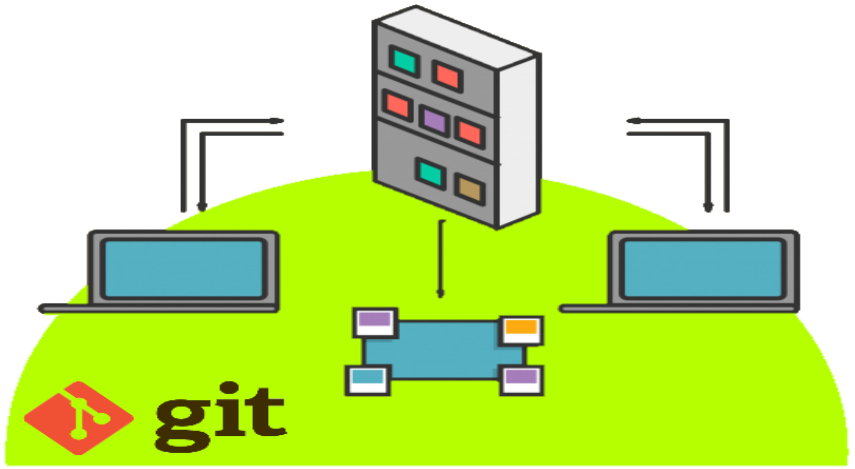
Entornos de Desarrollo

Tarea 2: Controlar cambios Java



Universidad Europea

Raúl Rodríguez Mercado

raul.rodriguez@universidadeuropea.es

Área de Arquitectura, Ingeniería y Diseño

Tarea 2: Controlar cambios Java

CONFIGURACIÓN INICIAL

Configura Git por primera vez

- ✓ Una vez hecho el paso anterior y dado que tienes Git en tu sistema, vamos a personalizarlo.
- ✓ Esta configuración sólo es necesario hacerla una vez; se mantendrán hasta que vuelvas a cambiarlo.
- ✓ Git trae una herramienta llamada **git config** que te permite obtener y establecer variables de configuración, que controlan la configuración, el aspecto y funcionamiento en general de Git.

Tarea 2: Controlar cambios Java

CONFIGURACIÓN INICIAL – 10%

Tu Identidad

- ✓ Lo primero que debes hacer cuando instalas Git es establecer tu nombre de usuario y dirección de correo electrónico.
- ✓ Esto es importante porque las confirmaciones de cambios (commits) en Git usan esta información, y es introducida de manera inmutable en los commits que envías.
- ✓ Ve al terminar de comandos git (bash) y utiliza el comando git config para configurar tu nombre y dirección de correo electrónica con la siguiente sintaxis:

$ git config --global user.name <*nombre y primer apellido*>
$ git config --global user.email <*email*>

Nota: sustituye el texto entre < > con tus datos, pero no pongas dichos símbolos en la instrucción.

- ✓ Comprueba el cambio con la siguiente instrucción:

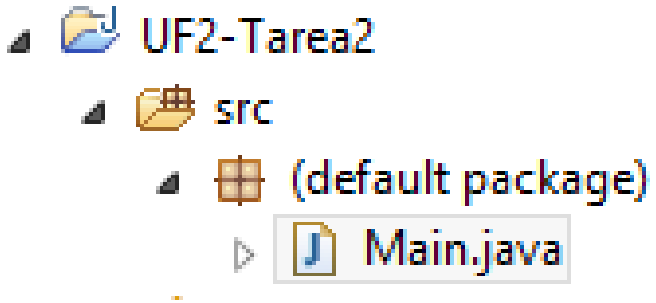
$ git config --list

Tarea 2: Controlar cambios Java

INTRODUCCIÓN - 15%

Proyecto Java

- ✓ En esta práctica vas a realizar el control de versiones de un proyecto Java.
- ✓ Para ello, lo primer que debes hacer es crear en tu directorio de trabajo el proyecto “UF2-Tarea2” y dentro de él la clase llamada “Main” con la función “main” en su interior, de la siguiente manera:




```
public class Main {
    public static void main(String[] args) {
        System.out.println("");
    }
```

- ✓ Introduce en el mensaje del println tu propio nombre (sin apellidos) y prueba que se escribe correctamente por consola.

Tarea 2: Controlar cambios Java

CREAR REPOSITORIO - 20%

Iniciar Git en un directorio

- ✓ Utilizando el comando “cd” debes llegar hasta la carpeta de proyecto “UF2-Tare2” que has creado para el proyecto java y escribir:

$ git init

- ✓ Esto crea un nuevo subdirectorio llamado .git que contiene todos los archivos necesarios del repositorio —un esqueleto de un repositorio Git. Pero recuerda que todavía no hay nada en tu proyecto que esté bajo seguimiento (**untracked**), compruébalo mirando el estado, escribiendo:

$ git status

- ✓ El siguiente paso es que incluyas todo el proyecto en el control de versiones de Git, para eso escribe:

$ git add .

- ✓ Vuelve a comprobar el estado del proyecto (git status) y observa las diferencias con el estado anterior.

Tarea 2: Controlar cambios Java

PRIMER COMMIT - 30%

Guardando cambios en el repositorio

- ✓ Con la anterior instrucción viste que aparecían ficheros "Modificados pero no actualizados" ("Changes not staged for commit") —esto significa que hay cambios en el staging área que no han sido confirmados todavía. Es el momento de hacer nuestro primer commit

$ git commit –m "Primera versión"

- ✓ Ahora volver a solicitar el estado, ejecutando:

$ git status
On branch master
nothing to commit, working directory clean

- ✓ Todo ha ido bien, y podemos ver nuestro primer commit:

$ git log

- ✓ Y comprueba toda la información que obtienes…**ya tienes tu primer commit!**
- ✓ Vuelve a comprobar el estado del proyecto (git status) y observa las diferencias con el estado anterior.

Tarea 2: Controlar cambios Java

SEGUNDA MODIFICACIÓN – 35%

Incluye tu primer apellido

✓ Vuelve a Java e incluye una nueva línea para imprimir tu primer apellido :

System.out.println("<apellido1>");

Nota: sustituye el texto entre < > con tu propio apellido, pero no pongas dichos símbolos en la instrucción.

✓ Ahora debes volver a hacer todo lo necesario para crear un segundo commit con un mensaje que ocupe dos líneas:

"Segunda versión
Se ha incluido el primer apellido"

✓ Por lo tanto, en lugar de utilizar la opción –m de commit, tendrás que utilizar el editor **vi** para hacerlo.
✓ Ahora comprueba las diferencias entre los dos commits con:

$ git diff

Tarea 2: Controlar cambios Java

TERCERA MODIFICACIÓN - 40%

Incluye el segundo apellido

- ✓ Nos acabamos de dar cuenta de que queríamos incluir también el segundo apellido, así que lo cambiamos en el código java:

System.out.println("<apellido1 apellido2>");

Nota: sustituye el texto entre < > con tus propios apellidos, pero no pongas dichos símbolos en la instrucción.

- ✓ Lo que sucede es que no queremos que este cambio se haga en nuevo commit, por lo que vamos a utilizar el comando

$ git commit –-amend

- ✓ Y utilizando el editor **vi** debes cambiar el comentario a:

“Segunda versión
Se ha incluido los dos apellidos”

Tarea 2: Controlar cambios Java

INTRODUCCIÓN DE UN ERROR 1 – 50%

Incluye una tercera línea de código

- ✓ Ahora ve al código java e introduce una tercera impresión:

System.out.println(“Esto es el primer error");

- ✓ Ejecútalo en tu entorno y luego vuelve a git para deshacer el error con:

$ git checkout -- .

Nota: Te en cuenta que el “.” es un comodín que sirve para indicar “todos los ficheros”

- ✓ Ve a eclipse y comprueba tu código, ¿qué ha pasado?

Tarea 2: Controlar cambios Java

INTRODUCCIÓN DE UN ERROR 2 -60%

Incluye una tercera línea de código

- ✓ Ahora ve al código java e introduce la impresión:

System.out.println(“Esto es el segundo error");

- ✓ Ejecútalo en tu entorno y luego vuelve a git para ejecutar:

$ git add .

- ✓ Ahora debes ejecutar un comando para eliminar el cambio del staging área, ¿recuerdas qué comando era?
- ✓ Ejecuta el comando para ver el estado de tu proyecto:

$ git status

- ✓ Utilizando la misma instrucción que para el error 1, elimina la línea de tu código.

Tarea 2: Controlar cambios Java

COMPROBANDO DIFERENCIAS – 70%

Diferencias entre commits

- ✓ Escribe la instrucción para comprobar las diferencias entre el primer y el commit actual utilizando el comodín HEAD:

$

- ✓ Escribe la instrucción para comprobar las diferencias entre el primer y el segundo commit utilizando el comodín HEAD:

$

- ✓ Ahora muestra todos los commits en una sola línea y que incluya las etiquetas master y head.

$

Tarea 2: Controlar cambios Java

INTRODUCCIÓN DE UN ERROR 3 – 80%

Revert

- ✓ Ahora ve al código java e introduce la impresión:

System.out.println(“Esto es el tercer error");

- ✓ Haz todo lo necesario para conseguir tener un nuevo commit y después comprueba que tienes 3 commits

$

- ✓ Utilizando **revert** debes deshacer lo que has hecho.

$

- ✓ Comprueba que es correcto, si todo ha ido bien debes tener 4 commits diferentes.

Tarea 2: Controlar cambios Java

ELIMINAR UN COMMIT - 90%

Reset

- ✓ Aunque no sea algo muy conveniente, vamos a borrar parte de nuestro historial de cambios para pasar de 4 commits a sólo 3, por lo tanto debes eliminar el último commit con la opción **reset**:

$

- ✓ Comprueba ahora que sólo tienes 3 commits

$

- ✓ Vete ahora a java y comprueba el código que tienes son tres líneas

Tarea 2: Controlar cambios Java

CAMBIAR DE VERSION - 100%

Checkout

- ✓ Si llegas a este paso es que has hecho todo lo anterior correctamente, ¿cuántas versiones tienes de tu proyecto?
- ✓ Con el comando checkout puedes cambiar de versión, haciendo que el HEAD sea diferente. Ahora debes comprobarlo cambiando el head a la primera versión, es decir, al primer commit que hiciste.

$

- ✓ Comprueba tu situación con

$ git log --decorate

- ✓ ¿Cuántas versiones ves, cómo puedes explicarlo?
- ✓ Ahora vuelve a la última versión utilizando la etiqueta MASTER

$

Tarea 2: Controlar cambios Java

¡LO HAS CONSEGUIDO!

100% = 10 puntos

Universidad Europea

Madrid Valencia Canarias