**RECURSOS**

**APUNTS MÒDUL ENTORNS DE DESENVOLUPAMENT**

Git i GitHub

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**1. Git i GitHub**

**1.1 Què és Git?**

Git és un sistema de control de versions que permet realitzar el seguiment i gestionar els canvis realitzats en el codi font d’un projecte al llarg del temps. És una eina essencial per al desenvolupament de programari, especialment en projectes col·laboratius, perquè permet:

* Desar cada versió del codi: Els desenvolupadors poden tornar a versions anteriors si es detecten errors.
* Facilitar el treball en equip: Cada membre pot treballar en una còpia independent, i després unir els canvis al projecte principal.

Exemple: Imagina un document de text en el qual diferents persones han de treballar. Si no existeix un control de versions, qualsevol canvi que es faci es sobreescriu. Amb Git, en canvi, podem "guardar" versions en diferents moments, de manera que qualsevol canvi queda registrat i es pot recuperar en qualsevol moment.

**1.2 Què és GitHub?**

GitHub és una plataforma en línia que permet emmagatzemar els repositoris Git al núvol i proporciona eines addicionals per facilitar la col·laboració entre desenvolupadors. És especialment útil per compartir codi i treballar en equip.

A GitHub, els repositoris es poden compartir amb altres usuaris, assignar permisos, obrir issues per gestionar problemes o millores, i realitzar pull requests per sol·licitar que un conjunt de canvis s'integri en el projecte principal.

Exemple: Si estem treballant en un projecte de desenvolupament d'una aplicació web, podem fer un repositori a GitHub, i cada membre de l’equip podrà veure el codi, fer canvis, i proposar millores. Això permet tenir sempre una versió actualitzada accessible per a tots els membres.

**1.3 Conceptes Bàsics de Git**

* Repositori: És el lloc on es guarda el projecte. Pot ser un repositori local (a l'ordinador) o un repositori remot (per exemple, a GitHub).
* Commit: És una acció que desarà els canvis amb un missatge descriptiu, creant una nova versió del projecte.
* Branca (branch): Una branca és una còpia del codi on es poden fer modificacions sense afectar la branca principal, sovint anomenada main o master. Això permet desenvolupar noves característiques sense interferir en el codi estable.
* Merge: És el procés de combinar els canvis d’una branca amb una altra.
* Pull request: És una petició per integrar els canvis d'una branca (normalment, la de desenvolupament) amb la branca principal.

Exemple de Conceptes:

* Repositori: Crea una carpeta per a un projecte i converteix-la en un repositori amb git init.
* Commit: Quan afegeixes un fitxer nou (per exemple, README.md), fas git add README.md i després git commit -m "Afegir el fitxer README".
* Branca: Pots crear una nova branca amb git branch nom-branca i canviar-hi amb git checkout nom-branca per treballar en una característica nova.

**1.4 Exemple Pràctic: Crear un Projecte amb Git i GitHub**

**1.4.1 Preparar un repositori local**

Crea una carpeta nova per al projecte:

mkdir projecte-exemple

cd projecte-exemple

Inicialitza un repositori Git:

git init

Aquest comandament crea una carpeta oculta .git on es desarà tota la informació de les versions del projecte.

**1.4.2 Primer Commit**

Crea un fitxer README.md amb una descripció del projecte.

Afegeix aquest fitxer a la “zona d'indexació”:

git add README.md

Fes el primer commit per guardar aquest canvi:

git commit -m "Primer commit: afegir README"

**1.4.3 Crear un repositori remot a GitHub**

Ves a GitHub.com i crea un nou repositori (sense afegir-hi cap fitxer inicial).

Afegeix l’enllaç del repositori remot al teu repositori local:

git remote add origin https://github.com/usuari/projecte-exemple.git

**1.4.4 Pujar els canvis al repositori remot**

Pujar el commit al repositori remot:

git push -u origin main

Ara el projecte es troba tant al teu ordinador com a GitHub, on pots compartir-lo amb altres desenvolupadors.

**2. Configuració inicial a Visual Studio Code**

Visual Studio Code és un entorn de desenvolupament integrat (IDE) lleuger i extensible que facilita la gestió de projectes amb Git. Mitjançant les extensions, és possible gestionar repositoris Git, crear commits, veure l’historial de canvis, i treballar amb GitHub sense sortir de l’editor.

**2.1 Instal·lació de Git**

* Descarregar Git: Abans de configurar Git a Visual Studio Code, és necessari instal·lar Git al sistema. La descàrrega es pot fer des de [la pàgina oficial de Git](https://git-scm.com/downloads), on es pot escollir la versió corresponent per a cada sistema operatiu (Windows, macOS o Linux).
* Instal·lar Git: Segueix els passos de la instal·lació. A Windows, assegura’t de marcar l’opció de “Add Git to PATH” per poder executar Git des de qualsevol terminal.
* Verificar la instal·lació: Després de la instal·lació, obre una terminal i executa:

git --version

Si s’ha instal·lat correctament, apareixerà el número de versió de Git.

**2.2 Configuració de Git al sistema**

* Configurar el nom d’usuari i l’adreça de correu: Aquestes dades s’associen a cada commit que es realitzi.

git config --global user.name "Nom de l'Alumne"

git config --global user.email "correu@exemple.com"

* Verificar la configuració: Per comprovar la configuració de Git, utilitza:

git config --list

**2.3 Configuració de Visual Studio Code per a treballar amb Git**

* Integra Visual Studio Code amb Git: Si Git està instal·lat correctament, Visual Studio Code el detectarà automàticament.
* Obrir la terminal integrada: Pots utilitzar la terminal integrada de Visual Studio Code amb la combinació de tecles Ctrl + ~ (o Cmd + ~ a macOS) per executar els comandaments de Git sense sortir de l’editor.

**2.4 Extensions recomanades per a Git i GitHub a Visual Studio Code**

Les extensions són essencials per a gestionar els fluxos de treball amb Git i GitHub directament des de Visual Studio Code.

* GitLens:
  + Aquesta extensió ofereix funcions avançades per visualitzar l'historial de canvis, la responsabilitat de cada línia de codi i els commits relacionats amb cada fitxer.
  + Funcions destacades:
    - Visualitzar l’autor de cada línia de codi (qui ha fet el canvi i quan).
    - Revisar l’historial de canvis d’un fitxer específic.
    - Comparar el codi actual amb versions anteriors.
* GitHub Pull Requests and Issues:
  + Aquesta extensió permet gestionar les pull requests i issues de GitHub directament des de Visual Studio Code, facilitant el treball col·laboratiu.
  + Funcions destacades:
    - Crear i revisar pull requests.
    - Gestionar issues (problemes o millores).
    - Visualitzar notificacions i activitats relacionades amb GitHub.

**2.5 Configuració de les Extensions**

* Instal·lació de les extensions:
  1. Ves al menú d’extensions a Visual Studio Code (Ctrl + Shift + X o Cmd + Shift + X a macOS).
  2. Cerca GitLens i fes clic a “Instal·lar”.
  3. Repeteix el procés amb GitHub Pull Requests and Issues.
* Configuració de GitLens:
  1. GitLens es configura automàticament, però es pot ajustar per adaptar-lo a les preferències de cada alumne.
  2. Opcionalment, poden activar l’opció de mostrar detalls addicionals a la barra de l’editor (com ara el temps de cada canvi) per tenir més informació sobre l’historial de canvis.
* Configuració de GitHub Pull Requests and Issues:
  1. Una vegada instal·lada, l’extensió demanarà permisos per accedir al compte de GitHub. Segueix els passos d’autenticació.
  2. Configura la pestanya d’issues per veure els problemes assignats i les notificacions importants del repositori.

**2.6 Exemple pràctic: Configurar Git i GitHub a Visual Studio Code**

Exemple de configuració:

1. Configurar Git al sistema:
   * Defineix el nom i el correu, tal com es mostra a l’apartat 2.2.
2. Obrir un projecte amb control de versions Git:
   * Crea una carpeta nova per al projecte i inicialitza Git amb:

git init

* + Crea un fitxer nou, com ara README.md, i afegeix una descripció del projecte.
  + Afegeix i fes el primer commit:

git add README.md

git commit -m "Primer commit amb README"

1. Connectar el projecte a GitHub:
   * Crea un repositori nou a GitHub i afegeix-lo com a repositori remot:

git remote add origin https://github.com/nomusuari/nomrepositori.git

* + Puja el commit al repositori remot:

git push -u origin main

1. Utilitzar GitLens:
   * Obre el fitxer README.md i observa com GitLens mostra qui ha realitzat l'últim canvi i quan.
   * Revisa l’historial de commits del fitxer fent clic dret al fitxer i seleccionant “Open File History” (Historial del fitxer).
2. Crear una issue amb GitHub Pull Requests and Issues:
   * A Visual Studio Code, obre la pestanya d’issues (icono de GitHub).
   * Crea una nova issue descrivint una característica o millora per al projecte. Això permet als alumnes practicar la gestió de tasques.

**3. Configuració de Git a Visual Studio Code**

Un cop instal·lat Git i les extensions a Visual Studio Code, el següent pas és configurar l’entorn per treballar amb control de versions de manera eficient. Aquesta configuració ens permetrà fer commits, gestionar branques i pujar canvis a GitHub directament des de Visual Studio Code.

**3.1 Configuració de Git**

Visual Studio Code detecta automàticament la instal·lació de Git. Si Git està correctament instal·lat i configurat al sistema, l’editor mostrarà les opcions bàsiques de control de versions.

Verificar la integració de Git:

Obre qualsevol carpeta amb un projecte o crea’n un de nou, i assegura’t que apareix la pestanya de Source Control (Control de Codi Font), representada per una icona de tres cercles connectats.

**3.2 Com fer el primer Commit a Visual Studio Code**

* Pas 1: Crea o obre un fitxer dins d’un projecte que tingui control de versions amb Git.
* Pas 2: Realitza canvis en el fitxer i des de la pestanya Source Control, Visual Studio Code et mostrarà els fitxers modificats. Clica al fitxer per veure els canvis.
* Pas 3: Afegeix el fitxer a l’index per fer el commit:
* A la pestanya de Source Control, passa el cursor per sobre del fitxer modificat i fes clic al símbol “+” per afegir-lo. Alternativament, fes clic a “+” al costat de “Changes” per afegir tots els fitxers modificats.
* Pas 4: Escriu un missatge de commit i fes clic a la icona de “Confirmar” (✓) o prem Ctrl + Enter (Windows/Linux) o Cmd + Enter (macOS).

Exemple de Commit: Suposem que has creat un fitxer README.md i has escrit una petita descripció del projecte. Afegeix el fitxer i fes el commit amb el missatge: “Primer commit: Afegir descripció del projecte”.

**3.3 Gestió de branques a Visual Studio Code**

Les branques permeten treballar en diferents parts del projecte sense afectar el codi principal (normalment a la branca main o master).

Crear una branca nova:

1. A la part inferior esquerra de Visual Studio Code, fes clic al nom de la branca actual (per exemple, main).
2. Selecciona “Create New Branch...” i escriu el nom de la nova branca (per exemple, feature-nova-funcio).
3. Canviar de branca:
4. Clica al nom de la branca actual i selecciona la branca a la qual vols canviar.
5. Fusionar branques (Merge):
6. Canvia a la branca principal (main).
7. A la pestanya de Source Control, selecciona “Merge Branch...” i escull la branca que vols fusionar amb main.

Exemple de Branca: Suposem que vols afegir una nova funció. Crea una branca feature-nova-funcio per treballar-hi sense modificar el codi de la branca principal. Un cop acabada la funció, fes un commit a feature-nova-funcio i fusiona-la amb main.

**3.4 Enviar els canvis a GitHub (Push)**

Pas 1: Connectar el repositori local amb GitHub:

Si el repositori no té un enllaç remot, pots afegir-lo obrint la terminal a Visual Studio Code (Ctrl + ~) i executant:

git remote add origin https://github.com/usuari/nomrepositori.git

Pas 2: Enviar els canvis (Push) a GitHub:

Després de fer un commit, fes clic a la pestanya Source Control i selecciona la icona “Sync Changes” o executa el comandament:

git push origin main

Això pujarà els canvis a GitHub.

Exemple de Push: Després de fer el commit “Afegir funció de càlcul”, selecciona “Sync Changes” per pujar el commit a GitHub, permetent que altres membres de l’equip el vegin.

**3.5 Exemple Pràctic: Treball complet amb Git a Visual Studio Code**

1. Iniciar un nou projecte amb control de versions:

Crea una carpeta nova i inicialitza Git:

git init

Crea un fitxer nou README.md amb una descripció del projecte.

2. Crear una branca i fer canvis:

Crea una nova branca per treballar en una característica (feature-afegir-funcio):

git branch feature-afegir-funcio

git checkout feature-afegir-funcio

Afegeix codi o informació al fitxer README.md i fes el commit:

git add README.md

git commit -m "Afegir informació sobre la nova funció"

3. Fusionar la branca amb main:

Torna a la branca principal:

git checkout main

Fusiona els canvis de feature-afegir-funcio a main:

git merge feature-afegir-funcio

4. Pujar els canvis a GitHub:

Puja els canvis de main a GitHub amb:

git push origin main

Resultat de l’Exemple: Ara tens un projecte amb un fitxer README.md, has creat una branca, hi has afegit informació, l’has fusionada amb la branca principal i has pujat els canvis a GitHub.

**3.6 Ús de GitLens per visualitzar l’historial i els canvis**

GitLens ajuda a tenir una vista completa de l’historial de commits i els canvis fets en cada línia d’un fitxer.

Visualitzar l’historial de canvis d’un fitxer:

A Visual Studio Code, obre el fitxer README.md.

A la barra superior, selecciona la pestanya de GitLens i fes clic a “File History” per veure l’historial de canvis del fitxer.

Visualitzar qui ha fet cada canvi:

Obre un fitxer i observa a l’esquerra cada línia de codi. GitLens mostra informació sobre qui va fer el canvi i quan.

Això és especialment útil per entendre qui ha fet cada contribució en un projecte col·laboratiu.

**4. Crear un repositori local**

Un repositori local és l'espai a l'ordinador on es guarda el projecte amb control de versions, permetent fer canvis, desar-los amb commits i, posteriorment, pujar-los a un repositori remot com GitHub. A continuació, es descriu com crear un repositori local, fer el primer commit, i gestionar els canvis.

**4.1 Iniciar un nou repositori amb Git**

Crear la carpeta del projecte:

Decideix on vols guardar el projecte al teu ordinador i crea-hi una carpeta nova. Per exemple, projecte-exemple.

mkdir projecte-exemple

cd projecte-exemple

Inicialitzar Git al projecte:

Un cop dins de la carpeta, inicialitza el repositori amb:

git init

Aquest comandament crea una carpeta .git oculta on es guarda tota la informació del repositori i els seus canvis. A partir d’aquí, qualsevol fitxer dins d’aquesta carpeta es pot afegir al control de versions.

Exemple: Suposem que estàs treballant en una aplicació de llista de tasques. Crees la carpeta projecte-llista-tasques i hi inicialitzes Git amb git init.

**4.2 Afegir fitxers al repositori**

Crear un fitxer inicial: Crea un fitxer al projecte, per exemple README.md, amb una petita descripció.

echo "# Projecte Llista de Tasques" > README.md

Afegir el fitxer al control de versions:

Per indicar a Git que vols fer el seguiment del fitxer README.md, utilitza git add:

git add README.md

Això afegeix el fitxer a l’"index" de Git, preparant-lo per al següent commit.

Exemple: Has creat un fitxer README.md amb el títol del projecte. Ara, el pots afegir amb git add README.md per començar-ne el seguiment.

**4.3 Fer el primer commit**

Un commit és com una fotografia de l'estat actual dels fitxers que afegeixes al control de versions. Cada commit té un missatge que descriu els canvis fets.

Crear un commit:

git commit -m "Primer commit: Afegir README amb la descripció del projecte"

Això desa l’estat actual del fitxer README.md al repositori local amb el missatge especificat.

Exemple: Després de crear el fitxer README.md i afegir-lo, fas el commit inicial amb git commit -m "Primer commit: Afegir README amb la descripció del projecte".

**4.4 Afegir i confirmar més fitxers**

Per a qualsevol canvi posterior en els fitxers, caldrà repetir els passos de add i commit.

Afegir més fitxers: Suposem que ara crees un fitxer nou index.py per començar a escriure el codi.

echo "print('Hola, món!')" > index.py

git add index.py

git commit -m "Afegir fitxer principal amb primer codi de l'aplicació"

Exemple: Ara has creat un fitxer index.py amb un simple print i l'has afegit amb git add index.py, seguit d'un commit amb git commit -m "Afegir fitxer principal amb primer codi de l'aplicació".

**4.5 Visualitzar l’historial de commits**

Per revisar l'historial de tots els commits realitzats al projecte, pots utilitzar el comandament:

git log

Això mostrarà una llista de tots els commits, amb el missatge, l'autor i la data.

Exemple: Després de fer diversos commits, executa git log per veure els diferents punts de la història del projecte. Podràs veure informació com “Primer commit: Afegir README amb la descripció del projecte” i “Afegir fitxer principal amb primer codi de l'aplicació”.

**4.6 Ignorar fitxers amb .gitignore**

Hi ha fitxers que no volem incloure al control de versions, com ara fitxers temporals o dades sensibles. Aquests fitxers es poden excloure amb un fitxer .gitignore.

Crear un fitxer .gitignore:

Crea el fitxer .gitignore i escriu-hi els noms o patrons dels fitxers que vols ignorar. Per exemple:

echo "\*.pyc" > .gitignore

echo "\_\_pycache\_\_/" >> .gitignore

git add .gitignore

git commit -m "Afegir .gitignore per ignorar fitxers temporals"

Exemple: En un projecte Python, és comú ignorar els fitxers .pyc i la carpeta \_\_pycache\_\_. Creant un .gitignore amb aquestes exclusions, Git no afegirà aquests fitxers al control de versions.

**4.7 Exemple complet: Crear i configurar un repositori local**

A continuació es resumeixen els passos per crear un repositori local amb Git i afegir-hi alguns fitxers:

Crear la carpeta del projecte:

mkdir projecte-exemple

cd projecte-exemple

Inicialitzar el repositori:

git init

Afegir el primer fitxer:

echo "# Projecte Exemple" > README.md

git add README.md

git commit -m "Primer commit: Afegir README amb el títol del projecte"

Afegir un segon fitxer:

echo "print('Hola, món!')" > main.py

git add main.py

git commit -m "Afegir el fitxer main.py amb el codi inicial"

Crear .gitignore per ignorar fitxers innecessaris:

echo "\*.log" > .gitignore

git add .gitignore

git commit -m "Afegir .gitignore per ignorar fitxers .log"

Resultat de l’Exemple: Ara tens un repositori local amb dos fitxers (README.md i main.py) i un fitxer .gitignore que exclou fitxers .log del control de versions. Aquests passos et permetran afegir i confirmar qualsevol canvi de manera ordenada.

**4.8 Consells per a un bon ús de Git**

Commit petits i freqüents: És millor fer commits sovint amb canvis petits. Això facilita el seguiment de cada canvi i simplifica la resolució de problemes.

Missatges clars i descriptius: Cada commit ha de tenir un missatge que descrigui clarament els canvis fets. Això és útil per recordar què s’ha fet en cada commit.

No afegir informació sensible: Evita incloure contrasenyes o dades personals en el repositori. Usa .gitignore per excloure aquests fitxers.

**5. Operacions bàsiques amb Git**

Les operacions bàsiques de Git inclouen afegir canvis, fer commits, treballar amb branques i fusionar canvis. Aquests conceptes són fonamentals per gestionar els projectes de manera eficient i organitzada.

**5.1 Afegir fitxers (git add)**

Quan es realitza un canvi en un fitxer, aquest no es desa automàticament en el control de versions. Cal afegir-lo primer a l'índex (o "stage") abans de fer un commit. Això es fa amb el comandament git add.

Afegir un fitxer específic:

git add nom\_fitxer

Afegir tots els fitxers modificats:

git add .

Exemple: Has modificat el fitxer index.py i creat un nou fitxer utils.py. Per afegir-los tots dos de cop:

git add .

**5.2 Fer un commit (git commit)**

Un commit desa els canvis afegits a l'índex amb un missatge que descriu què s'ha fet. Aquest és un pas crucial, ja que el commit actua com una "fotografia" del projecte en aquest moment.

Fer un commit amb un missatge:

git commit -m "Descripció del canvi"

Exemple de missatges de commit:

"Afegir funció de suma a utils.py"

"Corregir error en la funció de càlcul de descompte"

Exemple complet: Després d’editar index.py per afegir-hi una funció nova, pots afegir el fitxer i fer un commit:

git add index.py

git commit -m "Afegir funció de benvinguda a index.py"

**5.3 Consultar l'historial de commits (git log)**

Git log mostra l'historial de tots els commits fets fins al moment, ordenats cronològicament. Això permet veure quan i per què es van fer els canvis.

Veure l’historial complet:

git log

Veure un historial més compacte:

git log --oneline

Exemple: Després de fer diversos commits, executa git log --oneline per veure un resum dels canvis, amb només l’identificador del commit i el missatge.

**5.4 Treball amb branques (git branch, git checkout)**

Les branques permeten desenvolupar diferents funcionalitats o solucionar problemes sense afectar el codi de la branca principal (main). Un cop feta la modificació, es pot fusionar la branca amb la branca principal.

Crear una nova branca:

git branch nom\_brancha

Canviar a una altra branca:

git checkout nom\_brancha

Crear i canviar de branca al mateix temps:

git checkout -b nom\_brancha

Exemple: Suposem que vols afegir una nova funcionalitat al projecte. Crea una branca anomenada feature-nova-funcio i canvia a aquesta branca per treballar-hi:

git checkout -b feature-nova-funcio

**5.5 Fusionar branques (git merge)**

Quan finalitzes els canvis en una branca, és possible fusionar-la amb la branca principal per incorporar-hi les modificacions.

Pas 1: Canvia a la branca principal (main o master):

git checkout main

Pas 2: Fusiona la branca amb els canvis:

git merge nom\_brancha

Exemple: Després d’afegir la funcionalitat a feature-nova-funcio, fusiona aquesta branca amb main:

git checkout main

git merge feature-nova-funcio

**5.6 Resoldre conflictes de fusió**

Quan Git no pot combinar automàticament els canvis de dues branques diferents, es produeix un conflicte de fusió. Aquests conflictes són comuns quan dos canvis afecten la mateixa línia d'un fitxer.

Passos per resoldre conflictes:

1. Després de fer un git merge que genera conflictes, Git assenyala els fitxers amb conflictes.
2. Obre els fitxers en conflicte i revisa les parts marcades per Git (les trobaràs delimitades per <<<<<<<, =======, i >>>>>>>).
3. Selecciona quin codi vols mantenir i guarda els canvis.
4. Afegeix els fitxers resolts a l’índex amb git add.

Finalment, completa el merge fent un commit:

git commit -m "Resoldre conflictes en la fusió de feature-nova-funcio amb main"

Exemple de conflicte: Si a la branca main has canviat una línia a index.py i, a la vegada, la branca feature-nova-funcio té un canvi diferent a la mateixa línia, al fer la fusió apareixerà un conflicte. Revisa i tria el codi correcte, fes git add i després git commit.

**5.7 Eliminar branques (git branch -d)**

Una vegada s’ha fusionat una branca amb la branca principal, és habitual eliminar-la per mantenir l’organització del projecte.

Eliminar una branca:

git branch -d nom\_brancha

Exemple: Després de fusionar feature-nova-funcio amb main, pots eliminar la branca amb:

git branch -d feature-nova-funcio

**5.8 Exemple complet d'operacions bàsiques**

Imagina un projecte on has de fer el següent:

Crear un fitxer i fer el primer commit:

echo "# Projecte Exemple" > README.md

git add README.md

git commit -m "Primer commit: Afegir README amb el títol del projecte"

Crear una branca per afegir una nova funcionalitat:

git checkout -b feature-funcio1

echo "print('Funció 1')" > funcio1.py

git add funcio1.py

git commit -m "Afegir funció 1"

Tornar a la branca principal i fusionar els canvis:

git checkout main

git merge feature-funcio1

Eliminar la branca feature-funcio1 després de fusionar-la:

git branch -d feature-funcio1

Resultat final: Has creat una branca per desenvolupar una funcionalitat nova, l’has fusionada amb main i l’has eliminada després de completar la fusió.

**6. Connectar el repositori local a GitHub**

Connectar un repositori local amb GitHub permet desar el projecte en un repositori remot, facilitant la col·laboració i l’accés al codi des de qualsevol lloc. Quan el repositori està connectat amb GitHub, es poden pujar (push) els canvis locals i baixar (pull) els canvis realitzats per altres col·laboradors.

**6.1 Crear un repositori remot a GitHub**

Pas 1: Accedeix a GitHub.com i inicia sessió o crea un compte si no en tens.

Pas 2: Fes clic a “New” (Nou) per crear un nou repositori.

Pas 3: Omple la informació del repositori:

Nom: Escriu un nom per al repositori (per exemple, projecte-exemple).

Descripció: Opcionalment, afegeix una descripció breu del projecte.

Public o Private: Tria si vols que el repositori sigui públic o privat.

Pas 4: Crea el repositori fent clic a “Create repository”.

Nota: No afegeixis cap fitxer (README, .gitignore o llicència) des de GitHub si ja tens el repositori local creat i inicialitzat, perquè això pot causar conflictes en la sincronització inicial.

**6.2 Afegir l'enllaç remot al repositori local**

Una vegada creat el repositori a GitHub, GitHub et proporcionarà una URL per accedir-hi. Aquesta URL s'utilitzarà per connectar el repositori local amb el repositori remot.

Obrir el projecte a la terminal: Ves a la carpeta del projecte amb la terminal (o la terminal integrada de Visual Studio Code).

Afegir l'enllaç remot: Usa el següent comandament per vincular el repositori local amb GitHub:

git remote add origin https://github.com/nomusuari/projecte-exemple.git

Verificar la connexió: Pots comprovar que l’enllaç remot s’ha afegit correctament amb:

git remote -v

Exemple: Suposem que el teu nom d'usuari de GitHub és jordi123 i el nom del projecte és llista-tasques. Afegiries l’enllaç remot amb:

git remote add origin https://github.com/jordi123/llista-tasques.git

6.3 Pujar els canvis inicials a GitHub (git push)

Ara que el repositori local està vinculat amb el repositori remot a GitHub, pots pujar els canvis.

Pujar els canvis amb git push: El primer cop que pugis els canvis, usa el següent comandament per establir la branca principal (main):

git push -u origin main

A partir d’aquí, per a pujar els canvis futurs només hauràs de fer git push.

Exemple: Has fet alguns commits en el projecte localment. Executa git push -u origin main per pujar tots els canvis a GitHub i establir main com la branca de treball per defecte.

**6.4 Sincronitzar canvis entre el repositori local i GitHub**

Pujar nous canvis a GitHub: Cada cop que facis nous commits, pots pujar-los amb:

git push

Baixar canvis de GitHub (git pull): Si algú ha fet canvis en el repositori remot, pots baixar-los i sincronitzar-los amb el repositori local amb:

git pull

Exemple: Després de fer alguns canvis al projecte, fas git push per actualitzar el repositori remot a GitHub amb els canvis nous. Si un col·laborador afegeix una modificació, executa git pull per actualitzar el teu repositori local.

**6.5 Exemple complet: Connectar i sincronitzar un projecte amb GitHub**

Suposem que tens un projecte anomenat projecte-exemple al teu ordinador i vols pujar-lo a GitHub per primera vegada.

Crear el repositori remot a GitHub:

Ves a GitHub, crea un repositori nou anomenat projecte-exemple i copia la URL proporcionada.

Afegir l'enllaç remot al projecte local:

Obre la terminal al directori del projecte i afegeix l'enllaç remot:

git remote add origin https://github.com/nomusuari/projecte-exemple.git

Pujar el projecte a GitHub:

Puja els canvis a la branca main:

git push -u origin main

Afegir canvis i pujar-los:

Fes alguns canvis al projecte, com ara afegir una línia al README.md.

Afegeix i confirma els canvis:

git add README.md

git commit -m "Afegir nova línia a README"

Pujar els canvis:

git push

Resultat final: Ara tens un projecte completament sincronitzat amb GitHub. Qualsevol canvi que facis i desis amb git commit es podrà pujar amb git push, i qualsevol canvi remot es podrà baixar amb git pull.

**6.6 Clonar un repositori de GitHub (git clone)**

Quan vulguis baixar un projecte complet de GitHub a un altre ordinador o compartir-lo amb altres persones, es pot fer mitjançant git clone.

Clonar un repositori:

git clone https://github.com/nomusuari/projecte-exemple.git

Això crearà una nova carpeta amb tots els fitxers i l'historial del repositori.

Exemple: Si el teu col·laborador vol treballar en el projecte projecte-exemple, pot executar:

git clone https://github.com/nomusuari/projecte-exemple.git

Això crearà una còpia completa del repositori al seu ordinador.

**6.7 Consells per treballar amb repositoris remots**

Pujar canvis amb freqüència: Això ajuda a mantenir el projecte actualitzat i accessible per a tots els col·laboradors.

Sincronitzar abans de fer canvis importants: Executa git pull abans de fer canvis per assegurar-te que tens l’última versió del projecte.

Utilitzar missatges descriptius: Els missatges de commit han de descriure clarament el canvi fet, especialment en projectes col·laboratius.

**7. Treballar amb branques**

Les branques permeten treballar en diferents versions o funcionalitats d’un projecte de manera independent. Això facilita fer canvis sense afectar el codi principal, ja que es poden desenvolupar noves funcionalitats o solucionar errors en una branca separada. Un cop la branca estigui llesta, es pot fusionar amb la branca principal (main).

**7.1 Crear una nova branca (git branch)**

Una nova branca és una còpia del projecte en el moment en què es crea. Els canvis en aquesta branca no afectaran la branca principal fins que es faci una fusió.

Crear una branca nova:

git branch nom\_brancha

Crear i canviar automàticament a la nova branca:

git checkout -b nom\_brancha

Exemple: Si vols treballar en una nova funcionalitat per a l’aplicació, pots crear una branca feature-nova-funcio:

git checkout -b feature-nova-funcio

**7.2 Canviar de branca (git checkout)**

Per moure’s entre branques, utilitza el comandament git checkout. Això et permetrà canviar entre diferents versions del projecte sense perdre cap canvi.

Canviar a una altra branca:

git checkout nom\_brancha

Exemple: Si has creat una branca feature-nova-funcio i vols tornar a la branca principal (main), executa:

git checkout main

7.3 Fusionar branques (git merge)

Quan els canvis en una branca estan llestos, es poden incorporar a la branca principal mitjançant una fusió (merge). Això aplica tots els canvis fets en la branca secundària a la branca de destinació (normalment main).

Pas 1: Canviar a la branca de destinació (per exemple, main):

git checkout main

Pas 2: Fer la fusió amb la branca que vols incorporar:

git merge nom\_brancha

Exemple: Després de treballar en feature-nova-funcio, vols afegir els canvis a main. Canvia a main i fes la fusió:

git checkout main

git merge feature-nova-funcio

**7.4 Solucionar conflictes de fusió**

Quan Git troba canvis conflictius entre dues branques, es produeix un conflicte de fusió. Aquests conflictes apareixen quan els canvis fets en diferents branques afecten les mateixes línies d’un fitxer. Per resoldre’ls:

Obre el fitxer en conflicte: Git marca les seccions en conflicte amb <<<<<<<, ======= i >>>>>>>.

Resol el conflicte manualment: Selecciona o modifica el codi de manera que representi la versió final que vols.

Marca el conflicte com a resolt:

git add nom\_fitxer

Completa el merge amb un commit:

git commit -m "Resoldre conflicte entre main i feature-nova-funcio"

Exemple de conflicte: Si main conté canvis a index.py i feature-nova-funcio té altres canvis a la mateixa línia de index.py, quan intentes fer la fusió, Git et mostrarà un conflicte. Revisa i resol el conflicte, afegeix el fitxer amb git add index.py i completa el merge amb git commit.

**7.5 Crear i gestionar Pull Requests a GitHub**

Una Pull Request (PR) és una sol·licitud per incorporar canvis d’una branca a una altra a GitHub. És una eina essencial en projectes col·laboratius, ja que permet revisar i discutir els canvis abans de fer la fusió.

Pas 1: Pujar la branca al repositori remot:

git push -u origin nom\_brancha

Pas 2: A GitHub, ves al repositori i trobaràs l’opció de crear una Pull Request per la nova branca.

Pas 3: Omple els detalls de la Pull Request:

Títol i descripció: Afegeix una descripció clara dels canvis que conté la PR.

Revisió: Assenyala altres col·laboradors per revisar el codi.

Pas 4: Fusionar la Pull Request:

Després de la revisió, fes clic a “Merge pull request” per incorporar els canvis a la branca principal.

Exemple de Pull Request: Després de fer canvis a la branca feature-nova-funcio, l’has pujat a GitHub amb git push -u origin feature-nova-funcio. A GitHub, crea una PR per revisar i fusionar els canvis a main.

**7.6 Eliminar branques després de la fusió**

Després de fusionar una branca amb la branca principal, pots eliminar-la per mantenir el projecte net i organitzat.

Eliminar una branca localment:

git branch -d nom\_brancha

Eliminar una branca de GitHub:

git push origin --delete nom\_brancha

Exemple: Després de fusionar feature-nova-funcio amb main, pots eliminar la branca amb:

git branch -d feature-nova-funcio

git push origin --delete feature-nova-funcio

**7.7 Exemple complet de treball amb branques**

Imagina que estàs treballant en un projecte i vols afegir-hi una nova funcionalitat en una branca separada:

Crear una branca per a la nova funcionalitat:

git checkout -b feature-afegir-funcio

Fer canvis i commits en aquesta branca:

Edita o crea fitxers segons sigui necessari.

Afegeix i fes commit dels canvis:

git add .

git commit -m "Afegir nova funcionalitat al projecte"

Pujar la branca a GitHub:

git push -u origin feature-afegir-funcio

Crear una Pull Request a GitHub:

Ves al repositori a GitHub i crea una Pull Request per feature-afegir-funcio cap a main.

Sol·licita a altres col·laboradors que revisin els canvis.

Fusionar la Pull Request i eliminar la branca:

Un cop revisada, fes la fusió de la Pull Request.

Després, elimina la branca localment i de GitHub:

git branch -d feature-afegir-funcio

git push origin --delete feature-afegir-funcio

Resultat final: Has creat una nova funcionalitat en una branca, has fet una Pull Request, has fusionat els canvis a main i has eliminat la branca per mantenir el projecte organitzat.

**7.8 Consells per treballar amb branques**

Utilitza noms descriptius: Per facilitar el seguiment, dóna noms a les branques que descriguin el canvi o funcionalitat, com ara feature-xxx per a noves funcionalitats o fix-xxx per solucionar errors.

Fes Pull Requests petites i freqüents: Això facilita la revisió del codi i evita grans conflictes.

Sincronitza amb la branca principal: Abans de començar una nova funcionalitat, fes git pull a main per assegurar-te que treballes amb l'última versió.

**Data actualització i autor/res:** Ignacio López-Aylagas 06/11/2024