PRÁCTICA DE GIT POR COMANDOS

En clase de Desarrollo de Interfaces (a partir de ahora, DI) hemos hecho una serie de prácticas relacionadas con Git, uno de los programas de control de versiones más utilizados. De hecho, es el que se utiliza para el desarrollo del kernel de Linux. En esta primera entrada realizaremos un caso práctico basado en la programación por parejas. La situación es la siguiente:

Dos desarrolladores, John y Jessica, están colaborando para desarrollar un programa en Java. Para ello, utilizan Git para el control de versiones. Ambos utilizan la versión de Linux que funciona por comandos en el terminal. Para llevar a cabo la programación por parejas, han creado un repositorio remoto vacío en GitHub llamado actividadJohnJessica. A continuación se relatarán una serie de pasos que han ido siguiendo ambos durante el desarrollo.Lo primero que hacen John y Jessica es clonar el repositorio remoto cada uno en su equipo. Por defecto, los repositorios tanto remotos como locales sólo tienen una rama llamada master. De momento con esta rama tienen suficiente. Al realizar la clonación tendrán una copia en local de los archivos del proyecto conjunto. Dicho proyecto se copiará en la rama master del repositorio local. Para ello en el terminal de Linux escriben:

git clonehttps://github.com/GitParejas/actividadJohnJessica.git

Esta dirección URL se encuentra visible en el repositorio remoto precisamente para permitir la clonación.

John crea un archivo HolaMundo.java en su repositorio local y lo sube al repositorio remoto. Para ello escribe en su terminal:

git add HolaMundo.java

git commit -m “Subo HolaMundo.java”

git push origin

Para subirlo tendrá que escribir el usuario y la contraseña de la cuenta compartida en GitHub con Jessica.

Ahora Jessica hace *pull* (descarga) del repositorio remoto. Para ello escribe:

git pull

Así lo que está haciendo es bajarse las últimas versiones de los archivos del proyecto.

John modifica el archivo HolaMundo.java de nuevo y lo vuelve a subir al repositorio remoto, por lo que escribe estos comandos:

git add HolaMundo.java

git commit -m "Subo HolaMundo.java otra vez"

git push origin

Para subirlo tendrá que escribir el usuario y la contraseña de la cuenta compartida en GitHub con Jessica.

No hay problemas al hacer pull, ya que la versión de HolaMundo.java que estaba en el repositorio remoto era consistente con la que tenía John en local.

Mientras tanto, Jessica modifica el archivo HolaMundo.java que tenía en su repositorio local. Al intentar subirlo obtendrá un mensaje de error. Esto es debido a que la versión de HolaMundo.java del repositorio remoto tiene modificaciones que no tenía la versión en local de Jessica antes de que ella empezara a modificarlo. Es decir, los cambios que realizó John anteriormente no los tiene Jessica.

Lo primero que tiene que hacer Jessica en bajarse una copia del repositorio remoto en local. Al hacerlo se creará una segunda rama en el repositorio local de Jessica, llamada origin/master. Esto lo hace escribiendo el siguiente comando en el terminal:

git fetch origin

A continuación comprueba las inconsistencias entre los archivos locales que están en la rama master con los que están en la rama origin/master. Para ello escribe:

git merge origin/master

Al hacer merge, Jessica puede ver que hay inconsistencias entre el archivo HolaMundo.java que había editado y el que se acaba de descargar en la rama origin/master. Ya sólo le queda aceptar y rechazar los cambios que crea oportunos. A continuación, Jessica sube el archivo corregido con los siguientes comandos:

git add HolaMundo.java

git commit -m "Resueltos conflictos en HolaMundo.java"

git push origin

Para subirlo tendrá que escribir el usuario y la contraseña de la cuenta compartida en GitHub con John.

Mientras tanto, John ha estado trabajando en una rama local aparte, llamada issue54. Para crear una rama y tenerla como activa, John ha escrito los siguientes comandos:

git branch issue54

git checkout issue54

Ahora John tiene dos ramas en su repositorio local; master e issue54. Al crear la rama issue54, lo que ha hecho es crear una copia de la rama master local. Al activarla, ahora cualquier cambio que haga lo hará en dicha rama, quedándose la rama master en local sin cambios.

John ha hecho dos cambios en el archivo HolaMundo.java (ha hecho dos adds y dos commits). Después de estar conforme con el trabajo que ha hecho, John tiene que fusionar la rama issue54 con la rama master en local. Para ello, primero tiene que "saltar" a la rama master y escribir:

git checkout master

A continuación fusionará ambas ramas para que la rama master tenga los cambios realizados en la rama issue54. Para ello escribe:

git merge issue54

Ahora tiene que subir los cambios al repositorio remoto. Lo primero tiene que hacer es un *fetch* con la rama master del repositorio remoto. Para ello escribe:

git fetch origin

De esta manera se crea la rama local origin/master con los archivos ubicados en la rama master del repositorio remoto.

Ya solo le queda fusionar los cambios entre ambas ramas con el siguiente comando:

git merge origin/master

Después de la ejecución del último comando, en el repositorio remoto, en la rama master, se encuentra el proyecto conjunto de John y Jessica totalmente consistente y con el trabajo de ambos actualizado.