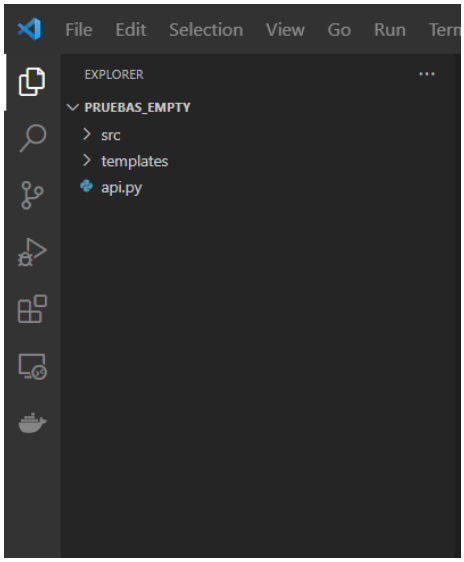
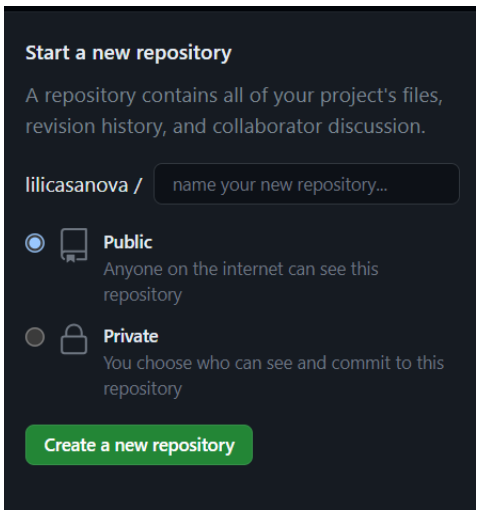
GUÍA EJERCICIO DEVELOP

1. Creamos nuestra carpeta con nuestro archivo api.py y carpetas src + templates. Siempre en un directorio que no tenga relación con el repo del profesor o cualquiera previamente vinculado con github. Abrimos únicamente ese folder a través de VSCode, de forma que quede así



2. Creamos un repo vacío en github.com, a través de nuestro usuario y mediante la página web.

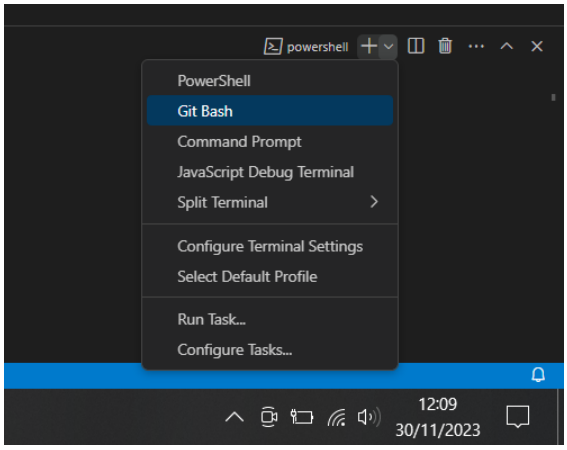


3. Tras especificar **nombre** de nuestro repositorio nuevo, se nos abre una página de git que nos permite copiar tres fragmentos de código diferentes. Copiamos el primero, para inicializar nuestro repo por primera vez

...or create a new repository on the command line

```
echo "# pruebas" >> README.md
  git init
  git add README.md
  git commit -m "first commit"
  git branch -M main
  git remote add origin https://github.com/lilicasanova/pruebas.git
  git push -u origin main
```

4. Inicializamos el repositorio a través de la terminal. Recordad abrir la terminal de Visual y especificar *git bash* (en Windows). Pegamos el fragmento de código que hemos sacado de github. ¡Ya está vinculado nuestro repo local con el remoto!



5. Ya hemos hecho el primer paso del ejercicio, para continuar con el segundo, tenemos que concretar los colaboradores en los ajustes del repositorio a través de la página web de github, añadiendo su nombre de usuario. Una vez acepten la petición, formarán parte del proyecto

6. Haremos nuestro siguiente push con los pasos que repetiremos continuamente, para añadir las carpetas al remoto. Con add y el punto añadimos todos los cambios nuevos. Con commit especificamos el mensaje del cambio. Con push lo subimos.

```
git add .
git commit -m “añadir carpetas iniciales”
git push -u origin main
```

7. Siguiente paso es crear la rama develop y a continuación pushearla para salvar el cambio. Con el primer comando al no existir la rama, se crea. Con checkout nos cambiamos a ella y con push, la subimos.

```
git branch develop-branch
git checkout develop-branch
git push origin develop-branch
```

8. Los siguientes pasos cada vez que tengamos que crear una nueva rama, repetiremos el punto 7.

9. Cuando tengas que empezar a pegar fragmentos de código, asegúrate de estar en la rama que se pide (cambiamos con *checkout*). Después de añadir cada fragmento de código, salvamos los cambios con lo que realizábamos en el punto 6. Cambiando cada vez el mensaje de commit especificando las modificaciones.

10. Antes de realizar los primeros merge y fusionar las ramas con contenido nuevo, tenemos que hacer los push de cada fragmento de código añadido. Para ello repetiremos paso 6, pero a partir de este momento quedará de esta forma:

```
git add .
git commit -m "fragmento código 1"
git push --set-upstream origin branch-ticket-1
```

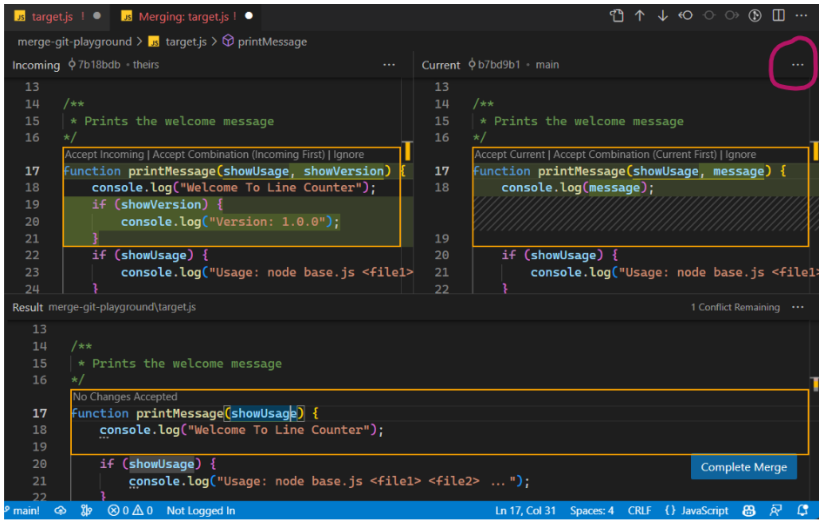
11. Directamente después, para hacer los **merge**, tendremos que pasarnos **A** la rama a la que queremos traer la información nueva y desde ella especificar la rama **DE** la que traemos información nueva. Veremos como en nuestro api.py vacío de epic branch, aparece lo que hemos colocado en branch-ticket-1

```
git checkout epic-branch-1
git merge branch-ticket-1
```

12. Repetimos los pasos con branch-ticket-2 y así en las sucesivas creaciones de sub-branches y merge

13. En el punto 3 de trabajador 2 y 3, cuando aparece un conflicto tras hacer dos merge con códigos que muestran la misma información, pero con diferentes archivos “html”, tenemos que escoger la opción de la derecha. En la que realmente se llama a un archivo existente en la carpeta templates (map existe, ÑAÑAÑAÑAÑA no).

    Para ello vamos a la ventana derecha de la división que se hace mostrándonos ambos códigos comparados y hacemos clic en las opciones de la ventana derecha, escogiendo “ACCEPT ALL CHANGES FROM RIGHT”



14. Los pasos siguientes consisten en repetir merge y commits de código. La diferencia única cuando entran developer 2 y developer 3, es que antes de poder empezar a trabajar tienen que clonar el repositorio que creó developer 1 con

    git clone https://github.com/enlacedelrepo

    cd -nombre repo-

15. Además de clonar, cuando cogemos el repositorio de developer 1 y queremos traer los cambios que se han ido haciendo desde ese repo, tenemos que ir creando los **pull request** cada vez que otro usuario haga git push origin, desde la interfaz de github y en la rama nuestra a la que queremos traer los cambios, haremos pull request. El otro usuario aceptará y se fusionarán las ramas.

16. Finalmente tendremos que asegurarnos de que todas las subramas de epic-branch están fusionadas con su rama origen. Tras esto, fusionar las ramas epic con develop-branch y hacer push de develop-branch

    git push origin develop-branch