

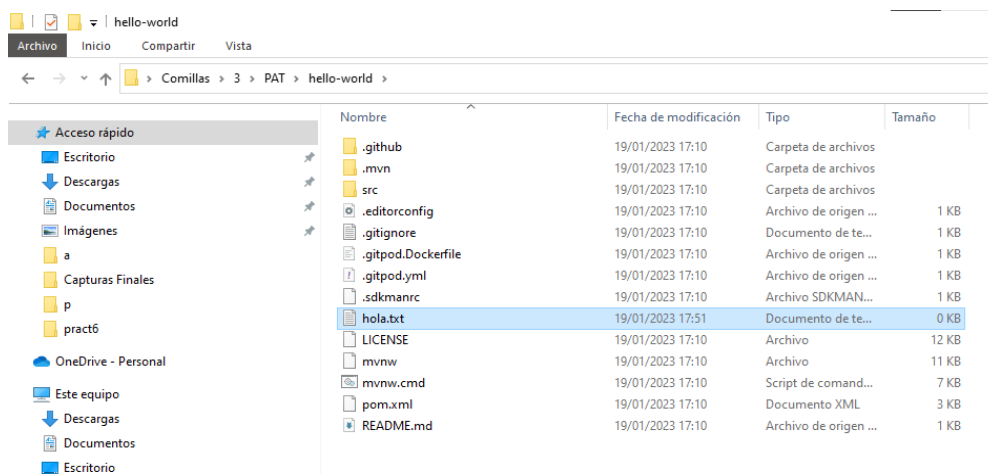
## Practica 1: Instalación de Herramientas

Al repositorio de Github dado en la práctica se le hizo un fork para tenerlo en el repositorio personal para poder editarlo directamente desde nuestro Github.

Una vez ya tenemos el repositorio en nuestro Github personal, clonamos el repositorio en nuestro ordenador utilizando el comando “git clone *url\_github*” para poder tener los ficheros en nuestro ordenador personal.

```
C:\Users\mhino\Desktop\Comillas\3\PAT>git clone https://github.com/gitt-3-pat/hello-world
Cloning into 'hello-world'...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
Receiving objects: 52% (20/38) 59.56 KiB | 451.00 KiB/s, done.
Receiving objects: 100% (38/38), 59.56 KiB | 451.00 KiB/s, done.
```

Una vez tenemos una copia del repositorio en nuestro PC, vamos a subir un fichero de prueba “hola.txt”. Para ello creamos un fichero en la copia del repositorio:



Una vez creado el fichero, utilizamos el comando “git add .” para preparar todos los archivos de la carpeta seleccionada de tu pc para subirlo al Github. Utilizando el comando “git commit -m *mensaje*” crear un nuevo commit, una instantánea de los archivos en un repositorio en un momento dado, en el repositorio local de Git con el mensaje especificado después del parámetro “-m”, proporcionando una descripción breve de los cambios realizados.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19044.2486]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\mhino>cd C:\Users\mhino\Desktop\Comillas\3\PAT\hello-world

















C:\Users\mhino\Desktop\Comillas\3\PAT\hello-world>git add .

C:\Users\mhino\Desktop\Comillas\3\PAT\hello-world>git commit -m "subo archivo hola.txt."
[main facbc21] subo archivo hola.txt.
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hola.txt
```

Por último, con el comando “git push” se envían los cambios realizados en el repositorio local al repositorio remoto de Github

```
C:\Users\mhino\Desktop\Comillas\3\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% (2/2), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/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/GitDeMike/hello-world.git
48fe276..facbc21 main -> main

C:\Users\mhino\Desktop\Comillas\3\PAT\hello-world>_
```

	GitDeMike subo archivo hola.txt.	facbc21 1 minute ago	 4 commits
	.github/workflows	Primera iteracion	last year
	.mvn/wrapper	Primera iteracion	last year
	src	Primera iteracion	last year
	.editorconfig	Primera iteracion	last year
	.gitignore	Primera iteracion	last year
	.gitpod.Dockerfile	Primera iteracion	last year
	.gitpod.yml	Primera iteracion	last year
	.sdkmanrc	Primera iteracion	last year
	LICENSE	Initial commit	last year
	README.md	Primera iteracion	last year
	hola.txt	subo archivo hola.txt.	1 minute ago
	mvnw	Primera iteracion	last year
	mvnw.cmd	Primera iteracion	last year
	pom.xml	Primera iteracion	last year

Para comprobar el estado actual del repositorio local de Git utilizamos el comando “git status”. Este muestra qué archivos están siendo seguidos, cuáles han sido modificados y aún no se han guardado en un commit, y cuáles están siendo seguidos pero no han sido modificados.

```
C:\Users\mhino\Desktop\Comillas\3\PAT\hello-world>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:   hola.txt

C:\Users\mhino\Desktop\Comillas\3\PAT\hello-world>_
```

Con el comando “git checkout” podemos cambiar entre distintas ramas del repositorio o incluso podemos restablecer versiones anteriores del repositorio y de documentos.

Versión de maven:

```
C:\Users\mhino>mvn -version
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: 17.0.5, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17.0.5
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Versión de java:

```
C:\Users\mhino>java -version
java version "17.0.5" 2022-10-18 LTS
Java(TM) SE Runtime Environment (build 17.0.5+9-LTS-191)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.5+9-LTS-191, mixed mode, sharing)
C:\Users\mhino>
```

## Versión Docker

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19044.2486]
(c) Microsoft Corporation. Todos los derechos reservados.

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

## Intellij

The screenshot shows an IDE with a project named 'game' in the file explorer on the left. The code editor on the right displays the implementation of a Tic Tac Toe game using a static class named 'game'. The code includes imports for Scanner and JOptionPane, a static Board class, and methods for initializing the board, getting the next player, and handling game logic like checking for wins or draws.

```
1 package com.connecta.models;
2
3 import java.util.Scanner;
4 import javax.swing.JOptionPane;
5
6 public class game {
7
8     private Board board;
9     private Turn turn;
10    private static final String TITLE = "TIC TAC TOE";
11    private static final String MESSAGE = "Do you want to continue?";
12
13    public static void main(String[] args) {
14        new game().playGame();
15    }
16
17    private game() {
18        this.board = new Board();
19        this.turn = new Turn(this.board);
20        this.reset();
21    }
22
23    private void reset() {
24        this.board.reset();
25        this.turn.reset();
26    }
27
28    private void playGame() {
29        do {
30            this.playGame();
31        } while (this.isFinished());
32    }
33
34    private void playGame() {
35        Console.getInstance().writeln(game.TITLE);
36        this.board.writeGame();
37        do {
38            this.turn.dropDown();
39            this.board.writeGame();
40        } while (!this.board.isFinished());
41        this.turn.writeResult();
42    }
43
44    private boolean isFinished() {
45        return this.board.isFinished() || this.turn.isFinished();
46    }
47}
```