Control de versiones

Ejercicio 1. Crea la siguiente estructura de ramas a partir de uno o varios ficheros de texto que te inventes con listas de la temática que consideres adecuada. Adjunta los comandos y capturas del proceso seguido (muestra de algún modo, mediante comandos git, la estructura generada de ramas para que quede claro que es correcta según la figura).

Primero creamos el repositorio llamado repo en el escritorio, y creamos el archivo .txt llamado deportes, dentro de él habrá una lista de deportes. Comprobamos la rama y el estado con "git status". Ejecutamos "git branch" para ver el numero de ramas, nos indica que solo esta la rama master. Para añadir una rama más ejecutamos "git branch -b Hotfix", "git Branch" de nuevo para comprobar el número de ramas, ahora aparecen dos, master y Hotfix. Añadimos el resto de ramas: "git branch -b (nombre de rama), se llamarán reléase, develop y feature.

Después, ejecutamos "git log –graph –all" para ver los commit de cada rama, solo hay un commit referenciado por las dos ramas (HEAD->master->hotfix). Ahora cambiamos de rama con "git checkout Hotfix", nos avisa GIT que estamos en la rama HOTFIX. "Git status" para comprobar el estado y nos indica que el árbol de trabajo está limpio. Después modificamos el archivo que estará en hotfix y hacemos "git add ." "git status". Realizamos git commit -m "mma"", nos indica que hemos añadido una línea nueva. "git status" y hemos modificado el archivo deportes.txt en la rama hotfix. Si ejecutamos

“git log –graph –all”, vemos como el commit de master sigue intacto mientras que el de la rama hotfix que acabamos de tocar tiene un commit posterior en fecha. Cambiamos de rama a “git checkout develop” y añadimos el archivo “git add .” “git status”, acabamos de añadir el archivo deportes.txt del primer commit de master a develop.

Cambiamos de rama a “git checkout master” y vemos que no tiene la línea mma. Añadimos otra línea en master con “git add .” “git commit -m “esgrima” y vemos como se ha añadido esgrima pero no mma en master. Añadimos otro elemento, sincronizada con “git add .” “git commit -m “sincronizada” y “git log –graph –all” para ver el gráfico de ramas como se ha modificado dos veces seguidas el commit de master pero hotfix sigue como antes.

Ahora vamos a añadir fusionar el fichero de hotfix con master, ejecutamos “git merge hotfix” y avanzan ambas ramas a la misma posición.

Cambiamos de nuevo de rama a la rama develop con “git checkout develop” añadimos con “git add.” “git commit “Triatlon”” y fusionamos de nuevo las ramas, esta vez hotfix y develop, por lo que ejecutamos “git merge hotfix” y se nos actualiza ambas ramas al mismo nivel.

De nuevo cambiamos de rama a feature “git checkout feature” y añadimos el archivo “git add .” “git status”, hacemos un commit “git add.” “git commit “boxeo”” y fusionamos con develop para que queden al mismo nivel “git merge develop” y a su vez fusionamos con reléase “git merge release” quedando las tres al mismo nivel.

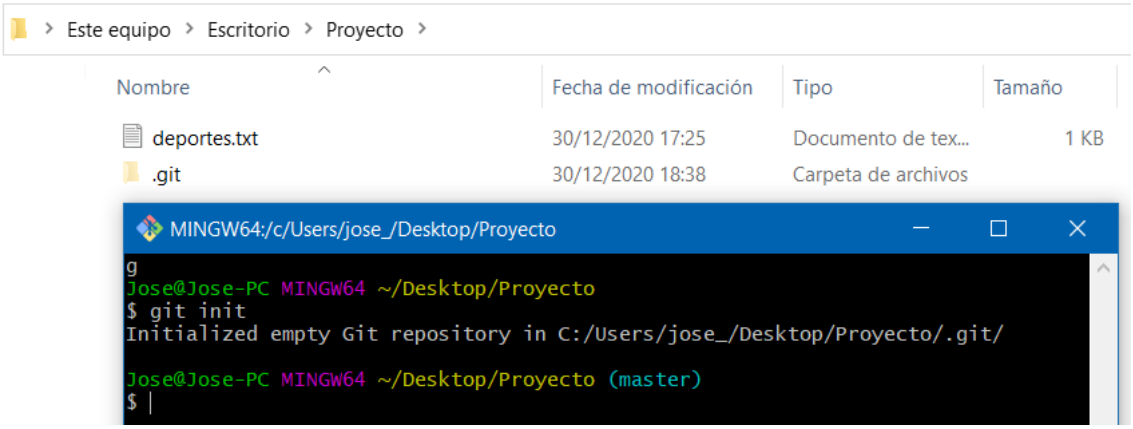
Nos posicionamos en reléase “git checkout release” y hacemos commit “git add.” “git commit “atletismo”” para posteriormente fusionar y quedar al mismo nivel reléase, develop y master. Para ello ejecutamos “git merge develop”, “git checkout release” y “git merge master”, de este modo están las tres ramas al mismo nivel y fusionadas, quedando feature dos puestos por debajo y hotfix cuatro puestos por debajo.

Ejercicio 2. Realiza los siguientes pasos y adjunta una captura de pantalla de cada paso:

• Crea un directorio llamado Proyecto

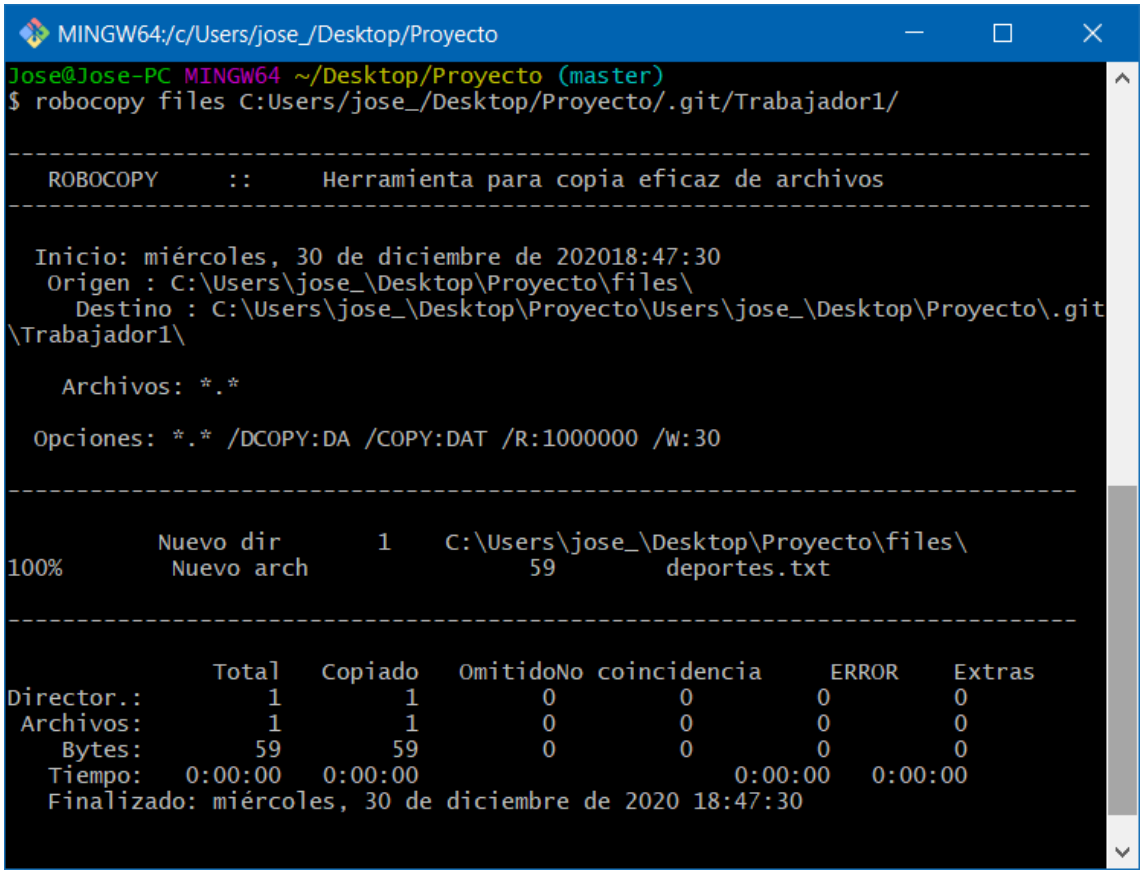
Proyecto-> "Git init"

• Inicializa en él un repositorio vacío



• Dentro de una carpeta llamada Trabajador1 crea una copia de ese proyecto

"robocopy files C:Users/jose_/Desktop/Proyecto/.git/Trabajador1/"



• Dentro de otra carpeta llamada Trabajador2 crea una copia de ese proyecto

“robocopy files C:Users/jose_/Desktop/Proyecto/.git/Trabajador2/”

• Trabajador1 añade al repositorio un proyecto web:

o Carpeta html con un fichero index.html con una capa centrada de color verde

o Carpeta css con los estilos

“git add .”

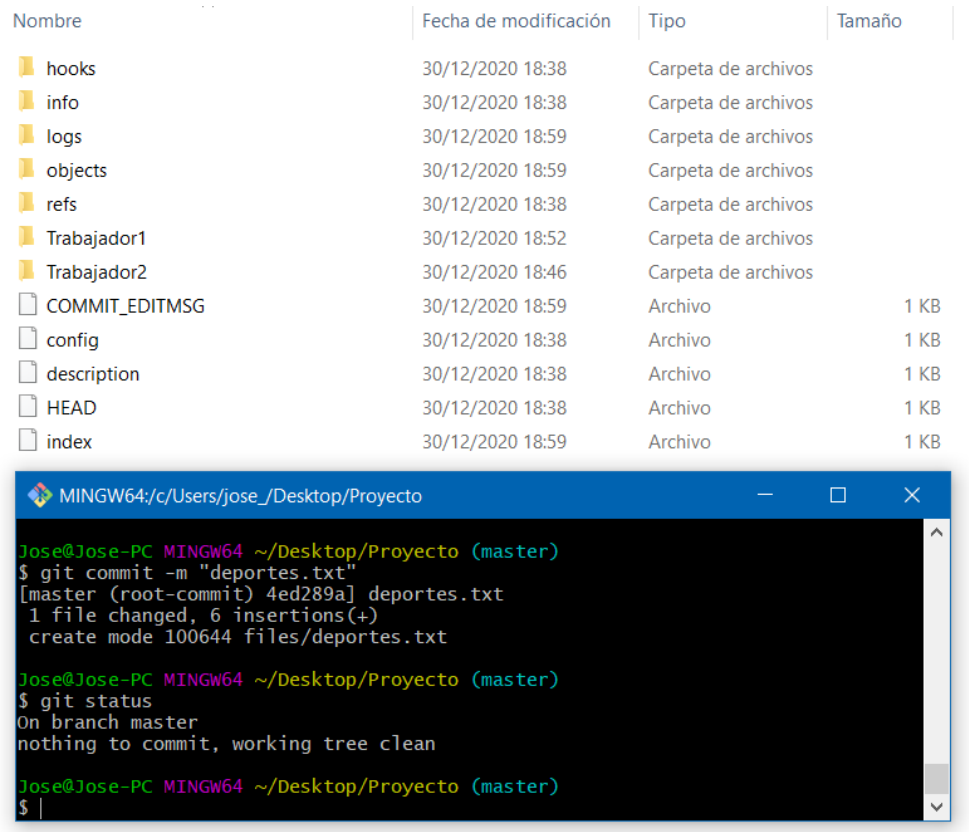
“git commit -m “index.html” y “css.css”

• Trabajador2 comprueba si hay cambios

“git branch -b trabajador2

“git checkout release”

“git status”



• Trabajador2 obtiene los cambios de Trabajador1

“git merge Trabajador1”,

Ejercicio 3. Trabajo con repositorios, issues, forks y pull requests.

• Haz un fork del repositorio localizado en la siguiente url:

https://github.com/alejandroSF-FP/UT2-Actividades-3

Adjunta la url de tu bifurcación.

https://github.com/Xele10/UT2-Actividades-3

• Realiza un primer commit a nivel de consola. Crea un fichero html con el nombre ejercicio3NombreYApellidos.html en el que incorpores una imagen. ¿Dónde haces ese commit, en el repositorio original o en la copia? Adjunta los comandos.

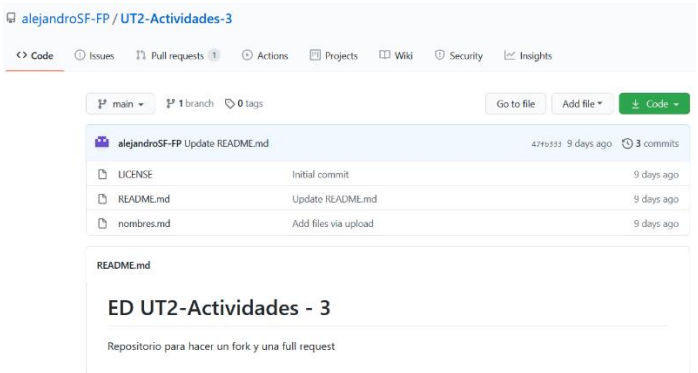
“git add.”

Hago el commit en la copia: “git commit -m “img””

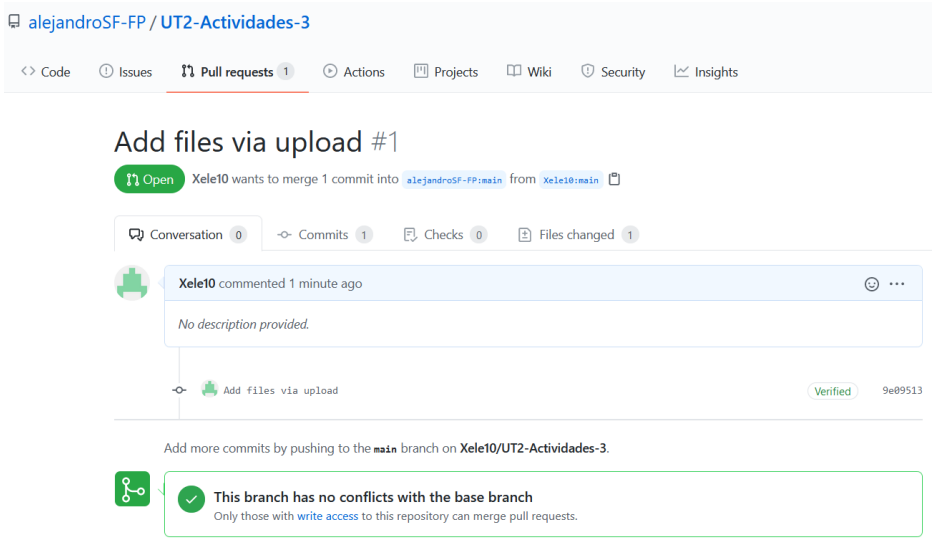
• Crea una tarea: añadir un título

• Añade el título a tu fichero html y realiza un commit cerrando la tarea con el mensaje del mismo.

“git status”, “git add –patch” y “git commit -m “titulo””

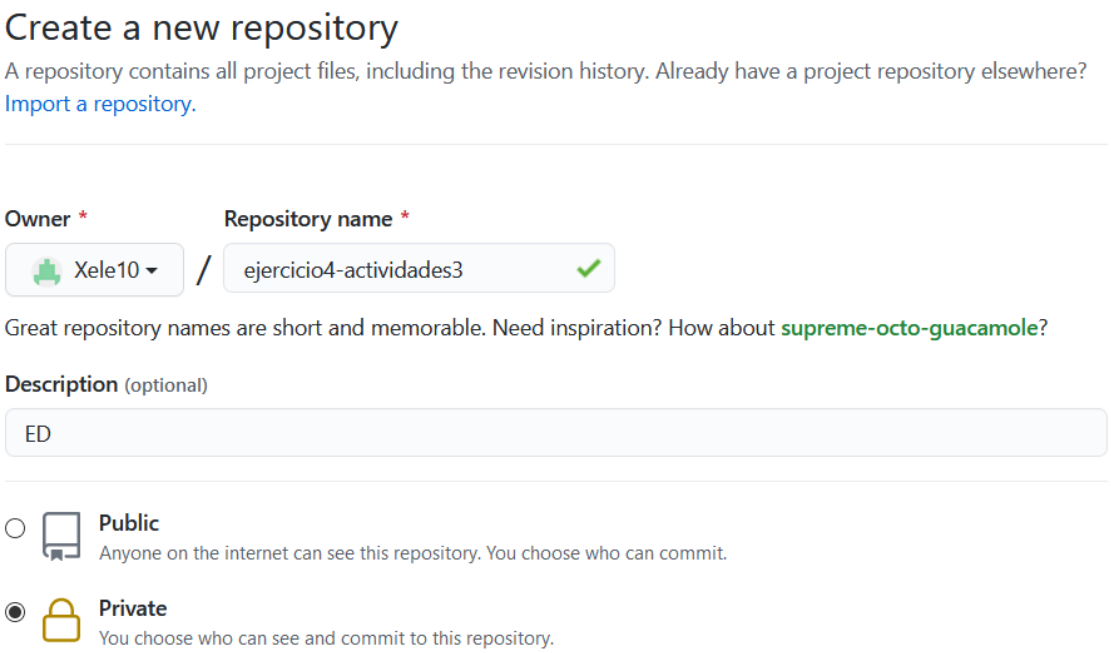


• Realiza una pull request indicando en el mensaje que has completado la tarea. ¿Qué acción has realizado con la pull request? Pídeme que lo acepte y revisa que este cambio se haya incorporado al repositorio original.



Ejercicio 4. Trabajo con repositorios y colaboradores. Adjunta una captura de pantalla de cada paso.

• Crea un repositorio que se llame ejercicio4-actividades3



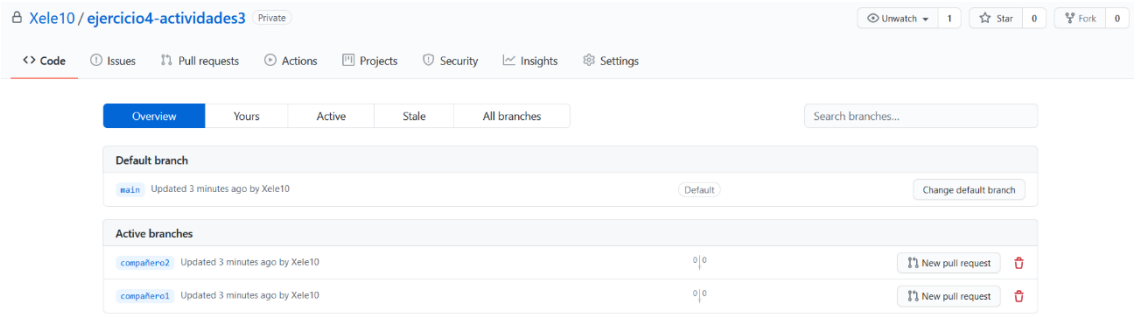
• Compártelo con dos compañeros de clase de manera que puedan tener acceso completo a él

https://github.com/Xele10/ejercicio4-actividades3.git

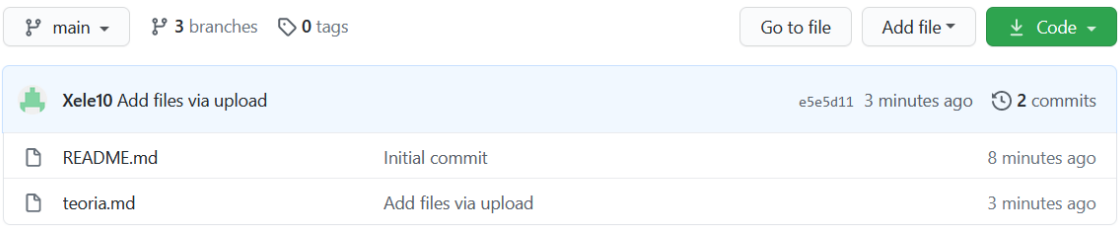
• ¿Qué diferencia hay entre este ejercicio y el anterior?

En el anterior había que hacer una copia de otro repositorio y en este doy la copia yo a quien quiero.

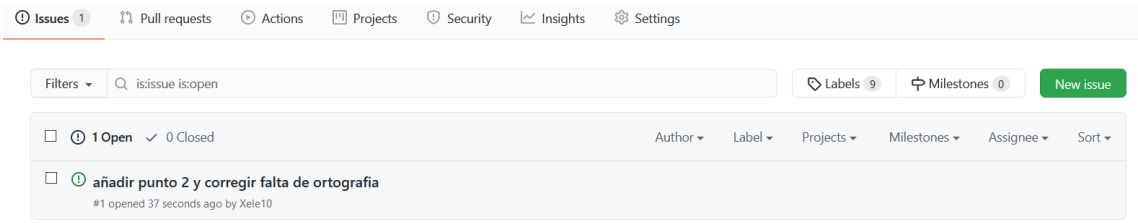
• Crea una rama para cada compañero



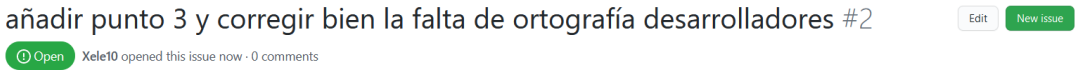
• Añade un primer fichero llamado teoría.md donde copies la introducción de la teoría del tema de ED de control de versiones en formato MarkDown pero cambia desarrolladores por "desaroyadores".



• Crea una tarea asignada a un compañero donde copia el punto 2. Además, este compañero debe corregir la palabra "desaroyadores" por "desarroyadores". Sube los cambios a su rama.



• Crea una tarea asignada a un compañero donde copia hasta el punto 3.1. Además, este compañero debe corregir la palabra "desaroyadores" por "desarolladores". Sube los cambios a su rama.



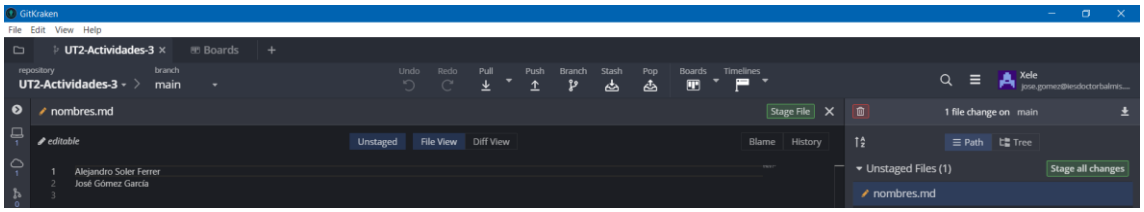
• Ambos compañeros actualizan tu repositorio. Mezclas las ramas. ¿Qué ocurre?

Se realiza "git merge" y se fusionan las tres ramas quedando al mismo nivel.

Ejercicio 5. Utiliza el cliente GitKraken para conectarte a GitHub, al repositorio remoto https://github.com/alejandroSF-FP/UT2-Actividades-3 y añade tu nombre al fichero nombres.md. Documenta los pasos seguidos.

Descargar e instalar GitKraken. Crear cuenta o autorizar acceso desde nuestro GitHub. Ir a File->Clone Repo, seleccionar carpeta de destino y poner la dirección compartida de github.

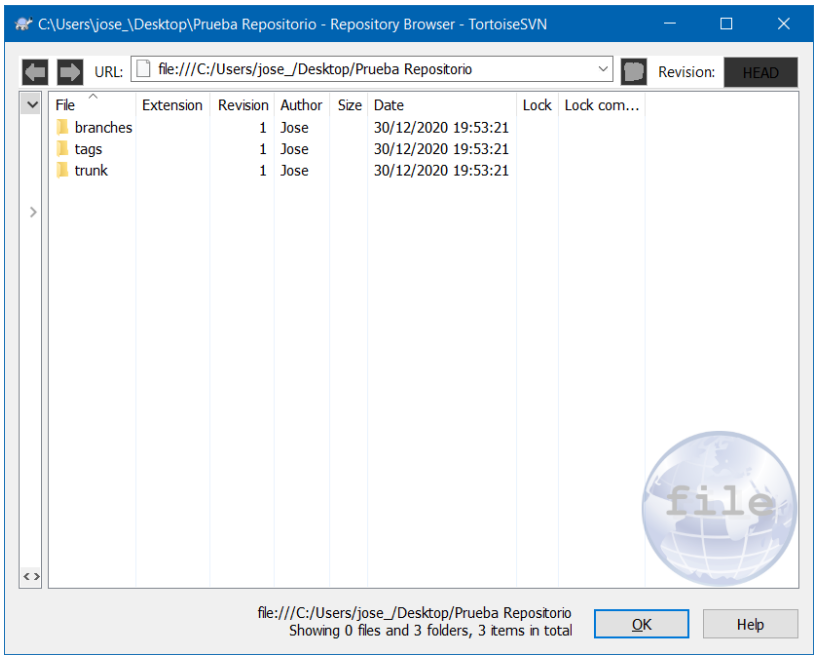
Después, click en add files via upload-> abrir nombres.md-> guardary push para pedir al administrador que realice el cambio de línea y fusione.



SVN

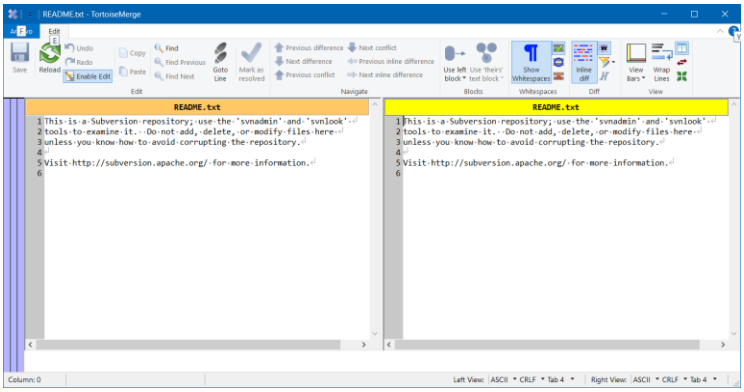
Ejercicio 6. ¡Aprende a usar SVN tú mismo! Documenta el proceso seguido y adjunta las capturas de pantalla que consideres adecuadas, puedes entregar también un comprimido con el repositorio svn final generado.

Entra a la siguiente url y descárgate la última versión del cliente SVN http://tortoisesvn.net/ Crea una carpeta vacía en tu ordenador llamada "Prueba repositorio" Con botón derecho, selecciona TortoiseSVN-> Crear repositorio aquí. Puedes elegir crear la estructura interna de la carpeta o no. Importa el proyecto fuentes al repositorio: Dirígete al directorio fuentes Con botón derecho selecciona la opción IMPORT Como mensaje le pones "Importando el proyecto al repositorio" Como url del respositorio la ruta absoluta del directorio "Pruebas repositorio" Importar

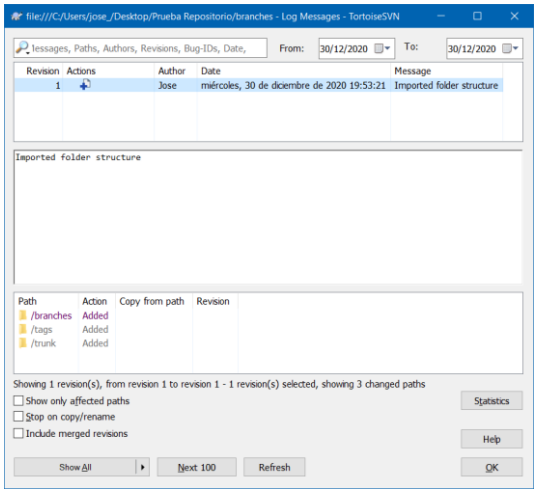


En el repositorio se almacena el proyecto, pero para trabajar con el proyecto debemos hacer una copia del mismo, y una vez realizados los cambios, actualizaremos nuestra copia. Créate una carpeta que se llame Carpeta-trabajo Con botón derecho en la carpeta, selecciona la acción SVN-Checkout En la url del repositorio debería aparecer la url de dónde habéis creado el repositorio En checkout-repositorio, la carpeta o directorio donde queremos hacer la copia del proyecto para trabajo. Presionar sobre ok Vamos a hacer un cambio en nuestra copia de trabajo Dirígite a la Carpeta-trabajo Origen1

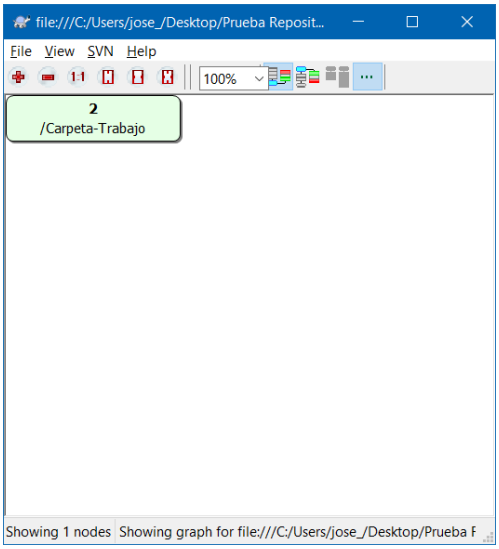
Src Edita con Notepad++ el fichero main.cc, cambia la línea 26, el texto “Algo va mal”, por “Este no es el camino correcto” Imagínate que este fichero no lo has cambiado tú, ha sido un compañero de tu grupo de trabajo, y quieres saber qué ha cambiado. Simplemente sobre el archivo, con botón derecho, selecciona la opción Diff, que te informará de las diferencias entre este fichero y el que tienes en tu repositorio.



Si ahora quisiéramos actualizar la copia de trabajo, para tener la última versión… Sobre el directorio Carpeta-trabajo, seleccionamos opción Actualizar (SVN Update) con botón derecho En algún momento podemos necesitar añadir más archivos, bien porque el proyecto está expandiéndose, bien porque se está durante la fase de desarrollo y el repositorio ha sido creado antes de que se haya establecido la estructura. Para poder añadir ficheros: Dentro de la Carpeta-trabajo, origen1 crea un fichero README.txt Escribe en el fichero “Haciendo pruebas de añadir ficheros” Con botón derecho sobre el fichero, seleccionar la opción TortoiseSVN->add En todo momento se puede ver el historial de cambios y de versiones realizados, consultando la opción Show Log.



Siempre podemos volver a una versión anterior, usando la opción Revert Crea un fichero paraborrar.txt Añádelo a tu repositorio Muestra el log para comprobar que se ha añadido Elíminalo de tu repositorio con la opción Revert



Imagina esta situación:

Este es el problema de compartir archivos, de trabajar en un mismo proyecto sobre los mismos archivos. Varios desarrolladores modifican la misma información.

Según el tipo de repositorio tenemos varias soluciones: Bloquear el archivo que un desarrollador está modificando y desbloquear cuando deje de usarlo Realizar una copia del archivo a cada desarrollador, fusionar sus copias una vez actualicen las versiones Ambas soluciones (lo que podemos hacer con Tortoise SVN) http://tortoisesvn.net/docs/release/TortoiseSVN_es/tsvn-basics-svn.html