PLATAFORMAS DE CONTROL DE VERSIONES

Actividad UF1 – 01

Documentación

Un trabajo para abordar puntos clave de las plataformas de control de versiones.

DAW1 – Marco Valiente Rodríguez

Entornos de Desarrollo

Índice

Tabla de contenido

1. Introducción

En este trabajo, vamos a ver el desarrollo elaborado sobre lo que es denominado como sistemas de control de versiones, de los cuales, se va a hacer una ronda de investigaciones para poder documentar sobre estos puntos cuatro aspectos principales sobre estos.

- Propiedades y funcionalidades de las plataformas de control de versión.
- Plataformas más importantes: características y diferencias.
- Principales aplicaciones de clientes de control de versiones
- Control de versiones integrados en los IDES (plugins)

Antes de esto, vamos a tener que hacer una breve introducción a qué es una plataforma de control de versiones, ya que no nos sirve de mucho hablar de un concepto del cual no se desarrolla ni es explicado de antemano, la cual formará parte del desarrollo y una extensión al apartado de sus propiedades y funcionalidades.

Luego, veremos múltiples ejemplos de varias de estas plataformas, yendo en detalle en sus características y diferencias, como hemos elaborado en el punto anterior, pero también evaluando un análisis algo más detallado en los puntos de cada una de estas, y cómo se están contextualizando en tiempos modernos, como por ejemplo con el auge de las Inteligencias Artificiales.

Lo siguiente a ver son los usos que un usuario o compañía tiene para este tipo de herramientas, incluido con las alternativas a éstas, y cómo es que estas aplicaciones pueden suponer una mejora al usuario en cuestión, si las hay. Con esto, también podremos entrar en detalle de en qué escenarios es más conveniente usarlo.

Por último, veremos los IDES, qué significan, cómo son, y que sirven, y cómo suelen ser a niveles prácticos tanto en como de bien o mal funcionan, así como cuál suele ser el posible nivel de estrés que le puede suponer a una aplicación tener esto.

Por último, un final nivel de conclusiones antes de todas las citas hechas en este respecto, en el cual recopilaremos las opiniones generales después de todo lo que hemos visto.

2. Los Controles de Versiones

A. ¿Qué son los Controles de Versiones?

Empezamos abordando la pregunta más básica que se puede hacer una vez que se puede llegar a hacer una vez que lea la portada, un elemento que merece, al menos, una pequeña introducción.

Los códigos fuente, un elemento para los programadores de mayor importancia, es la culminación del trabajo para el desarrollo de un programa, un elemento que es creado y mejorado a medida que más trabajo es puesto en este. Y a medida que más trabajo es dirigido a esta clave esencial de funciones o conocimiento, ¿Qué es lo que pasa cuando este código se pierde por un cambio accidental? ¿Cómo podrían dos personas trabajar de forma eficiente sobre este mismo código pudiéndose pasar los avances que una persona haga para convivir en colaboración?

En el 1972, debido a estas cuestiones, se empezaron a elaborar los SCCS (Sistemas de control de código fuente), que sirvieron como repositorios para programadores a medida que un código evolucionaba. De estas, fueron evolucionando hasta los sistemas de control de versiones.

Su propósito actual es, en términos generales, el control de las versiones después que los cambios que los archivos están sujetos, y la protección del deterioro o de accidentes que puedan ser causados por accidentes humanos, o incluso por inteligencia artificial en los casos contemporáneos.

La necesidad de trabajo en equipo, también, sirve como un elemento que elementa la estructuración de estas, generalmente en una forma de árbol de archivos, con sus jerarquías, permitiendo cambiar código no relacionado.

B. Propiedades de los Sistemas de Control de Versiones

En los puntos descritos por arriba, tenemos que los sistemas de controles proporcionan diferentes tipos de ayudas gracias a sus propiedades. Los elementos principales que estas poseen son los siguientes, a ser descritos a continuación:

1. Completo Historial de Versiones.
2. Creación de ramas y fusiones.
3. Trazabilidad.
4. Permisividad de Colaboración.
5. Control de Versiones

La primera forma en la que estos sistemas son útiles para este tipo de problema, son la capacidad de realizar en estos árboles de información un seguimiento de todos los cambios individuales que cada colaborador pueda realizar por sí mismo, así como evitar que este trabajo lleve a conflicto entre dos versiones. Como el código es esquematizado en esta estructura arbórea, permite la edición de cambios específicos en los archivos sin la necesidad que el programa entero sea reconstituido o actualizado.

Además, al tener en detalle las versiones previas, si una sufre un fallo, se puede usar la antigua.

Los cambios que se realicen en un solo archivo de este árbol pueden resultar incompatibles a través archivos, con, por ejemplo, los cambios que otro contribuidor esté realizando en otra parte del software. La disposición de ambas versiones en sus estados más actualizados al mismo tiempo que las versiones previas suponen una gran ventaja, ya que permite detectar y solucionar los problemas de compatibilidad sin necesidad de tener que desechar o tener de manera no funcional una de estas versiones, lo que facilita que se puedan solucionar de manera coordinada y que coexistan independientemente hasta que esté solucionado y se necesite publicar una nueva versión global del programa, que las establezca de nuevo como base una vez que estén completadas.

Los mejores de estos softwares, son, por tanto, diseñados con la intención de soportar flujos de trabajo sin obstrucciones o imposiciones, y generalmente con compatibilidad de trabajo entre diferentes sistemas operativos para permitir la cooperación, aunque no siempre es el caso.

Por último, a la hora de almacenar estos archivos, estas pueden tener varias herramientas a su disposición, como puede ser el almacenamiento de instantáneas completas para los archivos interesados, mientras otros como imágenes pueden ser seleccionados para solo guardar el más reciente, algo útil a la hora de no malgastar espacio indebido, y algunos programas más modernos usan almacenamiento por cambios (deltas), de forma que solo los cambios sean las diferencias entre varias versiones, lo cual permite reducir el tamaño de guardado y soporte de versiones antiguas.

Estos sistemas pueden tener varias funciones clave, que, por general, son conocidas por su nombre inglés. Como funciona creando repositorios que existen tanto de forma local como:

- **Fetch:** La función de esta instrucción es la de coger todos los archivos que estén en el repositorio digital y actualizar el local con estos. Esto permite que los cambios hechos por solo el usuario sin tomar los cambios hechos por dispositivos remotos.
- **Pull:** Similar a Fetch, excepto que también arrastra cambios de dispositivos remotos, permitiendo que esté todo actualizado, incluido si trabajas junto con un equipo, y al subir los cambios que otras personas han hecho, permite tener los archivos más modernos. Evitando conflictos.

- **Commit:** Esta función crea una copia local, que subes a tu control de versiones local, añadiéndole un comentario, y guardándolo, mostrando si existen conflictos y dando acceso a versiones anteriores si por algún casual estos archivos sufren algún tipo de daño.
- **Push:** Una vez que los cambios que tengas listos por el día y que hayas subido a tu repositorio local estén acabados, entonces puedes usar la función Push. Esta permite subir los archivos directamente a la nube, de forma que otros usuarios puedan coger tu trabajo y añadir a este sin tener que usar dispositivos USB para transferir archivos.
- **Branch:** Esta opción nos permite para dos líneas de trabajo que son paralelas, por archivos que no son compatibles en un determinado momento, a seguir siéndose desarrollados por diferentes desarrolladores sin tener que pausar y trabajar en la compatibilidad de forma directa y que cause más problemas de los necesarios.
- **Merge:** Una vez que una Branch esté lista y probada para la compatibilidad con la de su rama opuesta, se puede usar el botón Merge para finalmente acabar por unir la versión de los archivos.

C. Plataformas más importantes

La plataforma más conocida y popular, a ciertos niveles hoy en día, es GitHub, una plataforma en la nube que usa el sistema de almacenamiento de instantáneas completas, sin almacenar solo los cambios.

Con esto, se hace con los modelos más estándares que hemos visto en el sistema anterior, aunque se necesitan de aplicaciones externas para las interfaces de usuarios que permitan el manejo más casual de estas mismas, una labor que suele venir o por programas que GitHub comparte como útiles para esto como GitHub Desktop, o como otros tal que SourceTree, los cuales son capaces de trabajar con GitHub.

La tecnología que estos programas usan es Git, que es el sistema que permite que estos programas de control de versiones funcionen de la manera que funcionen.

Sin embargo, también es importante mencionar que GitHub, aunque famoso y popular, es parte de la corporación de Microsoft desde 2018, lo que ha llevado a algunas controversias, como que Microsoft está pensando en usar este como una base de datos para sus compañías de Inteligencia Artificial, y que posea como material de aprendizaje los archivos que los usuarios hayan subido a sus repositorios.

Esta posible brecha en la seguridad ha hecho que los programadores de partes del mundo se estén desplazando a copias de este con algunos cambios pequeños, que son claramente clones, pero sin estas desventajas, y que se suelen centrar en comunidades pequeñas.

Unos ejemplos de estas son sourcehut.org o git.gay, está primera siendo advertida por sí misma como hacker-friendly al mantener la privacidad sin ningún tipo de rastreo, y la segunda estando centrada en la comunidad LGTB.

Estos ejemplos, son, obviamente, no muy populares, pero síntoma de que las acciones que GitHub está tomando está moviendo a usuarios a buscar alternativas que sean open source, sin seguimiento del trabajo de los usuarios por parte de empresas.

Otra que, en contraposición, es más famosa, es Mercurial, una plataforma que es conocida por su capacidad para poder manejar desde proyectos pequeños hasta proyectos bastante grandes, sin perder mucho rendimiento. Una de las ventajas que este programa tiene es que es fácil de aprender, pero acaba siendo no tan pulido como podría ser Git, necesitando de varias extensiones para hacer cosas que Git permite usar por defecto, como Git permite mover los Commit a una nueva base, creando un historial limpio y sencillo, mientras que, para Mercurial, usan un historial para que los Commit o ediciones del historial se mantengan guardados de forma paralela, necesitando la extensión MQ para poder hacer actualizaciones tan fácilmente.

Una de las últimas más famosas, es BitBucket, que suele ser mencionada en conjunción con GitHub. Una de sus funciones era que BitBucket permitía subir no solo perfiles de GitHub, sino que también de Mercurial al mismo tiempo, aunque fue discontinuado en 2020 a medida que seguían creciendo.

De etas funciones, para BitBucket, es la creación de infinito repositorios privados, mientras que, en la versión gratuita de GitHub, esto está limitado, así como la habilidad de poner revisiones de código de eso que se ha subido. Como ejemplo, el sistema de interfaz de SourceTree, pertenece a la compañía de BitBucket, aunque esta no es exclusiva a BitBucket. Aunque, por ejemplo, GitHub tiene en paralelo algunas funciones como la capacidad de ver diferencias entre ramas, mientras que BitBucket maneja permisos para las ramas para evitar y limitar accidentes al subir archivos. Además, BitBucket no marca el texto de código, a pesar de que es una de las cosas más básicas de GitHub cuando se leer código.

Estas diferencias hacen que no siempre sea una opción mejor, sino que hay diferentes opciones y atributos para que cada persona pueda escoger que tipo de sistema y que tipo de estructura quiere a la hora de escoger cómo hacer repositorios.

D. Principales aplicaciones de clientes de control de versiones

Como habíamos mencionado anteriormente, sus mayor y principal función es la del manejo del código por un equipo o una sola persona, controlando los posibles errores y el historial de todas las versiones anteriores, guardando cambios o haciéndolos difíciles de ver, también colaborar y compartir código con otros usuarios por medios simples como es un link.

Sin embargo, los programadores encuentran conforte en la estructura y funcionalidad de GitHub, y, al tener la capacidad de publicar diferentes tipos de archivos, como por ejemplo serían los documentos Word, Excel, o PowerPoint, permite de ser usado para algo más que simplemente una herramienta para escribir código, y tener siempre disponibles desde cualquier computadora con internet todos tus archivos.

Estas son ventajas que podrían beneficiar a un autor, funciones que OneDrive no puede realizar.

E. Controles de Versiones integrados en los IDES (Plugins)

Los IDES son los programas que permiten el estudio y la ejecución de código, así como de su fácil edición y sistemas de autocompletación. Estos son, por ejemplo, Visual Studio Code, Eclipse, IntelliJ IDEA, y más.

Estos sistemas pueden tener integrados dentro de estos plugins, como Git, el cual es famoso por su capacidad de ser integrado en casi cualquier IDE. Permitiendo la personalización de nuestras herramientas y tenerlo ajustado como se desee. Las integraciones de sistemas de control de versiones permiten manejar estos cambios en el dispositivo de una forma en segundo plano, mientras se mantiene un flujo de trabajo en el proyecto en el cual estés concentrado.

Los plugins de forma general, permiten que incluso cuando estos programas no tengan soporte directo, puedan seguir teniendo la capacidad de soporte indirecto teniendo la capacidad de escoger desde un ecosistema extenso de plugins, que estos programas permiten a sus usuarios instalar, que pueden ser creados por comunidades o terceros agentes sin necesidad de ser desarrollado por las propias empresas.

Sus mayores utilidades están en cómo estos plugins permiten usar herramientas clásicas de sistemas de control de versiones como pueden ser la confirmación de cambios en el código, o crear ramas o unirlas.

Igual, uno de los mejores fuertes de esto, es que todas estas funciones descritas arriba son completamente opcionales y disponibles para todos los usuarios, a tomar y dejar a su placer, de manera gratuita y sin mayores niveles de compromisos. Puede gustarte tenerlo ahí, o tenerlo separado para no mezclar tareas y tenerlo todo controlado de forma tradicional, e independientemente de esto, seguir permitiendo a otros usuarios a manejarlos a su propia manera.

3. Conclusiones

Las herramientas de sistemas de control de versiones son bastante potentes y especialmente útiles en sus ámbitos de desarrollo, permitiendo compartir y guardar archivos, así como de tener ciertos niveles de seguridad, y al tener naturaleza de código abierto, permiten la capacidad de ser trabajadas y mejoradas por desarrolladores externos, ofreciendo una amplia libertad de cómo quieren los usuarios customizar lo que ellos quisiesen hacer.

El uso de estas en el ámbito de código, especialmente en equipo, resulta de lo más conveniente por todas las medidas que se han descrito previamente, y su uso extensivo para ámbitos cotidianos pueden ofrecerlos como alternativa a otros documentos para aquellas personas que se acostumbren a estos.

De forma última, acaba siendo la elección del propio equipo o usuario de cómo va a funcionar y a ser útil, lo cual es lo más importante a la hora de establecer un entorno de trabajo en equipo, sean estas de las más grandes, o cerradas a pequeñas comunidades.

4. Recursos

- https://www.atlassian.com/es/git/tutorials/what-is-version-control
- https://codicesoftware-es.blogspot.com/2010/11/la-historia-del-control-de-versiones.html
- https://docs.github.com/es/get-started/start-your-journey/about-github-and-git
- https://www.vice.com/en/article/github-users-want-to-sue-microsoft-for-training-an-ai-tool-with-their-code/
- https://sourcehut.org/
- https://git.gay/
- https://hackernoon.com/lang/es/top-10-sistemas-de-control-de-versiones-4d314cf7adea
- https://www.mercurial-scm.org/
- https://book.mercurial-scm.org/read/changing-history.html#:~:text=First%20of%20all%2C%20changing%20or,edit%20history%20for%20those%20changes
- https://www.atlassian.com/blog/bitbucket/sunsetting-mercurial-support-in-bitbucket#targetText=After%20much%20consideration%2C%20we've,get%20migration%20resources%20and%20support
- https://www.upguard.com/blog/bitbucket-vs-github
- https://es.linkedin.com/advice/1/what-version-control-systems-provide-most-octhc?lang=es&lang=es
- https://aws.amazon.com/es/what-is/ide/