Actividad grupal: *pull requests* en GitHub

Objetivos

En esta actividad aprenderéis la funcionalidad de *pull requests* de GitHub y el manejo de repositorios remotos. El GitHub *workflow* está cada vez más extendido en entornos empresariales, por lo que estar familiarizado con él es muy ventajoso a la hora de enfrentarse a un nuevo proyecto.

Pautas de elaboración

El objetivo es aplicar el flujo de GitHub sobre el código del caso práctico. Uno de vosotros actuará como administrador del repositorio principal, mientras que el resto actuará como desarrolladores individuales que deben implementar una funcionalidad diferente.

Todos debéis tener cuentas personales en [GitHub;](https://www.github.com.) si aún no la tenéis, daos de alta. También debéis tener acceso a una máquina con Git. Seguid las instrucciones del caso práctico del tema 3, «Flujos de trabajo con Git», para instalarlo. Deberán decidid quién va a actuar como administrador del repositorio. Este último debe inicializar un repositorio que actuará como principal:

* Crea un repositorio nuevo en GitHub. **Debe estar** **marcado como público.** Puedes llamarlo practica-unir-eiec, por ejemplo.
* Clona el repositorio en local con git clone <url>.
* Descarga el código desde la página de [Github](https://github.com/srayuso/unir-git) como un ZIP. Copia los archivos del ZIP a la carpeta del repositorio.
* Ejecuta git add . (cuidado con el punto), git commit -m "primer commit" y git push origin master.
* También puedes clonar el repositorio unir-git directamente y añadir tu repositorio como remoto. El objetivo es que tu repositorio no sea un *fork* del usado en el caso práctico para no despistar durante la creación de *pull requests.*

Una vez inicializado el repositorio, el administrador compartirá la URL con el resto del grupo. No debe dar acceso de escritura a nadie. El resto del grupo deberá seguir los pasos del caso práctico:

* Crear un *fork* a partir del repositorio del administrador.
* Una vez clonado en local, crear una rama nueva, implementar la funcionalidad, *commit.* Al clonar en local, comprobad que la carpeta contenga los archivos correctos (el README.md, el fichero Python, etcétera).
* Subir la rama a su *fork* y crear la *pull request.*

Podéis implementar las funcionalidades que queráis. Aquí tenéis algunas ideas:

* Actualizar el README.md con una ejecución de prueba para demostrar cómo se usa la aplicación.
* Traducir los mensajes que se imprimen por pantalla al inglés.
* Aceptar un nuevo parámetro de línea de comandos que indique si el orden debe ser ascendente o descendente.
* Aceptar un nuevo parámetro de línea de comandos que indique si es necesario eliminar palabras duplicadas de la lista antes de imprimirlas (en este caso habrá que modificar la función sort\_list).
* Añadir una directiva en el Makefile que ejecute el comando en local, sin arrancar un contenedor de Docker.
* Añadir un fichero de ejemplo, palabras.txt, que incluya varias palabras, una por línea, desordenadas, y modificar el Makefile para que use el nombre del fichero como parámetro de línea de comandos.

El objetivo de esta práctica no es aprender a escribir Python ni Makefiles, por lo que no os preocupéis si el código no funciona perfectamente. La idea es que trabajéis todos en paralelo sobre el mismo repositorio. Algunos editaréis los mismos ficheros, pero lo estaréis haciendo en repositorios diferentes (tanto el local como cada *fork* son independientes del repositorio principal).

El administrador deberá implementar también una funcionalidad y abrir una *pull request* contra su propio repositorio. **Recuerda crear una nueva rama antes de añadir tus cambios.** De lo contrario, no podrás abrir una *pull request.* Imagina que añades los cambios a master: cuando subas master a tu repositorio con git push origin master, ya estarás integrando tus cambios en esa rama. No tiene sentido abrir una *pull request* de master contra master.

Todos debéis iniciar vuestro *fork* a la vez para partir del mismo código y facilitar la revisión de *pull requests.* Cuando estas estén abiertas, entre todos debéis acceder a ellas. Comprobad el código de vuestros compañeros.

Una vez revisado el código de todas las *pull requests,* el administrador podrá empezar a fusionarlas. Si hay algunas que hayan modificado las mismas líneas de un fichero, es posible que solo se pueda fusionar una de ellas. Al intentar fusionar la siguiente, GitHub indicará que hay conflictos y no ofrecerá la opción de fusionar. No pasa nada, fusionad las que podáis, la resolución de conflictos no entra en el alcance de esta práctica.

Aunque solo uno actúe como administrador, ayudadle en los primeros pasos para inicializar el repositorio. Durante la implementación de funcionalidades nuevas, ayudaos entre vosotros si tenéis dudas en el código Python, en cómo ampliar el Makefile, etcétera. Aunque no sea el objetivo de la práctica, trabajar sobre código que funciona es siempre más gratificante.

En cuanto a la organización y gestión de equipos, en el foro «Pregúntale al profesor» de la asignatura encontrarás un nuevo tema específico para la organización de equipos donde el profesor explicará todos los detalles.

Una vez cerrado el equipo de trabajo os podéis poner en contacto a través de vuestras cuentas @comunidadunir.net y comenzar a trabajar. Puedes ampliar la información sobre el trabajo en equipo, consultando los [Tutoriales de trabajo en grupo.](http://tv.unir.net/secciones/3967/4883/0/0/0/0/)

Aquellos estudiantes que no comiencen su trabajo dentro de los siete primeros días, contados a partir del día de inicio de la actividad, quedarán excluidos de la actividad sin poder tomar parte en ella. Se trata de una actividad colaborativa, por lo que unos estudiantes no pueden beneficiarse del trabajo que hayan realizado sus compañeros.

Al finalizar el trabajo, **todos los miembros del equipo entregarán la misma actividad** a través del apartado Envío de actividades del aula virtual. El documento debe ir nombrado de la siguiente manera:

**APELLIDO1\_APELLIDO2\_NOMBRE\_Titulo\_actividad (sin tildes ni apóstrofes ni ningún otro carácter que pudiera resultar conflictivo).**

Indica en la actividad el nombre de todos los componentes del equipo y cumplimenta la siguiente tabla de valoración individual:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sí** | **No** | **A veces** |
| **Todos los miembros se han integrado al trabajo del grupo** |  |  |  |
| **Todos los miembros participan activamente** |  |  |  |
| **Todos los miembros respetan otras ideas aportadas** |  |  |  |
| **Todos los miembros participan en la elaboración del informe** |  |  |  |
| **Me he preocupado por realizar un trabajo cooperativo con mis compañeros** |  |  |  |
| **Señala si consideras que algún aspecto del trabajo en grupo no ha sido adecuado** |  |  |  |

Extensión y formato

No es necesario entregar el código, pero sí incluir capturas de pantalla de:

* El repositorio principal en GitHub.
* El *fork* de cada desarrollador en GitHub.
* El comando git push origin <rama> de cada desarrollador (la captura debe ser de la ventana de la terminal).
* Cada *pull request,* una vez abierta. Debe aparecer el nombre de la rama y el *fork* desde el que se ha abierto. La captura tiene que incluir la pestaña de *Files changed.*
* Lista de *pull requests* en el repositorio principal.
* El árbol de *commits* del repositorio principal tras fusionar todas las *pull requests posibles.* Para ello, acceded a la ruta /commits/master del repositorio (por ejemplo, (https://github.com/<id-usuario-admin>/practica-unir-eiec/commits/master). Deben aparecer los *commits* de las *pull requests* que se han fusionado y los autores.

Rúbrica

|  |  |  |  |
| --- | --- | --- | --- |
| Actividad grupal: *pull requests* en GitHub | Descripción | Puntuación máxima  (puntos) | Peso  % |
| Criterio 1 | El administrador ha conseguido crear un repositorio principal. | 3 | 30% |
| Criterio 2 | Cada desarrollador ha creado su *fork* y lo ha clonado en su máquina. | 3 | 30% |
| Criterio 3 | Cada desarrollador ha subido su rama y ha abierto una *pull request* contra el repositorio principal. | 4 | 40% |
|  |  | **10** | **100 %** |