

Control de versions en IntelliJ amb Git

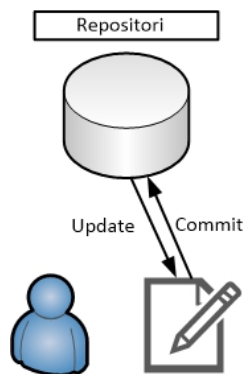
1. Què és el control de versions?

El control de versions es la gestió de totes aquelles variants produïdes durant la vida d'una aplicació, ja sigui en el desenvolupament o durant el manteniment d'aquesta.

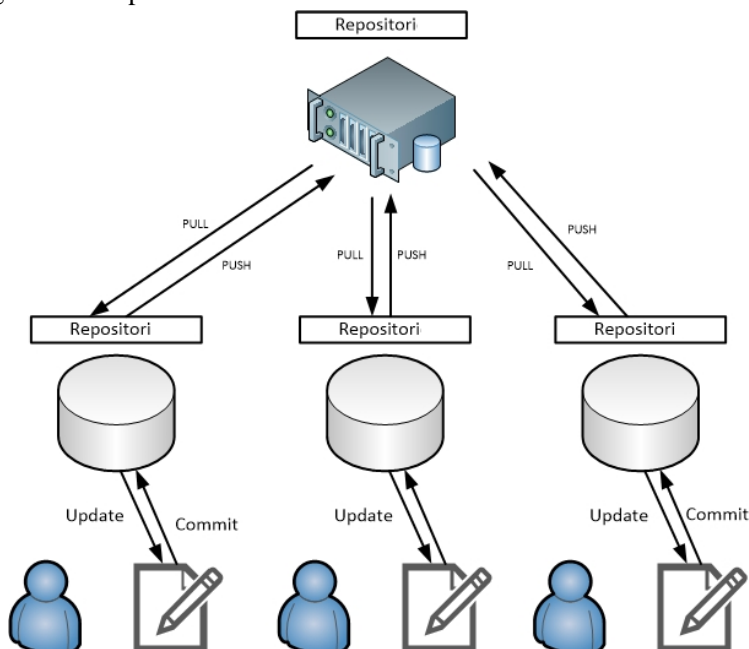
Els VCS (Version Control System) ens molt útils només per guardar tot l'historial dels fitxers de la aplicació sinó que a més ens permeten treballar de forma coordinada entre desenvolupadors al treballar sobre el mateix projecte. Ens poden ajudar a evitar o alertar de conflictes en la edició, feta pels desenvolupadors, sobre les mateixes parts de codi.

Podem trobar diferents tipologies d'eines VCS:

- VCS amb repositori local on es treballa sobre un repositori no distribuït.



- VCS Client-Servidor on el codi es compartit mitjançant l'ús d'un únic repositori compartit.
- VCS Distribuïts, la solució més estandarditzada, on cada desenvolupador treballa en un repositori local i el canvis es propaguen a un repositori central.



Disseny de Software.

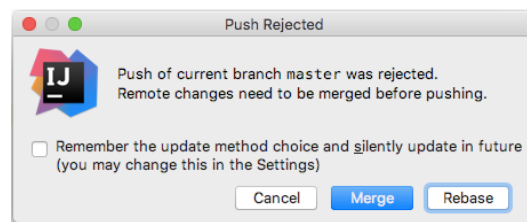
Pràctiques de laboratori.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

2. Com es treballa en grup?

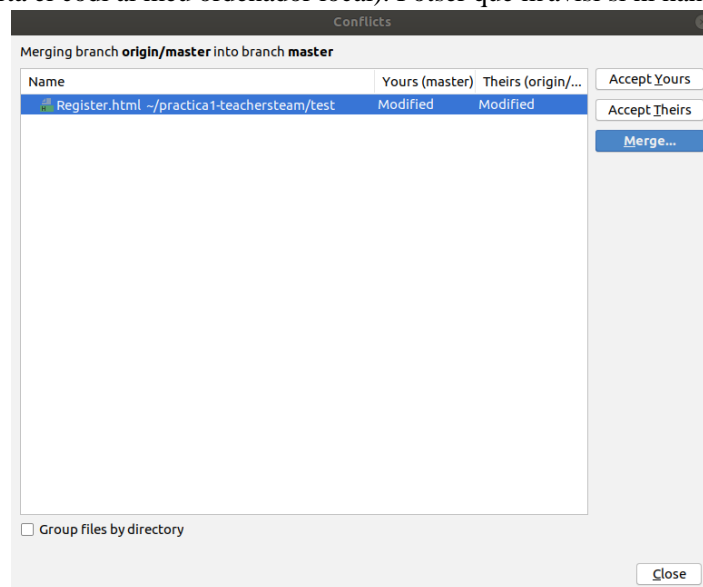
En els sistemes centralitzats (com git), tenim una única manera de treballar. Un repositori o punt central guarda el codi font (github); i tothom sincronitza el seu treball amb ell. Les desenvolupadores són nodes de treball i es sincronitzen amb aquest punt central. Això vol dir que, si dues desenvolupadores clonen des del punt central, i totes dues fan canvis; tan sols la primera d'elles, en enviar els seus canvis de tornada, ho podrà fer netament. La segona desenvolupadora haurà de fusionar prèviament el seu treball amb el de la primera, abans d'enviar, per evitar el sobre escriure els canvis de la primera. Git s'encarrega d'evitar el que es sobre escriu els uns als altres.

Si una de les desenvolupadores clona, fa canvis i després intenta enviar i una altra desenvolupadora ha enviat canvis durant aquest temps; el servidor rebutjarà els canvis de la segona desenvolupadora. Avisarà que estàs intentant enviar (push) canvis no directes (non-fast-forward changes), i que no podrà fer-ho fins que recuperi (fetch) i fusioni (merge) amb els canvis preexistents.



Així, quan treballis en equip, cadascun dels membres de l'equip poden estar treballant en una còpia local i avançar per separat en el projecte. Per a evitar conflictes i pèrdues d'informació cal que cadascú, abans de pujar res al git, faci els següents passos:

1. Comprovar que el projecte funciona en local, compila i s'executa fins al test que s'estigui provant.
2. Es poden anar fent Commits dels canvis fets (el commit es una operació local i per tant no està interferint amb el github remot).
3. Fer la comanda **Pull** (porta el codi al meu ordinador local). Potser que m'avisi si hi han conflictes:



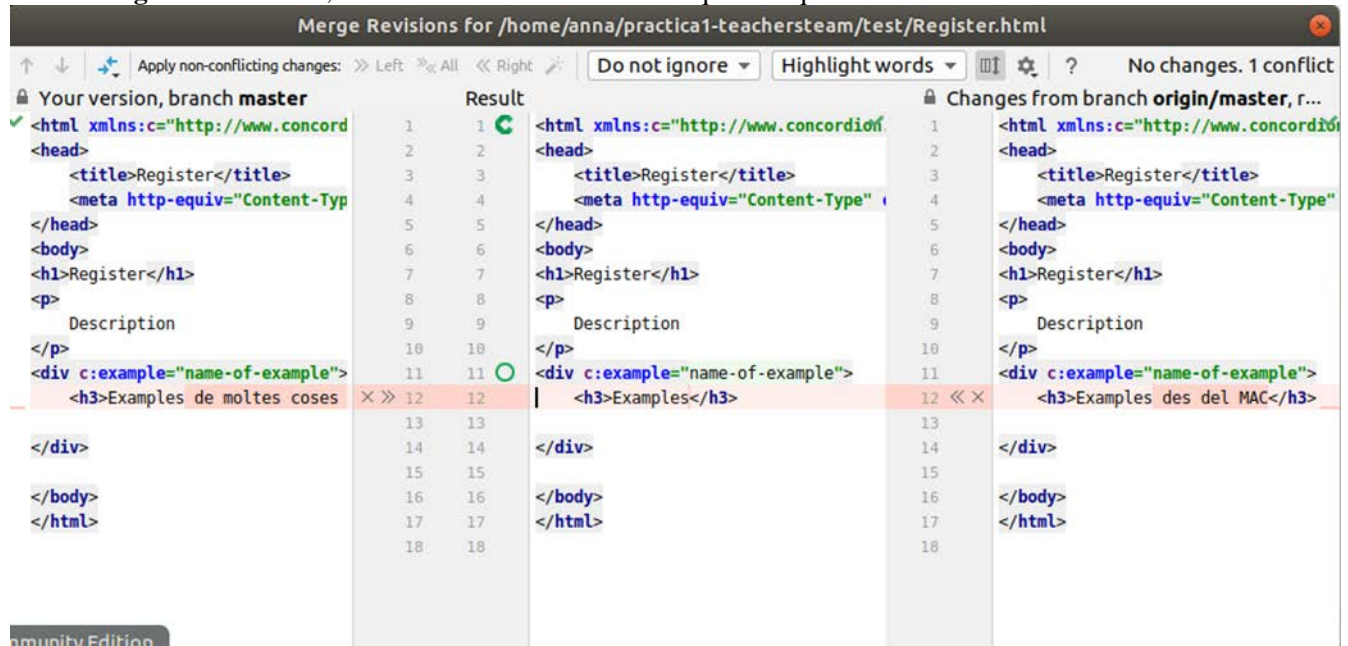
Disseny de Software.

Pràctiques de laboratori.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB

Curs 2019-20

4. Fer **Merge** si és necessari, seleccionant els canvis amb els que vull quedar-me.



5. Finalment fer **Push** al repositori remot.

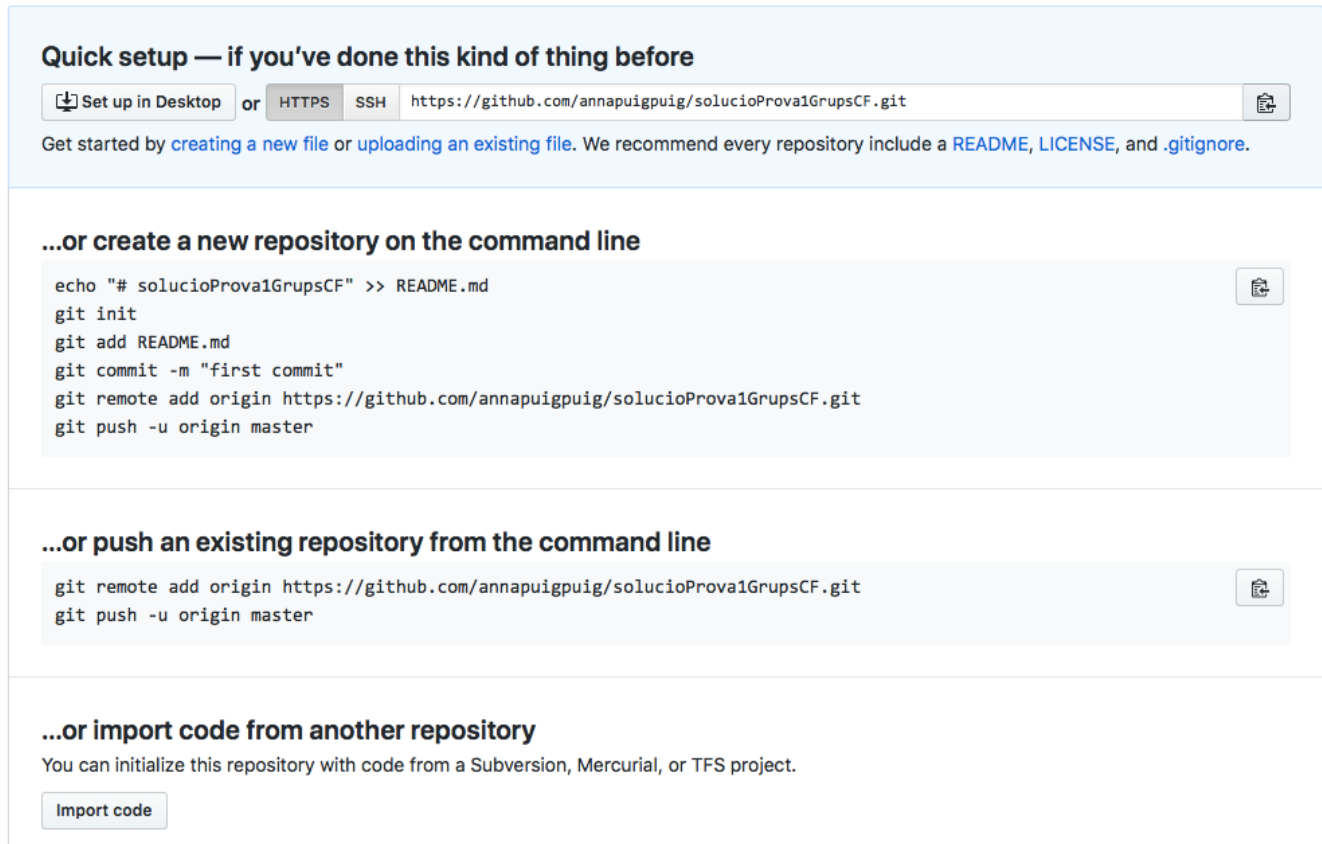
Disseny de Software.

Pràctiques de laboratori.



Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

3. Sincronització d'un repositori buit de github amb IntelliJ

Com usuaris de github podeu crer nous repositoris inicialment buits o bé accedir a alguna tasca del github classroom que tot i que el repositori estigui creat en una tasca, no contingui res. En el github us trobareu un missatge com:




Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** <https://github.com/annapuigpuig/solucioProva1GrupsCF.git> 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).


...or create a new repository on the command line

```
echo "# solucioProva1GrupsCF" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/annapuigpuig/solucioProva1GrupsCF.git
git push -u origin master
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/annapuigpuig/solucioProva1GrupsCF.git
git push -u origin master
```



...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

El més segur és fer un nou repositori en la línia de comandes, tal i com indica la pantalla de github. En la versió d'IntelliJ Ultimate permet directament fer un nou projecte des del menú File -> New -> Project From Existing Sources indicant l'adreça web del repositori. Si ho feu des del IntelliJ heu de recordar-vos de fer Commit i després Push per a que quedi tot integrat en el projecte del github.

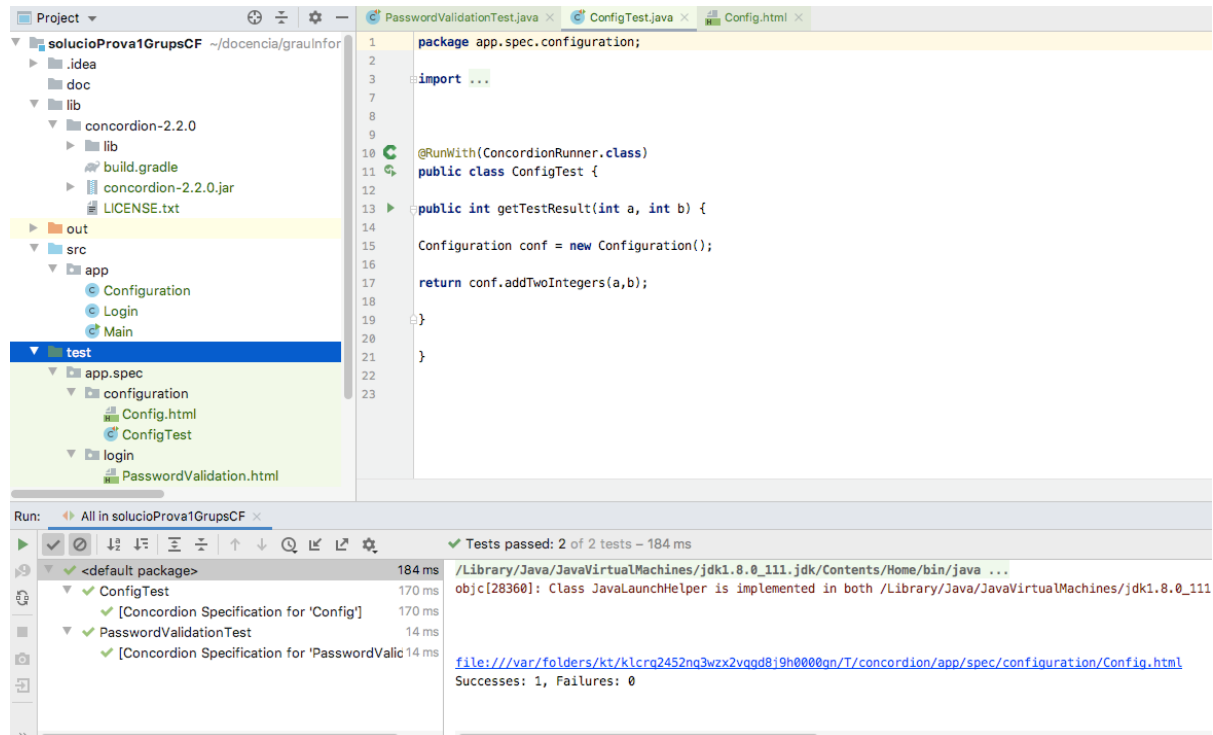
Com a primera prova aneu al vostre repositori Concordion de github i feu Download Zip. Copieu els continguts del zip a la carpeta local del nou repositori. Comproveu que tot funciona correctament i està ben configurat amb la llibreria de concordion. Passeu els tests fent que funcionin.

Disseny de Software.

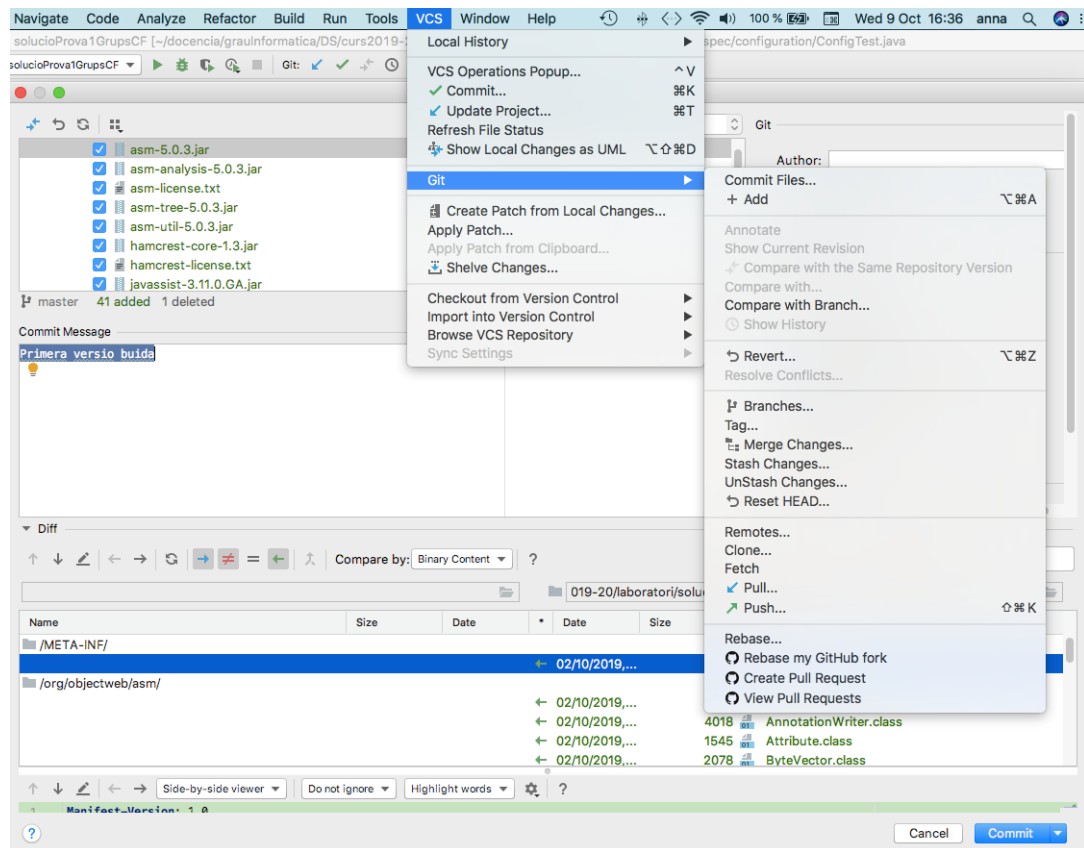
Pràctiques de laboratori.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB

Curs 2019-20



Feu **COMMIT** per a actualitzar el projecte localment i **PUSH** per pujar tot el contingut al github.

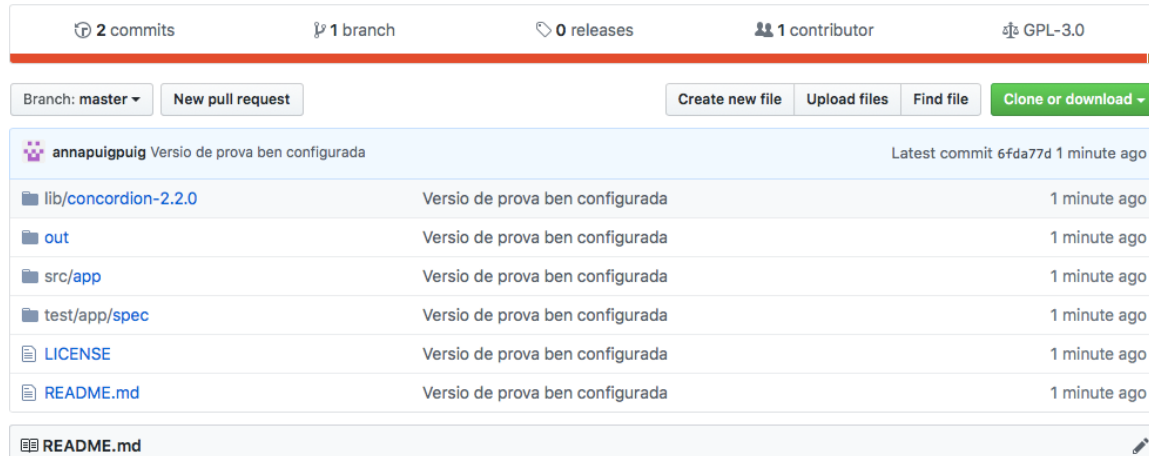


Disseny de Software.

Pràctiques de laboratori.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

Si recarregueu la plana de github, veureu que s'han pujat tots els nous fitxers i carpetes.

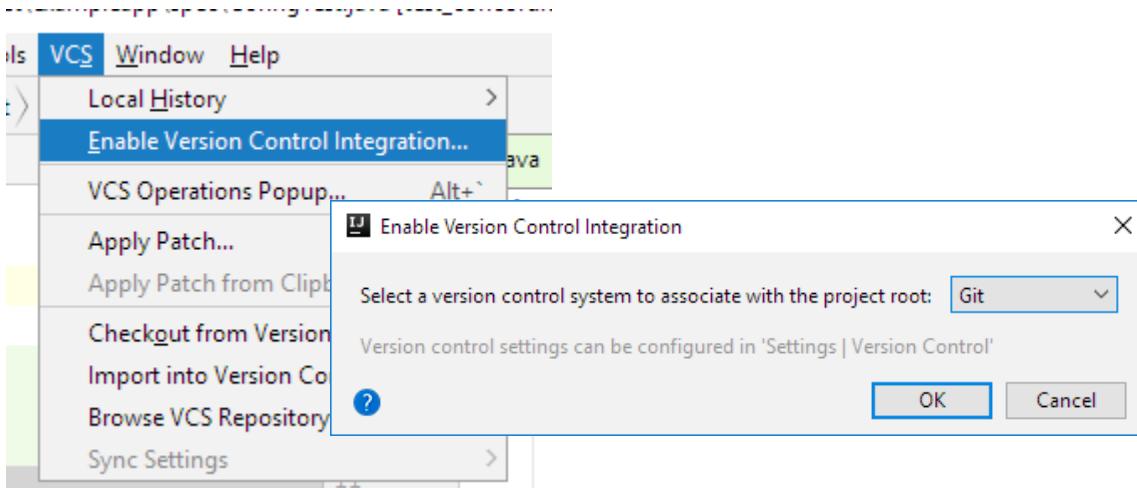


Ara, si baixessiu o obrissiu aquest projecte des d'IntelliJ des de qualsevol ordinador, us hauria de funcionar tot correctament.

4. Habilitar el control de versions en un nou projecte local i fer un push inicial a GitHub

Les següents accions en permetran habilitar el control de versions i configurar un repositori remot a GitHub.

1. Obrim el menú VCS → Enable Version Control Integration i configurem Git com a eina VCS.

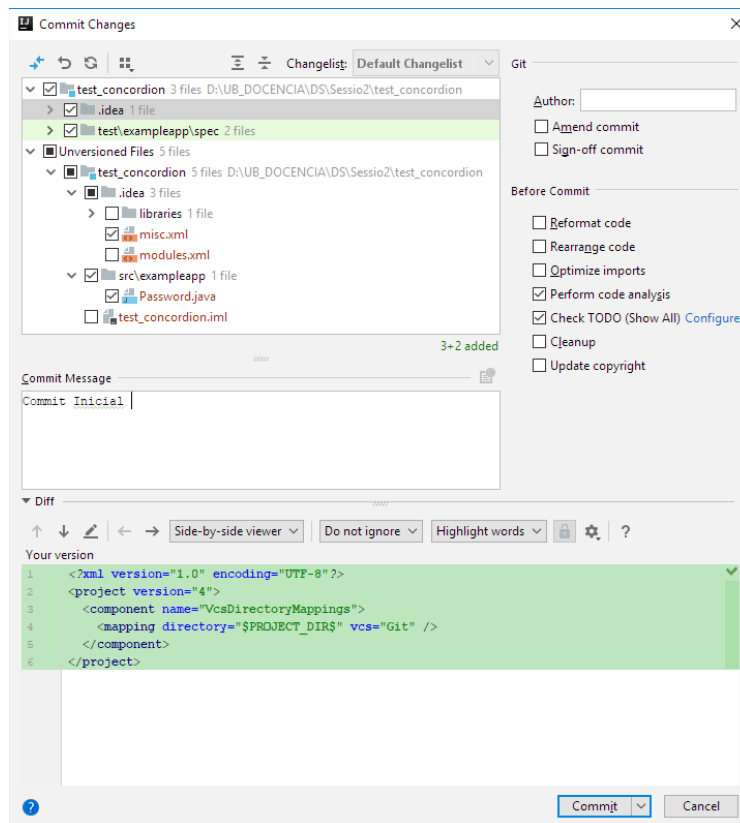
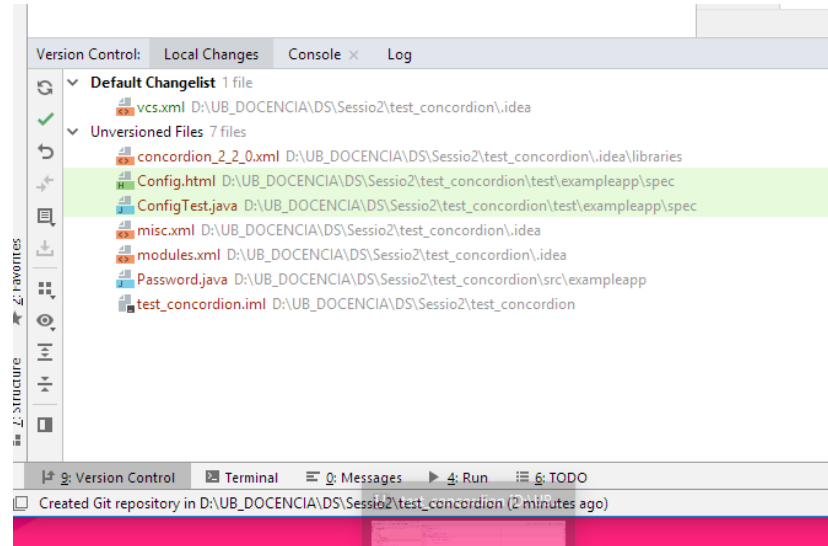


Disseny de Software.

Pràctiques de laboratori.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

2. En el pas anterior hem configurat el nostre Git local com a eina VCS. En aquest moment si volguéssim fer un commit dels nostres canvis, es farien només en aquest repositori, es a dir, en local.

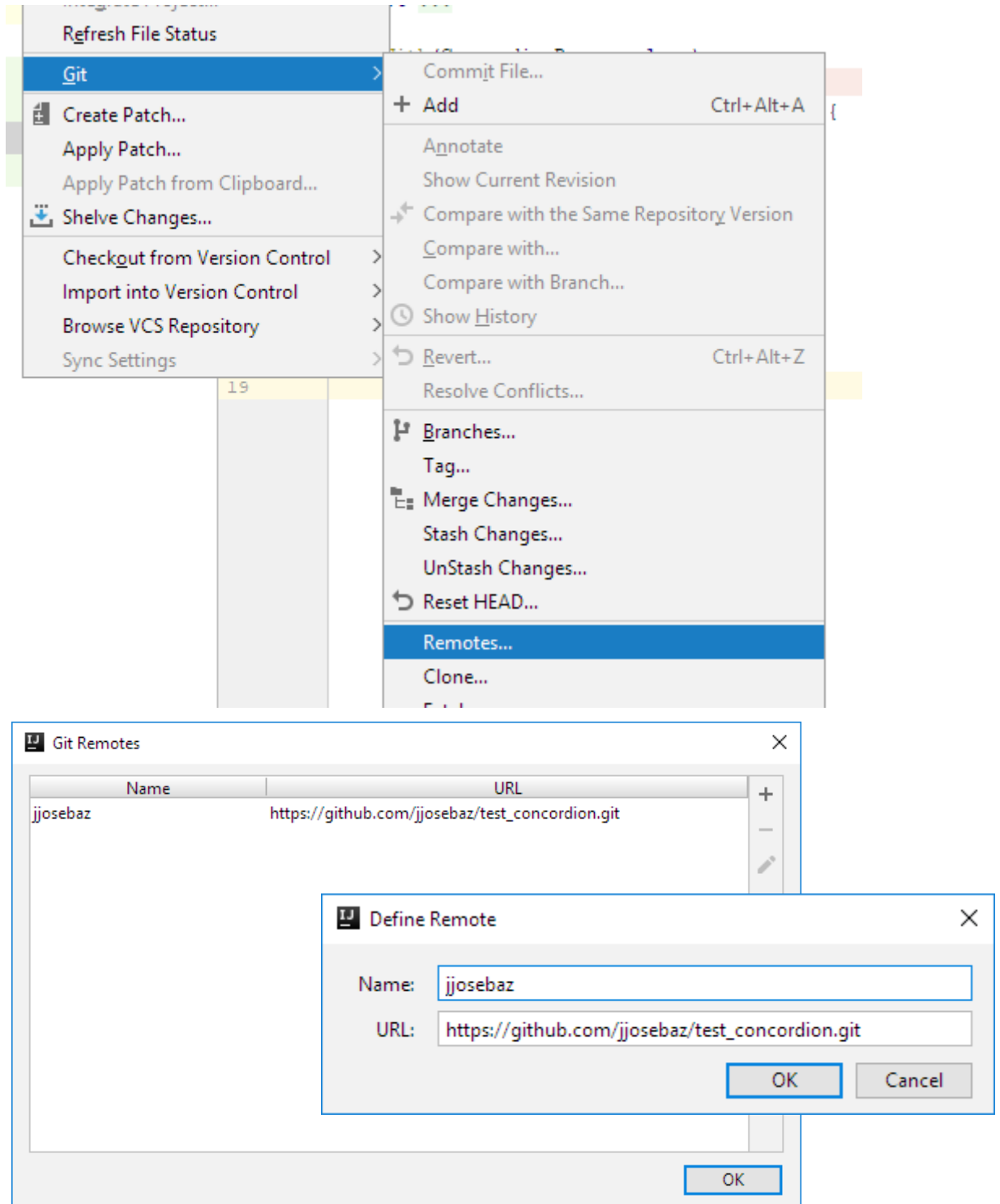


Disseny de Software.

Pràctiques de laboratori.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

3. Fer un commit equival pujar al repositori local els últims canvis fets. Per a fer un push a GitHub, ens cal primer crear i configurar el repositori remot. Per l'exemple he creat amb compte un repositori públic. Per a les vostres entregues caldrà doncs configurar el repositori privat corresponent a cada grup.



4. Un cop configurat el repositori remot ja podreu fer Push a GitHub del estat actual del projecte.

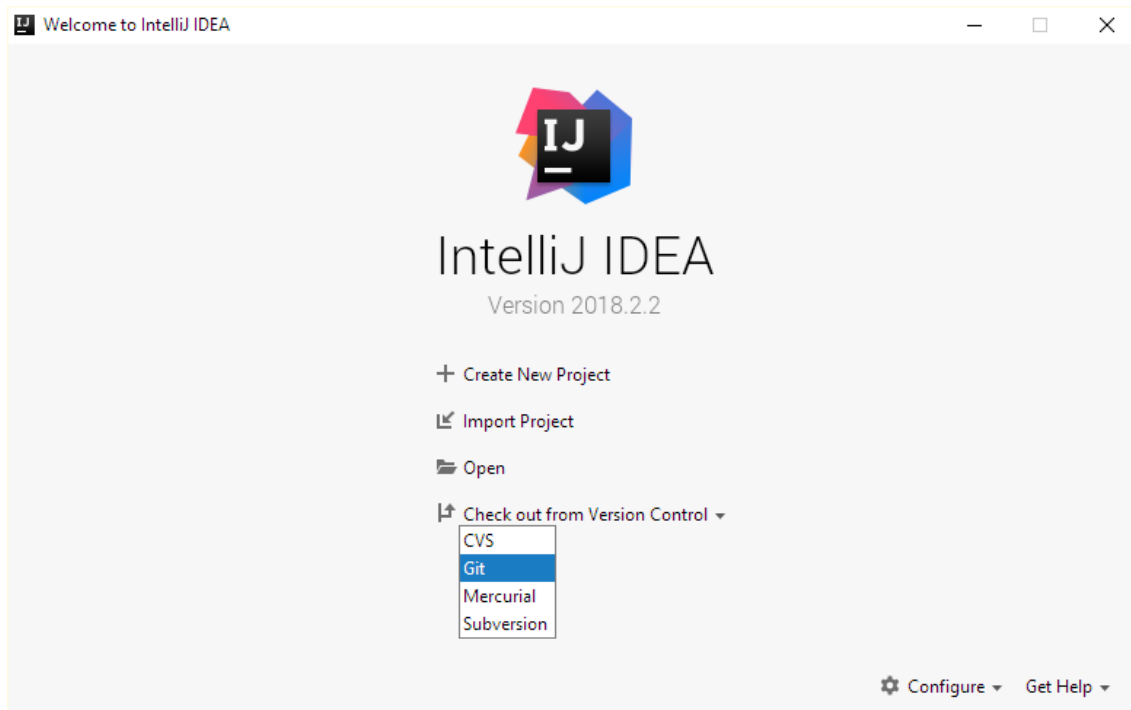
Disseny de Software.

Pràctiques de laboratori.

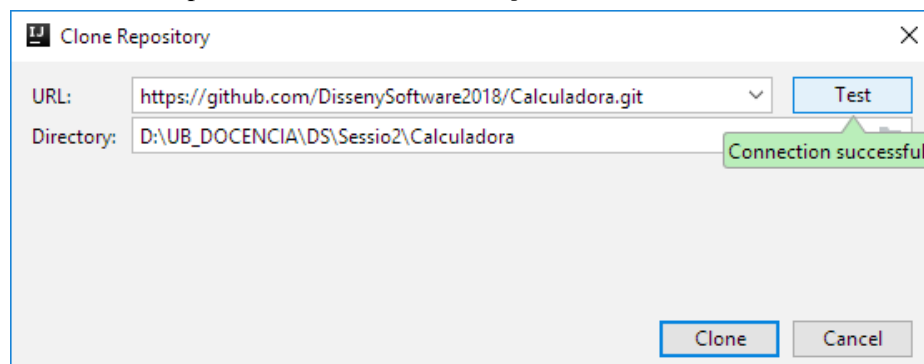
Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

5. Importar un projecte des de GitHub

1. Per a importar un projecte des de GitHub caldrà fer un *Check out From Version Control* des de el menú inicial amb l'eina Git.



2. Connectarem amb el repositori remot i indicarem el *path* on volem ubicar localment el nostre projecte.

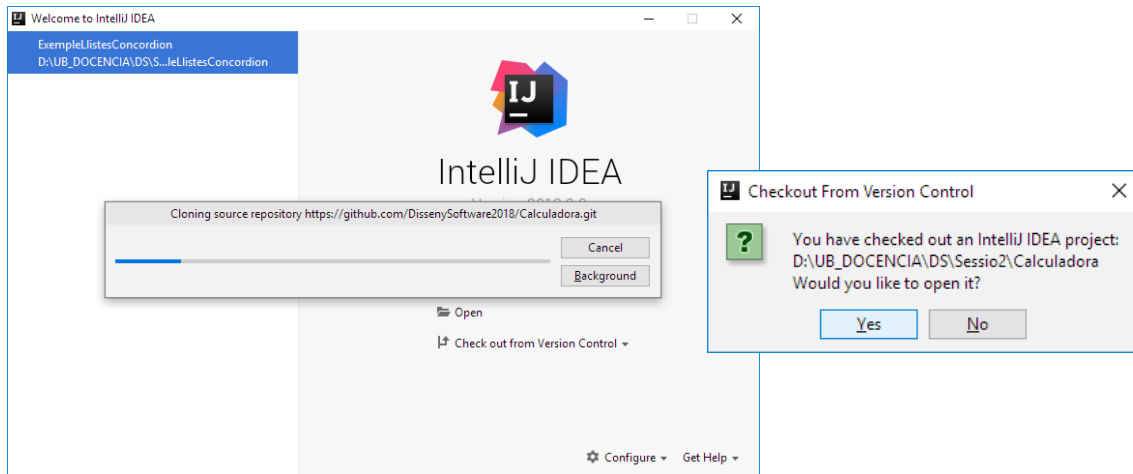


Disseny de Software.

Pràctiques de laboratori.

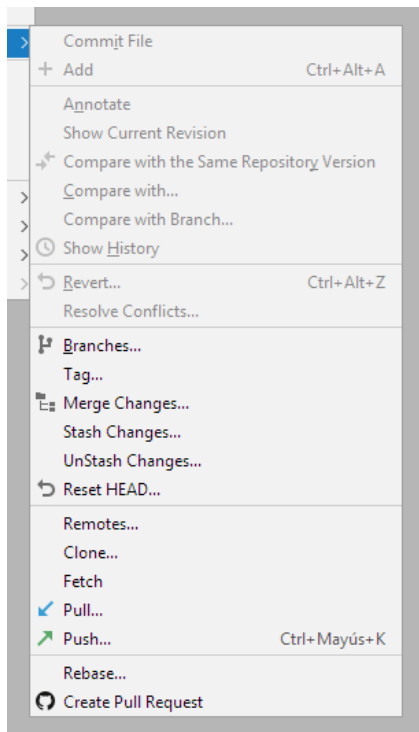
Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

3. Un cop clonat el repositori remot al repositori local ens demanarà si volem obrir el projecte.



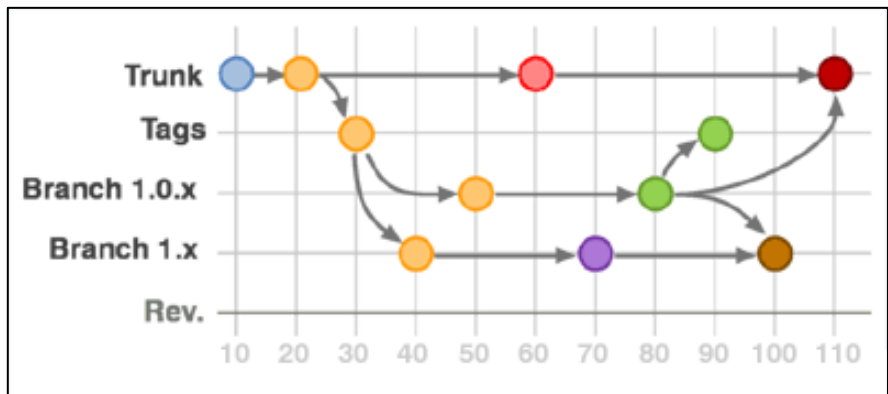
6. Funcionalitats a destacar dels VCS

6.1 – Algunes funcionalitats clau per a la coordinació dels desenvolupadors



A la majoria de VCS trobarem un conjunt de funcionalitats molt útils per a treballar en equip.

- Push / Pull: Ens servirán per replicar els canvis entre el repositori local i remot.
- Commit / Update: Ens serviran per pujar baixar els canvis en els nostres repositoris locals.
- Trunk: Branca principal de la aplicació.
- Branches: Ens permet crear branques paral·leles de desenvolupament
- Tags: Permet agrupar un conjunt de canvis sota una mateixa etiqueta a qualsevol branch o al Trunk
- Merge: Ens permet unificar branches amb altres branches o el mateix trunk, Avisant dels possibles conflictes.



Disseny de Software.

Pràctiques de laboratori.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

https://es.wikipedia.org/wiki/Archivo:SV_1

7. Referències i més documentació.

<https://www.jetbrains.com/help/idea/using-git-integration.html>

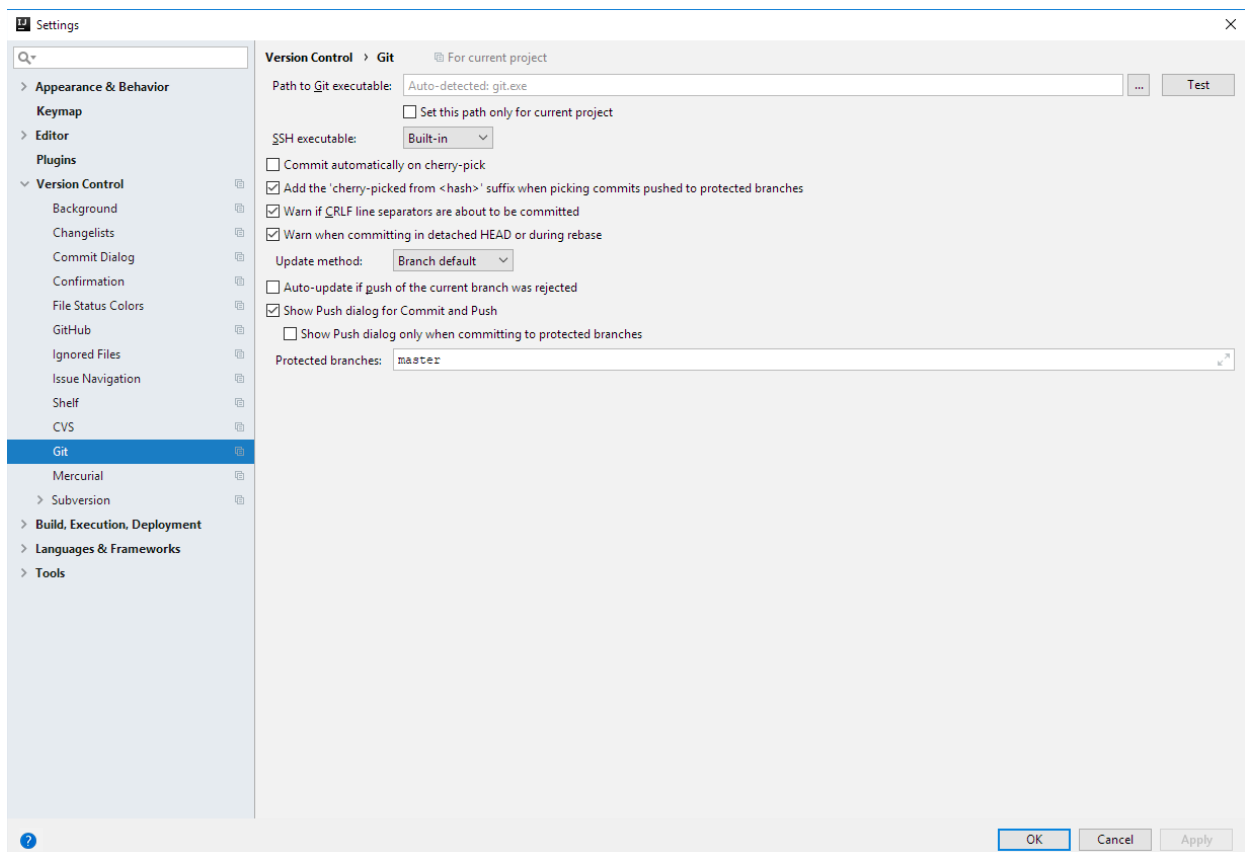
<https://www.jetbrains.com/help/idea/set-up-a-git-repository.html>

<https://git-scm.com/doc>

Appèndix – Configuració de Git a IntelliJ

En primer terme serà necessari instal·lar Git en la nostra màquina local (en cas que no la tinguem ja instal·lada). Podem trobar els executables per a diferents sistemes operatius al següent enllaç: <https://git-scm.com/downloads>. Aquesta instal·lació instal·larà un repositori local al nostre equip que posteriorment podrem enllaçar a un repositori remot.

1. Un cop instal·lat Git en el nostre equip caldrà anar a *File* → *Settings*, per a configurar la ruta del executables de Git.



Disseny de Software.

Pràctiques de laboratori.

Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2019-20

2. Un cop confirmat mitjançant el botó de test, la ubicació del executable passarem a configurar el nostre compte de GitHub. Això serà necessari per poder pujar els canvis al repositori. Per això al mateix menú Settings → GitHub configurarem les credencial d'usuari.

