Calachua Ramos, Gerson Alonso 20134549A

ENSAYO DEL CURSO DEL CURSO DE LENGUAJE DE PROGRAMACIÓN ORIENTADO A BJETOS

FIFTH ESSAY

Luego de un par de semanas agitadas y por qué no, algo relajadas en el tema de los ensayos, regresamos nuevamente con estos. En esta oportunidad y las que vienen los presentes ensayos serán dirigidos, en pocas palabras los temas serán seleccionados por el profesor y a nosotros los estudiantes nos competerá el buscar toda la información relacionada a los temas asignados.

En esta oportunidad hablaremos acerca del GIT y del GITHUB; pero para hablar de ambos antes tenemos que hacer un pequeño repaso de lo que son los sistemas de control de versiones.

Puede pasar que al ir desarrollando un software toques o muevas algo de tu código principal o que alguien más lo haga y tu programa ya no funcione; este es un problema muy molesto y que causan muchas complicaciones por lo cual se han creado programas para ayudarnos en este aspecto es así como iniciamos hablando de los **sistemas de control de versiones**, son programas que tienen como objetivo controlar los cambios que ocurran al desarrollar cualquier tipo de software, permitiendo conocer el estado actualizado del de un proyecto, así como los cambios efectuados a cualquier parte de este o saber quienes fueron las personas implicadas en este cambio.

Necesidad de un control de versiones

El control de versiones es una de las tareas fundamentales para la administración de un proyecto de desarrollo de software en general. Surge de la necesidad de mantener y llevar control del código que vamos programando, conservando sus distintos estados. Es absolutamente necesario para el trabajo en equipo, pero resulta útil incluso a desarrolladores independientes.

Alternativas y variantes de sistemas de control de versiones

Comenzaron a aparecer los sistemas de control del versionado del software por los años setenta, aunque al principio eran bastante elementales.

Tenemos principalmente dos tipos de variantes:

Sistemas centralizados: En estos sistemas hay un servidor que mantiene el repositorio y en el que cada programador mantiene en local únicamente aquellos archivos con los que está trabajando en un momento dado. Ese sistema centralizado es el único lugar donde está todo el código del proyecto de manera completa. Subversion o CVS son sistemas de control de versiones centralizados.

Sistemas distribuidos: En este tipo de sistemas cada uno de los integrantes del equipo mantiene una copia local del repositorio completo. Al disponer de un repositorio local, puedo hacer *commit* (enviar cambios al sistema de control de versiones) en local, sin necesidad de estar conectado a Internet o cualquier otra red. Es cierto que es local de momento, luego podrás compartirlo con otras personas, pero el hecho de tener un repositorio completo me facilita ser autónomo y poder trabajar en cualquier situación. Git es un sistema de control de versiones distribuido.

Luego de esa pequeña introducción comenzaremos con desarrollar y explicar todo lo relacionado al Git.

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o frond end (tema ya estudiado en ensayos anteriores). Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

La primera gran diferencia de Git con respecto a otros sistemas de control de versiones es la forma que tiene de manejar los cambios en los ficheros. Mientras que otros scv's, (del inglés *comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, almacenan los archivos originales, conservando una lista de los cambios realizados a dichos archivos en cada versión, Git guarda una "foto" (snapshot) del estado de cada archivo en un momento concreto. Si uno de los archivos no ha cambiado no crea una nueva copia del mismo, simplemente crea una referencia al archivo original.

La segunda es la eficiencia. Git se basa en que cada programador almacena una copia completa del repositorio en su máquina de forma local, incluido el historial de cambios. Esto implica que muchas de las operaciones realizadas sobre el código

fuente no tienen lugar en la red, permitiendo que la velocidad de proceso dependa únicamente en los recursos locales.

CARACTERÍSTICAS:

- Fuerte apoyo al desarrollo no lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones.
- Gestión distribuida. Al igual que Darcs, BitKeeper, Mercurial, SVK, Bazaar y Monotone, Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
- Los almacenes de información pueden publicarse por HTTP, FTP, rsync o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH. Git también puede emular servidores CVS, lo que habilita el uso de clientes CVS pre-existentes y módulos IDE para CVS pre-existentes en el acceso de repositorios Git.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.

APORTES:

- ✓ Auditoría del código: saber quién ha tocado qué y cuándo.
- ✓ Control sobre cómo ha cambiado nuestro proyecto con el paso del tiempo.
- ✓ Volver hacia atrás de una forma rápida.
- ✓ Control de versiones a través de etiquetas: versión 1.0, versión 1.0.1, versión 1.1, etc. Sabremos exactamente que había en cada una de ellas y las diferencias entre cualquiera de ellas dos.
- ✓ Seguridad: todas las estructuras internas de datos están firmadas con SHA1.
- ✓ No se puede cambiar el código sin que nos enteremos.
- ✓ Mejora nuestra capacidad de trabajar en equipo.

GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework Ruby on Rails por *GitHub, Inc.* (anteriormente conocida como *Logical Awesome*). Desde enero de 2010, GitHub opera bajo el nombre de *GitHub, Inc.* El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

En resumen es un hosting online para nuestros repositorios que utiliza Git para el mantenimiento y versionado del código fuente, añadiendo una serie de servicios extras para la gestión del proyecto y el código fuente.

Tanto Git como GitHub esconden mucho más detrás de estas sencillas explicaciones, pero tal vez ayuden a salvar la barrera que impida a muchos implicarse y empezar a colaborar en algún proyecto comunitario.

CARACTERÍSTICAS:

- Wiki para cada proyecto
- Página web para cada proyecto
- Gráfico para ver cómo los desarrolladores trabajan en sus repositorios y bifurcaciones del proyecto
- Funcionalidades como si se tratase de una red social, como por ejemplo: seguidores;
- Bueno para trabajo colaborativo entre programadores.

BIBLIOGRAFÍA:

- http://wpmallorca.com/2013/02/12/pero-que-es-github/
- http://www.desarrolloweb.com/articulos/introduccion-git-github.html
- https://barradevblog.wordpress.com/2013/01/21/que-es-gitgithub/
- http://es.wikipedia.org/wiki/Git
- http://es.wikipedia.org/wiki/GitHub