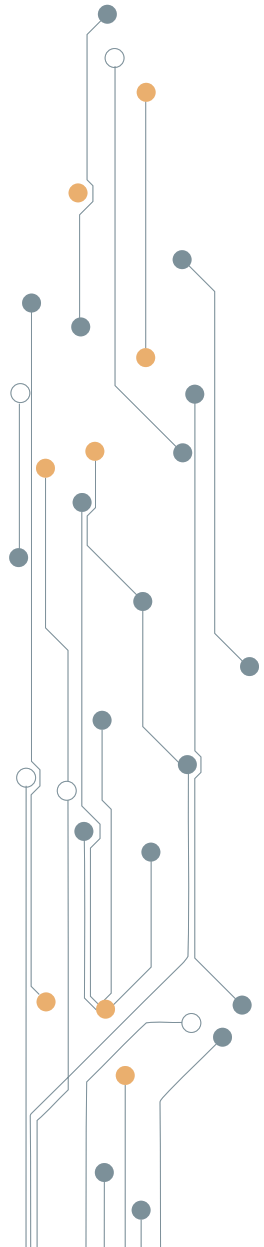
Completado
18/09/2019, 0:37:28

Trabajar en equipo

Telefónica
EDUCACIÓN DIGITAL

Índice



1. Trabajar en equipo

Existen herramientas para trabajar en equipo, que además nos permiten versionar nuestras aplicaciones. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente especifico). Ejemplos de este tipo de herramientas son entre otros Subversion y GIT

Para colaborar en un proyecto usando un sistema de control de versiones lo primero que hay que hacer es crearse una copia local obteniendo información del repositorio. A continuación el usuario puede modificar la copia. Existen dos esquemas básicos de funcionamiento para que los usuarios puedan ir aportando sus modificaciones:

- **De forma exclusiva:** En este esquema para poder realizar un cambio es necesario comunicar al repositorio el elemento que se desea modificar y el sistema se encargara de impedir que otro usuario pueda modificar dicho elemento. Una vez hecha la modificación, esta se comparte con el resto de colaboradores. Si se ha terminado de modificar un elemento entonces se libera ese elemento para que otros lo puedan modificar. Subversion, por ejemplo, permite este esquema de funcionamiento.
- **De forma colaborativa:** En este esquema cada usuario modifica la copia local y cuando el usuario decide compartir los cambios el sistema automáticamente intenta combinar las diversas modificaciones. El principal problema es la posible aparición de conflictos que deban ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas. Subversion o GIT implementan este esquema.

1.1 | Arquitecturas de almacenamiento

Podemos clasificar los sistemas de control de versiones atendiendo a la arquitectura utilizada para el almacenamiento del código:

- **Centralizados:** existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos). Se facilitan las tareas administrativas a cambio de reducir flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable. Subversion implementa esta arquitectura de almacenamiento.

Las principales ventajas de este sistema son:

- **En los sistemas centralizados** las versiones vienen identificadas por un número de versión. Sin embargo en los sistemas de control de versiones distribuidos no hay números de versión, ya que cada repositorio tendría sus propios números de revisión dependiendo de los cambios.

En lugar de eso cada versión tiene un identificador al que se le puede asociar una etiqueta (tag).

- **Distribuidos:** Cada usuario tiene su propio repositorio. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos. Es frecuente el uso de un repositorio, que esta normalmente disponible, que sirve de punto de sincronización de los distintos repositorios locales. GIT implementa este sistema.

Ventajas de sistemas distribuidos:

- Necesita menos veces **estar conectado a la red para hacer operaciones.** Esto produce una mayor autonomía y una mayor rapidez.
- **Aunque se caiga el repositorio** remoto la gente puede seguir trabajando.
- **Al hacer los distintos repositorios una réplica local** de la información de los repositorios remotos a los que se conectan, la información está muy replicada y por tanto el sistema tiene menos problemas en recuperarse si por ejemplo se quema la máquina que tiene el repositorio remoto. Por tanto hay menos necesidad de copias de seguridad (backups). Sin embargo los backups siguen siendo necesarios para resolver situaciones en las que cierta información todavía no haya sido replicada.
- **Permite mantener repositorios centrales más limpios** en el sentido de que un usuario puede decidir que ciertos cambios realizados por él en el repositorio local, no son relevantes para el resto de usuarios y por tanto no permite que esa información sea accesible de forma pública. Por ejemplo es muy útil se pueden tener versiones inestables o en proceso de codificación o también tags propios del usuario.

- **El servidor remoto requiere menos recursos** que los que necesitaría un servidor centralizado ya que gran parte del trabajo lo realizan los repositorios locales.
- **Al ser los sistemas distribuidos más recientes** que los sistemas centralizados y, al tener más flexibilidad por tener un repositorio local y otro/s remotos. Estos sistemas han sido diseñados para hacer fácil el uso de ramas (creación, evolución y fusión) y poder aprovechar al máximo su potencial. Por ejemplo se pueden crear ramas en el repositorio remoto para corregir errores o crear funcionalidades nuevas.

Pero también se pueden crear ramas en los repositorios locales para que los usuarios puedan hacer pruebas y dependiendo de los resultados fusionarlos con el desarrollo principal o no. Las ramas dan una gran flexibilidad en la forma de trabajo

Algunos de los programas para controles de versiones más extendidos son:

- **CVS (Concurrent Versions System):** sistema de libre distribución bajo licencia GNU, que funciona bajo el esquema de cliente-servidor. Al igual que el resto de sistemas de control de versiones, su finalidad es proveer a un equipo de desarrollo de una herramienta que permita gestionar los distintos cambios realizados por el equipo de desarrollo y generar un único código final en la medida de lo posible.
- **Subversion:** sistema de libre distribución bajo licencia Apache y desarrollada por Apache Software Foundation. Está basado en CVS, pero lo mejora en muchos aspectos como la gestión de ficheros, envíos atómicos de los cambios (todos los cambios son tratados como un único cambio), etc.
- **Clear Case:** solución propietaria desarrollada por IBM. Se trata de una herramienta principalmente enfocada a trabajar dentro del ecosistema de aplicaciones de desarrollo de IBM.
- **VSS (Visual SourceSafe):** sistema de control de versiones de Microsoft (desarrollado inicialmente por la empresa One Tree Software). Su sistema de transferencia de archivos se basa en SMB (el sistema de compartición de recursos de Microsoft). Se integra con el entorno de desarrollo Visual Studio.
- **StarTeam:** sistema de control de versiones propietario de la empresa Borland (previamente desarrollado por Starbase Corporation y posteriormente adquirido por Borland). Es una aplicación cliente / servidor cuya parte servidora se implementa mediante una base de datos relacional (Oracle o SQL Server) y un cliente pesado desarrollado en Java.

También existen otros clientes como un cliente de línea de comandos o el cliente implementado en el entorno de desarrollo Jbuilder.

- **GIT, Bazaar y Mercurial:** a diferencia de los anteriores, no siguen el esquema cliente-servidor, si no que son sistemas de control de versiones distribuidos.

Telefónica

EDUCACIÓN DIGITAL