LP00

Trabajo

- Ensayo N° 04

Nombre del profesor:

- Hancco Carpio, Rony Jordan

Nombre del alumno:

- Negrete Girano, Luis Carlo

Lima, 22 de mayo del 2015

CUARTO ENSAYO SOBRE LENGUAJE DE PROGRAMACIÓN ORIENTADOS A OBJETOS

Este ensayo, será un poco diferente a los demás ya que es sobre dos temas en particular, específicamente, sobre “Git” y Github”. Sin más preámbulo, pasemos a hablar sobre ellos.

En primer lugar, qué es Git, una respuesta sería: es un software de control de versiones diseñado por Linus Torvalds, creador de Linux, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Asimismo, Git es un sistema distribuido de control de código fuente o SCM (en inglés Source Code Management). Ahora bien, qué es un SCM, un SCM es una herramienta que nos permite resolver una serie de problemas a todos aquellas personas que tienen que trabajar con código fuente. Al referirnos a “código fuente” podemos referirnos a muchas cosas, por ejemplo:

- Ficheros HTML / CSS / Javascript
- Ficheros PHP
- Ficheros de configuración
- Documentación, etc.

Al parecer su funcionamiento es simple, pero no se trata de solo eso, además, como bien sabemos, una persona que se dedica a trabajar en este campo, tiene que actualizar constantemente el código fuente, el SCM nos permite saber en qué momento hemos modificado nuestro código, que partes de este específicamente hemos modificado, de tal manera, que si ha ocurrido un error, simplemente para corregirlo o depurarlo, nos enfocamos en las líneas que han sido modificadas y así la labor será mucho más sencilla.

Así también, las características de Git son las siguientes:

- ✓ Gran apoyo en el desarrollo de código no lineal.
- ✓ Desarrollo distribuido.
- ✓ Compatibilidad con los sistemas y protocolos actuales.
- ✓ Nos permite un manejo eficiente de proyectos largos.
- ✓ No se puede cambiar el código sin que lo notemos.
- ✓ Realmacenamiento periódico en paquetes(ficheros).

Respecto a las implementaciones para los cuales se usa el Git, fundamentalmente fue creado para ser aplicado en Linux, sin embargo, también se puede utilizar en otros sistemas operativos tales como BSD, Solaris, OS X y Microsoft Windows. Cabe mencionar a “JGit”, que es una implementación Git de librería en Java que se puede utilizar para cualquier aplicación Java; así como también a “Dulwich”, que es una implementación de Git en Python.

Debemos considerar órdenes básicas que nos permiten trabajar con el Git, estos son:

- **git fetch:** Descarga los cambios realizados en el repositorio remoto.
- **git merge <nombre_rama>:** Impacta en la rama en la que te encuentras parado, los cambios realizados en la rama “nombre_rama”.
- **git pull:** Unifica los comandos fetch y merge en un único comando.
- **git commit -am "<mensaje>":** Confirma los cambios realizados. El “mensaje” generalmente se usa para asociar al commit una breve descripción de los cambios realizados.
- **git push origin <nombre_rama>:** Sube la rama “nombre_rama” al servidor remoto.
- **git status:** Muestra el estado actual de la rama, como los cambios que hay sin commitear.
- **git add <nombre_archivo>:** Comienza a trackear el archivo “nombre_archivo”.
- **git checkout -b <nombre_rama_nueva>:** Crea una rama a partir de la que te encuentres parado con el nombre “nombre_rama_nueva”, y luego salta sobre la rama nueva, por lo que quedas parado en ésta última.

- **git checkout -t origin/<nombre_rama>:** Si existe una rama remota de nombre "nombre_rama", al ejecutar este comando se crea una rama local con el nombre "nombre_rama" para hacer un seguimiento de la rama remota con el mismo nombre.
- **git branch:** Lista todas las ramas locales.
- **git branch -a:** Lista todas las ramas locales y remotas.
- **git branch -d <nombre_rama>:** Elimina la rama local con el nombre "nombre_rama".
- **git push origin :<nombre_rama>:** Elimina la rama remote con el nombre "nombre_rama".
- **git remote prune origin:** Actualiza tu repositorio remoto en caso que algún otro desarrollador haya eliminado alguna rama remota.
- **git reset --hard HEAD:** Elimina los cambios realizados que aún no se hayan hecho commit.
- **git revert <hash_commit>:** Revierte el commit realizado, identificado por el "hash_commit".

Por último, debemos considerar que existen cuatro ramas en el Git, las cuales son Master, contiene el repositorio que se encuentra publicado en producción; Development, son todas las nuevas funcionalidades se deben integrar en esta rama; Features, cada nueva funcionalidad se debe realizar en una rama nueva; y, Hotfix, que son bugs que surgen en producción.

Ahora, nos toca hablar sobre Github, este es es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git, el cual fue explicado anteriormente. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Se debe resaltar que el código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.
Github te facilita toda la infraestructura para trabajar en equipos distribuidos a través de una interfaz web.

Por qué usar Github, por la sencilla razón de que si tienes una copia de tu código fuente en Github, tienes un backup de todo el proyecto completo. Ese backup incluye no sólo el código que tienes ahora sino también de todo el historial de modificaciones que el código ha sufrido desde el primer día. Esta copia la puedes recuperar en cualquier momento y continuar trabajando desde cualquier ordenador como si nada hubiese sucedido.

En la actualidad, GitHub es mucho más que un servicio de alojamiento de código. Además de éste, se ofrecen varias herramientas útiles para el trabajo en equipo. Entre ellas, caben destacar:

- Una wiki para el mantenimiento de las distintas versiones de las páginas.
- Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
- Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
- Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

Para poder empezar a utilizar, debemos ir a la web Githum.com y desde ahí crear una cuenta, recordemos que puede ser gratuita, como también puede pagarse en ella. Luego, para poder iniciar proyectos dentro de él es necesario crear repositorios, pero qué es un repositorio, es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos. Este se puede crear como público (permite la modificación por parte de otras personas) o como privado (solo puede ser modificado por el creador del repositorio).

Dentro del mismo, se pueden agregar diversos tipos de archivos, carpetas, etc. Cabe mencionar que es necesario utilizar una serie de comandos Git, para poder realizar determinadas acciones, como los que se verán a continuación:

Para crear un proyecto:

```
touch README.md
git init
git add README.md
git commit -m "comentario"
git remote add origin https://github.com/LuchoCastillo/Repositorio.git
git push -u origin master
```

Para subir archivos:

```
git add archivo
git commit -m "comentario"
git push
```

Finalmente, para culminar con la presentación, recordemos que así como nos pueden ayudar en nuestros proyectos, nosotros también podemos aportar en proyectos ajenos, siempre y cuando estén como públicos. La idea fundamental del Github, es permitirnos trabajar en equipo para así optimizar los proyectos que se realizan en él.