GIT avec
GIT GUI, GITK et GPS

Git est un système de contrôle des versions (Version Control System - VCS) du code source d'un logiciel. La particularité de GIT est qu'il est distribué, ce qui a pour conséquence de posséder en local l'ensemble des versions sans dépendre d'un serveur distant. Son système de gestion des branches est très particulier et déroute à coup sûr lors des premiers contacts. Il est très différents de SVN, par exemple.

Depuis 2005, GIT est notamment utilisé pour le code source du noyau Linux. Git est un logiciel libre et ouvert, conçu pour gérer tout depuis des petits aux très grands projets avec rapidité et efficacité.
Source http://git-scm.com.

1) Installation pour Mac

Bien que disponible en standard pour MacOS 10.8, la version présente ne comporte pas GIT GUI, l'interface utilisateur graphique. Bien que fonctionnel en ligne de commande, nous préférons utiliser l'interface utilisateur en récupérant la dernière version de GIT disponible sur le site de GIT.

Récupérez l'archive git-1.8.1.3-intel-universal-snow-leopard.dmg (ou une plus récente) sur le site :
http://code.google.com/p/git-osx-installer/downloads/list.

Ouvrir l'archive ainsi que le paquet d'installation git-1.8.1.3-intel-universal-snow-leopard.pkg. Git s'installe sous /usr/local/git.

Documentation :

Le livre Pro Git de Scott Chacon (July 29, 2009).
https://github.s3.amazonaws.com/media/progit.en.pdf

Une visualisation interactive des commandes :
http://ndpsoftware.com/git-cheatsheet.html

2) Configuration

Configurez l'emplacement de GIT GUI et lancez l'interface graphique à partir du Terminal avec les deux commandes suivantes :

```
$ git config --global --add alias.gui '!open /usr/local/git/share/git-gui/lib/Git\ Gui.app'
$ git gui
```

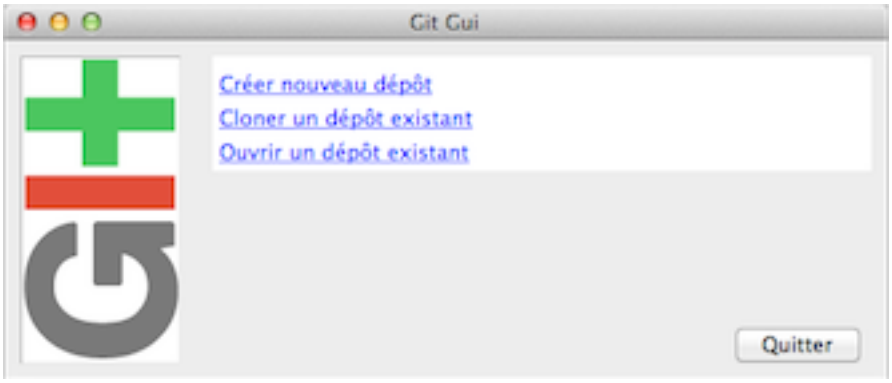


Figure 1 : L'interface utilisateur GIT GUI

Allez dans les préférences et modifiez dans la partie Globales (à droite) le nom de l'utilisateur et l'adresse de messagerie puis cliquez sur "Sauvegarder". Les informations sont conservées dans le fichier ~/.gitconfig.

Les informations de la partie Dépôt (à gauche) ne sont valables que pour le dépôt courant. Elles sont conservées dans le fichier .git/config.

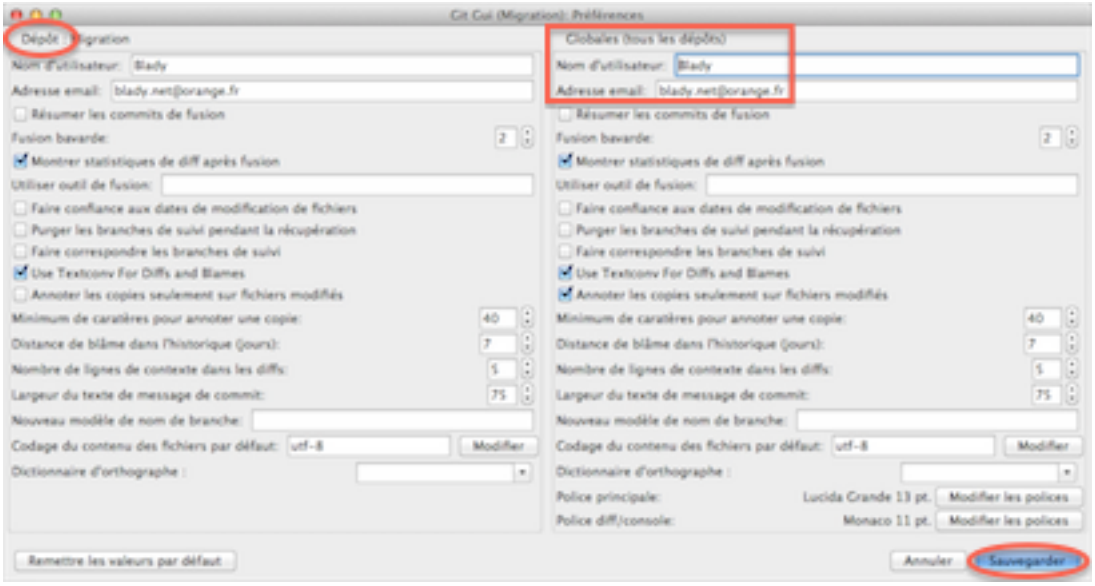


Figure 2 : la configuration de l'utilisateur

3) Création d'un dépôt

Cliquez sur "Créer nouveau dépôt" (voir figure 1) puis dans la fenêtre qui s'ouvre indiquez directement l'emplacement ou utilisez le bouton "Naviguer" pour sélectionner un répertoire existant ou en créer un nouveau, puis cliquez sur "Créer".

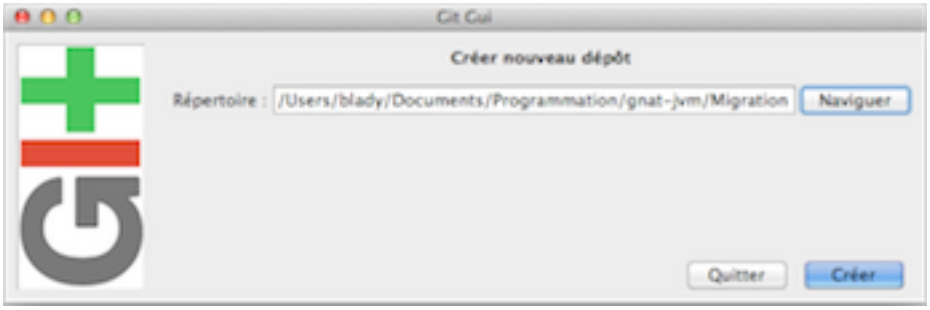


Figure 3 : Emplacement du dépôt à créer

La fenêtre de navigation principale apparait avec la liste des fichiers de notre répertoire dans la zone des fichiers non-indexés.

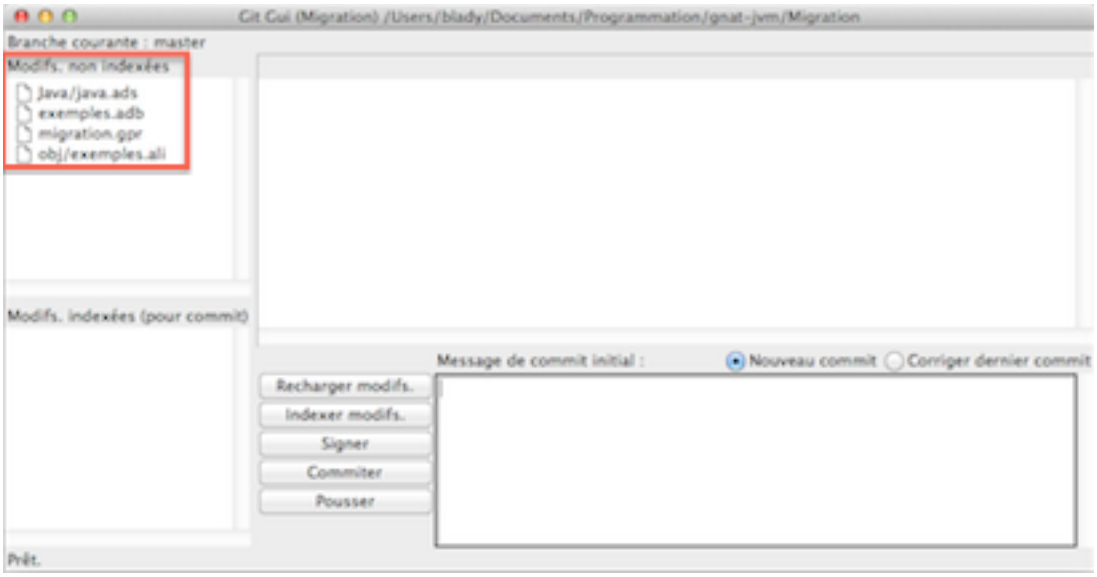


Figure 4 : notre répertoire avec son contenu

Certains fichiers ou répertoires ne nous intéressent pas, nous allons demander à GIT de les ignorer (nom de fichier ou nom avec "/" pour un répertoire), pour cela entrez la commande suivante :

```
$ cat > .gitignore
.*
Java/
obj/
ChangeLog
*.npp
```

La première ligne empêche GIT de référencer le fichier .gitignore, ce qui n'est pas forcément une bonne idée. Il peut être intéressant de référencer d'éventuelles modifications en ajoutant la ligne : "!.gitignore".

Cliquez sur le bouton "Recharger modifs", les fichiers contenus dans les 2 répertoires ont disparus de la zone des fichers non-indéxés.

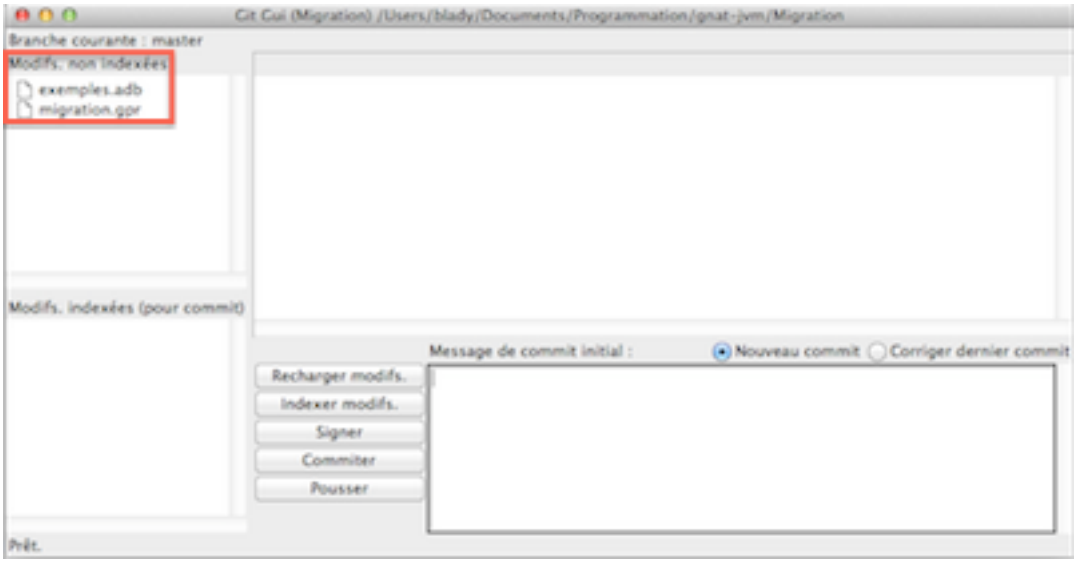


Figure 5 : seuls les fichiers filtrés apparaissent

4) Indexer puis référencer le projet et les codes sources

L'opération suivante est d'indexer les fichiers pour qu'ils soient connus de GIT en les sélectionnant l'un après l'autre puis en cliquant sur le menu Commit->Indexer, nous sélectionnons le projet GPS ainsi que les sources Ada, les modifications sont alors visibles.

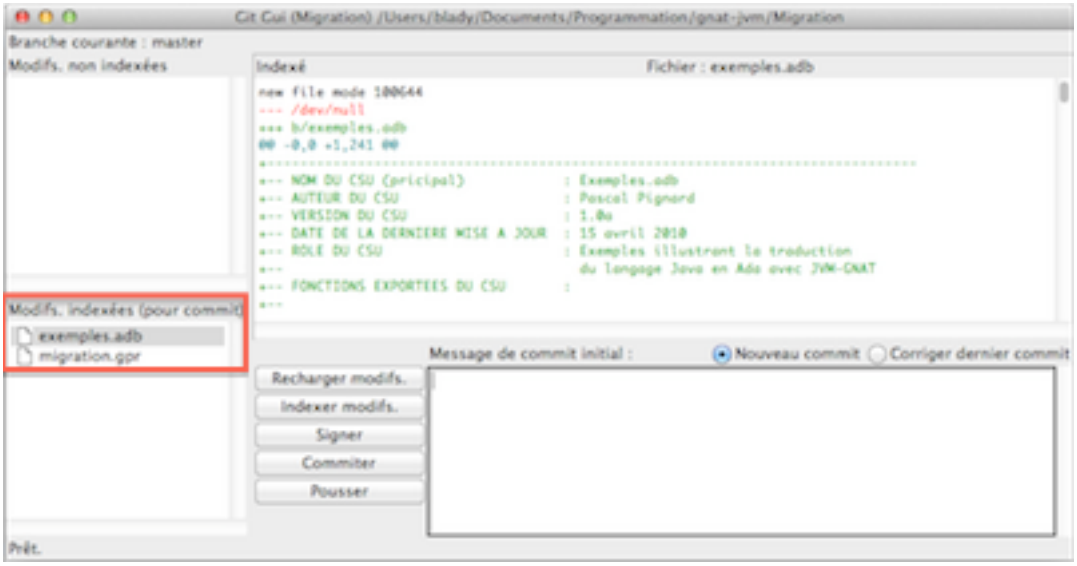


Figure 6 : les fichiers sont indéxés

L'état correspondant est visible sous GPS en affichant la fenêtre des projets avec le menu VCS->Explorer :

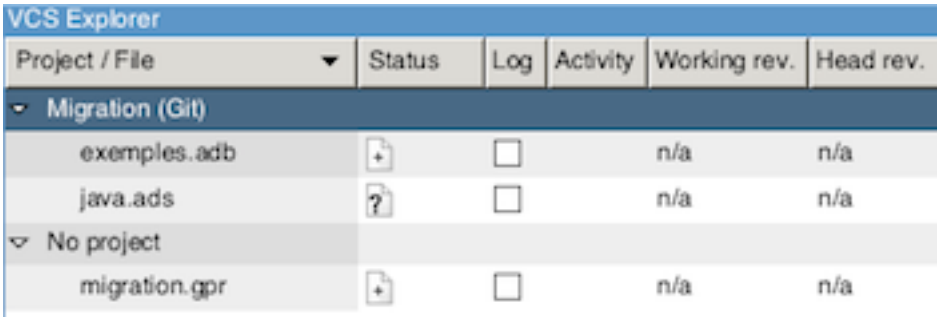


Figure 7 : le + indique les fichiers indexés

Après avoir saisi un commentaire approprié, cliquez sur le bouton "Commiter", tout a disparu et a été référencé dans GIT :

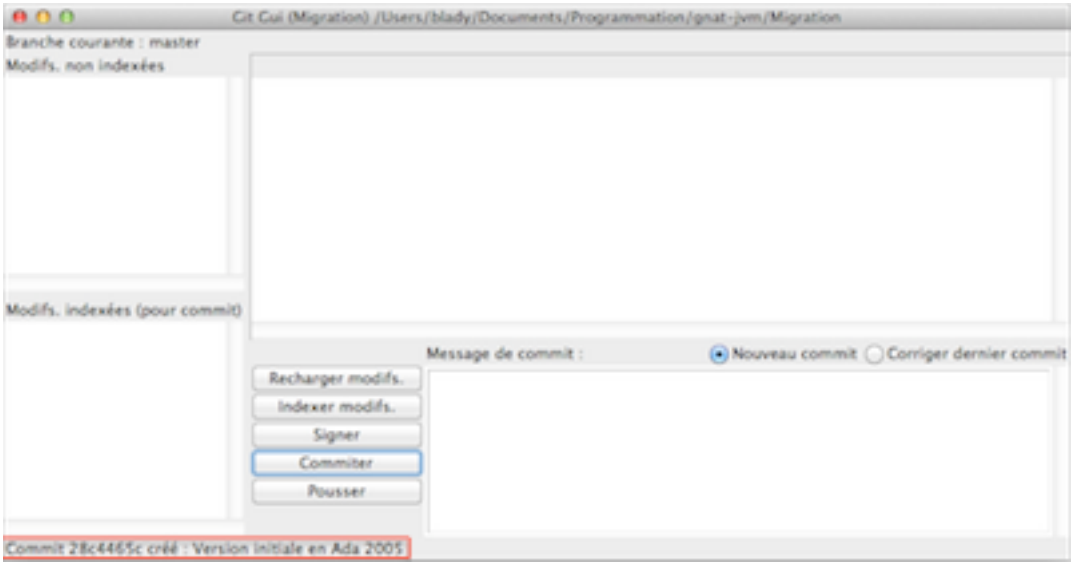


Figure 8 : les fichiers sont maintenant référencés par GIT

L'état correspondant est visible dans la fenêtre des projets GPS :

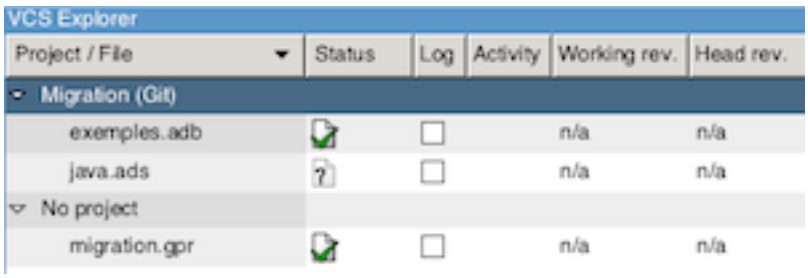


Figure 9 : le tick vert indique les fichiers référencés

5) Modifications et historique

Modifions le fichier exemples.adb, dans GIT GUI cliquez sur le bouton "Recharger modifs", nous voyons les différences apportées :

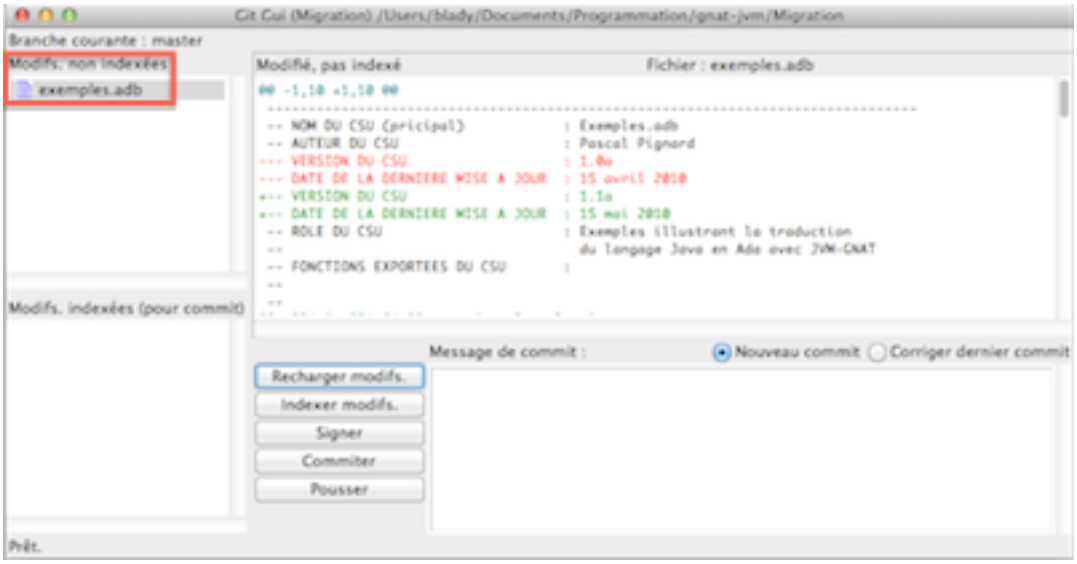


Figure 10 : les modifications apparaissent en rouge (ancien) et vert (nouveau)

Cliquez sur le menu Dépôt->Voir l'historique de toutes les branches, GITK s'ouvre et nous voyons notre premier commit ainsi que nos modifications :

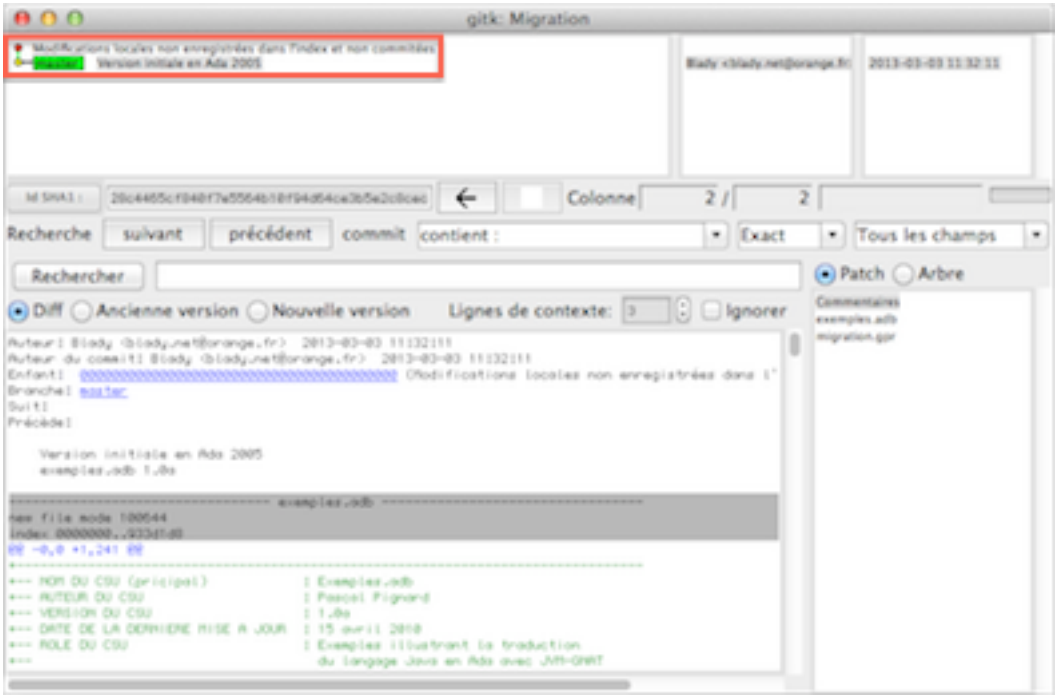


Figure 11 : historique des modifications dans GITK

L'état correspondant est visible sous GPS :

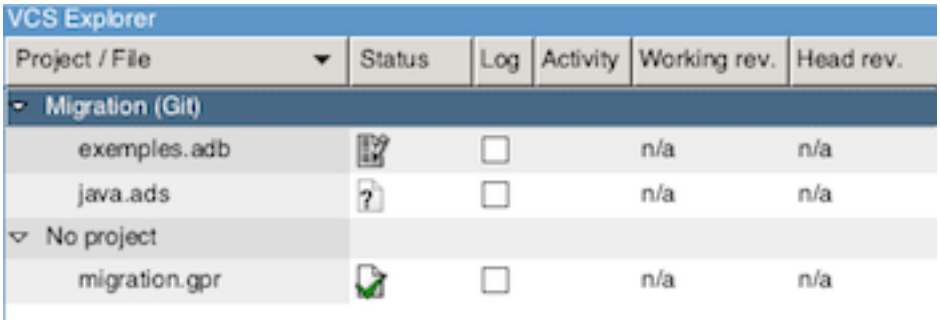


Figure 12 : le crayon indique le fichier modifié

Indexer les modifications avec le bouton "Indexer modifs" (figure 13-1), donnez un commentaire (figure 13-2) puis référencer les modifications indexées avec le bouton "Commiter" (figure 13-3) :

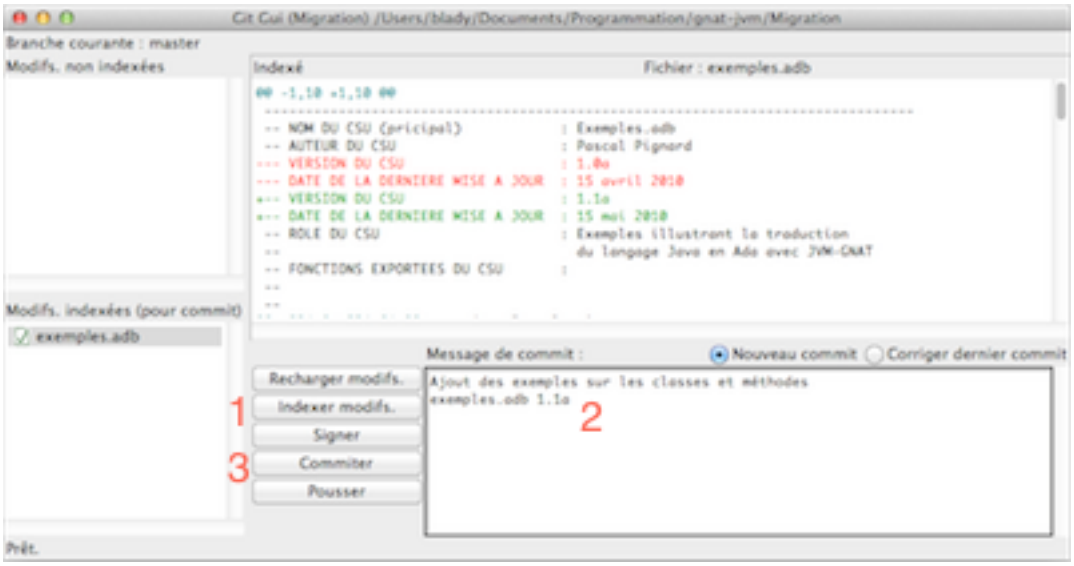


Figure 13 : le code source modifié a été indexé et référencé

Dans GITK, rafraîchir la vue en cliquant sur le menu Fichier->Recharger :

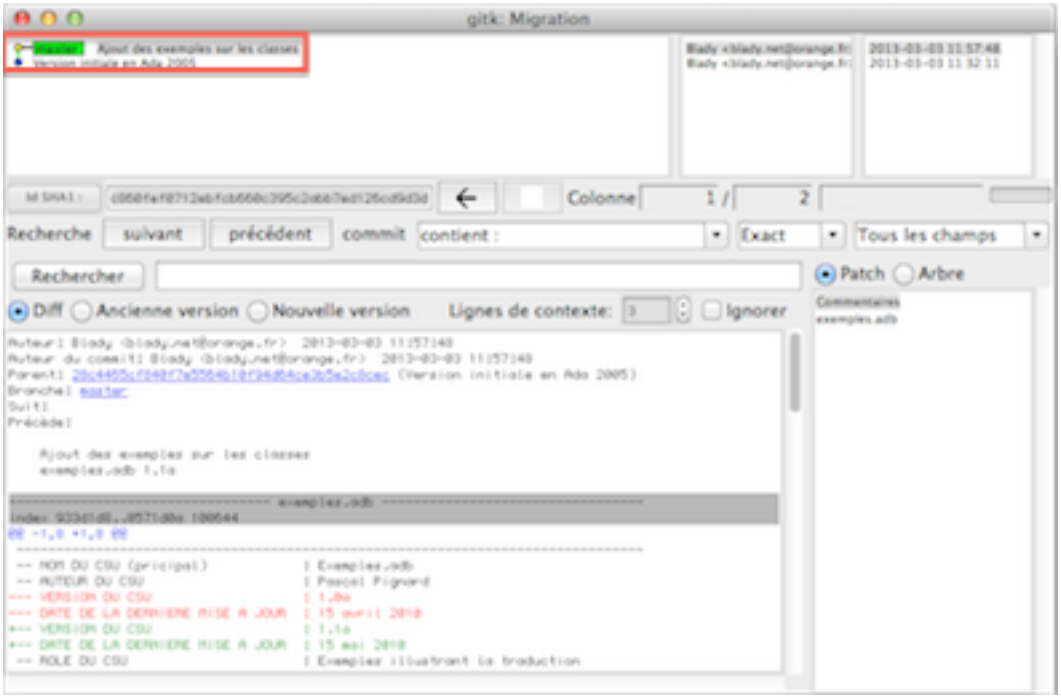


Figure 14 : l'historique des modifications

En cas d'erreur, sur le commentaire, par exemple, dans GIT GUI, sélectionner "Corriger dernier commit" et modifiez le commentaire puis cliquer sur "Commiter". Vérifez dans GITK que le commentaire modifié a remplacé le précédent, c'est très pratique.

Dans GPS, modifions à nouveau le fichier exemples.adb et le projet GPR, puis créez une activité "Constructeurs" (menu VCS->Activities et clic-droit sur la vue des activités) et ajoutez les deux avec en ctrl-clic et clic-droit dans la vue VCS Explorer :

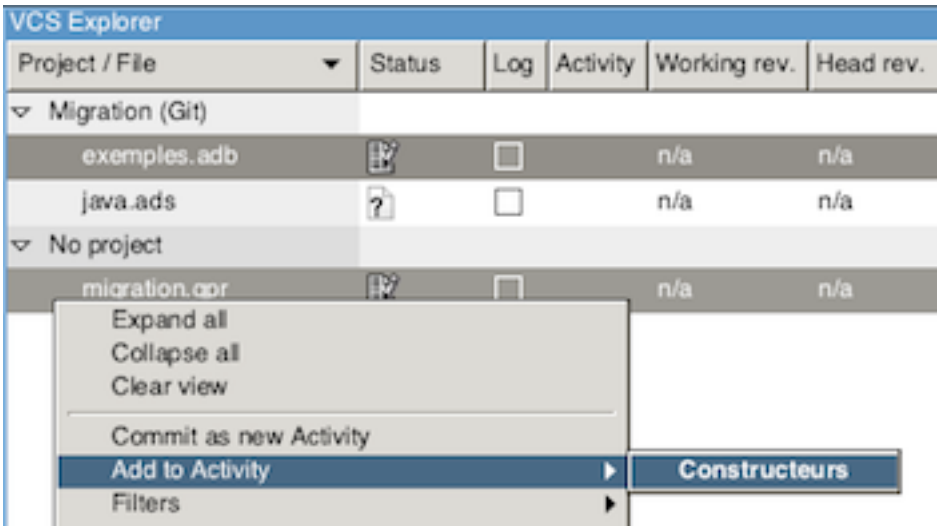


Figure 15 : ajout des fichiers sélectionnés à une activité dans GPS

Dans la vue des activités, indexez les deux fichiers l'un après l'autre avec un clic-droit->Version Control->Add, no commit :

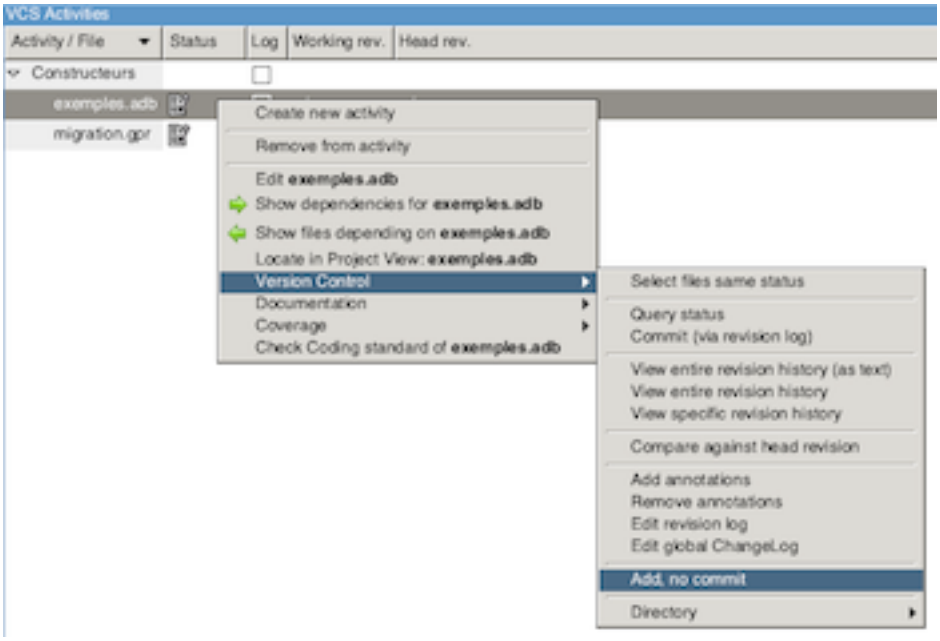


Figure 16 : le fichier est indexé

Sélectionnez l'activité et ajoutez un commentaire avec clic-droit->Edit revision log puis clic-droit->Commit activity :

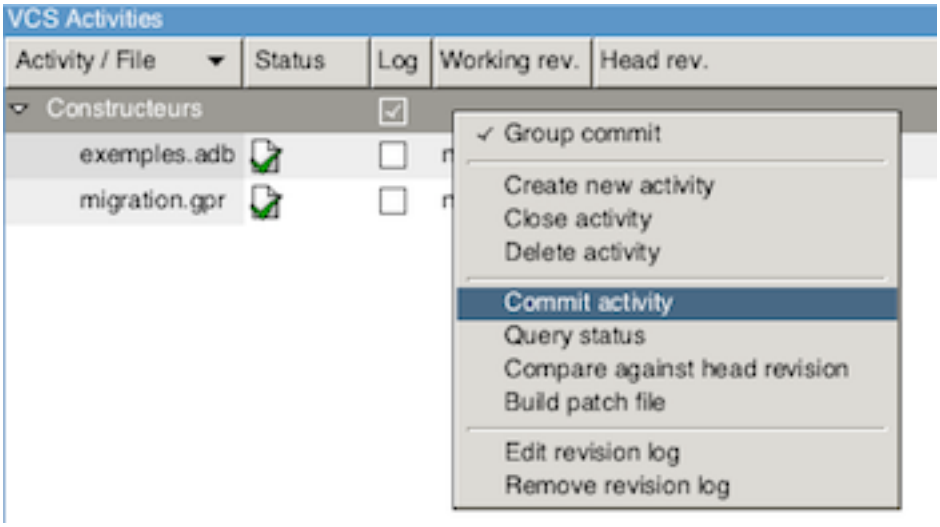


Figure 17 : les fichiers indexés sont référencés

Dans GITK, rafraîchir la vue en cliquant sur le menu Fichier->Recharger :

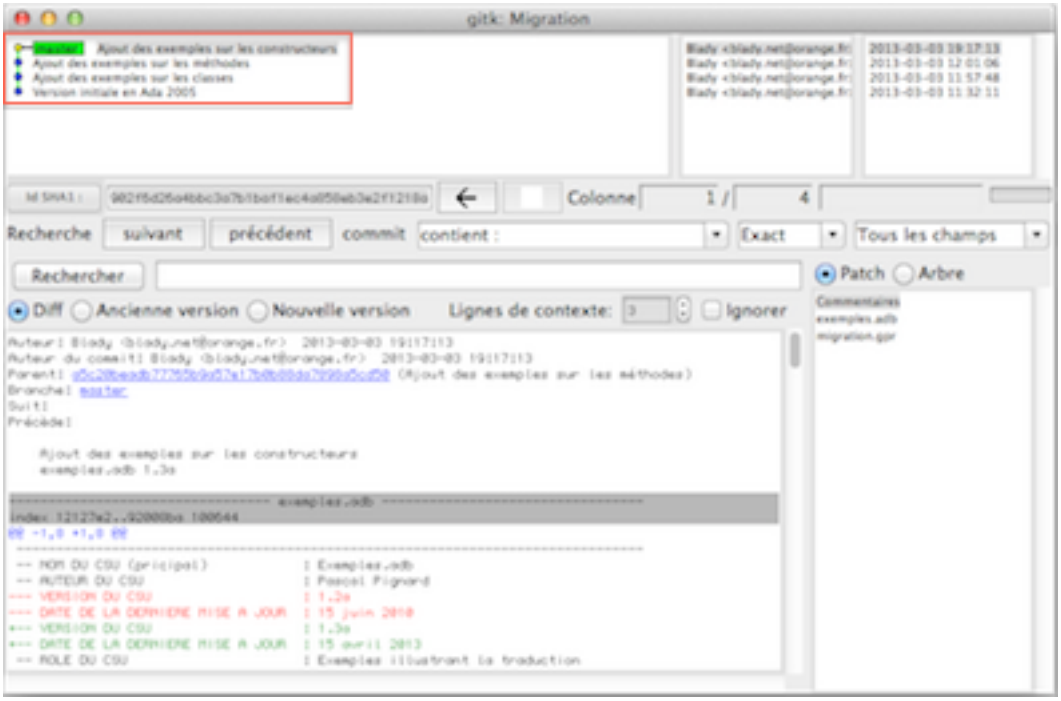


Figure 18 : historique des modifications

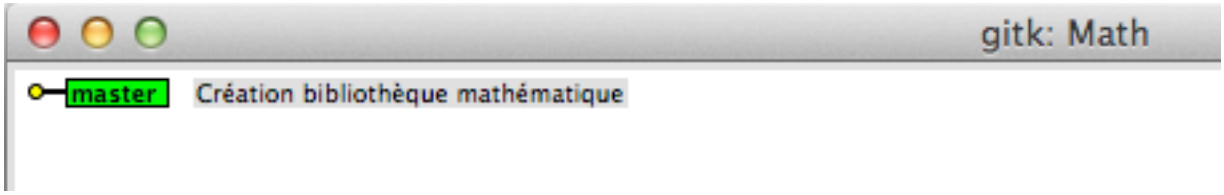
6) Création des branches

Dans un développement l'enchainement des versions n'est pas forcément linéaire, par exemple, à la suite de la version 1.5 courante, la version 2.0 sort mais on souhaite se réserver la possibilité de sortir ensuite une version 1.6. Une fois référencée dans GIT la version 2.0, GIT nous permet de revenir à la version 1.5 pour construire la version 1.6 mais où la référencer puis que la version 2.0 est déjà référencée ? C'est ici qu'intervient la notion de branche. À partir du référencement de la version 1.5 nous allons créer une branche permettant d'accrocher la version 1.6. En réalité, c'est plutôt l'inverse qui se passe. À la création du projet , nous préparons une branche V1.x puis avant de réaliser les modifications pour la version 2.0, nous créons une branche V2.x. Ainsi, nos deux versions peuvent être référencées indépendamment l'une de l'autre.

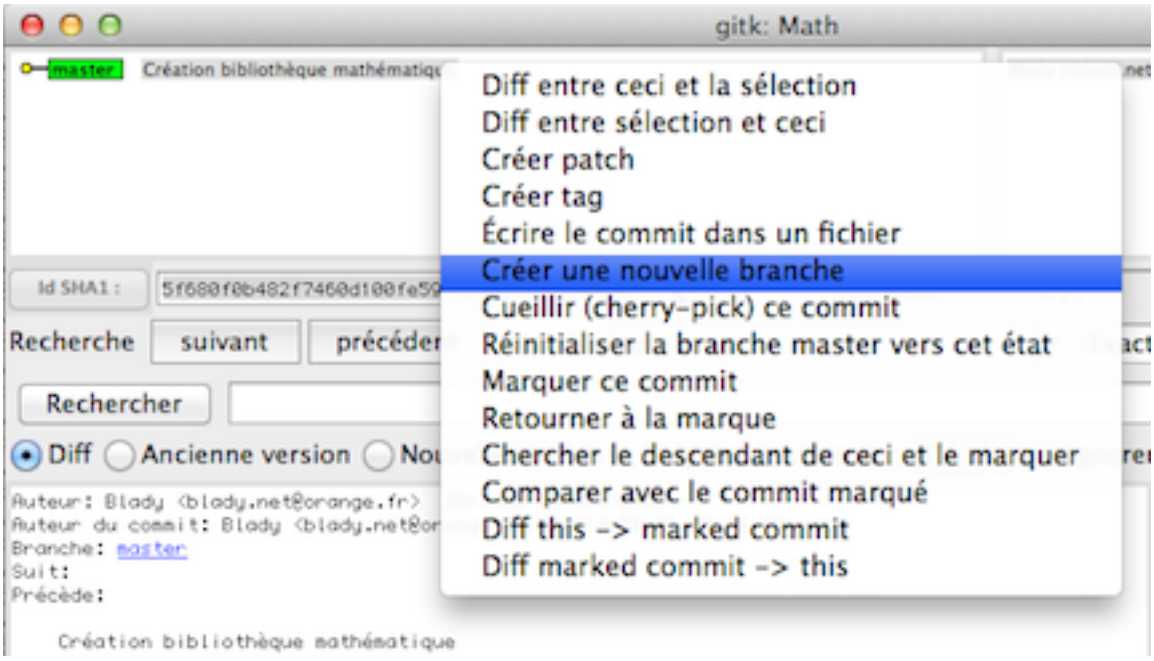
Les branches peuvent servir également à référencer des variantes d'un programme. Dans l'exemple suivant nous avons une première variante d'une programme Ada qui suit le standard Ada 2005 et une deuxième variante qui va exploiter les nouveautés du standard Ada 2012, mais nous souhaitons faire évoluer les deux variantes indépendamment, nous créons donc deux branches.

Voici quelques création de branches (en vert) avec GITK :

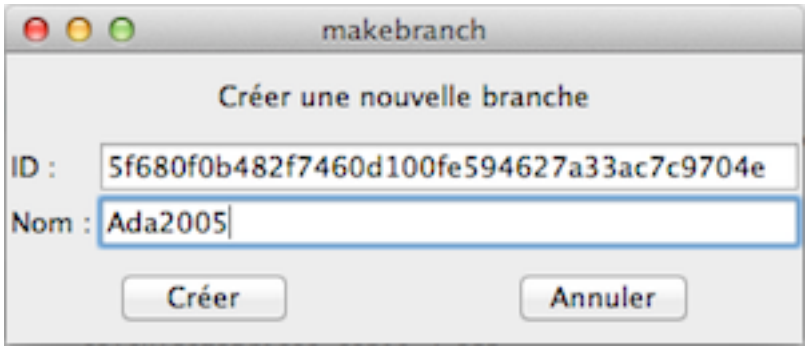
- notre projet initial avec sa branche master par défaut



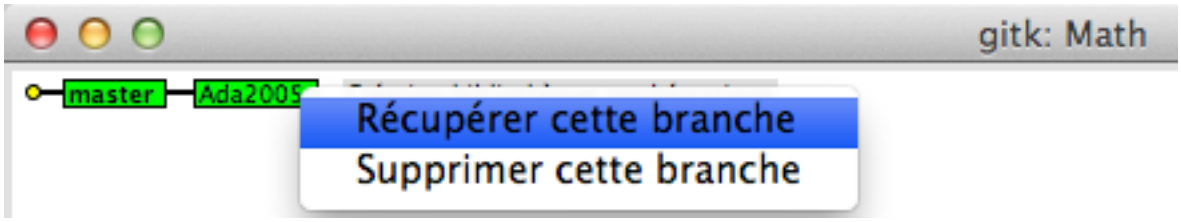
- sélectionnons la branche master avec un clic-droit et un clic sur "Créer une nouvelle branche"



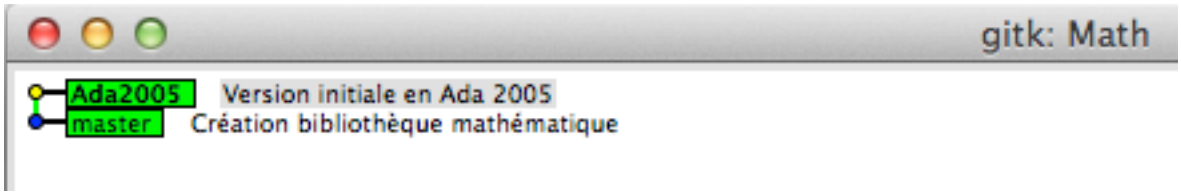
- entrons le nom de la branche



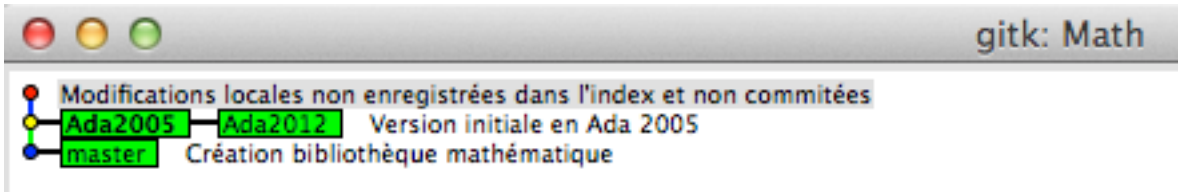
- sélectionnons la branche créée avec un clic-droit et clic sur "Récupérer cette branche.



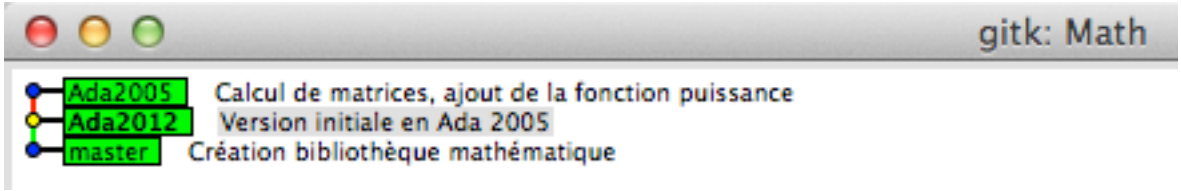
- la branche active est en gras, nous pouvons y référencer nos modifications



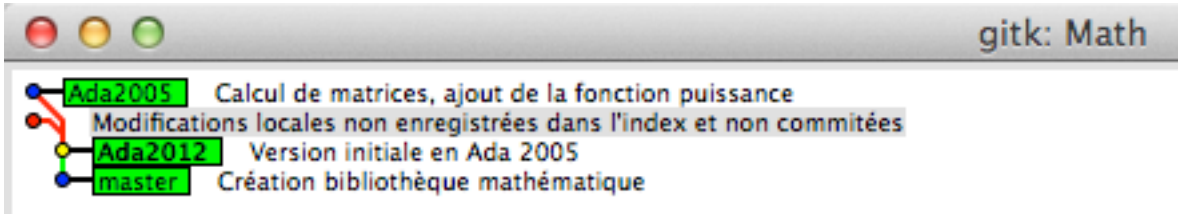
- à partir de ce point nous préparons la version suivante en créant la branche Ada2012 sans la récupérer



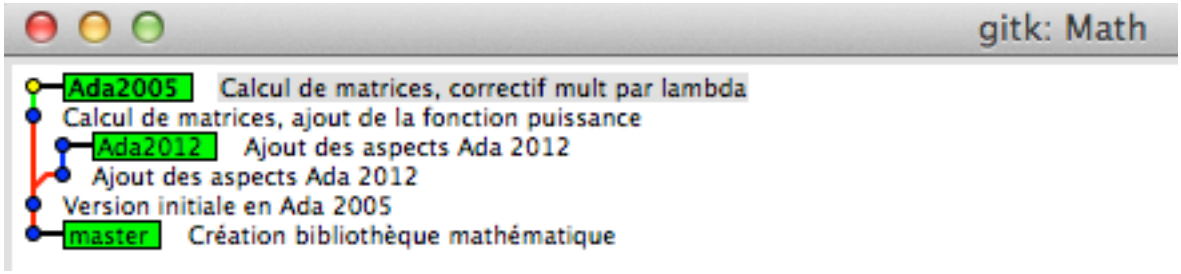
- nous référençons d'autres modifications sur la branche Ada2005



- récupérons la branche Ada2012



- le projet après plusieurs modifications sur l'une ou l'autre branches



La création de branche peut également s'effectuée dans GIT GUI. Sélectionnez le menu "Branche -> Créer...", sur le panneau qui s'est ouvert, entrez le nom de la branche (figure 19.1) et sélectionnez la branche de de départ (figure 19.2). Si vous ne voulez pas récupérer tout de suite la branche alors décochez la case "Charger (checkout) après création" (figure 19.3).

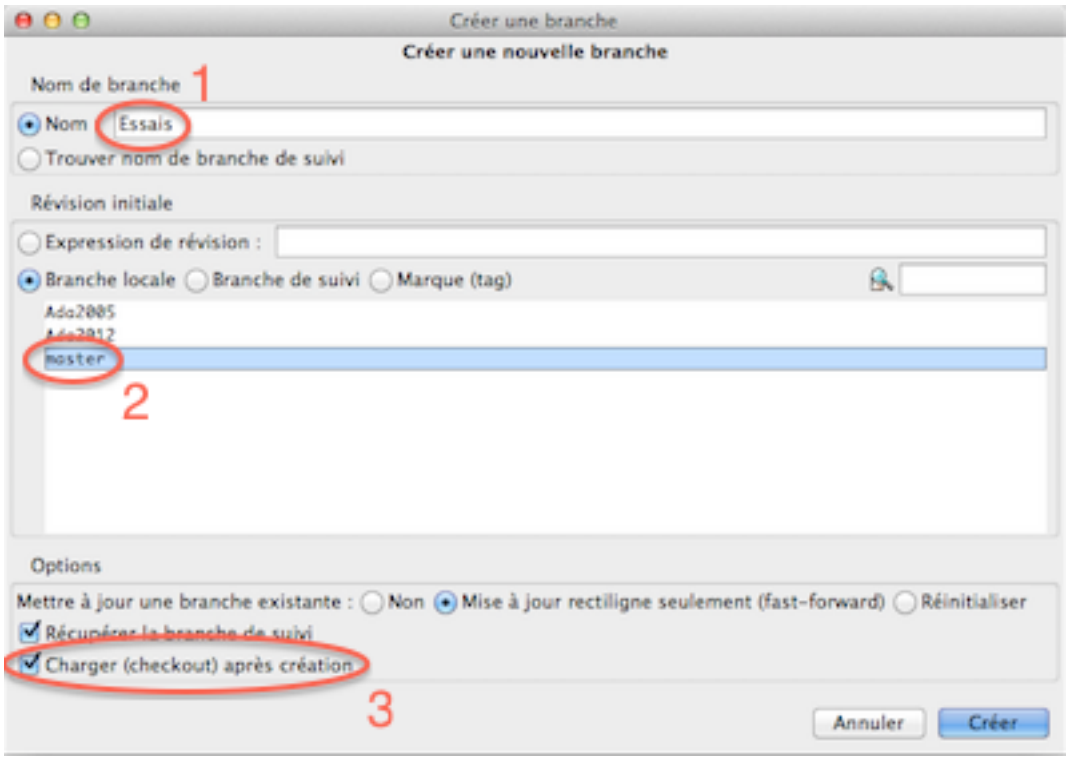


Figure 19 : panneau de création d'une branche

Une branche peut être récupérer ensuite en sélectionnant le menu "Branche -> Charger (checkout)...". Elle peut être supprimée ou renommée toujours avec le menu Branche.

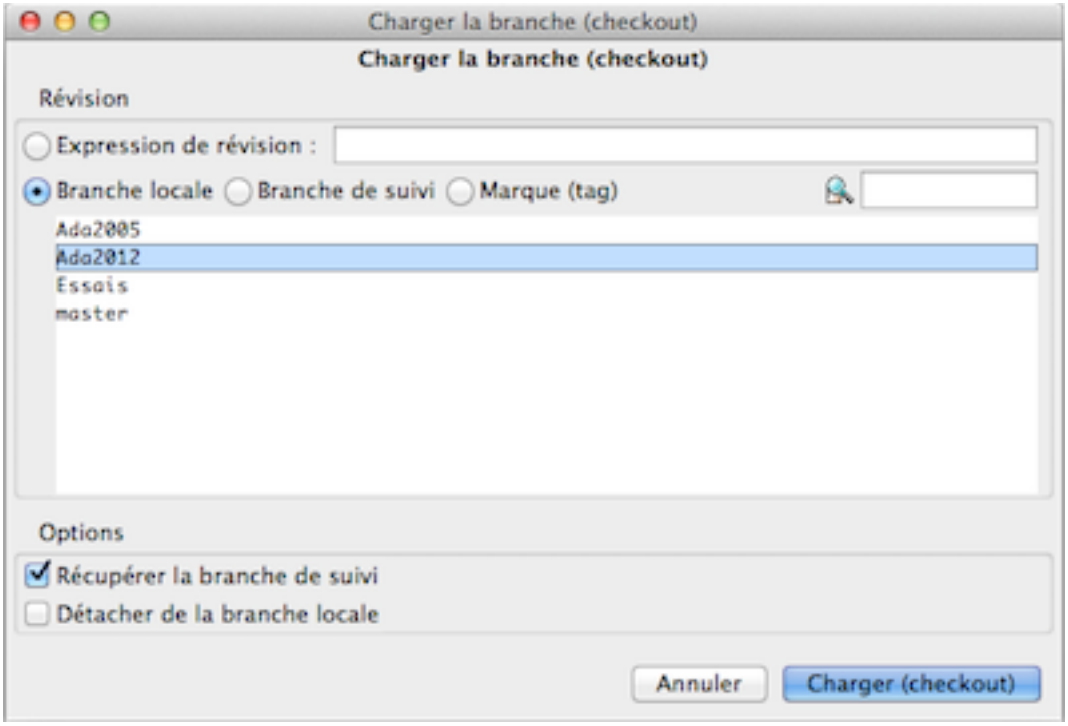


Figure 20 : panneau de récupération d'une branche

6) Création d'un tag

Le tag est un raccourci posé sur un commit enregistré pour pouvoir le récupérer très facilement avec le même menu que les fenêtres (voir figure 20).

Par contre, pour créer un tag, nous devons utiliser GITK. Faites un clic-droit sur le commit choisi et sélectionner "Créer tag". Dans la fenêtre qui s'affiche, entrez le nom du tag puis cliquez sur le bouton Créer. C'est tout, notre tag s'affiche sur fond jaune dans l'historique de modifications :

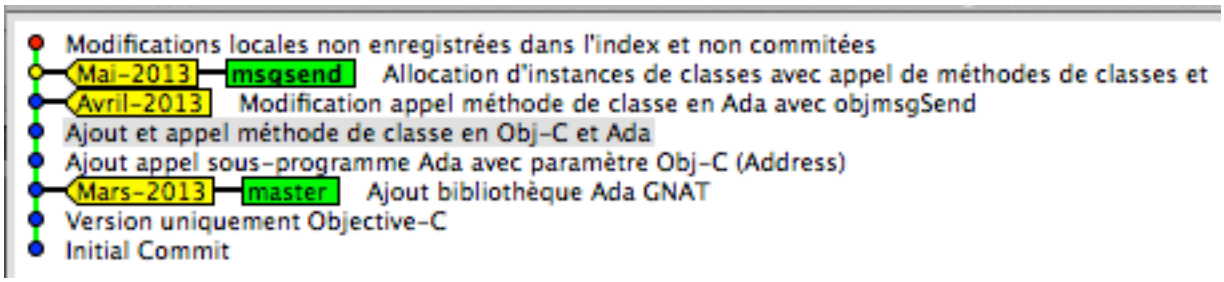


Figure 21 : l'historique des modifications avec 3 tags

Pascal Pignard, mars-mai 2013.