Git: mooc_git-entrega3_rebase2

Objetivo

Ver nuevos usos de GitHub y de "git rebase". En concreto clonar repositorios con fork y "git clone …", así como rehacer la rama master con "git rebase --interactive …". Este ejercicio muestra también como Git puede gestionar diferencias entre ficheros de texto plano, que no sean programas.

Resumen de la práctica

Duplicar el repositorio https://github.com/jquemada/tf_agenda con Fork en su cuenta en GitHub y clonarlo en un directorio local del mismo nombre. Este repositorio contiene solo el fichero tf-agenda.txt con texto plano. Es una agenda telefónica muy simplista, con 3 teléfonos, creada en 4 commits, cuyo único objetivo es practicar con "git rebase --interactive …"

```
$ cat tf_agenda.txt
John: 913-677-899;
Eva:  915-768-455;
Mary: 918-789-221;
$
$ git log --oneline
9eaa103 Add Mary tf
71e69ce Add Eva tf
1204dc8 Add Eva pending-tf
f6e660e Add John tf
$
```

En el primer commit se introdujo el teléfono de John. En el segundo solo el nombre de Eva, pero se dejó el teléfono pendiente tal y como indica el mensaje del commit. En el tercero se añade el teléfono de Eva solamente. Y en el cuarto se introduce el teléfono de Mary.

En esta práctica se rehacen los commits de la rama **master** utilizando "git rebase --interactive …". Se deben juntar los commits 2 (1204dc8 Add Eva pending-tf) y 3 (71e69ce Add Eva tf) en uno solo, y corregir el teléfono de Mary (918-789-221) por el número 918-555-555. Debe quedar así:

```
$ cat tf_agenda.txt
John: 913-677-899;
Eva:  915-768-455;
Mary: 918-555-555;
$
$ git log --online
2eb1703 Add Mary tf   fixed
4716ce5 Add Eva tf    integrated
f6e660e Add John tf
$
```

Finalizar subiendo la rama **master** local (regenerada) a la rama **corrected_tf_agenda** del repositorio origin, que se creará porque no existe. El repositorio origen en Github tendrá ahora tanto la rama **master**, como la nueva **corrected_tf_agenda**.

Prueba de la práctica

Para comprobar que la práctica ha sido realizada correctamente hay que utilizar el validador de este repositorio

https://github.com/practicas-ging/mooc_git-entrega3_rebase2

Recuerde que para utilizar el validador se debe tener node.js (y npm) (https://nodejs.org/es/) y Git instalados. El proyecto se descarga, instala y ejecuta en el ordenador local con estos comandos:

```
$                                              ## El proyecto debe clonarse en el ordenador local
$ git clone https://github.com/practicas-ging/mooc_git-entrega3_rebase2
$
$ cd mooc_git-entrega3_rebase2                 ## Entrar en el directorio de trabajo
$
$ npm install                                   ## Instala el programa de test
$
$ npm run checks  [nombre_de_la_cuenta]     ## Pasa los tests sobre el repositorio en github
.....                                           ## indicando que partes  están correctamente
.....                                           ## implementadas y cuales no.
... (resultado de los tests)
$
```

Debe cambiar [nombre_de_la_cuenta] por el nombre de su cuenta en GitHub.

Entrega y evaluación

La entrega consiste en subir a MiriadaX el nombre de la cuenta en GitHub donde se ha copiado con Fork el repositorio https://github.com/<su_cuenta>/tf_agenda. Recuerde que antes de hacer la entrega debe haber actualizado dicho repositorio con la nueva rama master modificada.

¡Cuidado! Compruebe que el nombre de la cuenta subido es el correcto y que el repositorio esta actualizado con la última versión.

El evaluador debe comprobar que la entrega es correcta buscando en GitHub el nombre de cuenta entregado y comprobando que contiene el repositorio pedido con las características solicitadas.

RÚBRICA: La resolución de cada uno de estos puntos dará un el % indicado de la nota total:
10%: Existe el repositorio my_calculator
20%: El primer commit de la rama master el original: f6e660e Add John tf
35%: El segundo commit de master es "Add Eva tf integrated" y contiene los 2 originales integrados
35%: El tercer commit de master es "Add Mary tf fixed" y contiene el original corregido

El objetivo de este curso es sacar el máximo provecho al trabajo que están dedicando, por lo que les recomendamos que utilicen la evaluación para ayudar a sus compañeros enviando comentarios sobre la corrección del código, su claridad, legibilidad, estructuración y documentación. Dado que es un curso para principiantes, ante la duda les pedimos que sean benevolentes con sus compañeros, porque muchos participantes están empezando y los primeros pasos siempre son difíciles.

¡Cuidado! Una vez enviada la evaluación, está no se puede cambiar. Piensen bien su evaluación antes de enviarla.

Desarrollo de la práctica

Seguir los siguientes pasos para la realización de esta practica.

Paso 1) Copiar el repositorio https://github.com/jquemada/tf_agenda en su cuenta de GitHub usando el botón de Fork.

Paso 2) Clonar el repositorio <su _cuenta>/tf_agenda (de su cuenta en GitHub) en su ordenador local.

Sugerencia de comandos a utilizar

```
$ git clone     https://github.com/<su _cuenta>/tf_agenda
$
```

Paso 3) Utilizar "git rebase --interactive f6e660e" para juntar los commits 2 (1204dc8 Add Eva pending-tf) y 3 (71e69ce Add Eva tf) en uno solo, y corregir el teléfono de Mary (918-789-221) por el número 918-555-555.

La opción --interactive (equivalente a -i) permite rehacer interactivamente los 3 últimos commits de la rama **master**. f6e660e es equivalente a HEAD~3: referencia al tercer commit del grafo de commits, en dirección a la raíz, relativo al que está en el directorio de trabajo. Al invocar este comando, se abre el editor por defecto (normalmente nano o vi) con este script (los comentarios (empiezan por #) contienen instrucciones)

```
$ git rebase -i f6e660e
.....

========== script del rebase ==============
pick 1204dc8 Add Eva pending-tf
pick 71e69ce Add Eva tf
pick 9eaa103 Add Mary tf

# Rebase f6e660e..9eaa103 onto f6e660e
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
================================
```

El script está en las 3 primeras líneas, que siguiendo las instrucciones (comentarios) modificamos para que junte los commits 2 y 3, y además nos abra el editor en el commit 4.

========== script del rebase ==========
```
pick 1204dc8 Add Eva pending-tf
squash 71e69ce Add Eva tf
edit 9eaa103 Add Mary tf

.............
```
==========

Al cerrar la edición del fichero con el nuevo script, Git integra los commits 2 y 3 automáticamente (sabe que tiene que dejar el código del commit 3), pero nos abre el editor para que editemos el mensaje asociado al nuevo commit en la historia

========== contenido del fichero ==========
```
# This is a combination of 2 commits.
# The first commit's message is:

Add Eva pending-tf

# This is the 2nd commit message:

Add Eva tf

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Fri Jan 11 18:58:54 2019 +0100
#
# rebase in progress; onto f6e660e
# You are currently editing a commit while rebasing branch 'master' on
'f6e660e'.
#
# Changes to be committed:
#       modified:   tf_agenda.txt
#
```
==========

Dejamos el segundo mensaje añadiendo la palabra "integrated" al final (para indicar que se han integrado los dos commits) y cerramos la edición para que el rebase continúe. El script indica que queremos editar el commit 4 por lo que Git devuelve control a la shell (sin finalizar el rebase) dando el siguiente mensaje

```
.............
[detached HEAD 4716ce5] Add Eva tf    integrated
 Date: Fri Jan 11 18:58:54 2019 +0100
 1 file changed, 1 insertion(+)
Stopped at 9eaa1031b1c5eb7d52a19970a3a0967193d0b5a3... Add Mary tf
You can amend the commit now, with

     git commit --amend
```

```
Once you are satisfied with your changes, run

     git rebase --continue

$
```

Git nos deja en el commit 4 de la rama master original, indicándonos que debemos modificar el código de dicho commit con un "amend" y continuar el rebase. La opción --amend rehace el commit anterior, envés de crear uno nuevo. Editamos el fichero tf_agenda.txt para cambiar el teléfono de Mary por 918-555-555 y una vez modificado hacemos amend al commit 4 con:

```
$
$ git add tf_agenda.txt
$ git commit --amend
.....
```

Antes de cerrar el nuevo commit 4, Git nos abre el editor con el mensaje asociado al commit para que lo podamos modificar, le añadimos la palabra "fixed" al final (para indicar que hemos corregido el commit) y cerramos la edición para que el amend finalice

```
.....
[detached HEAD 2eb1703] Add Mary tf   fixed
 Date: Fri Jan 11 19:01:44 2019 +0100
 1 file changed, 1 insertion(+)
$
```

Y con el commit 4 corregido con el amend, continuamos el rebase para que finalice

```
$
$  git rebase --continue
Successfully rebased and updated refs/heads/master.
$
```

Estado final de la rama master:

Después del rebase, tanto el contenido de la agenda en tf_agenda.txt, como los commits de la rama master han quedado tal y como se pedía: commits 2 y 3 integrados y commit 4 corregido

```
$ cat tf_agenda.txt
John: 913-677-899;
Eva:  915-768-455;
Mary: 918-555-555;
$
$ git log --online
2eb1703 Add Mary tf   fixed
4716ce5 Add Eva tf    integrated
f6e660e Add John tf
$
```

Si alguna vez se equivoca uno al rehacer una rama, se puede utilizar el reflog para arreglarlo: https://git-scm.com/docs/git-reflog.

Paso 4) Finalizar subiendo la rama **master** local (regenerada) a la rama **corrected_tf_agenda** del repositorio origin, que se creará porque no existe. El repositorio origen en Github tendrá ahora tanto la rama **master**, como la nueva **corrected_tf_agenda**.

Si hubiésemos querido subir la nueva rama master del repositorio local a la rama master del repositorio origin (repositorio origen de la clonación) deberíamos usar el comando "git push --force …" porque los commits son incompatibles. Utilizando la opción --force o -f se sobre-escriben los commits antiguos.

¡Atencion! los commits antiguos se pierden al sobre-escribirlos y no podrán ser recuperados en ese repositorio. En un desarrollo real no se deben compartir repositorios, ni ramas que vayan a ser sobre-escritas posteriormente. Los commits añadidos por terceros a las copias del repositorio no sobre-escrito serán incompatibles con los commits nuevos creados sobre las antiguas ramas.