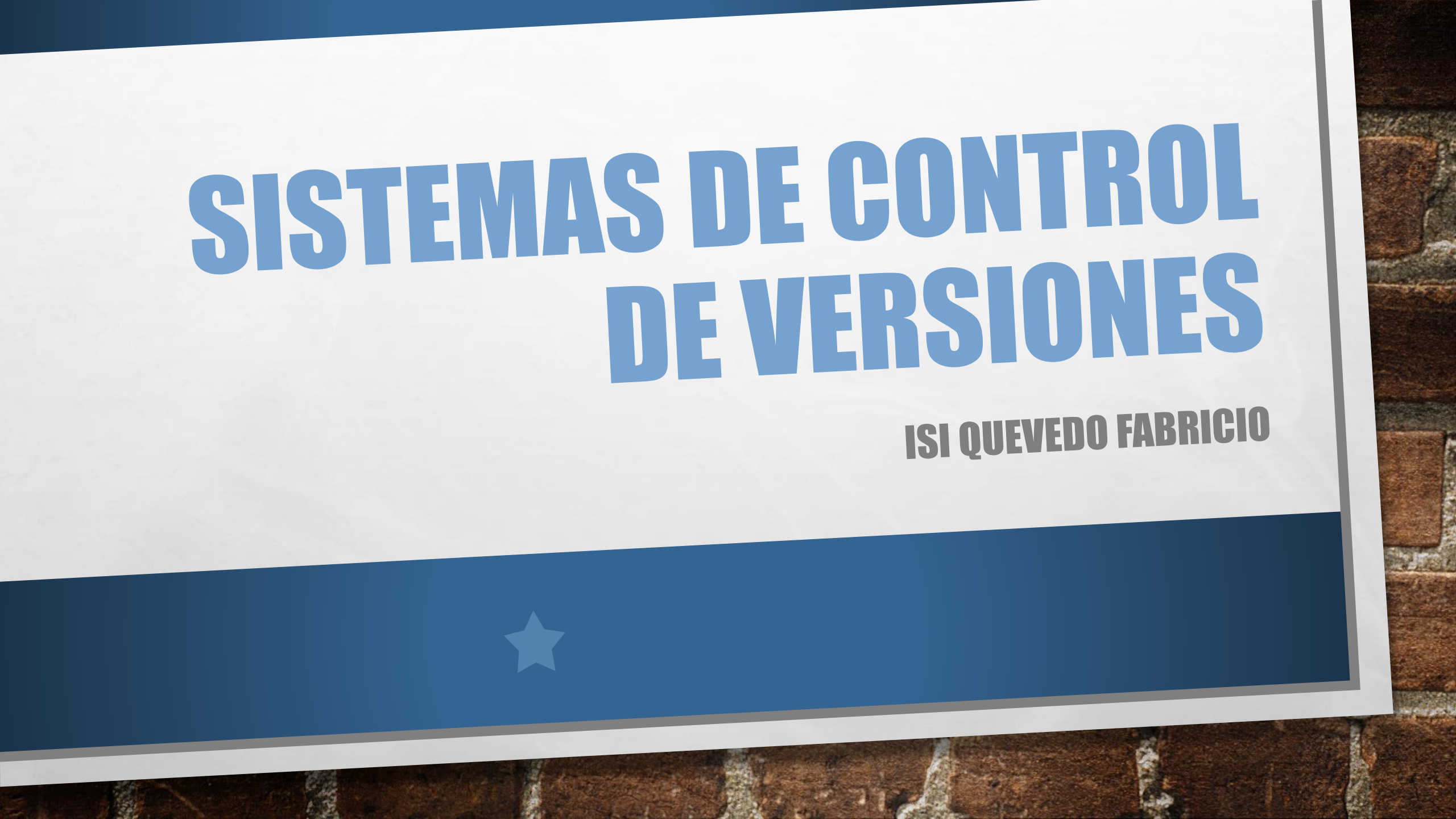
SISTEMAS DE CONTROL DE VERSIONES

ISI QUEVEDO FABRICIO



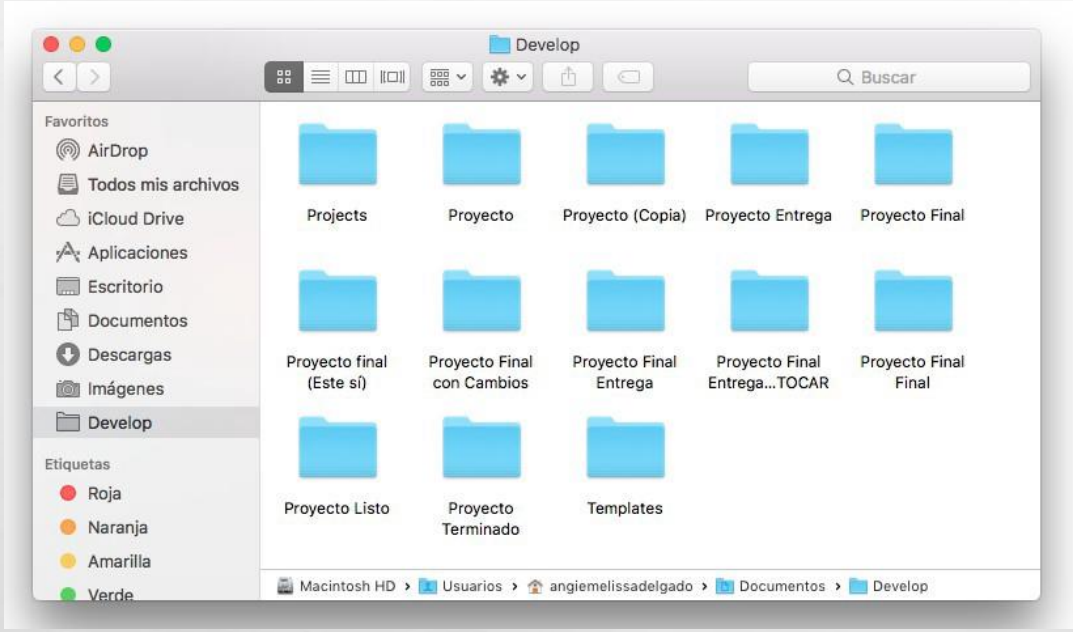
QUE ES UN SISTEMA DE CONTROL DE VERSIONES?

- UN SISTEMA DE VERSIONADO DE CÓDIGO (SVC) EN ABSTRACTO ES LO QUE NOS PERMITE **COMPARTIR EL CÓDIGO** FUENTE DE NUESTROS DESARROLLOS Y A LA VEZ **MANTENER UN REGISTRO DE LOS CAMBIOS** POR LOS QUE VA PASANDO.
- HABITUALMENTE PARA GESTIONAR LAS **DISTINTAS VERSIONES** POR LAS QUE PASA EL CÓDIGO FUENTE DE LAS APLICACIONES, LO QUE NOS PERMITE SABER **QUIÉN REALIZA QUÉ CAMBIOS** Y PODER VOLVER A ELLOS EN UN DETERMINADO MOMENTO. LOS MÁS UTILIZADOS EN ESTE CAMPO A LO LARGO DEL TIEMPO (AUNQUE EXISTEN BASTANTE MÁS) SON **CVS, SUBVERSION Y GIT**

PORQUÉ USAR UN CONTROL DE VERSIONES NOS HARÁ FELICES?

- PROPORCIONA COPIAS DE SEGURIDAD AUTOMÁTICAS DE LOS FICHEROS.
- PERMITE VOLVER A UN ESTADO ANTERIOR DE NUESTROS FICHEROS.
- AYUDA A TRABAJAR DE UNA FORMA MÁS ORGANIZADA.
- PERMITE TRABAJAR DE FORMA LOCAL, SIN CONEXIÓN CON SERVIDOR. (EN DISTRIBUÍDOS).
- PERMITE QUE VARIAS PERSONAS TRABAJEN EN LOS MISMOS FICHEROS.
- PERMITEN TRABAJAR EN VARIAS FUNCIONALIDADES EN PARALELO SEPARADO (RAMAS).

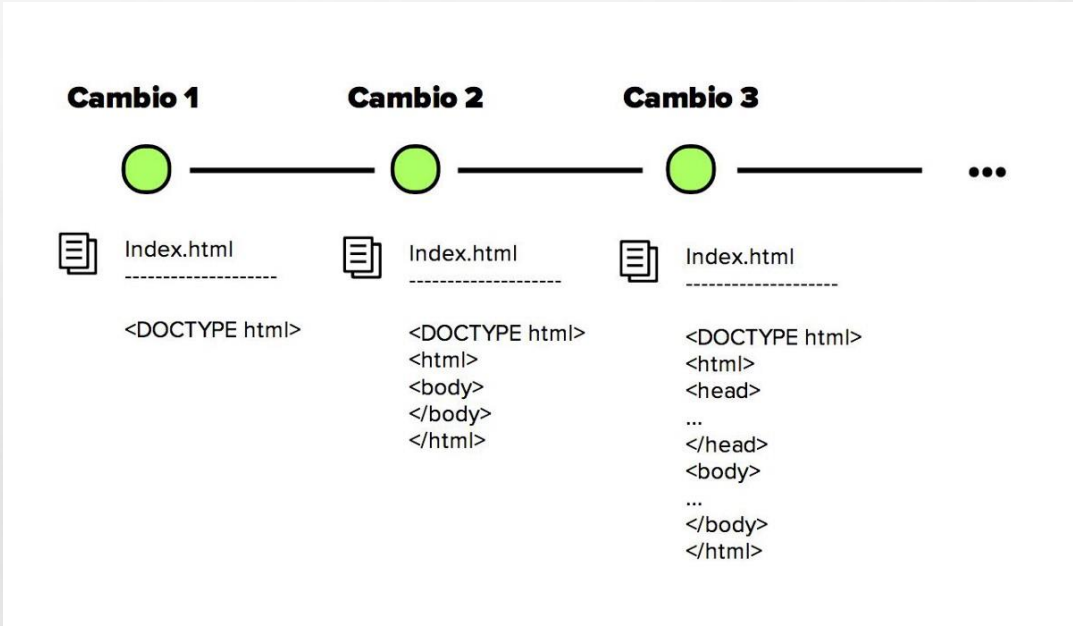
ANTES Y EL DESPUÉS DE CONOCER LOS SISTEMAS DE CONTROL DE VERSIONES



- EN ALGÚN MOMENTO DE NUESTRAS VIDAS NUESTRA CARPETA DE DOCUMENTOS LUCIERA COMO LA DE LA IMAGEN Y NOS HAYA TOCADO RECURRIR A TENER MUCHAS COPIAS DE NUESTROS PROYECTOS.
- TIEMPO DESPUÉS POR ALGÚN ACCIDENTE DEL DESTINO CONOCEMOS LOS SISTEMAS DE CONTROL DE VERSIONES Y NO PODEMOS NEGARLO, NUESTRAS VIDAS CAMBIAN Y ENTRAMOS A UNA ERA DONDE TODO ES MUCHO MÁS BONITO: EL DESPUÉS.

SISTEMA DE CONTROL DE VERSIONES

- EL CONTROL DE VERSIONES ES UN SISTEMA QUE REGISTRA LOS CAMBIOS REALIZADOS SOBRE UN ARCHIVO O CONJUNTO DE ARCHIVOS A LO LARGO DEL TIEMPO DE TAL MANERA QUE SEA POSIBLE RECUPERAR VERSIONES ESPECIFICAS MÁS ADELANTE.



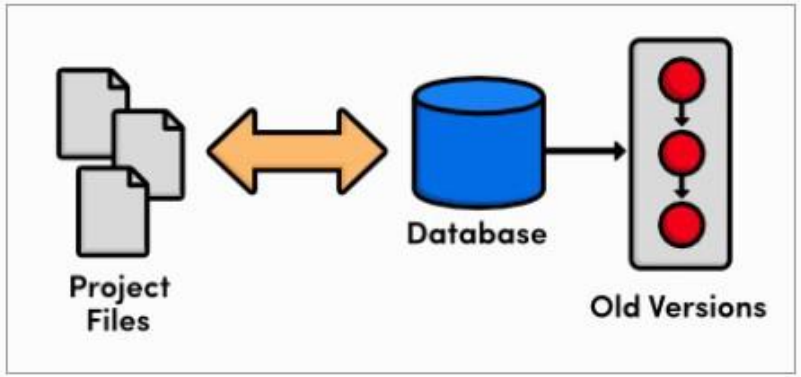
PODEMOS CLASIFICARLOS

3 TIPOS:

- **LOCALES,**
- **CENTRALIZADOS**
- **DISTRIBUIDOS.**

SISTEMAS DE CONTROL DE VERSIONES LOCALES

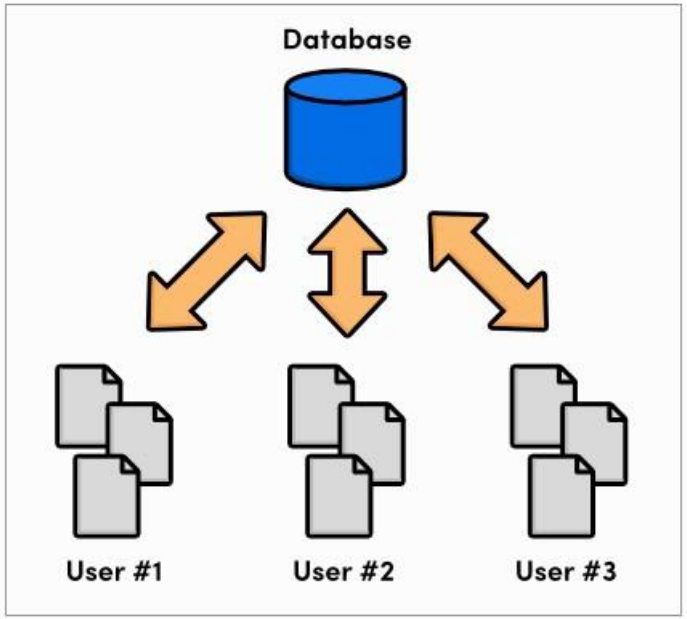
- EN VEZ DE MANTENER LAS VERSIONES COMO ARCHIVOS INDEPENDIENTES, LOS ALMACENABAN EN UNA BASE DE DATOS.
- EN CUALQUIER MOMENTO SOLO SE TENIA UNA COPIA DEL PROYECTO, ELIMINANDO LA POSIBILIDAD DE CONFUNDIR O ELIMINAR VERSIONES.
- EN ESTE PUNTO EL CONTROL DE VERSIONES SE LLEVABA A CABO EN EL COMPUTADOR DE CADA UNO DE LOS DESARROLLADORES Y NO EXISTÍA UNA MANERA EFICIENTE DE COMPARTIR EL CÓDIGO ENTRE ELLOS



Local version control

SISTEMAS DE CONTROL DE VERSIONES CENTRALIZADOS

- PARA FACILITAR LA COLABORACIÓN DE MÚLTIPLES DESARROLLADORES EN UN SOLO PROYECTO LOS SISTEMAS DE CONTROL DE VERSIONES EVOLUCIONARON: EN VEZ DE ALMACENAR LOS CAMBIOS Y VERSIONES EN EL DISCO DURO DE LOS DESARROLLADORES, ESTOS SE ALMACENABAN EN UN SERVIDOR.

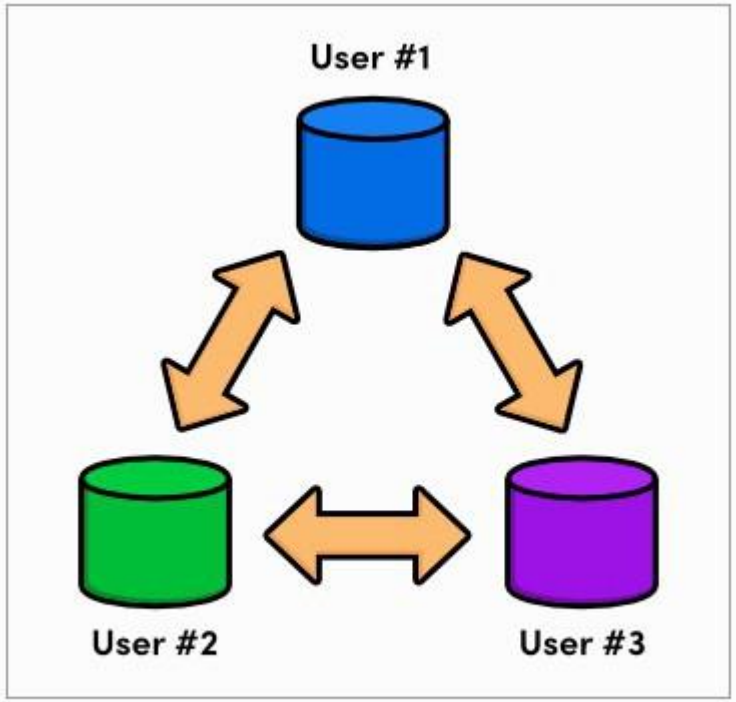


Centralized version control

- LOS SISTEMAS CENTRALIZADOS TRAJERON CONSIGO NUEVOS RETOS: ¿CÓMO TRABAJABAN MÚLTIPLES USUARIOS EN UN MISMO ARCHIVO AL MISMO TIEMPO?
- LOS SISTEMAS DE CONTROL DE VERSIONES CENTRALIZADOS ABORDARON ESTE PROBLEMA IMPIDIENDO QUE LOS USUARIOS INVALIDARAN EL TRABAJO DE LOS DEMÁS. SI DOS PERSONAS EDITABAN EL MISMO ARCHIVO Y SE PRESENTABA UN CONFLICTO ALGUIEN DEBÍA SOLUCIONAR ESTE PROBLEMA DE MANERA MANUAL Y EL DESARROLLO NO PODÍA CONTINUAR HASTA QUE TODOS LOS CONFLICTOS FUERAN RESUELTOS Y PUESTOS A DISPOSICIÓN DEL RESTO DEL EQUIPO.
- ESTA SOLUCIÓN FUNCIONÓ EN PROYECTOS QUE TENÍAN RELATIVAMENTE POCAS ACTUALIZACIONES Y POR ENDE POCOS CONFLICTOS PERO RESULTO MUY ENGORROSO PARA PROYECTOS CON DOCENAS DE CONTRIBUYENTES ACTIVOS QUE REALIZABAN ACTUALIZACIONES A DIARIO.

SISTEMAS DE CONTROL DE VERSIONES DISTRIBUIDOS

- ESTE SCV OPTÓ POR DARLE A CADA DESARROLLADOR UNA COPIA LOCAL DE TODO EL PROYECTO, DE ESTA MANERA SE CONSTRUYO UNA RED DISTRIBUIDA DE REPOSITORIOS, EN LA QUE CADA DESARROLLADOR PODÍA TRABAJAR DE MANERA AISLADA PERO TENIENDO UN MECANISMO DE RESOLUCIÓN DE CONFLICTOS MUCHO MÁS ELEGANTE QUE UN SU VERSIÓN ANTERIOR.



Distributed version control

SCV DISTRIBUÍDOS

- AL NO EXISTIR UN REPOSITORIO CENTRAL, CADA DESARROLLADOR PUEDE TRABAJAR A SU PROPIO RITMO, ALMACENAR LOS CAMBIOS A NIVEL LOCAL Y MEZCLAR LOS CONFLICTOS QUE SE PRESENTEN SOLO CUANDO SE REQUIERA. CÓMO CADA USUARIO TIENE UNA COPIA COMPLETA DEL PROYECTO EL RIESGO POR UNA CAÍDA DEL SERVIDOR, UN REPOSITORIO DAÑADO O CUALQUIER OTRO TIPO DE PERDIDA DE DATOS ES MUCHO MENOR QUE EN CUALQUIERA DE SUS PREDECESORES.

CVS

- CVS (CONCURRENT VERSIONS SYSTEM) ES UNO DE LOS PRIMEROS SISTEMAS DE CONTROL DE VERSIONES. TIENE UNA ARQUITECTURA CLIENTE-SERVIDOR, EN LA QUE EL CÓDIGO ESTÁ ALMACENADO EN UN SERVIDOR CENTRAL Y CON EL SOFTWARE CLIENTE PODEMOS HACER UNA COPIA DEL CÓDIGO LOCAL PARA HACER CAMBIOS Y POSTERIORMENTE VOLVER A SUBIRLA AL SERVIDOR.
- PERMITE EL TRABAJO CONCURRENTE ENTRE DISTINTOS PROGRAMADORES, PERO EL SERVIDOR SOLO ACEPTA ACTUALIZACIONES DE LOS FICHEROS QUE ESTÉN EN LA ÚLTIMA VERSIÓN, DE FORMA QUE LOS PROPIOS USUARIOS DEBEN ACTUALIZAR SUS COPIAS LOCALES ANTES DE SUBIRLAS AL SERVIDOR. SOPORTA ADEMÁS EL TRABAJO EN DISTINTAS RAMAS.
- SE HIZO BASTANTE POPULAR ENTRE LA COMUNIDAD DE SOFTWARE LIBRE POR SER LANZADO BAJO LA LICENCIA GNU.

SVN

EL PROYECTO DE SUBVERSION SURGIÓ EN EL AÑO 2000 CON EL OBJETIVO CREAR UN SISTEMA DE CONTROL DE VERSIONES CON LA MISMA FILOSOFÍA QUE CVS, PERO ARREGLANDO PROBLEMAS Y CUBRIENDO CARENCIAS DEL MISMO:

- COMMITS ATÓMICOS, EN CVS UN COMMIT INTERRUMPIDO PUEDE DEJAR DATOS INCONSISTENTES.
- LA CREACIÓN DE RAMAS ES MÁS EFICIENTE, CON COMPLEJIDAD CONSTANTE A DIFERENCIA DE CVS QUE ES LINEAL (AUMENTA CON EL NÚMERO DE RAMAS).
- MANEJO DE ARCHIVOS BINARIOS COMO TAL, CVS LOS TRATA COMO ARCHIVOS DE TEXTO.
- ENVÍA LOS INCREMENTOS DE LOS FICHEROS EN LA COMUNICACIÓN CLIENTESERVIDOR, EN LUGAR DE LOS FICHEROS COMPLETOS COMO CVS.

SVN

- LA ESTRUCTURA MÁS HABITUAL EN UN REPOSITORIO SVN:
  - **TRUNK** CON LA VERSIÓN DEL CÓDIGO PRINCIPAL.
  - **TAGS** PARA ALMACENAR LAS DISTINTAS VERSIONES DE UNA APLICACIÓN QUE NO VOLVERÁN A MODIFICARSE.
  - **BRANCHES** PARA GESTIONAR LAS DISTINTAS VERSIONES DE CÓDIGO QUE SE DESARROLLAN PARALELAS AL **TRUNK**

  AL SER TAMBIÉN SOFTWARE LIBRE, HA SIDO ADOPTADO POR MULTITUD DE PROYECTOS, INCLUSO EN GRANDES CORPORACIONES, CONVIRTIÉNDOSE EN UN REFERENTE.

GIT

- EN 2005, LINUS TORVALDS INICIÓ EL DESARROLLO DE GIT COMO ALTERNATIVA A BITKEEPER (QUE HABÍA PASADO A SER SW PROPIETARIO), EL SISTEMA DE CONTROL DE VERSIONES QUE SE USÓ HASTA EL MOMENTO PARA DESARROLLAR EL KERNEL DE LINUX.
  POR ESTO, GIT NACE CON LAS PRINCIPALES NECESIDADES DE SER RÁPIDO, EFICIENTE Y DISTRIBUIDO:

● RÁPIDO: COMBINANDO EL TRABAJO SOBRE EL REPOSITORIO LOCAL Y LA POSTERIOR DISTRIBUCIÓN REMOTA.

● EFICIENTE: PENSADO PARA MANEJAR GRANDES PROYECTOS CON MUCHOS FICHEROS (LINUX) Y NO SE VUELVE LENTO A MEDIDA QUE LA HISTORIA DEL PROYECTO CRECE.

● DISTRIBUIDO: FUNDAMENTAL PARA EL TRABAJO CONCURRENTE Y EN REMOTO DE MUCHOS USUARIOS. CADA USUARIO TIENE UNA COPIA LOCAL EN LA QUE TRABAJAR, QUE POSTERIORMENTE SINCRONIZA CON EL RESTO DE USUARIOS. ES TAMBIÉN ESPECIALMENTE EFECTIVO A LA HORA DE MEZCLAR LOS CAMBIOS HECHOS POR DISTINTOS USUARIOS.

GIT CARACTERISTICAS

- SU **SENCILLEZ** ES LO QUE LO DIFERENCIA DE OTROS SISTEMAS DE CONTROL DE VERSIONES Y LO QUE LE HACE VERDADERAMENTE POTENTE.
- LA MAYORÍA DE SISTEMAS DEL MOMENTO ALMACENABAN LOS CAMBIOS (DELTAS) EN LOS FICHEROS ENTRE DISTINTAS VERSIONES Y REQUERÍAN DE COMPLEJOS ALGORITMOS DE APLICACIÓN DE ESOS DELTAS PARA RECUPERAR UN DETERMINADO ESTADO DEL REPOSITORIO. GIT EN CAMBIO ES UNA SENCILLA BASE DE DATOS CLAVE-VALOR, EN LA QUE CADA VERSIÓN APUNTA A UN ÁRBOL DE NODOS QUE REPRESENTA LA ESTRUCTURA DE DIRECTORIOS Y CONTENIDOS COMPRIMIDOS DE LOS FICHEROS. ESTO LE PERMITE RECUPERAR ESTADOS ÚNICAMENTE APUNTANDO A UN DETERMINADO ÁRBOL Y FACILITA LAS TAREAS DE DISTRIBUCIÓN Y MEZCLADO DE CONTENIDOS.

GIT CARACTERISTICAS

- **DISTRIBUIDO**: CADA USUARIO DISPONE DE UN REPOSITORIO COMPLETO DE FORMA LOCAL. ESTO PERMITE TRABAJAR SIN NECESIDAD DE CONEXIÓN A UN SERVIDOR PARA REALIZAR LAS DISTINTAS ACCIONES, PUDIENDO MÁS TARDE SINCRONIZAR NUESTROS DATOS CON EL SERVIDOR.
- OTRA DE LAS PARTICULARIDADES DE GIT ES EL **ÁREA DE STAGE**. LA GRAN MAYORÍA DE SISTEMAS ALMACENAN LA INFORMACIÓN EN DOS SITIOS, LA COPIA LO AL DE FICHEROS Y DIRECTORIOS DEL USUARIO, Y EL ALMACENAMIENTO DE VERSIONES.
- GIT PROPORCIONA UNA TERCERA OPCIÓN CON EL ÁREA DE STAGE. CONSISTE EN UN ÁREA INTERMEDIA ENTRE LAS OTRAS DOS, DONDE IR COLOCANDO LOS CAMBIOS DE LA COPIA LOCAL CON LAS QUE QUEREMOS HACER UN NUEVO COMMIT. DE ESTA FORMA SE CONSIGUE MUCHA MÁS VERSATILIDAD Y CONTROL A LA HORA DE IR GUARDANDO VERSIONES DEL CÓDIGO.