# PRÁCTICA 2: CONTROL DE VERSIONES CON GIT

## 1. Introducción

Un sistema de control de versiones permite mantener una historia de los artefactos involucrados en el desarrollo de un sistema software, además mantiene el orden, la concurrencia y la sincronización entre los integrantes del equipo de trabajo y la distribución entre equipos de trabajo deslocalizados geográficamente.

¿Cuándo creéis que es necesario un sistema de control de versiones?

a. Cuando el tamaño del sistema a construir crece considerablemente.
b. Cuando el equipo de trabajo lo conforma más de una persona.
c. Cuando disponemos de equipos de trabajo deslocalizados.
d. Cuando existen problemas de concurrencia/sincronización entre versiones.
e. Cuando trabajamos solos pero somos despistados o no tenemos claro el alcance.
f. Todas las anteriores, por los siglos de los siglos, ¡amén!

Esto es necesario aprenderlo desde ¡ya!. ¿Cuándo es necesario un sistema de control de versiones? ¡Siempre! Sin ningún tipo de duda. Pero claro, es necesario haberse tropezado con el problema, para darse cuenta de la importancia y la necesidad imperativa de un sistema de estas características.

Por lo tanto, como parece que esto es algo importante, vamos a aprender a manejar uno de los más utilizados.

*Git* (https://git-scm.com/) es una herramienta de control de versiones de gran popularidad y difusión, cuyo objetivo es el de aliviar en la medida de lo posible los problemas hasta ahora mencionados. El objetivo de la presente práctica es proporcionar al alumno una introducción guiada al manejo de Git.

## 2. Objetivos

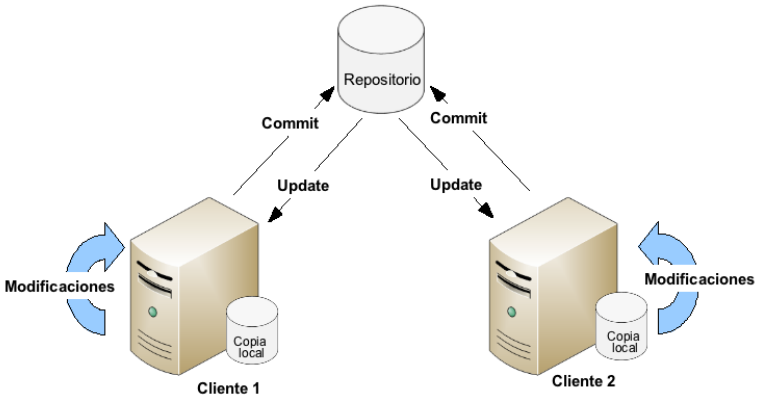
El objetivo de la presente práctica es familiarizar al alumno con los procesos de control de versiones usando *Git*. Concretamente, el alumno aprenderá a:

(1) Crear un repositorio local.
(2) Crear un proyecto a partir de un repositorio remoto.
(3) Sincronizar cambios con el repositorio.
(4) Resolver conflictos entre versiones.

Para alcanzar dichos objetivos, el alumno deberá realizar la lista de actividades que se describen en la sección 4.

## 3. Conceptos previos

En un equipo de trabajo distribuido generalmente se dispone de un servidor donde se almacena el código fuente y de una serie de clientes que desarrollan código fuente y lo suben a dicho servidor.




En este caso, el rol de servidor lo va a jugar un repositorio remoto de Git, que será proporcionado por Github (https://github.com/).

Para los clientes, existen diversos clientes que simplifican la interacción con Git, como por ejemplo, SourceTree (https://www.sourcetreeapp.com/) o el plugin que Eclipse proporciona para el manejo de Git. Más adelante podremos utilizar dicho plugin, pero en esta práctica, utilizaremos Git a nivel de línea de comandos.

## 4. Desarrollo de la práctica

En todos los pasos intermedios, debéis ir invocando `git status` para comprobar los mensajes que os muestra Git.

Formaremos parejas. Un miembro de la pareja será el Alumno A y otro el Alumno B.

Creación de un repositorio local

1. Alumno A crea un repositorio Git local en su máquina y añade un fichero de texto con las siguientes líneas al directorio de trabajo

```
Esta es la primera línea del Alumno A
Esta es la segunda línea del Alumno A
```

2. Alumno A pone el fichero bajo seguimiento y lo actualiza en su repositorio local.

Creación y sincronización con repositorios remotos

3. Alumno A crea un repositorio en Github de nombre Practica 2 y sube el contenido de su repositorio local a dicho repositorio remoto (utilizaremos la URL con protocolo https). Nota: Crear el repositorio vacío, sin fichero README.
4. Alumno A invita a Alumno B a su repositorio de Github.
5. Alumno B clona el repositorio remoto en su máquina.
6. Alumno B añade el siguiente texto al fichero de texto:

```
Esta es la primera línea del Alumno B
Esta es la segunda línea del Alumno B
```

7. Alumno B actualiza tanto su repositorio local como el remoto.
8. Alumno A sincroniza su repositorio con el repositorio remoto.

Actualizaciones fuera de sincronía

9. Alumno A y Alumno B modifican el fichero en sus RESPECTIVAS primeras líneas, por ejemplo:

```
Esta es la primera línea MODIFICADA del Alumno A
```

10. Alumno A sube su modificación al repositorio remoto.
11. Alumno B sube su modificación al repositorio remoto. Git debería informar que la versión del repositorio se ha modificado después de su última actualización (non-fast-forward change).
12. Alumno B fusiona los cambios del repositorio remoto con su repositorio local y a continuación, vuelve a subir su modificación al repositorio remoto.

Actualizaciones con conflicto

13. Alumno A y Alumno B hacen cambios sobre la misma línea (la segunda línea, por ejemplo) y actualizan sus repositorios locales.
14. Alumno B sube la nueva versión al repositorio remoto.
15. Alumno A sube la nueva versión repositorio remoto. Recibirá un mensaje de fuera de sincronía.
16. Alumno A intenta fusionar su repositorio local con el remoto. Git debería informar de la existencia de conflicto.
17. Alumno A resuelve el conflicto. Para ello abrir el fichero con conflicto y observar cómo muestra Git los conflictos encontrados.
18. Alumno A vuelve a subir los cambios al repositorio remoto.
19. Alumno B sincroniza su repositorio local con el repositorio remoto.

## 4. Criterios de Evaluación y Aclaraciones

El alumno deberá entregar un pequeño documento de texto con los comandos Git ejecutados en cada paso de la práctica. Además, las preguntas de las pruebas evaluables escritas pueden estar relacionadas con cuestiones generales relacionadas con los contenidos de esta práctica.