PRÁCTICA 1: Git y GitHub

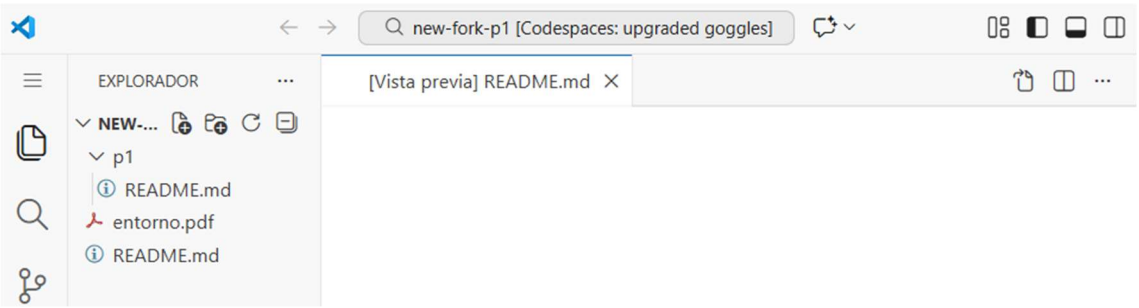
PROGRAMACIÓN DE APLICACIONES TELEMÁTICAS

Para la primera parte de la práctica, se nos pidió ejecutar una serie de comandos, explicando qué significan, para qué se han usado y los logs que han dejado en la consola tras su ejecución.

- **git clone** https://github.com/gitt-3-pat/p1

```
@MartinaOD →/workspaces/new-fork-p1 (main) $ git clone https://github.com/gitt-3-pat/p1
Cloning into 'p1'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 5 (from 1)
Receiving objects: 100% (6/6), done.
```

Al introducir "`git clone https://github.com/gitt-3-pat/p1`", copiamos o clonamos (como su propio nombre indica) en local el repositorio que está en GitHub y que se indica con la URL, creándose así una nueva carpeta o repositorio en nuestro directorio /workspaces/new-fork-p1:



La respuesta en terminal muestra el proceso de clonado donde se crea una carpeta local llamada 'p1' y en ella copia el repositorio. Posteriormente, lista los archivos y cambios que existen en el repositorio remoto, contando todo aquello que va a enviar hasta concluir en un total de 6 objetos. Finalmente, recibe todos los objetos, copiando el repositorio completo en local.

Además, para poder trabajar en local y luego subir sin problemas los cambios en mi repositorio, se ha de eliminar el fichero oculto '.git/', de esta manera, trabajaremos con una carpeta y no con un repositorio que apunte o esté ligado al remoto del cual se clonó:

```
@MartinaOD →/workspaces/new-fork-p1 (main) $ cd p1
@MartinaOD →/workspaces/new-fork-p1/p1 (main) $ ls -la
total 16
drwxrwxrwx+ 3 codespace codespace 4096 Jan 23 15:55 .
drwxrwxrwx+ 4 codespace root      4096 Jan 23 15:55 ..
drwxrwxrwx+ 7 codespace codespace 4096 Jan 23 15:55 .git
-rw-rw-rw-  1 codespace codespace  572 Jan 23 15:55 README.md
@MartinaOD →/workspaces/new-fork-p1/p1 (main) $ rm -rf .git/
@MartinaOD →/workspaces/new-fork-p1/p1 (main) $ cd ..
```

- **git status**

```
@MartinaOD →/workspaces/new-fork-p1 (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   p1/README.md
```

Con este comando podemos ver el estado actual de nuestro repositorio. Al introducirlo en la línea de comandos, no se produce ningún cambio, simplemente se informa del estado.

En primer lugar, nos indica en qué rama estamos del repositorio, la cual es 'main'. Posteriormente, nos indica que la rama está actualizada, es decir, que no hay cambios pendientes de subir o de traer.

Este comando, en la práctica, lo realicé después de aplicar el comando 'git add', por lo que las últimas líneas que se muestran en consola, indican que los cambios ya están preparados para el 'git commit'.

Si hubiéramos ejecutado antes del 'git add' este comando, las últimas líneas mostrarían algo como:

```
Changes not staged for commit:

new file: p1/README.md
```

Lo cual sería indicador de que hay archivos nuevos, con los cuales se ha detectado el cambio, pero aún no se han guardado.

- **git add .**

```
@MartinaOD →/workspaces/new-fork-p1 (main) $ git add .
```

El comando introducido sirve para indicar a Git qué cambios queremos guardar en el 'commit' que se realice posteriormente. En este caso, con el punto al final, se encarga de seleccionar todo cambio realizado.

Este es necesario para poder guardar los cambios introducidos y poder posteriormente subir dichos cambios locales al repositorio en remoto.

En cuanto a los logs en pantalla, no se muestra nada.

- **git commit -m "TU MENSAJE"**

```
@MartinaOD →/workspaces/new-fork-p1 (main) $ git commit -m "finally"
[main 88ab4e1] finally
 1 file changed, 24 insertions(+)
 create mode 100644 p1/README.md
```

Este comando toma todo lo seleccionado con 'git add' y guarda todo ello, pasando así a formar parte de la historia del repositorio. Si utiliza como paso consecutivo a 'add' para guardar la modificación realizada al inicio cuando decidimos clonar 'p1' en nuestro *fork*.

Los logs nos muestran la rama donde se hizo el 'commit' con el mensaje 'finally', mostrando el cambio que se produjo: 1 archivo afectado con 24 líneas añadidas. En dicha carpeta creada se crea a su vez 'README.md' con permisos normales.

- **git push**

```
@MartinaOD →/workspaces/new-fork-p1 (main) $ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 349 bytes | 349.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/MartinaOD/new-fork-p1
   ed46266..88ab4e1  main -> main
```

'git push' es el comando encargado de subir los cambios que ya han sido guardados y realizados en local a nuestro repositorio remoto, se encarga de enviarlo fuera. Es por este motivo que interesa ejecutar este comando ahora, para poder actualizar nuestro repositorio remoto al cual podrán acceder otros usuarios.

Dentro de los logs presentes, podemos apreciar (por orden) cómo se enumeran y cuentan los commits y archivos que van a ser enviados, los comprime y escribe los objetos comprimidos en el remoto. Una vez hecho, se muestra el enlace de a dónde fueron enviados con el comando a remoto, habiendo enviado la rama local 'main' al 'main' remoto.

- **git checkout -b feature/1**

```
@MartinaOD →/workspaces/new-fork-p1 (main) $ git checkout -b feature/1
Switched to a new branch 'feature/1'
```

Este comando crea una nueva rama de nombre 'feature/1' y se cambia a dicha rama. Esta misma acción se refleja en el log que se muestra en la terminal.

- **git checkout main**

```
@MartinaOD →/workspaces/new-fork-p1 (feature/1) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Este comando permite regresar a nuestra rama principal 'main'. Los logs nos muestran dos líneas: en la primera se muestra el cambio de rama a 'main', mientras que, en la segunda, se refleja que nuestra rama local está sincronizada con la rama remota 'origin/main' (por lo que no hay nada que se debe enviar o recibir con un 'push' o 'pull').