UF4: CONTROL DE VERSIONS I DOCUMENTACIÓ

**CAS PRÀCTIC 3 – METODOLOGIA GITFLOW**

**NOM DE L’ALUMNE/s: Elyass el Jerari**

**OBJECTIU**

* **Treballar de forma introductòria amb la metodologia *GitFlow***

**INSTRUCCIONS**

* **L’activitat és realitza de manera individual però té una part que caldrà realitzar amb una parella, això si, cadascú haurà d’entregar la seva part.**
* **Si algú ho vol fer sol, simplement s’haurà de crear un segon usuari a GitHub per realitzar la part de la ‘parella’.**
* **Cal utilitzar les màquines de l’anterior UF o que compleixin els mínims necessaris.**
* **Recordeu que a la segona entrega el màxim que es pot aconseguir és un 7,5.**
* **Us aconsello fer un *snapshot* com a punt de partida del cas pràctic i també al final.**
* **Per defecte, cal que justifiqueu les respostes amb captures de pantalla.**
* **Si a la captura no hi ha cap valor que la identifiqui de forma única, cal que es vegi el fons d’escriptori, *notepad* o eines similars amb el vostre nomcognom!**
* **Totes les captures que mostrin les comandes han d’incloure, a part del resultat, la comanda i/o els paràmetres, per tal de veure com la feu.**
* **En apartats on hi hagi un alt volum de comandes, si n’hi ha més de les que es demanen, cal ressaltar-les o ‘borrar’ les que no tenen vinculació amb l’apartat o son errònies.**
* **El fet de no fer-ho pot anul·lar l’apartat parcial o completament.**

[*Gitflow*](https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow) (manera simplificada de *Git Workflow*) és una metodologia de gestió de branques que proporciona una estructura detallada per organitzar el desenvolupament de software en diferents etapes clarament definides.

El model està dissenyat per mantenir un flux de treball estructurat i mantenible, especialment en projectes de software complexos amb diverses funcionalitats i equips de desenvolupament.

*Gitflow* defineix diverses branques principals, cadascuna amb una finalitat específica, i estableix regles clares sobre com i quan fusionar les branques entre elles. Les branques principals són:

* *Master*: La branca "master" és la branca principal del projecte i conté sempre la versió més estable i desplegable del codi. Només s'hi accedeix per a integracions finals i desplegaments a producció.
* *Develop*: La branca "develop" és la branca de desenvolupament principal, on es fusionen totes les funcionalitats en curs de desenvolupament. És una branca compartida i sempre ha d'estar en un estat funcional, és a dir, que està prou estabilitzada i depurada per permetre el desenvolupament i la col·laboració sense interrupcions importants.

A més de les branques principals, *Gitflow* també inclou branques de suport per a desenvolupaments específics, com:

* *Feature Branches:* Les branques de funcionalitat són branques temporals creades per desenvolupar funcionalitats o característiques específiques del projecte. Sorgeixen de la branca "develop" i s'integren de nou en aquesta una vegada finalitzades.
* *Release Branches*: Les branques de release són utilitzades per preparar i realitzar proves finals en vista d'una nova versió del software. Es creen a partir de la branca "develop" i s'integren en les branques "master" i "develop" una vegada finalitzades.
* *Hotfix Branches*: Les branques de hotfix es fan servir per corregir errors crítics en producció. Sorgeixen de la branca "master", es fusionen amb "master" i "develop" un cop aplicades les correccions.

Amb *Gitflow*, cada tipus de branca té una finalitat específica i un cicle de vida ben definit, el que facilita la gestió dels canvis i la col·laboració en equips de desenvolupament. Utilitzar aquest model en la pràctica us proporcionarà una comprensió més profunda del flux de treball de Git i les seves aplicacions en el desenvolupament de software.



La comanda *git* es pot acompanyar pel paràmetre *flow* per tal d’utilitzar aquesta metodologia, però en aquesta pràctica d’iniciació queda totalment prohibit ja que es busca comprendre les diferents parts d’aquesta metodologia.

Per treballar amb aquesta metodologia doncs, el primer que farem és agafar un repositori públic, en aquest cas, <https://github.com/dbancells/laundryDAWUF4>, que prèviament jo també he replicat d’un altre usuari, i “copiar-lo” al vostre compte de GitHub.

En aquest cas però ho farem aprofitant les característiques de GitHub i no farem un *git clone* i posteriorment un *git remote/push*, sinó que directament farem un *fork*.

Fer un *fork* implica crear una còpia completa d'un repositori de Git, incloent-hi la història dels *commits*, les branques i altres metadades, però ho fas en un compte de Git separat. Aquesta còpia és independent del repositori original, i es manté en un compte de Git diferent, tot i que pot tenir un origen comú.

Per tant, amb el vostre usuari de GitHub utilitzat en activitats anteriors, feu un *fork* del projecte indicat. Podeu mantenir-li el nom.

1. Un cop fet, adjunteu una captura des de GitHub conforme es veu el vostre usuari i nom del projecte i conforme consta ‘*forked from dbancells/laundryDAWUF4*’. *0,5 punts.*

Captura de pantalla de un celular

Descripción generada automáticamente

Tal com veieu, aquest projecte ve amb una única branca (*master*) i ho volem ‘moure’ a metodologia *GitFlow*, de forma que caldrà crear una nova branca *develop* amb el mateix contingut que a *master*. Penseu que ho haureu de baixar a local, fer el procés de crear la branca de *develop* igual que *master* i després pujar-ho al vostre repositori remot.

1. Adjunteu la/les captura/es de tot el procés amb comandes de *git*. Vigileu a no deixar-vos cap part. *1 punt*.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Per començar a testejar la metodologia *Gitflow*, caldrà que imaginem que hem trobat un *bug* a la branca en producció i per tant haurem d’aplicar un *hotfix*.

Caldrà doncs que creeu una nova branca que es digui ‘*hotfix/bug1*’, on feu una modificació al fitxer index.php (afegiu un comentari per exemple) i després caldrà que apliqueu la metodologia de *Gitflow* per aplicar el canvi a les corresponents branques i feu el corresponent esborrat. Penseu a més que aquestes han de quedar actualitzades al vostre repositori en remot.

1. Adjunteu la/les captura/es de tot el procés amb comandes de *git*. Vigileu a no deixar-vos cap part. *1 punt*.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente



Texto

Descripción generada automáticamente



Si seguim amb la metodologia *GitFlow*, ens cal ‘bloquejar’ la branca *master*. El que volem dir és que cal configurar una restricció que impedeixi fer *commits* directament a la branca *master*.

Això es pot aconseguir mitjançant funcionalitats proporcionades per les plataformes de gestió de repositoris com GitHub o GitLab, que ofereixen opcions per protegir branques específiques amb polítiques de control d'accés.

La manera més típica de fer-ho és a través de *Pull Requests (PR)*. Aquesta és una sol·licitud per fusionar canvis d'una branca de desenvolupament a una branca principal en un repositori Git. Serveix com a punt de discussió i revisió, permetent als membres de l'equip avaluar els canvis, fer comentaris i sol·licitar modificacions abans de la seva fusió. És una pràctica comuna en el desenvolupament col·laboratiu de programari, facilitant el control de qualitat i la col·laboració efectiva en projectes de codi obert o en equip.

En aquest cas doncs, tocarà fer-ho dins l’entorn de GitHub. Caldrà que apliqueu regles sobre la branca *master* per tal que només s’hi puguin afegir els canvis via PR però aquests no hagin de ser aprovats.

1. Adjunteu la captura dins de GitHub que demostra aquesta protecció i a quin repositori i branca s’aplica. *0,5 punts*.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

Un cop fet, és el moment de comprovar-ho amb fets. Per tant, caldrà que creeu una nova branca, anomenada *featureXXX*, on XXX son les vostres inicials, que sortirà de *develop*. Aquí caldrà que modifiqueu el contingut de *README.md* i feu el corresponent *commit*. Aleshores caldrà fer el *merge* corresponent amb la branca de *develop* i penjar la branca de *develop* al repositori remot dins de GitHub.

1. Adjunteu una captura amb les comandes de git corresponents a tot el procés indicat. *0,5 punts.*

Texto

Descripción generada automáticamente



El pas de penjar la branca de *develop* al podria ser prescindible en situacions reals però el necessitem per poder fer la comprovació dels següents apartats.

Ens faltarà fer el *merge* del *develop* al *master* i penjar-ho al remot.

1. Adjunteu la captura amb les corresponents comandes de *git. 0,5 punts*.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Si us hi fixeu, us ha sortit un missatge conforme cal crear un *Pull Request*. No obstant, si aneu al repositori de GitHub veureu que el canvi s’ha penjat. Ho podeu comparar amb la branca de *develop* i veureu que son iguals.

Això és perquè per defecte dins GitHub els administradors (com és el nostre cas) es poden saltar les normes. Així doncs, caldrà que editeu la regla anterior sobre la branca *master* per tal de fer que els administradors no es puguin saltar les regles.

Un cop fet, feu un nou canvi a *develop* (aquest cop ens saltarem el *GitFlow* i no farem una branca *feature)*, feu el corresponent *merge* a *master*, i pengeu la branca *develop* i *master* al repositori de GitHub.

1. Adjunteu les captures amb les corresponents comandes de *git. 0,5 punts.*

Texto

Descripción generada automáticamente



Si us hi fixeu, novament per *develop* no ha tingut problemes per penjar la branca però en canvi ara per la de *master* us ha saltar un error i us obliga a fer un *pull request*.

Això s’haurà de fer directament des de l’entorn web de *GitHub*. Si aneu a l’apartat de *Pull Request* caldrà crear-ne un de nou.

Seleccioneu la branca *master* del vostre repositori i la de *develop* i automàticament us marcarà una diferència per poder fer el PR. Creeu-lo i un cop dins veureu que ell mateix detecta si es pot fer *merge* o no.

1. Adjunteu una captura del PR conforme us diu que no té conflictes, i una altra un cop s’ha fet correctament. *0,5 punts.*

Captura de pantalla de un celular

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Ara és el moment de fer una simulació més real, on més d’un programador treballarà amb el mateix projecte. Per això caldrà que afegiu un company a vostre projecte com a col·laborador i vosaltres us haureu d’afegir al projecte d’un company (pot ser el mateix company o un de diferent) com a col·laborador.

La idea és que tots dos treballeu amb una branca de desenvolupament que es dirà XXX-YYY, on XXX son les vostres inicials al projecte que sou el propietari, i YYY les inicials quan esteu com a col·laborador.

El primer que caldrà fer doncs serà que el col·laborador sincronitzi el projecte a un nou directori del seu espai de treball local.

Posteriorment, l’encarregat de crear la branca és el propietari del projecte, de forma que l’ha de crear i sincronitzar al repositori de GitHub.com.

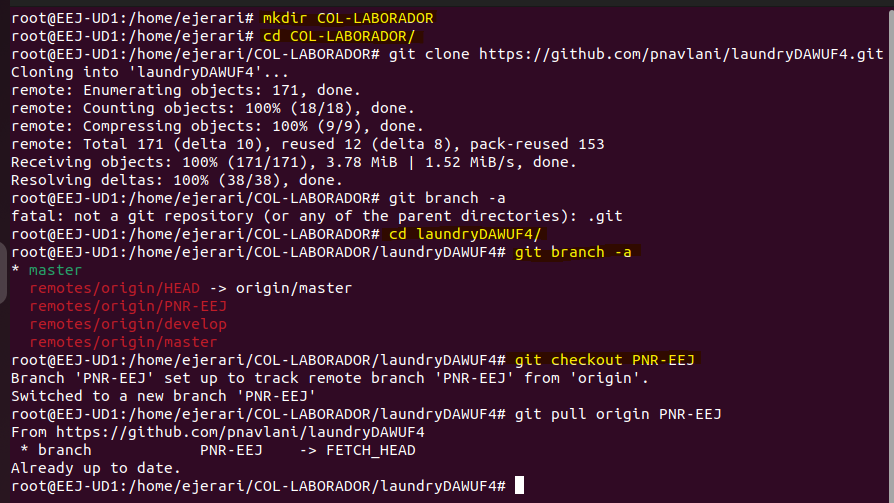
1. Adjunteu com a propietari les comandes corresponents a la premissa anterior. *0,5 punts*.

Texto

Descripción generada automáticamente

Ara, actuant com a col·laborador, llanceu les corresponents comandes de *git* per tal de veure aquesta nova branca i sincronitzar-la en local.

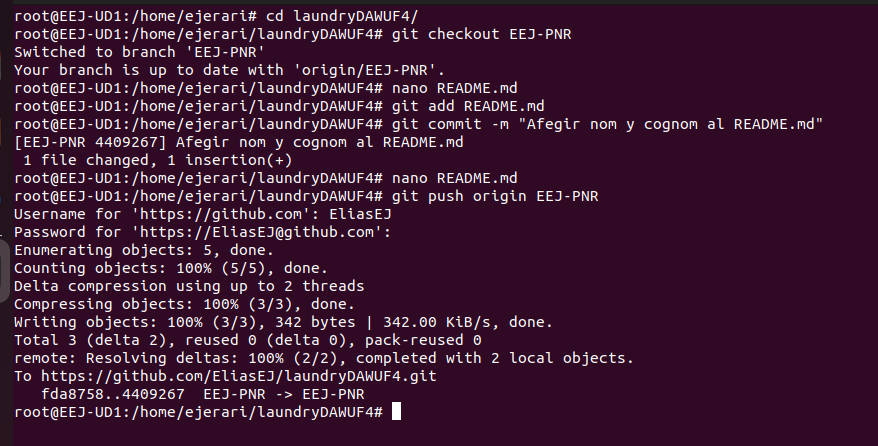
1. Adjunteu la captura que demostra les premisses anteriors. *0,5 punts.*



Arribats a aquest punt tots dos teniu al vostre repositori local la branca XXX-YYY i per tant podeu fer canvis al fitxer README.md en paral·lel.

Així doncs, afegiu una línia amb el vostre nom i cognom i el rol que teniu en aquesta acció (col·laborador o propietari). Feu el corresponent commit com a propietari i pengeu-lo al repositori remot de GitHub. Veureu com no us donarà problemes.

1. Adjunteu la captura amb les corresponents comandes de *git. 0,5 punts.*





Com a col·laborador feu el corresponent canvi a README.md i pengeu el *commit* a GitHub. Us ha de retornar un error ja que el repositori local és diferent del remot.

1. Adjunteu la captura amb les corresponents comandes de *git* i l’error corresponent. *0,5 punts.*

Texto

Descripción generada automáticamente



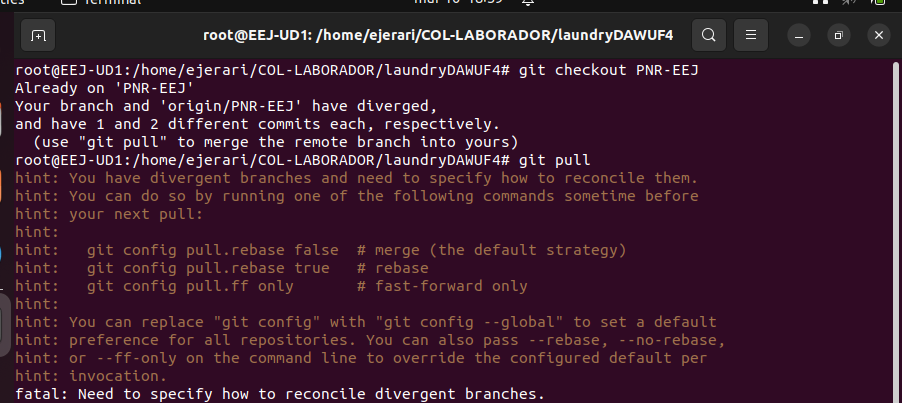
Texto

Descripción generada automáticamente



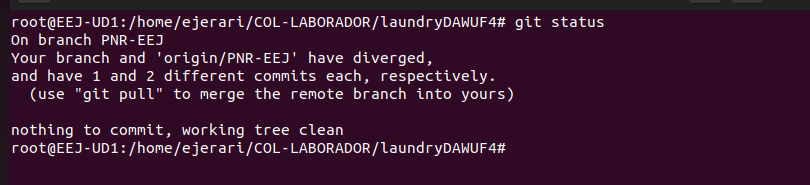
Ara us caldrà tornar a sincronitzar el repositori local amb les novetats que hi ha al remot que ha fet el propietari. Feu-ho i a més intenteu de fer el *merge*.

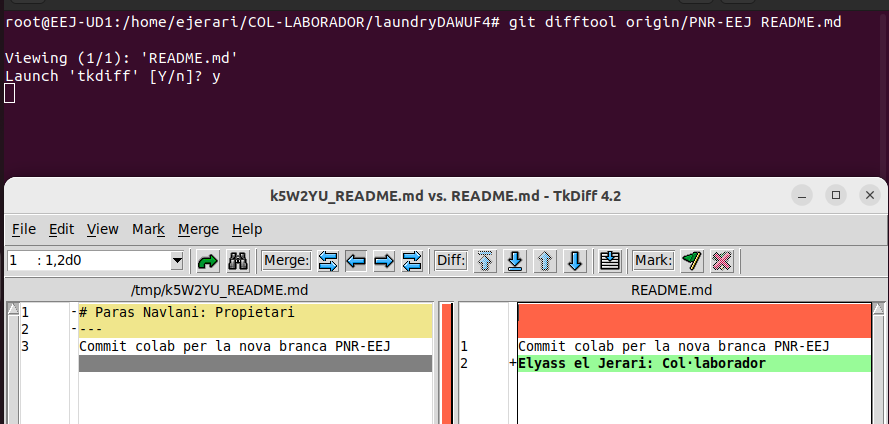
1. Adjunteu les captures corresponents amb les seves comandes de *git* que al final us ha de mostrar conforme hi ha un conflicte que no es pot solucionar automàticament. *0,5 punts.*



Com a col·laborador, llanceu un *git status* que us marcarà allà on hi ha divergències. També obriu el fitxer README.md i veureu com s’indiquen les diferències existents.

1. Adjunteu la captura corresponent al git status i al contingut de README.md on es veuen les diferències. *0,5 punts.*





Feu els canvis manualment per deixar correcte el fitxer README.md i pengeu els canvis a GitHub actuant com a col·laborador.

1. Adjunteu la captura corresponent per tal de penjar el repositori en remot. *0,5 punts.*

Com a propietari, aneu al web de GitHub i feu un cop d’ull a Activity. Allà veureu com hi ha el commit amb l’usuari col·laborador i el propietari.

1. Adjunteu la captura que ho demostra. *0,5 punts.*

Texto

Descripción generada automáticamente

Com a propietari però encara no tenim sincronitzada la branca local XXX-YYY tal com està en remot, pel que caldrà fer-ho.

Posteriorment, llanceu un *git blame* del fitxer README.md i feu un grep ‘2024’ i així es veuran les canvis sobre el fitxer i qui n’és el responsable.

1. Adjunteu la corresponent captura. *0,5 punts.*

Per acabar, com a propietari, i directament des de GitHub, llanceu una *release* corresponent a la versió beta que es digui XXX-CP3-UF4 on XXX son les vostres inicials.

1. Adjunteu la corresponent captura des de GitHub.com. *0,5 punts.*

Captura de pantalla de un celular

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente