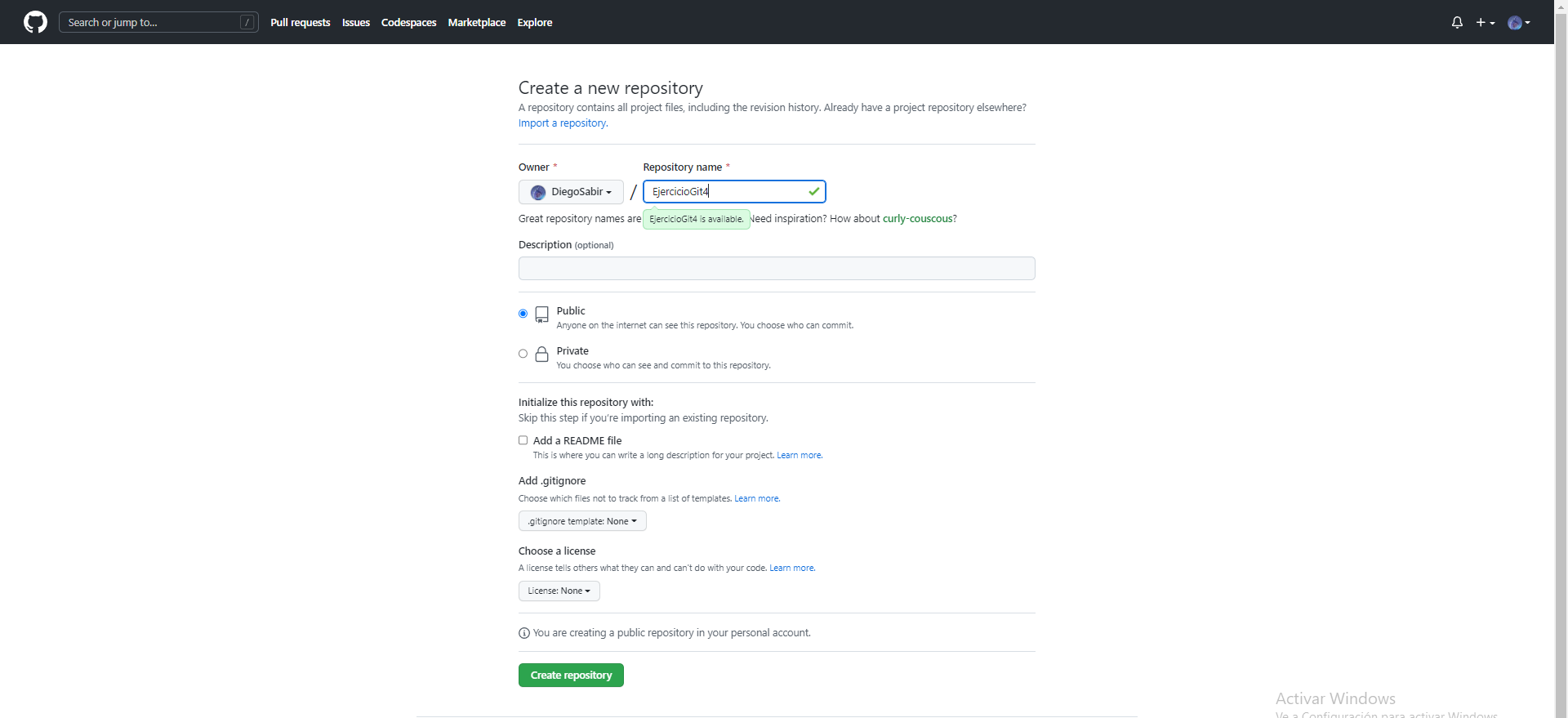
1.2 → Este README.md para ir editándolo y creando el registro de trabajo con capturas.

1.3 → Una carpeta "img" para las imágenes que serán referenciadas desde el README

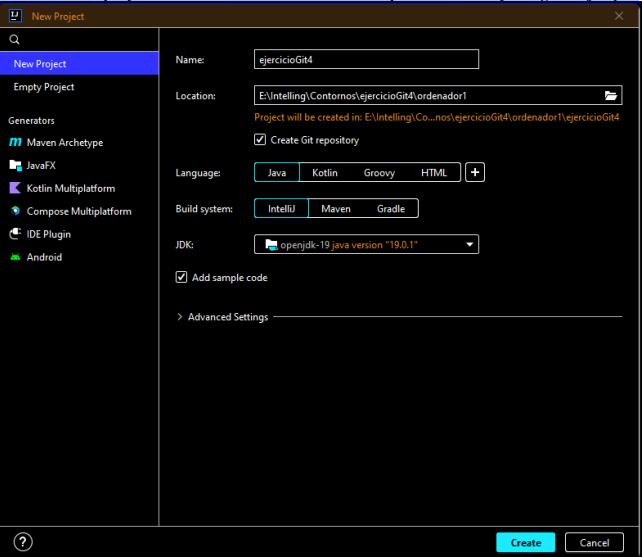




**2. Creamos un repositorio vacío (sin README ni gitignore) en GitHub.**

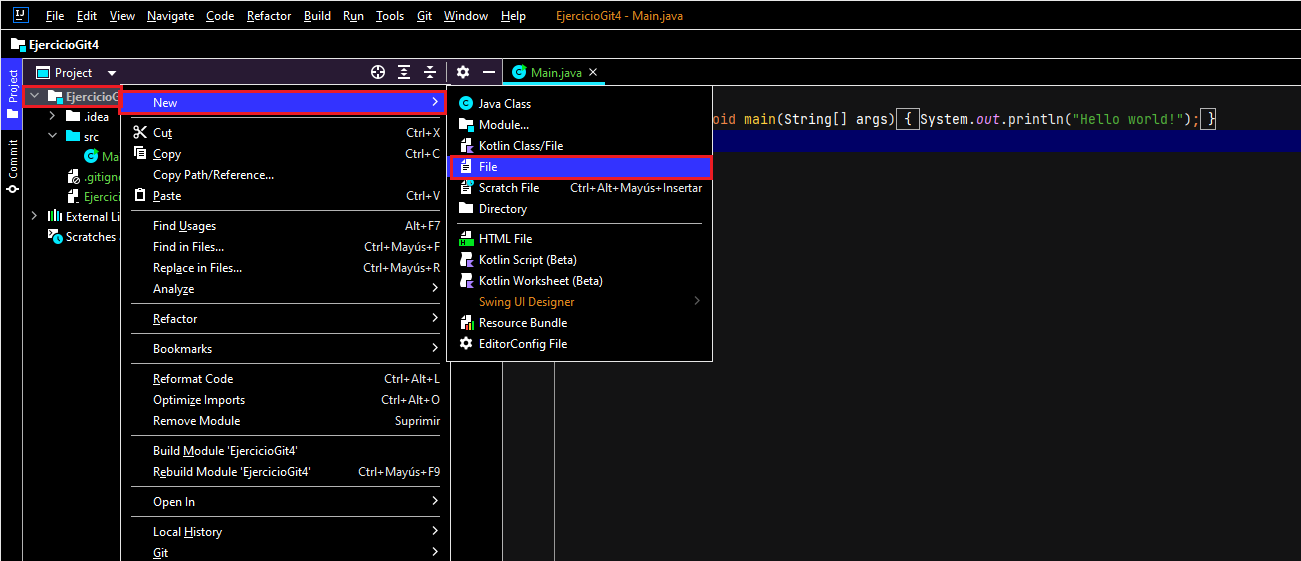
****

**3- Creamos un proyecto de IntelliJ en Ordenador1 con repositorio de Git y código de ejemplo.**

****

**4- Creamos un nuevo README.md localmente (no el presente con el registro de trabajo, que se añadirá al final sustituyendo a ese.)**

4.1 → Nos ponemos encima del proyecto, le damos al botón derecho del ratón, new “File”.

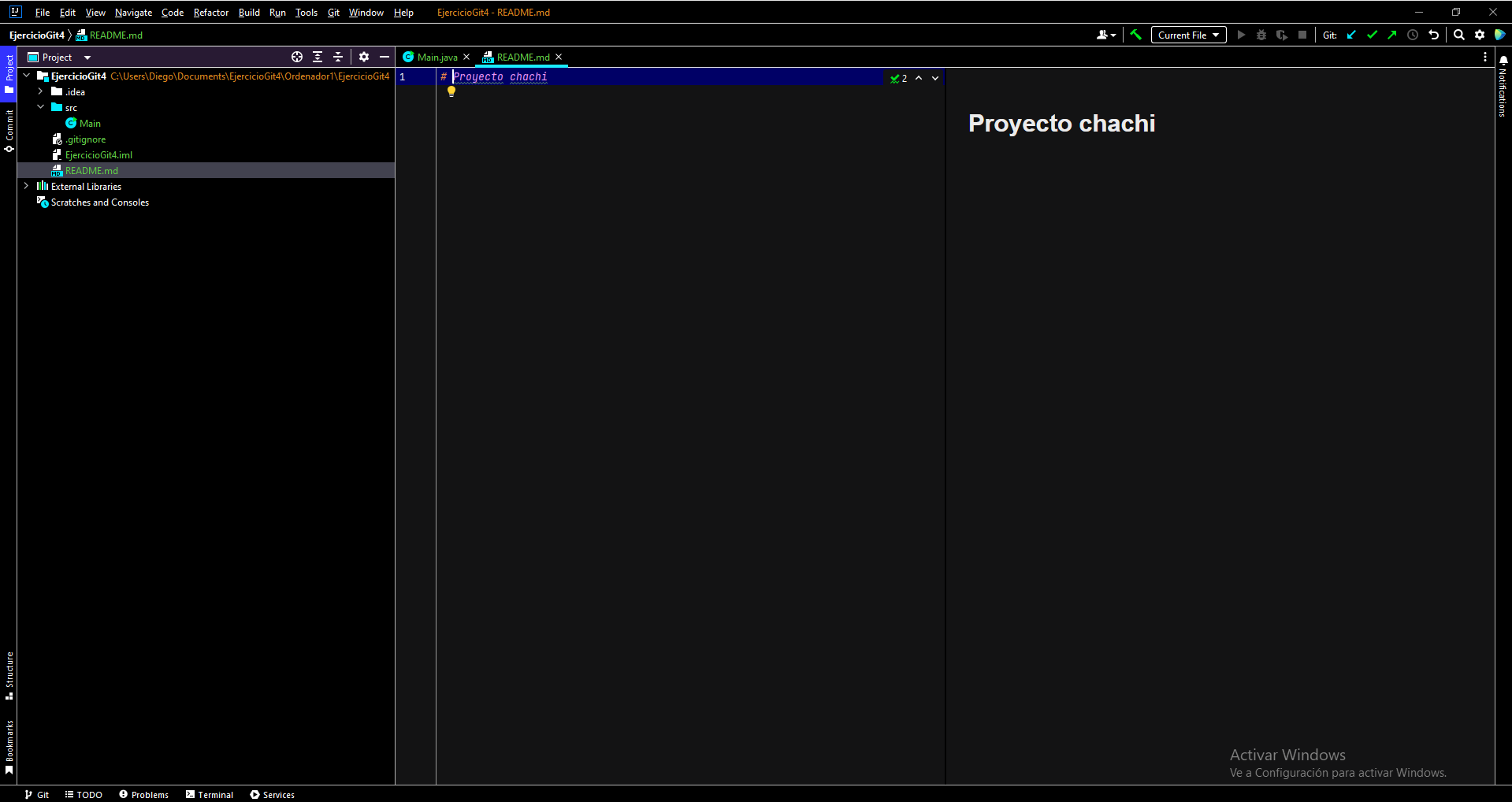


4.2 → Nos saldrá este cuadro, ponemos “README.md” (para crear el archivo README.md)



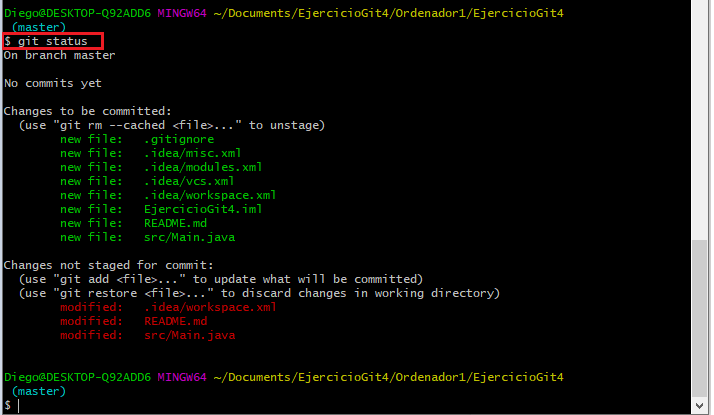
**5. Realizamos un primer commit y push de la base del proyecto. NOTA: ved mi log para comprobar que hacéis commits idénticos a los míos.**

5.1 → Editamos el archivo README.md NOTA: (Se guarda automáticamente)

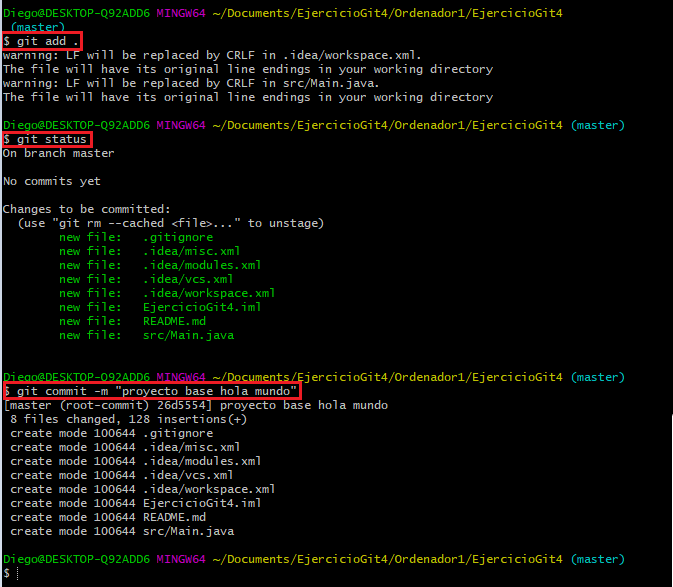


5.2 → Comprobamos el estado de los archivos antes de preparar los archivos con el comando →

*git status*



5.3 → Añadimos los archivos con *git add .* y comprobamos que se añadido con *git status* y creamos el primer commit con *git commit -m “proyecto base hola mundo”*



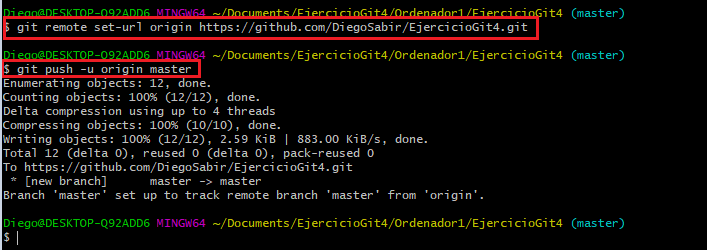
5.4 Una vez creado el commit ahora nos toca subirlo al repo remoto →

*git remote add origin https://github.com/DiegoSabir/EjercicioGit4.git*

Para subir el proyecto al repositorio → *git push -u origin master*

NOTA: Si eres retrasado como yo y fallas el URL pon el siguiente comando para corregir →

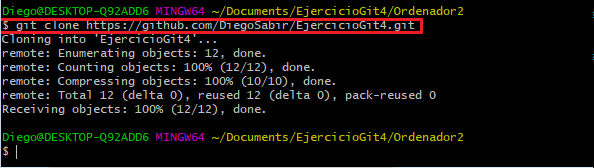
*git remote set-url origin https://github.com/DiegoSabir/EjercicioGit4.git*

**

**6. Clonamos el proyecto dentro de Ordenador2 y lo abrimos.**

NOTA: Antes de clonar, tenemos que estar el directorio Ordenador2

*git clone https://github.com/DiegoSabir/EjercicioGit4.git* → para clonar vuestro repo en local

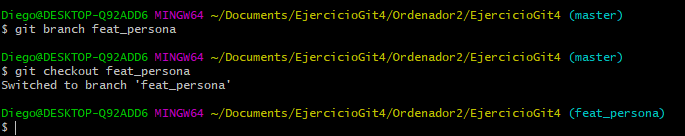


**7. Creamos una nueva funcionalidad, para lo cual creamos una rama "feat\_persona" y la desarrollamos (ver mi log de commits). Una vez lista subimos la rama (git push feat\_persona). Después, nos movemos a master para hacer un merge de "feat\_persona" localmente. Al no haber nuevos commits en master, el merge debería ser automático y sin conflictos. Después del merge, hacemos un push de master con los nuevos cambios añadidos.**

7.1 Primer paso crear una rama “feat\_persona” y cambiar a rama feat\_persona

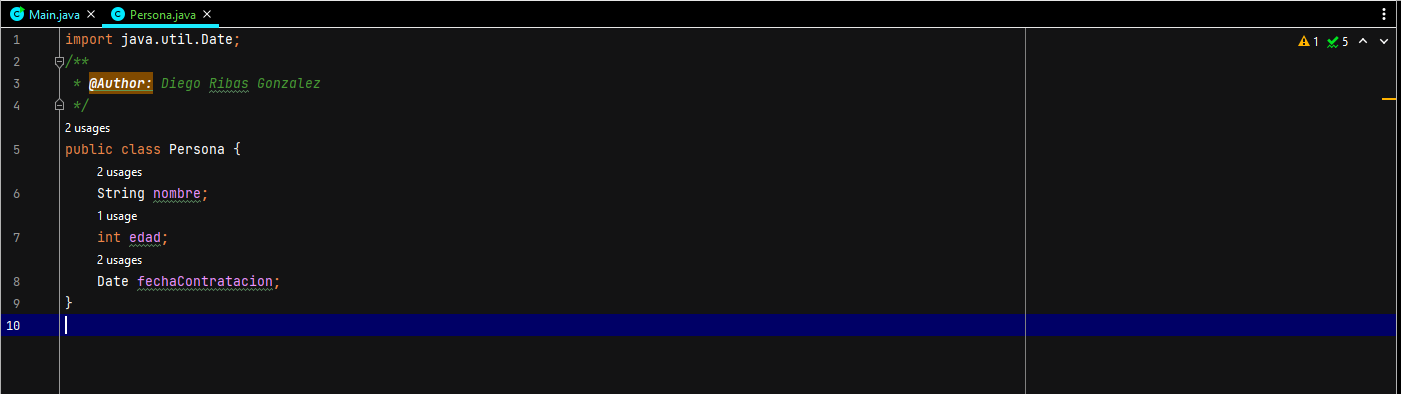
*git branch feat\_persona* → para crear una rama

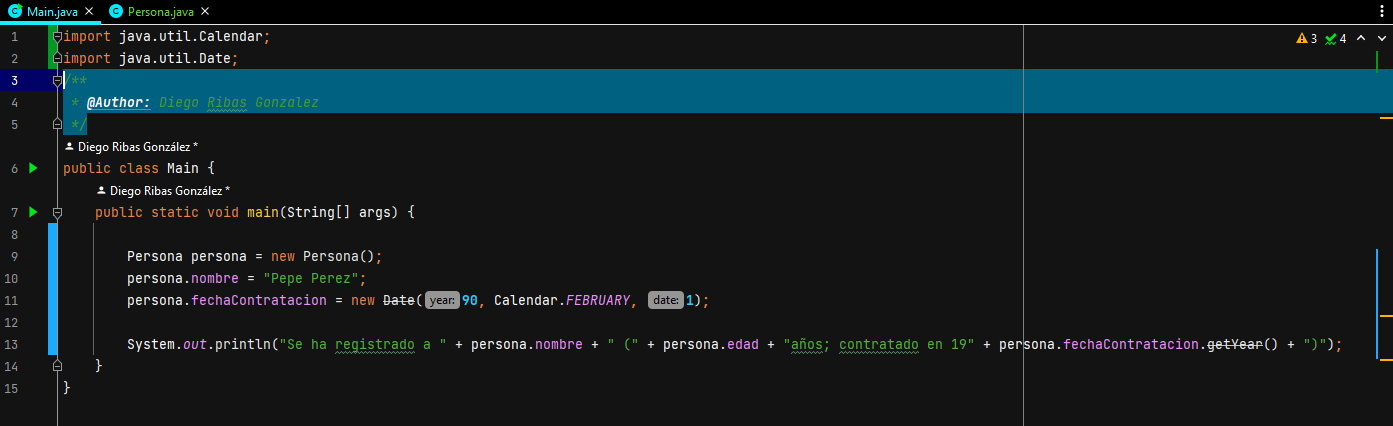
*git checkout feat\_persona* → para cambiar de rama



7.2 Segundo paso entrar en el proyecto en el Ordenador2/ejercicioGit4 ; creamos el objecto persona “Persona.java” y editamos “main.java”

NOTA: Modificamos el main “como está en commit probando un objeto Persona”

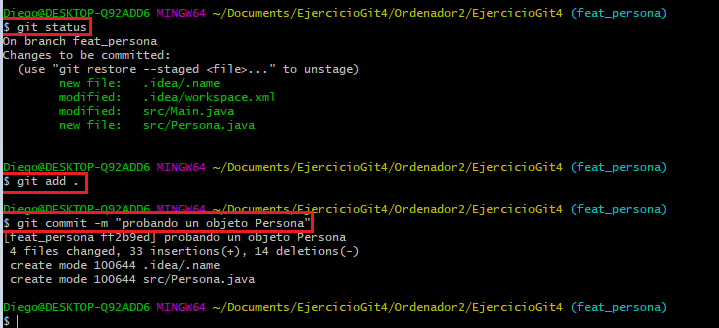




*git status* → Para comprobar estado de los archivos

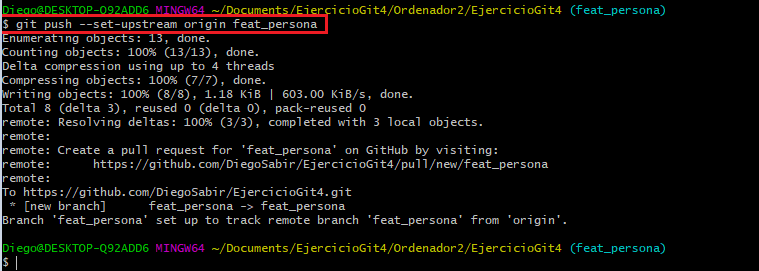
*git add .* → Para preparar los archivos existentes

*git commit -m “probando un objecto Persona”* → Ahora lo tenemos preparado para el commit



7.3 Ahora nos toca subir al remoto a la nueva rama feat\_persona

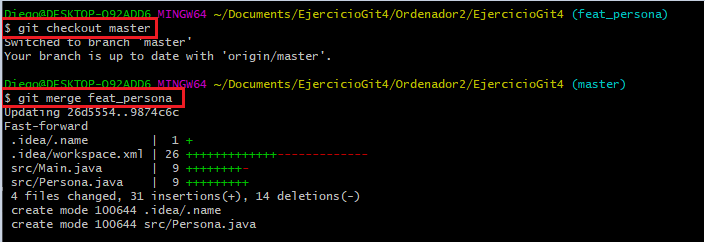
*git push* *--set-upstream origin feat\_persona* → Porque no lo tenemos configurado automáticamente, utilizamos este comando.



7.4 Ahora nos toca cambiar de rama a master y mergear(fusionar ramas) de la rama feat\_persona a master

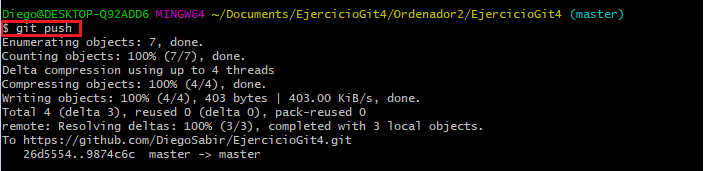
*git checkout master* → Cambiar de rama a master

*git merge feat\_persona* → Para fusionar la ramas



Por último nos toca subir el commit que mergeamos de feat\_persona

*git push* → Subir repo a remoto a la rama master

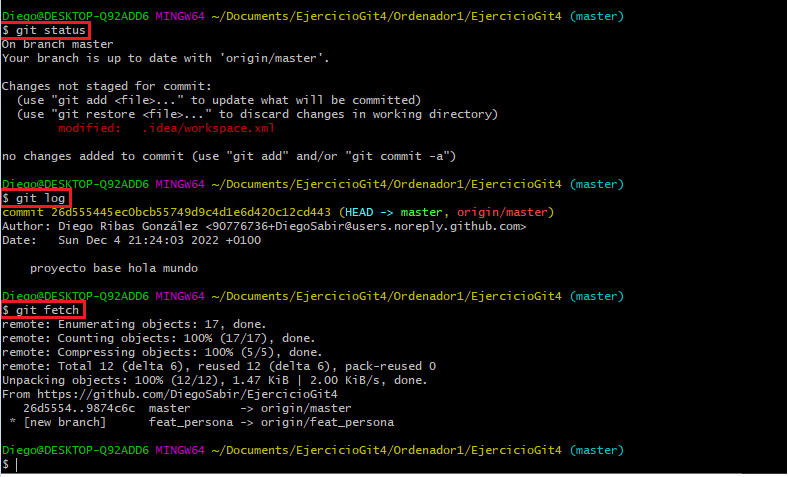


**8- Volvemos a Ordenador1. Recordamos el estado de nuestro proyecto en este ordenador haciendo un git status y un git log. Si no nos indica que el master local está por detrás del remoto (origin/master), será necesario hacer un git fetch.**

*git status* → Comprobar el estado del archivos

*git log* → Comprobar los commits

*git fetch* → Actualizar los cambios del repositorio

****

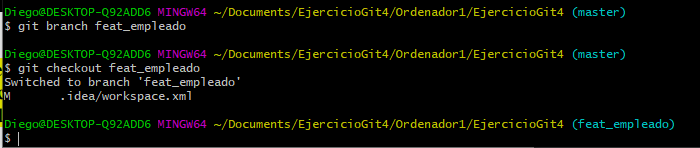
**9- Para poder seguir trabajando desde Ordenador1, deberíamos hacer antes de nada un pull para traernos el nuevo commit que tiene master en remoto. Sin embargo, vamos a probar qué sucedería si nos olvidamos de ello.**

**NOTA: Este paso nos saltamos, lo normal había que hacer un git pull “para que descargue el proyecto, ramas y commit de remto”.**

**10 - Desarrollamos una nueva funcionalidad "feat\_empleado" en su correspondiente rama.**

*git branch feat\_empleado* → creamos una nueva rama feat\_empleado

*git checkout feat\_empleado* → nos cambiamos a la rama feat\_empleado

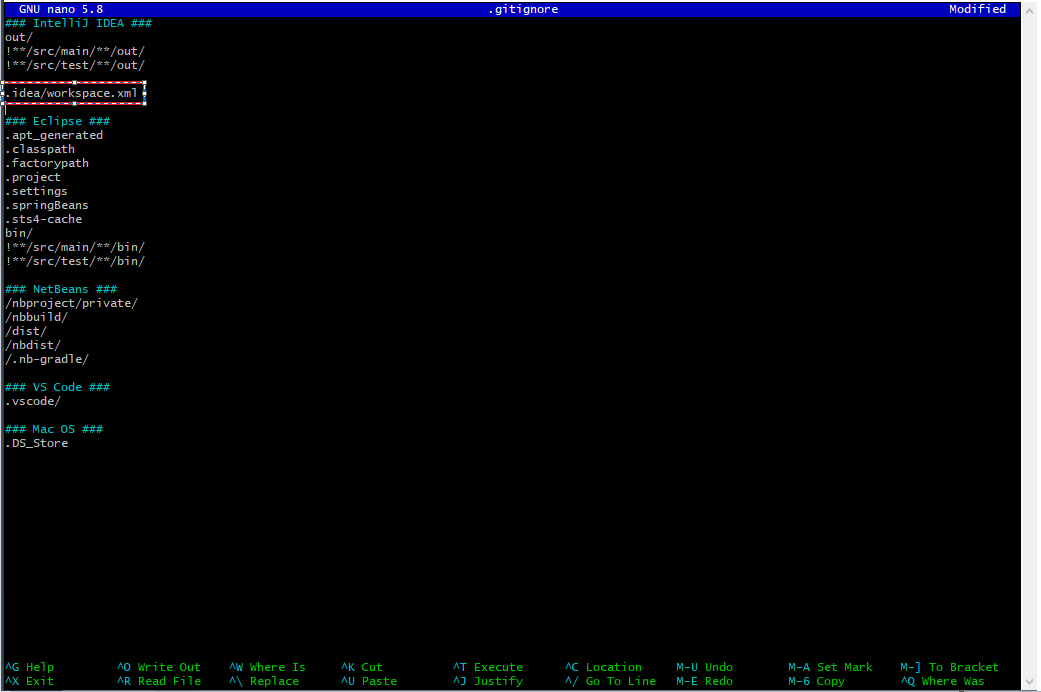


10.1 Durante el desarrollo, antes de hacer el primer commit en feat\_empleado, detectamos que el fichero .idea/workspace.xml cambia automáticamente a menudo al abrir el proyecto en IntelliJ (es un archivo de metadatos de proyecto de IntelliJ).

Editamos el archivo .gitignore del directorio ordenador1/ejercicioGit4:

Añadimos .idea/workspace.xml, para ignorarlo





10.2 Hacemos un primer commit para añadir .idea/workspace.xml al .gitignore. Además, como los ficheros de ese directorio ya se han añadido previamente al repositorio, tendremos que eliminarlos de él haciendo:

- Antes de crear el commit tenemos que borrar la cache archivo del workspace.xml

*git rm --cached .idea/workspace.xml* → utilizamos este comando para borrar la cache archivo y su cache

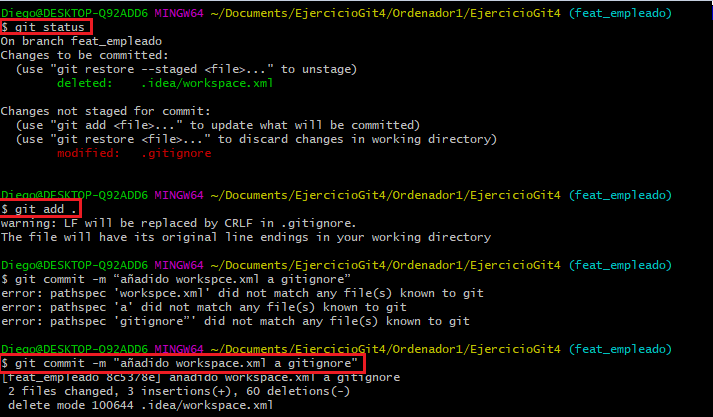


Ahora creamos el commit “añadido workspce.xml a gitignore”

• git status

• git add .

• git commit -m “añadido workspce.xml a gitignore”



10.3 Hacemos los otros 2 commits y pusheamos.

Primero creamos el objecto empleado Empleado.java, editamos el archivo Main.java y su commit “probando objeto empleado”

- Creamos el objecto empleado Empleado.java