Amaia ROTAECHE MONTERO

3.B GITT+BA

PAT: PRACTICA1

Link del repositorio:

https://github.com/AmaiaRM/hello-world

DESCARGAS:

En primer lugar, descargamos y comprobamos las herramientas necesarias para el desarrollo de la asignatura.

JAVA 17:

```
C:\Users\Amaia>java --version
java 16.0.2 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)
```

La versión de Java disponible en mi sistema es anterior a la requerida pero sus funcionalidades no varían luego mantendremos la versión 16 para evitar descargarnos el programa “duplicado”.

MAVEN:

```
C:\Users\Amaia>mvn --version
Apache Maven 3.8.7 (b89d5959fcde851dcb1c8946a785a163f14e1e29)
Maven home: C:\Program Files (x86)\apache-maven-3.8.7
Java version: 16.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-16.0.2
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

DOCKER:

```
C:\Users\Amaia>docker --version
Docker version 20.10.22, build 3a2c30b
```

INTELLIJ:

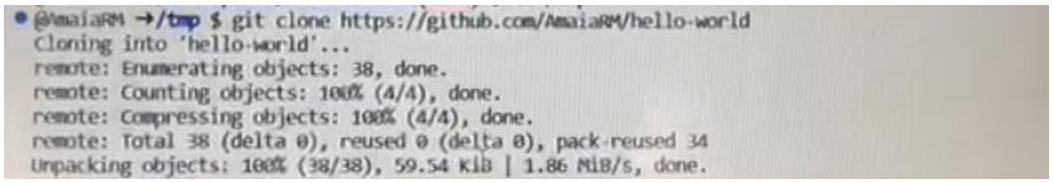
Se trata de un ejemplo de proyecto realizado en el editor de código intelliJ.

Desarrollo en GITHUB

En primer lugar, hacemos un fork de un repositorio ya creado por otro usuario. De esta forma, hacemos una copia del repositorio para poder realizar las modificaciones que deseemos desde nuestra propia cuenta de usuario, sin que estas afecten al repositorio de origen.

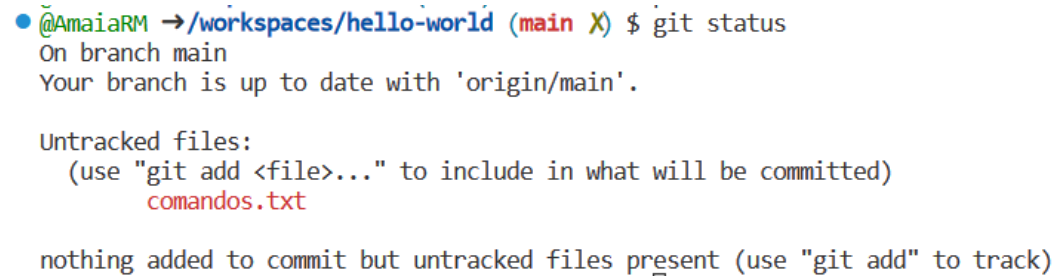
Git clone:

Como su nombre indica, este comando clona el repositorio al que apunta en un nuevo directorio local. De esta forma, todo el contenido el repositorio previamente “forkeado” y remoto se copiará en el directorio /tmp de nuestro equipo local.



Git status:

Comprueba el estado actual del directorio en el que se trabaja. Permite observar los cambios que se han realizado (creación de archivos) e informa sobre la falta de confirmación de archivos modificados.



Git add:

Añade el archivo especificado (sobre el que se ha realizado alguna modificación) al área de preparación. Los cambios se guardan en un "borrador" y no se registran en el directorio hasta que se ejecuta la confirmación definitiva con git commit.(Se pueden añadir todos los cambios con "gitt add .".)

```
● @AmaiaRM →/workspaces/hello-world (main X) $ git add .
```

Git commit:

Registra todos los cambios realizados y guardados en el área de preparación (borrador) en el repositorio de trabajo.

```
⊗ @AmaiaRM →/workspaces/hello-world (main) $ git commit -m "comandos.txt"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```

Git push:

Se carga el contenido desde el directorio local sobre el que se trabaja al directorio remoto de tal forma que el resto de usuarios puedan observar las modificaciones hechas sobre el repositorio.

```
● @AmaiaRM →/workspaces/hello-world (main) $ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 336 bytes | 336.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/AmaiaRM/hello-world
   48fe276..8bcb00e  main -> main
```

En definitiva, los cambios en GitHub funcionan en 3 pasos:

- Git add: se añaden como borrador en el repositorio local
- Git commit: se confirma el borrador y se registran dichos cambios en el repositorio local
- Git push: se cargan los cambios desde el repositorio local al repositorio remoto

Git checkout:

Permite desplazarte entre las diferentes versiones de código de un repositorio, tanto entre ramas como archivos y confirmaciones. Una de las ventajas de Git es el desarrollo mediante ramas. Si se desea probar cambios o intentar solucionar errores, se genera una nueva rama para alojarlos de tal forma que el "código prueba" no desestabilice el principal. Y "git checkout" permite desplazarte sobre estas ramas creadas.

```
● @AmaiaRM →/workspaces/hello-world (main) $ git checkout
Your branch is up to date with 'origin/main'.
```

Otro comando interesante es “git branch” que indica la rama sobre la que se está trabajando. En nuestro caso, como no hemos creado ninguna rama extra, seguimos sobre la principal: