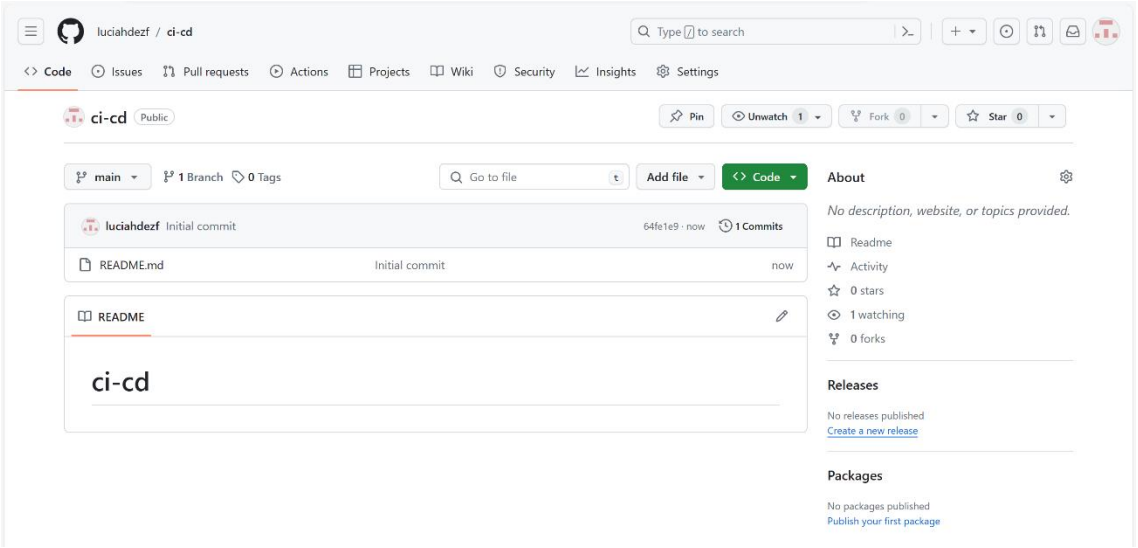
PRÁCTICA 1

GIT Y GITHUB:

En primer lugar, para poder llegar a cabo las prácticas y el desarrollo de la asignatura, he creado una cuenta en GitHub con mi correo de la universidad, cuyo nombre de usuario es $luciahdezf$.

Luego he creado un repositorio llamado $ci - cd$ en el que he probado los comandos para ver cuál era su función principal.



Siguiendo las instrucciones del Tema 1 en relación con GitHub fui probando todos los comandos para analizar cuál era la función de cada uno.

GIT CLONE

- El primer comando a probar es $git\ clone$, cuya función principal es hacer una copia (clonar) un elemento de otro repositorio o el repositorio completo, siempre y cuando este sea público.
- En primer lugar, he probado el comando que nos indicaban en la teoría

```
@luciahdezf ➜ /workspaces $ 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
Receiving objects: 100% (6/6), done.
```

- En este caso podemos comprobar que los elementos que hemos clonado son aquellos que se encuentran en https://github.com/gitt-3-pat/p1

GIT STATUS

- Este comando nos permite conocer cómo se encuentra el repositorio sobre el que estamos trabajando, es decir, nos indica el estado que tiene un repositorio concreto.

```
@luciahdezf ➜/workspaces $ cd /workspaces/ci-cd
it -m "feat: homepage"
git push origin main@luciahdezf ➜/workspaces/ci-cd (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

- En este caso, se puede comprobar que nos indica que el estado de nuestro repositorio es “up to date” lo que implica que podemos hacer cosas sobre el repositorio ya que todo lo que hemos realizado anteriormente se ha guardado de manera correcta
- En caso de que no nos indicase esa frase, podría significar que el repositorio no se ha guardado correctamente o que está en proceso de carga de datos.

GIT ADD

- Este comando nos permite establecer cambios en las zonas del área de repositorio que deseemos.
- En el caso de querer hacer cambios sobre un elemento concreto, se deberá incluir el nombre del archivo a continuación del comando

```
@luciahdezf ➜/workspaces/ci-cd (main) $ git add archivo
```

- Si queremos que el cambio se realice en todos los elementos el comando que utilizaremos será el que vimos en clase

```
@luciahdezf ➜/workspaces/ci-cd (main) $ git add .
```

GIT COMMIT

- Este comando tiene la función de confirmar que los cambios se han realizado en el área establecida con los comandos anteriores.
- Como vimos en clase, esto se puede confirmar a través de uan serie de mensajes en la línea de comandos

```
@luciahdezf ➜/workspaces/ci-cd (main) $ git commit -m "Mensaje confrmacion"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

- En este caso, el ejemplo nos indica que no hay nada que confirmar de cambios ya que no se ha realizado ningún cambio en el repositorio, es decir que no hay ningún archivo creado internamente en el repositorio.

GIT PUSH

- Este comando es importante para que los cambios que tú has realizado en un repositorio sean visibles para un repositorio remoto, es decir, de otra persona u otro repositorio diferente.
- Otro elemento importante para esto es el hecho de que se puede usar para hacer trabajos colaborativos, ya que sin este comando el resto de los miembros del equipo no serian capaces de ver los cambios que se realizan.

```
@luciahdezf ➜/workspaces/ci-cd (main) $ git push origin feat/add-body
```

- Aunque no es de este comando, otro elemento para este trabajo en el equipo es el comando de pull, que te permite obtener los documentos actualizados de un repositorio

```
@luciahdezf ➜/workspaces/ci-cd (main) $ git pull origin main
From https://github.com/luciahdezf/ci-cd
 * branch            main       -> FETCH_HEAD
Already up to date.
```

```
@luciahdezf ➜/workspaces/ci-cd (main) $ git add .
@luciahdezf ➜/workspaces/ci-cd (main) $ git commit -m "feat: homepage"
[main 5b993b6] feat: homepage
 1 file changed, 5 insertions(+)
 create mode 100644 src/index.html
@luciahdezf ➜/workspaces/ci-cd (main) $ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 385 bytes | 385.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/luciahdezf/ci-cd
   64fe1e9..5b993b6  main -> main
```

GIT CHECKOUT

- Este comando se usa para salir de un determinado elemento del repositorio, es decir, te permite cambiar de elemento en un mismo repositorio.
- Esto es importante para guardar los cambios realizados en el repositorio o para acabar de hacer los cambios establecidos

```
@luciahdezf ➜/workspaces/ci-cd (main) $ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
```

- En este caso indicamos que lo que queremos hacer es salir al main de nuestro repositorio, pero, como en nuestro caos partíamos de ese no supone ningún cambio.
- Esto nos lo indican con el mensaje “Already on main”

```
@luciahdezf ➜/workspaces/ci-cd (main) $ cd /workspaces/ci-cd

@luciahdezf ➜/workspaces/ci-cd (main) $ git checkout -b feat/add-body
Switched to a new branch 'feat/add-body'
@luciahdezf ➜/workspaces/ci-cd (feat/add-body) $
```

ENTORNO DE DESARROLLO

Una vez que ya hemos hecho la parte de Git, pasamos a comprobar que tenemos todos los elementos descargados en nuestro ordenador para poder ejecutarlos.

JAVA 17:

- Como para la asignatura del año pasado, tenía descargado Java 19, en las variables de entorno he dado prioridad a la versión anterior para que el sistema recurra a Java 17 como nos indicaba la práctica
- Esto nos ha permitido que al indicar la versión de java del sistema nos salga la versión 17

```
C:\Users\lucia>java --version
java 17.0.10 2024-01-16 LTS
Java(TM) SE Runtime Environment (build 17.0.10+11-LTS-240)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.10+11-LTS-240, mixed mode, sharing)
```

MAVEN:

- El primero es descargar Maven
- Una vez que lo hemos hecho, comprobamos que existe en nuestro sistema y que está con la versión correcta

INTELIJ:

- Esta aplicación tuve que instalármela ya que en ningún curso tuve que hacerlo por lo que me la instalé

VSCODE:

- Esta aplicación ya la tenia instalada con anterioridad en mi ordenador