Práctica 1

Programación Aplicaciones Telemáticas

El Comando fork nos permite trabajar en el repositorio destre nuestra máquina local, sin afectar a la versión original

Hacemos fork del repositorio: https://github.com/gitt-3-pat/p1y probamos los siguientes comandos

```
@Abmcolino →/workspaces/p1-fork (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
Receiving objects: 100% (6/6), done.
@Abmcolino →/workspaces/p1-fork (main) $
```

·***Git clone***: Este comando sirve para crear una copia local del repositorio **p1** en el directorio p1-fork de mi propia máquina.

```
@Abmcolino →/workspaces/p1-fork (main) $ cd p1
@Abmcolino →/workspaces/p1-fork/p1 (main) $
```

Con el comando cd, podré acceder a ese repositorio clonado

· ***Git Status***: Nos da información sobre el repositorio en el que estamos trabajando.

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        p1/

nothing added to commit but untracked files present (use "git add" to track)
```

Nos muestra también el estado de la rama. El comando nos dice que estamos trabajando en la rama “Main”. Está sincronizada con la rama main del repositorio original, todavía no hemos ningún cambio. Nos dice que el repositorio p1, antes clonado, no está siendo rastreado. Por tanto, nos sugiere usar a continuación el comando *Git Add*

·***Git Add***: Sirve para añadir los próximos cambios al área de preparación. Por ejemplo, antes no rastreaba **p1** y ahora sí que lo está haciendo. Verificamos su utilidad con el comando *git status*, que nos devuelve una información diferente a la de la última vez. Se ha añadido el archivo **p1** y ahora necesitamos confirmar estos cambios

```
@Abmcolino →/workspaces/p1-fork (main) $ git add p1
warning: adding embedded git repository: p1
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> p1
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached p1
hint:
hint: See "git help submodule" for more information.
@Abmcolino →/workspaces/p1-fork (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
```

·***Git Commit:*** Sirve para confirmar los cambios en el repositorio local. Si queremos que los cambios se vean en el repositorio remoto, nos falta un comando todavia (*git push)*

```
@Abmcolino →/workspaces/p1-fork (main) $ git commit -m "Añadimos repositorio p1"
[main 4274cb4] Añadimos repositorio p1
 1 file changed, 1 insertion(+)
 create mode 160000 p1
```

· ***Git Push***: Sirve para reflejar los cambios del repositorio local en el repositorio remoto. La rama "Main" del repositorio original debería reflejar los cambios. Los comandos anteriores están enfocados en hacer de github una plataforma colaborativa para los desarrolladores.

```
@Abmcolino →/workspaces/p1-fork (main) $ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 289 bytes | 289.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Abmcolino/p1-fork
   07720b5..4274cb4  main -> main
```

· ***Git checkout:***

```
@Abmcolino →/workspaces/p1-fork (main) $ git checkout -b feat/add/body
Switched to a new branch 'feat/add/body'
@Abmcolino →/workspaces/p1-fork (feat/add/body) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@Abmcolino →/workspaces/p1-fork (main) $
```

En este caso, el comando Git checkout nos sirve para cambiar de rama, por ejemplo, desde main hasta feat/add/body, proceso que luego invertimos para volver a la rama original. Aquí además, añadimos una funcionalidad extra con el –b. Es una abreviatura del comando git branch y se utiliza para crear una nueva rama

ENTORNO DESARROLLO JAVA

- **JAVA 17**

```
PS C:\Users\abmco> java -version
java version "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)
PS C:\Users\abmco>
```

- **Apache MAVEN**

```
C:\Users\abmco>mvn -version
Apache Maven 3.9.6 (bc0240f3c744dd6b6ec2920b3cd08dcc295161ae)
Maven home: C:\Users\abmco\Downloads\apache-maven-3.9.6-bin\apache-maven-3.9.6
Java version: 17.0.10, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users\abmco>
```

- **IntelliJ Community Edition 2023**

Ventana.java

```
1  import javax.swing.*;
2
   no usages
3  public class Ventana extends JFrame {
       no usages
4      public Ventana(){
5          new Ventana();
6
7      }
8  }
```

- **Visual Studio**