1 Introdução

Todo o trabalho prático da disciplina, centrado no desenvolvimento de software, é suposto ser desenvolvido cooperativamente e concorrentemente pelos diferentes elementos do grupo de trabalho constituído para o efeito. De forma a facilitar esta tarefa, vão utilizar o Git, um sistema de controlo de versões.

Os objetivos deste trabalho prático são que:

- os alunos pratiquem a utilização do Git para realizar o controlo de versões, consolidando os conceitos apresentados na aula teórico-prática
- os alunos se familiarizem com a utilização do Git para realizar o controlo de versões a partir do Eclipse IDE
- cada grupo se familiarize com um processo que vai usar em todas as entregas do trabalho prático da disciplina

A realização deste trabalho pressupõe o registo prévio do grupo, o qual deve ser feito através de `http://git.alunos.di.fc.ul.pt/css000/registo_de_grupos`.

2 Que devemos fazer?

Configuração do Git

Cada elemento do grupo tem de configurar o Git no ambiente onde vai realizar o trabalho prático de CSS (por exemplo, nos labs do DI ou nas suas máquinas pessoais).

1. Abra uma janela do terminal no Linux.
2. Configure o email e nome no Git com

```
$ git config --global user.name "PRIMEIRO-NOME ULTIMO-NOME"
$ git config --global user.email "EMAIL-DE-ALUNO"
```

3. Inspecione o conteúdo do ficheiro `.gitconfig` na sua *home*. Alternativamente pode utilizar o comando `git config --global -l`.
4. Se não estiver definido um ficheiro como sendo o `excludesfile` pode fazer

```
$ git config --global core.excludesfile ~/.gitignore_global
```

5. Edite o ficheiro que está definido como sendo o `excludesfile` de forma a definir que certos ficheiros não são adicionados a qualquer repositório git (pode definir regras mais finas para cada repositório, editando o ficheiro `.gitignore` respetivo).

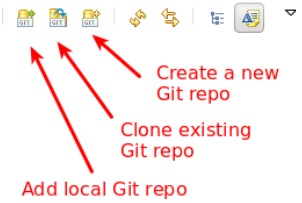
```
# ignore tmp, log, class and war files
*.tmp
*.log
*.class
*.war
```

A configuração do Git pode ser feita alternativamente com o Eclipse (*Preferences* > *Git* > *Configuration* > *User Settings*).

Instalação do Eclipse IDE for Java EE developers

Se seguiu as instruções dadas na primeira aula TP, nesta altura já deve ter instalado o Eclipse para Java EE developers. Entre outras coisas, este inclui a integração do Git para o Eclipse que vai ser exercitada neste TPC. O que a seguir se apresenta corresponde ao que está disponível no Eclipse Oxygen 3a; em versões mais recentes pode haver pequenas variações.

Criação e Utilização de Repositório Local

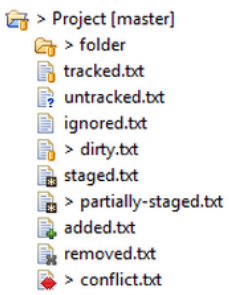


Através da consola ou do Eclipse experimente replicar o que vimos na aula TP:

1. Criar repositório local
2. Criar dois ficheiros de texto na *working directory*
3. Juntar alterações à *staged area*
4. Fazer *commit*
5. Ver o histórico
6. Fazer mais uma alteração a um dos ficheiros e colocar no repositório
7. Ver o histórico
8. Criar um ramo novo
9. Juntar três novos ficheiros ao ramo novo (note que tem de fazer uns *commits* e uns *checkouts*)
10. Juntar um novo ficheiro ao ramo *master*
11. No *master* fazer *merge* do ramo novo

Se usar o Eclipse convém começar por abrir uma perspetiva Git

Window > Open Perspective > Other > Git

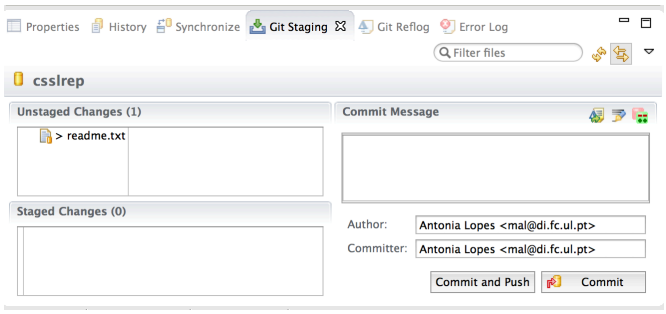


Nessa perspetiva, entre outras coisas úteis para o ajudar no seu trabalho com os repositórios, vai ficar com acesso direto à criação e clone de repositórios. Para criar fazer *File > New > Other > Git > Git Repository* ou então premir o botão *Create a new Git repo* como indicado na figura acima. Para clonar e importar de imediato os projetos para o *workspace* do Eclipse fazer *File > Import > Git > Projects from Git* (mais à frente é descrita uma forma alternativa de fazer clone, que é mais simples quando se tem no repositório projetos Java).

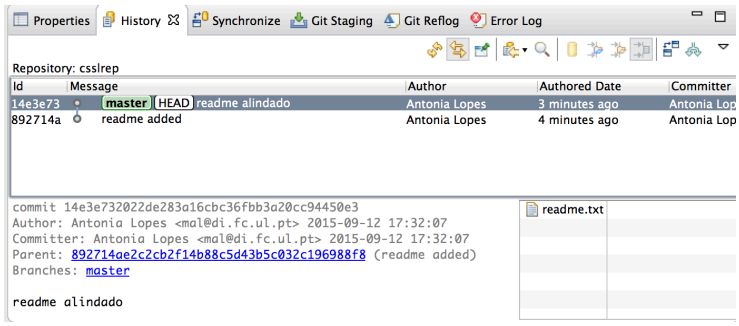
Depois do primeiro *commit* no repositório deve também explorar a integração do Git no *project explorer*. Possivelmente tem de começar por fazer

File > Import > Git > Projects from Git > Existing local repository

para que a *working directory* do projeto *git* lhe apareça neste explorador. Familiarize-se com a forma como o Eclipse representa o estado de cada ficheiro (figura acima) e veja como no menu de contexto através de *Team* tem disponíveis os comandos mais importantes.

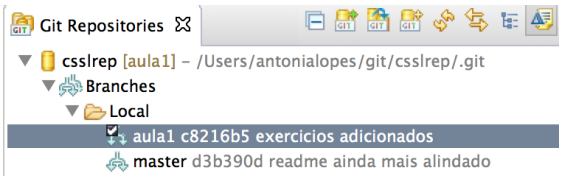


Explore depois o que existe na perspetiva Git nomeadamente os *tabs Git Staging* e *History*.



Explore os menus de contexto que existem quando se seleciona determinado *commit* na *tab History* (por exemplo, *checkout* e *create branch*) e quando se seleciona num determinado ficheiro de um *commit* (por exemplo, *compare with previous version*).

Depois de ter criado o ramo novo, veja o resultado no explorador do repositório e veja que pode mudar de ramo (*checkout*) dando dois cliques sobre o ramo e que a operação de *merge* com outro ramo está disponível no menu de contexto.



Na perspetiva Java do Eclipse, crie um projeto Java e, lá dentro, coloque uma classe Java à sua escolha. Depois no menu de contexto sobre o projeto faça

> *Team* > *Share Project*

indicando o repositório local (para este ficar também sujeito ao controle de versões).

Criação de Repositório no GitLab e dar acesso ao mesmo

Cada grupo deve criar um repositório no servidor GitLab do DI. Esta tarefa deve ser realizada **apenas por um** dos elementos do grupo.

1. Para aceder ao servidor do DI de GitLab abra o *browser* e introduza o endereço

   `http://git.alunos.di.fc.ul.pt/`

   Se não estiver nas máquinas dos laboratórios do DI, precisa de ligar a VPN (ver instruções em https://ciencias.ulisboa.pt/pt/vpn#toc2).

2. Introduza as suas credenciais e, uma vez já dentro do GitLab, crie um novo repositório no servidor GitLab escolhendo a opção *New Project* (lado direito em cima) e atribuindo-lhe o nome cssXXX em que XXX corresponde ao número do grupo. Defina a visibilidade como *private,* visto que os outros grupos NÃO PODEM ter acesso ao projeto.

Após a criação do projeto, o GitLab dá-lhe as instruções de como pode clonar esse projeto usando *http* (veja a próxima seção). Em alternativa pode também aceitar o conselho dado pelo GitLab: *"The repository for this project is empty. If you already have files you can push them using command line instructions below. Otherwise you can start with adding README file to this project."*. Clique no "*adding README*".

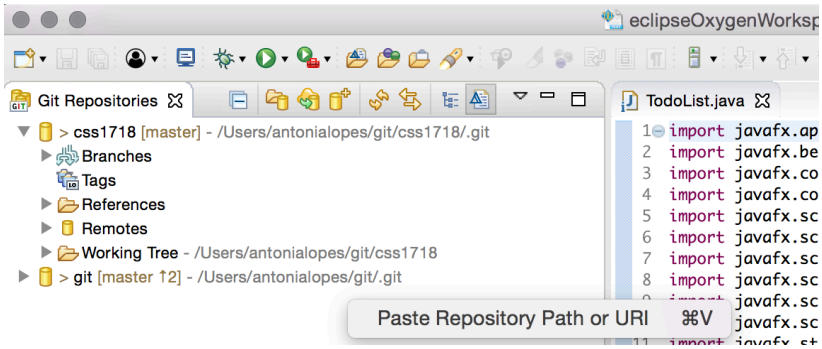
3. De forma a que os seus colegas de grupo possam ter acesso ao repositório criado, terá de dar permissão aos seus colegas de grupo (*maintainer*).

4. De forma a que os docentes da disciplina possam ver e avaliar o trabalho que vai guardar neste repositório, tem de adicionar o utilizador css000 como um membro do seu projeto. Para tal, no GitLab, deve selecionar o repositório e depois a opção *Settings* > *Members* do menu à esquerda; premir o botão *Add members* e adicionar o utilizador css000 com a permissão de *reporter*.

Utilização do Repositório Local + Repositório remoto

A ideia será replicarem o que fizémos na aula TP:

1. Clonar o repositório que criou no GitLab. Comece por copiar o endereço (http) do repositório. De dentro do Eclipse o mais fácil é, na perspetiva Git, usar a opção *paste repository path* acessível no menu de contexto. Desta forma o repositório não fica um projeto Eclipse, e vamos poder colocar lá dentro projetos Java.



2. Criar uma diretoria com o seu nome, um ficheiro lá dentro, adicioná-lo ao repositório local e empurrar (*push*) a mudança para o repositório remoto (a origem). Não se esqueça de associar o projeto ao repositório que acabou de clonar. Para tal, use as opções do menu de contexto *Team*.

3. Fazer mais umas mudanças nessa diretoria, fazer *commit* no repositório local e empurrar para o repositório.

4. Na sessão do GitLab que abriu no *browser* pode ir vendo o resultado das suas atividades.

Utilização do Repositório Local + Repositório remoto partilhado

Juntamente com os seus colegas de grupo, faça umas experiências semelhantes às que vimos na aula TP.

1. criar um ficheiro com o seu nome, adicioná-lo ao repositório local e empurrar a mudança para o repositório remoto (a origem);
2. trazer para o repositório local as mudanças que, entretanto, tenham sido feitas;
3. escolher com um colega um ficheiro já existente para modificar em simultâneo;

4. modificar o ficheiro escolhido, fazer *commit* no repositório local e empurrar para a origem;
5. se o seu colega empurrou as alterações dele primeiro, não vai conseguir empurrar as suas porque vai ter um conflito; chame o seu o colega para o ajudar a resolver o conflito, faça *commit* no repositório local e empurre para a origem;
6. repita os passos 1–5, mas sendo um dos seus colegas a usar o seu repositório (ou vice-versa).

Explore a interface *web* do GitLab para aceder à informação sobre o repositório. Veja nomeadamente os menus: *commits*, *network*, *activity* e *files*.

3 Como e quando entregamos?

Executaram os passos descritos anteriormente e empurraram o vosso repositório para o repositório remoto no GitLab? Está quase entregue. Identifiquem o *commit* como sendo a entrega final (`git tag entrega1`) e coloquem essa identificação no servidor gitlab (`git push origin entrega1`). Consideramos apenas este *tag*, se o *commit* tiver uma data anterior a **8 de março**, o *deadline* para entrega.