

Práctica 1 – Instalación de herramientas


Ignacio Ortega Falces

Instalación del software

Primero instalaremos los 4 programas solicitados, dejando evidencias del funcionamiento de cada una.

- Java 17
- Maven
- Docker
- IntelliJ

Comenzamos comprobando la version instalada de java en el PC

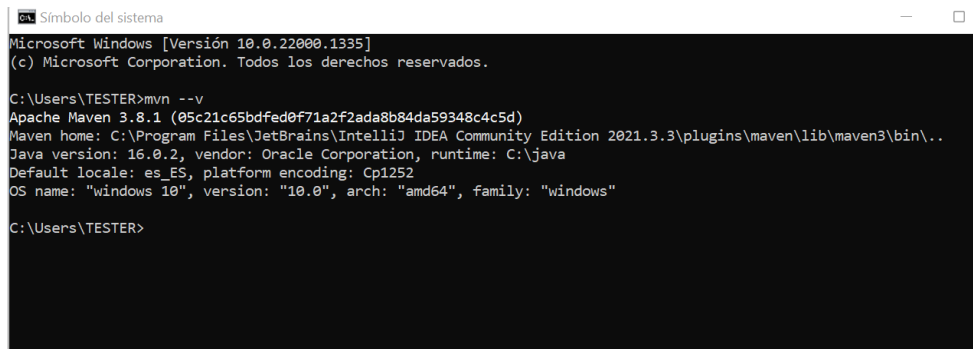


```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22000.1335]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\TESTER>java -version
java version "17.0.3.1" 2022-04-22 LTS
Java(TM) SE Runtime Environment (build 17.0.3.1+2-LTS-6)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.3.1+2-LTS-6, mixed mode, sharing)

C:\Users\TESTER>
```

Para seguir comprobaremos si tenemos Maven instalado

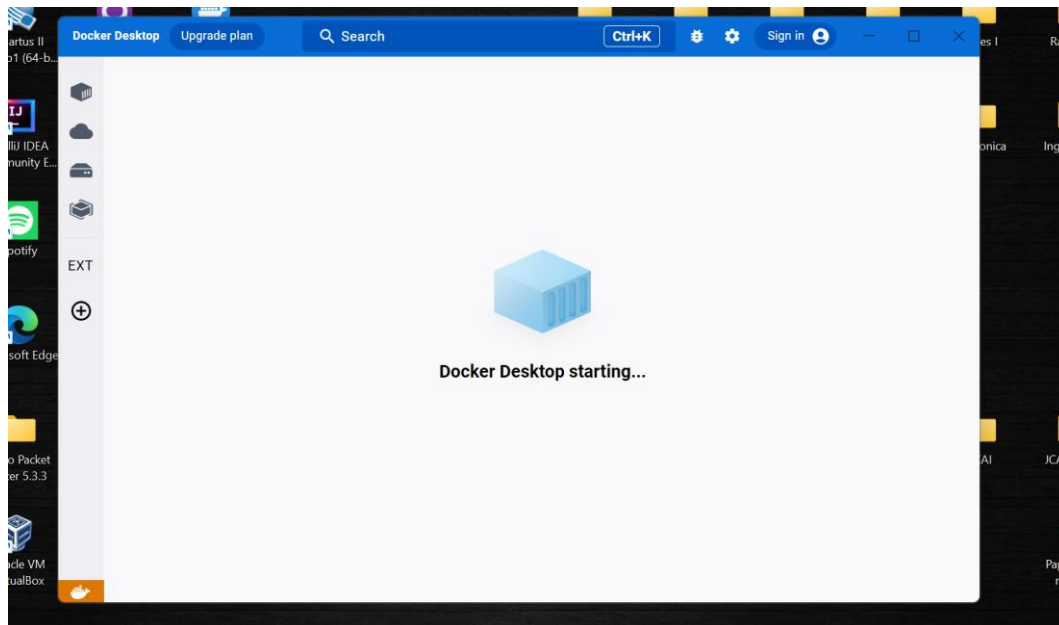


```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22000.1335]
(c) Microsoft Corporation. Todos los derechos reservados.

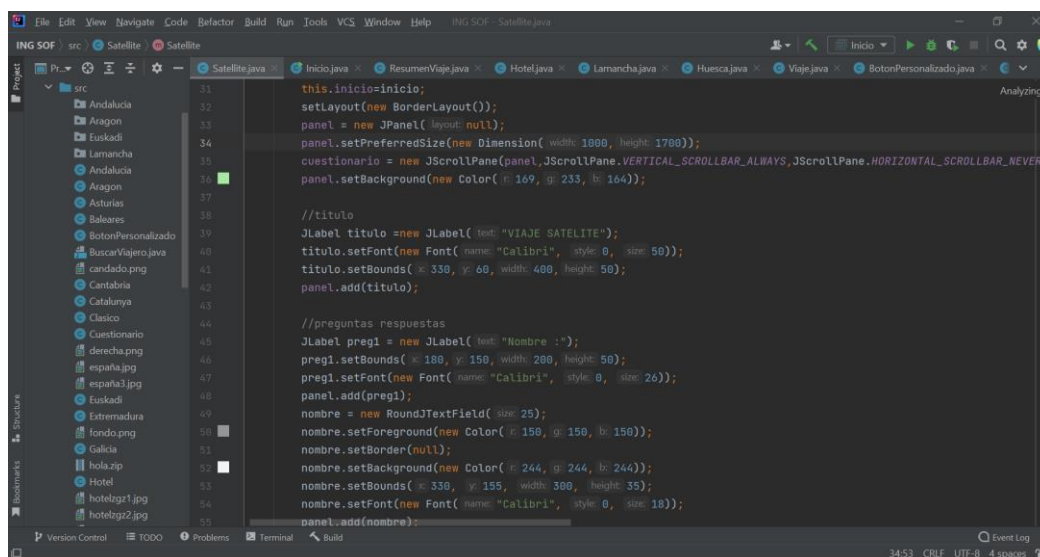
C:\Users\TESTER>mvn --v
Apache Maven 3.8.1 (05c21c65bdfed0f71a2f2ada8b84da59348c4c5d)
Maven home: C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.3.3\plugins\maven\lib\maven3\bin\..
Java version: 16.0.2, vendor: Oracle Corporation, runtime: C:\java
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Users\TESTER>
```

A continuacion instalaremos Docker y comprobamos funcionamiento



Y finalmente el editor de código fuente. En mi caso he elegido IntelliJ Community



Comandos a probar

Comenzaremos usando **git clone**. Este comando consiste en crear una copia de un repositorio de Git en otro lugar. Al clonar este, descargamos todos los archivos y también podremos trabajar en el como si estuviéramos en el original.

```
Simbolo del sistema
Microsoft Windows [Versión 10.0.22000.1455]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\TESTER>cd Desktop

C:\Users\TESTER\Desktop>cd PAT

C:\Users\TESTER\Desktop\PAT>git clone https://github.com/Nachoortegafalces/hello-world.git
Cloning into 'hello-world'...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 38 (delta 0), reused 0 (delta 0), pack-reused 34
Receiving objects: 100% (38/38), 59.56 KiB | 1.70 MiB/s, done.

C:\Users\TESTER\Desktop\PAT>
```

Clonamos el repositorio proporcionado por la practica para descargar el documento Hello world

A continuacion usaremos **git status**. Este consiste en un comando que nos dejara ver el estado actual del repositorio de Git. Mostrara informacion sobre archivos modificados, seguidos y sin seguir. Este comando nos sera util para ver que archivos han cambiado en el repositorio.

```
C:\Users\TESTER\Desktop\PAT>cd hello-world

C:\Users\TESTER\Desktop\PAT\hello-world>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\TESTER\Desktop\PAT\hello-world>
```

Git add. Este comando lo usaremos para preparar archivos para un commit. Cuando hacemos cambios en un proyecto, estos se agregaran primero con git add. De esta forma marcamos los cambios para ser incluidos en el proximo commit.

Como hemos mencionado antes, una vez realizados los cambios utilizaremos el comando **git commit**. Este se utiliza para confirmar y guardar los cambios en el repositorio de Git.

```

C:\Users\TESTER\Desktop\PAT\hello-world>
C:\Users\TESTER\Desktop\PAT\hello-world>git add .

C:\Users\TESTER\Desktop\PAT\hello-world>git commit -m "hola"
[main b543471] hola
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 pat.docx

C:\Users\TESTER\Desktop\PAT\hello-world>

```

El comando **git push** consiste en enviar cambios locales al repositorio. Después de realizar cambios en un proyecto, y confirmarlos con git commit, podremos usar git push para enviar esos cambios al repositorio remoto.

```

C:\Users\TESTER\Desktop\PAT\hello-world>git push
git: 'push' is not a git command. See 'git --help'.

C:\Users\TESTER\Desktop\PAT\hello-world>
C:\Users\TESTER\Desktop\PAT\hello-world>
C:\Users\TESTER\Desktop\PAT\hello-world>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 174.41 KiB | 8.30 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Nachoortegafalces/hello-world.git
 48fe276..b543471  main -> main

C:\Users\TESTER\Desktop\PAT\hello-world>

```

Git checkout

Finalmente el comando **git checkout**. Este se utiliza para cambiar entre ramas o descartar cambios en un archivo. El comando git checkout <branch_name> cambiara de la rama actual a la especificada. El comando git checkout <file_name> descartara los cambios no confirmados en el archivo especificado

```

C:\Users\TESTER\Desktop\PAT\hello-world>git checkout
Your branch is up to date with 'origin/main'.

C:\Users\TESTER\Desktop\PAT\hello-world>

```