Informe Técnico De Plan De Trabajo Para Construcción De Software

GA7-220501096-AA1-EV01

Proyecto Del Software – DOKI VET

Aprendiz: Stefanny Vanegas Marin

Tecnólogo De Análisis Y Desarrollo De Software

Instructor: Milton Ivan Barbosa Gaona

Ficha: 2977481

Centro de la Tecnología de Diseño y de la Productividad Empresarial

SENA (Servicio Nacional de Aprendizaje)

Girardot – Cundinamarca

Julio 22 del 2025

Contenido

Contenido de imágenes

Introducción

El desarrollo de software exige metodologías modernas y eficientes que permitan llevar un control preciso de los cambios realizados en los proyectos. En este contexto, la integración continua y los sistemas de control de versiones se han posicionado como herramientas fundamentales para garantizar la calidad del código, permitir la colaboración en tiempo real y detectar errores en fases tempranas del desarrollo. Este informe técnico se enfoca en describir dichos conceptos, su utilidad, tipos de versiona-miento, comandos básicos de Git y la elección de GitHub como plataforma ideal para la gestión del código fuente.

Justificación

En los procesos de desarrollo de software, es común realizar constantes modificaciones y mejoras en el código. Sin una herramienta que permita controlar dichos cambios, es fácil perder versiones, cometer errores irreversibles o duplicar esfuerzos. La integración continua facilita la implementación de nuevas versiones y pruebas automáticas en un flujo de trabajo ágil. Además, los sistemas de control de versiones como Git y plataformas como GitHub permiten almacenar, compartir y recuperar versiones anteriores del proyecto, promoviendo el trabajo colaborativo y la eficiencia del equipo de desarrollo. Por esta razón, es crucial su estudio y aplicación en la formación de futuros desarrolladores.

Objetivos

Objetivo general:

Comprender y aplicar conceptos de integración continua y sistemas de control de versiones utilizando herramientas como Git y GitHub en proyectos de desarrollo de software.

Objetivo Específico:

- Identificar las características principales de la integración continua.
- Describir el funcionamiento de los sistemas de control de versiones.
- Aplicar comandos básicos de Git para el manejo de repositorios.
- Seleccionar herramientas de control de versiones adecuadas para entornos de trabajo colaborativo.
- Implementar buenas prácticas en el control y gestión del código fuente.

¿Qué Es La Integración Continua?

Como lo menciona (Facilito, 2018); se trata de "una práctica de desarrollo, en la cual los desarrolladores integramos nuestro código a un repositorio central, de forma periódica"; y así, a la hora en que se deba ejecutar el código de forma automática, incluya cada nueva versión, teniendo en cuenta las pruebas unitarias. Este mismo autor sostiene que esta es una práctica común que facilita la detección de errores y bugs en etapas tempranas del desarrollo del producto de software, para corregirlos, a costos bajos, manejando código (mezclando líneas de código nuevas sobre las versiones anteriores); para tener un código de calidad. Dicho lo anterior, ahora profundizo en la definición de los repositorios, las versiones y los Sistemas de Control de Versiones, a continuación.

¿Qué Es Un Repositorio?

Se trata de un espacio compartido en el cual un grupo de desarrollo "puede ver y gestionar, de forma unificada, el trabajo propio y el de los compañeros del proyecto, a través de sistemas, conocidos como Sistemas de Control de Versiones" (García et al., s.f., párr. 4).

Aplicado a la informática, son los espacios donde se almacenan archivos, sistemas de control de metadatos para guardar los cambios realizados (García et al., s.f.) o sistemas de red formados por varios elementos importantes, como software, hardware, datos y procedimientos para almacenar, ejecutar y modificar el código guardado; que "contiene objetos digitales y metadatos; ofrece funciones de gestión, archivo, preservación y sistemas adecuados de seguridad para dichos objetos; al asignar un identificador único persistente a cada objeto para su identificación" (Bibliolab, 2011)

Estos lugares o espacios de almacenamiento de datos pueden contener información relativa a documentos académicos, gubernamentales, literarios, entre otro tipo de materiales, y, por supuesto, código escrito en algún tipo de lenguaje de programación, que necesita de cambios y su control, gracias a un sistema que registre y almacene dichos cambios.

Los Sistemas De Control De Versiones

Tal como lo menciona García et al., (s.f., párr. 1), se trata de un sistema que se encarga de registrar los cambios realizados a lo largo del tiempo a uno o a un conjunto de archivos, en versiones; de forma que puedan ser recuperados conforme los usuarios lo necesiten.

Los Sistemas de Control de Versiones (VCS) permiten que los equipos hagan una copia de seguridad del código fuente de sus proyectos, y también que puedan archivarlo, facilitando la labor de revisar y hacer modificaciones en el repositorio, o de restaurar las versiones anteriores, si es necesario (Technologies, 2025)

Al crearse un nuevo repositorio en el VSC, se crea lo que se conoce como rama principal o tronco maestro, que permite a la base del código ingresar a un canal, en el que se compila e implementa; para que, al crear ramas de tal código, los desarrolladores puedan introducir cambios a la parte del código que necesitan y mantenerlos hasta llegado el momento de que el mismo Sistema tome esas ramas y las una, para fusionarlos en la rama principal, donde está la versión mayor del software. (Technologies, 2025)

Versiona-miento

Las *versiones* son definidas por Versionado de software (2023, párr. 2) como “el proceso de asignación de un nombre, código o número único a un software, para indicar su nivel de desarrollo”; considerando, a la hora de mejorar y optimizar el código:

- Número: (versiona-miento semántico): que suele estar representado por una serie de 3 números divididos por un punto, cada uno indica una cosa diferente: **versión mayor** (versión principal del software, nuevo módulo o característica importante), **versión menor** (nuevas funcionalidades, corrección de errores) y **parche** (revisión del producto a causa de errores, y cada entrega del proyecto, después de su revisión).
- Estabilidad: los populares alfa y beta, que se refieren a lo siguiente:

  Alpha: una versión inestable del software, que tiene muchas posibilidades de mejorar, que debe ser probada y testada, y de esta forma encontrar errores o poner a prueba nuevas funcionalidades. El software "está casi listo".

  Beta: es una versión un poco más estable del software; y solo se desea realizar pruebas de rendimiento, usabilidad y funcionamiento de uno o más módulos, en un ambiente no tan controlado; antes de ser candidato a su lanzamiento.

  RC o Release Candidate: solo necesita un retoque fino, antes de salir al mercado para el cliente y los usuarios finales. (ED Team, s.f.).

Crear versiones, no solamente es aplicable al software, sino también a diferentes tipos de archivos que se pueden guardar en un repositorio, para su posterior consulta o uso como fuente de datos secundaria.

El registro de los cambios de un archivo o software, en palabras de García, et al., (s.f.), permite:

- Regresar a versiones anteriores.
- Comparar los cambios realizados entre las versiones.
- Obtener información de las personas que afectaron los archivos.

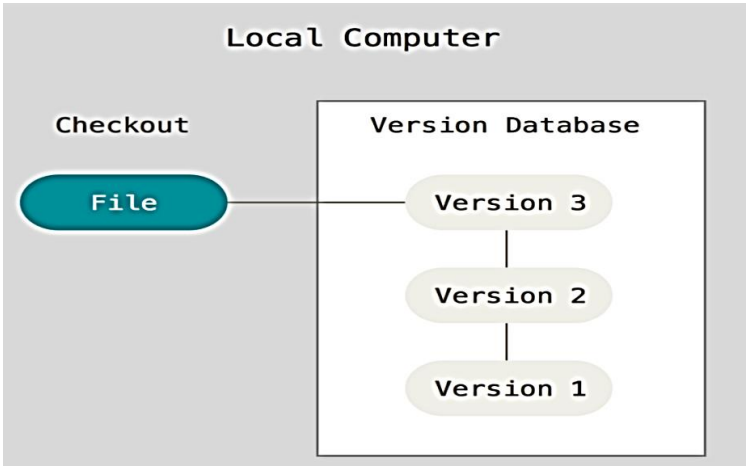
- ♥ Identificar la introducción de errores.
- ♥ Recuperar archivos defectuosos o eliminados (gracias a distintos mecanismos y herramientas de software).

Categorías de los Sistemas de Control de Versiones

Los Sistemas de Control de Versiones pueden ser categorizados en tres grupos:

Sistemas de Control de Versiones Locales: son sistemas que permiten guardar versiones de un archivo de forma manual en un directorio exclusivo o de fácil acceso; usando versiona-miento por numeración o fecha del versiona-miento. A pesar del hecho de que se pueden cometer errores al incluir la versión, o al olvidar la ubicación del directorio; hoy en día contamos con sistemas que contienen una BD local simple que registra información sobre los cambios realizados. Son usados por desarrolladores individuales, y los datos y las versiones, se almacenan en una sola computadora, disco duro o espacio en repositorio. Todos los cambios hechos al proyecto se almacenan como versiones.

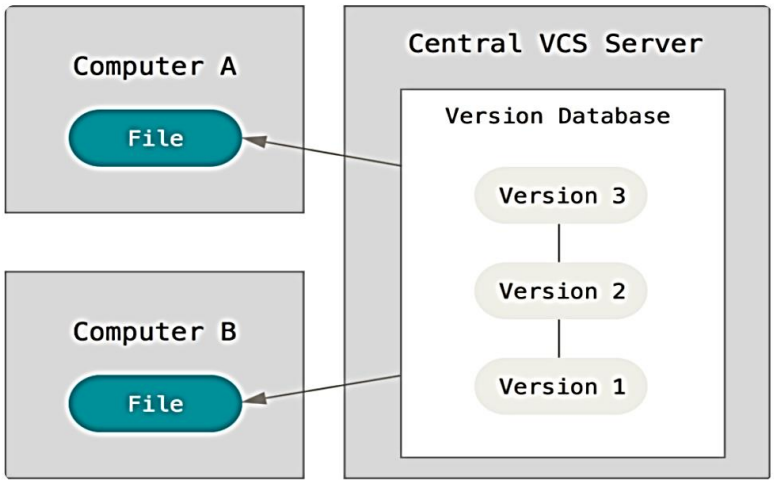
Imagen 1 Sistemas de control de versiones locales



Nota: imagen capturada en la guía de apoyo

Sistemas de Control de Versiones Centralizados: surgen para solucionar el problema del trabajo en equipo sobre un producto de software, a medida que se desarrollan e implementan cambios. Se trata de sistemas que disponen de un servidor donde se almacenan todos los archivos, y los clientes pueden realizar descargas del producto, a medida que salen nuevas versiones mejoradas y corregidas. Requieren una conexión a Internet y, si bien, pueden aparecer errores a la hora de fusionar el código en la rama principal, relacionados con el hardware o espacio en servidor Cloud, que puede retrasar o incluso hacer perder el trabajo, de no contar con copias de seguridad; sí que permite llevar un control más de las modificaciones de forma automática.

Imagen 2 Sistemas de control de versiones centralizados



Nota: imagen capturada en la guía de apoyo

Sistemas de Control de Versiones Distribuidos: son sistemas que "permiten subir código, crear ramas y fusionarlas, luego del trabajo de los desarrolladores, sin necesidad de conexión al servidor principal" (Unity, s.f.), porque cada miembro del equipo de desarrollo trabaja almacenando los cambios a un repositorio clonado guardado Cloud. Esto permite el

trabajo por separado, a gran velocidad. Eso sí, si un miembro del equipo desea descargar todo el proyecto, le puede tomar un periodo de tiempo considerable.

Git

Es un Sistema de Control de Versiones Distribuido (DVCS) que nació en 2005, como producto de la necesidad de desarrolladores del entorno Linux de "encontrar un mecanismo o herramienta que soportara las actualizaciones que los desarrolladores realizaban a su código, que dota a cada versión posee una copia que se puede descargar y en la que se puede trabajar, sin necesidad de conexión con el servidor central.

Ofrece seguridad a la hora de realizar y guardar cambios de las versiones, al usar códigos criptográficos que los detectan, siendo imposible que los cambios pasen desapercibidos por la herramienta y los miembros de los equipos que trabajan en el proyecto; permitiendo la recuperación de la información y la detección de los datos eliminados.

Imagen 3 Logo de Git

Nota: imagen capturada en la guía de apoyo

El Sistema de Control de Versiones de Git maneja tres estados principales para cada uno de los archivos de un proyecto:

1. Confirmado: committed.
2. Preparado: staged.
3. Modificado: modified.

En Git, todos los archivos tendrán dos estados, por defecto: **tracked files**, que están en la última versión del proyecto, independientemente de si han sido creados o modificados o no; y **untracked files**, que son aquellos archivos que NO formaban parte de la última versión del proyecto, y que tampoco están en el área de preparación de este.

Comandos Básicos de los Sistemas de Control de Versiones

Antes de trabajar con un Sistema de Control de Versiones como Git, el desarrollador debe conocer algunos comandos básicos que le posibilitarán: la creación de repositorios, directorios y subdirectorios, visualizar el estado de los archivos, rastrear archivos y directorios, entre otras funciones útiles:

a. $ git init para iniciar un repositorio desde un directorio existente.

b. $ git clone https://url_del_repositorio para crear un nuevo subdirectorio, el cual contendrá todos los archivos necesarios para tal repositorio.

c. $ git status para visualizar el estado actual de los archivos.

d. $ git add Nombre_archivo para rastrear archivos nuevos.

e. $ git add Directorio para rastrear directorios en Git.

f. $ git commit sirve para preparar archivos y marcar archivos como resueltos cuando entran en conflicto y este se resuelve.

g. $ git commit -m “En esta versión se arregló el archivo W” para confirmar el mensaje anterior.

h. $ git commit -a -m ‘comentario de esta confirmación’ esta operación de confirmación se encarga de preparar todos los archivos rastreados y luego confirmar.

i. $ git log para visualizar el histórico de las confirmaciones, desde la más reciente hasta la más antigua, realizadas sobre un repositorio.

Git en Entornos Remotos

Si estuviera trabajando en mi proyecto con un equipo de desarrolladores, y necesitara implementar los cambios, usando para esto el concepto de integración continua para un repositorio remoto; tendría que tomar en cuenta que aprender los siguientes comandos, sería mandatorio:

a. $ git remote para ver los repositorios remotos.

b. git remote add [nombre-remoto] [url] para definir un repositorio remoto y asociarlo a un nombre, como referenciación.

c. $ git remote add ref https://github.com/paulboone/ticgit Donde nombre-remoto corresponde al nombre con que se referencia el repositorio y URL es la ubicación lógica del mismo en un entorno de red o en una dirección de internet.

d. $ git fetch [nombre-remoto] para extraer los datos del repositorio remoto, de los cuales todavía no se tiene copia en el repositorio local.

e. git push [nombre-remoto] [nombre-rama] para enviar información desde el repositorio local hacia el servidor remoto.

f. $ git push origin master si se ha clonado un repositorio desde alguna ubicación, Git asigna el nombre original al servidor del que se ha realizado la clonación.

Herramienta de Sistemas de Control de Versiones a Trabajar

De acuerdo con García et al., (s.f.) hoy existen plataformas que permiten trabajar en la nube, y que permiten implementar procesos de integración contínua; que permiten crear repositorios públicos o privados que se integran con SCV, permiten el registro de miembros de grupos de trabajo, entre otras funcionalidades, basados en el sistema Git.

Basado en lo visto en el material de formación, y en lo estudiado y visto por mi cuenta, elijo GitHub como la herramienta de trabajo a utilizar.

Imagen 4 Logo de GitHub.

Nota: Fuente: García et al., (s.f.).

GitHub es una plataforma de alojamiento, propiedad de Microsoft, que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de versiones llamado Git; lo que me facilita la aplicación de los comandos a trabajar.

Facilita la organización de proyectos y permite la colaboración de varios desarrolladores en tiempo real; aunque voy a trabajar solo en mi proyecto. Es decir, me permitirá centralizar el contenido del repositorio para poder dejarlo ahí, o más adelante, crear repositorios en los cuales pueda trabajar en equipo.

Conclusión

El dominio de herramientas de integración continua y control de versiones es esencial en la formación de desarrolladores de software. Git y GitHub proporcionan soluciones eficientes para la gestión del código fuente, permitiendo a los equipos mantener la calidad del producto, documentar los cambios y colaborar sin conflictos. Su uso desde el inicio de un proyecto fortalece la organización y evita errores comunes durante el desarrollo.

Referencias

Bibliolab. (2011, 30 de noviembre). ¿Qué es un repositorio digital? Canal de YouTube. https://www.youtube.com/watch?v=z2xFHVX4N6E&t=47s

Código Facilito. (2018, 14 de diciembre). ¿Qué es integración continua? Canal de YouTube. https://www.youtube.com/watch?v=_BNtMVZiD8UE

ED Team. (s.f.). ¿Cómo se deciden las versiones del software? Página web. https://ed.team/blog/como-se-deciden-las-versiones-del-software

García, M., Lizcano R., Bastidas, H., Ramírez, D., Pinchao, P., Guevara, O., Lizcano, F., Ortíz, J., Vecino, J., Ruiz, Z., Arenales, W., Rodríguez, C., Tamayo, C., Litivin, O.,

Moreno, J., Arévalo, J., Santaella, A., Manchego, L., Bustillo, J… Villamil, M. (s.f). *Integración Contínua*. (Material de estudio). Tecnología en Análisis y Desarrollo de Software

ADSO. Servicio Nacional de Aprendizaje SENA.

Unity. (s.f.). ¿Qué es el Control de Versiones? Página web. https://unity.com/es/solutions/what-is-version-control

Versionado de software. (2023, 8 de noviembre). En Wikipedia. https://es.wikipedia.org/wiki/Versionado_de_software