Manual LAPR4

LAPR4 / LEI

Version 0.2

Alexandre Bragança
School of Engineering / Polytechnic Institute of Porto / Portugal,
atb@isep.ipp.pt,
www.isep.pt

Maio 2013

Revision History

Revision	Date	Author(s)	Description
0.2	13-05-2013	atb	correcções ortográficas e nota "Issues e commits quando temos 2 repositórios no grupo"
0.1	01-03-2013	atb	versão inicial publicada

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Este documento consiste num guia para o aluno de LAPR4, edição de 2012/13.

Pretende-se que este documento seja suficientemente completo de forma a ajudar o aluno a organizar o seu trabalho. É no entanto muito provável que questões novas possam surgir que não estão cobertas pela versão actual do documento. Nesse caso o aluno deve fazer uso dos fóruns do moodle dedicados ao projecto de LAPR4 ou contactar directamente algum dos docentes de LAPR4.

Relativamente ao documento todas as sugestões e criticas podem ser enviadas para atb@isep.ipp.pt.

É importante que todos os alunos leiam a totalidade deste documento antes de iniciarem o projecto.

1.1 Organização do manual

O Capítulo 2 descreve como é que se executam as operações mais significativas relativas à utilização do Bitbucket e, particularmente, à gestão de "issues" no contexto do projecto.

O Capítulo 3 é dedicado às ferramentas que os alunos podem usar nos seus computadores pessoais para trabalharem com as suas cópias dos repositórios. O projecto pode ser desenvolvido sem a utilização de qualquer IDE. No entanto, no Capítulo 3 descreve-se como se pode utilizar o NetBeans e o Eclipse. Qualquer um destes IDE funciona em Windows, Linux e Mac OSX. Também é apresentada a utilização do SourceTree, um ambiente gráfico para manipulação de repositórios Git e Mercurial. Neste caso, a ferramenta apenas está disponivel actualmente para Windows e MAC OSX.

O Capítulo 4 é dedicado a explicar as tarefas fundamentais que os alunos terão de executar no contexto do projecto. Apresentam-se também sugestões quando à organização do trabalho individual e do grupo.

O Capítulo 5 descreve a avaliação do projecto.

1.2 Contexto do Projecto

O projecto de LAPR4 tem um objectivo que se enquadra no âmbito das outras unidades curriculares de Laboratório/Projecto da LEI, tal como definido na Ficha da Unidade Curricular.

Em particular, este LAPR tem as seguintes caracteristicas que o destinguem dos outros:

- forte ênfase no processo de desenvolvimento de software justificado pelo facto de decorrer no mesmo semestre que EAPLI. Neste contexto, é valorizado o processo usado para se obter o produto (software) em detrimento do produto em si. Espera-se que o aluno consiga aplicar as melhores práticas estudadas em EAPLI, aplicando um processo ágil e iterativo. Para além dos princípios de análise e concepção de código "object oriented" espera-se que o aluno demonstre maturidade na utilização de sistemas distribuídos de controlo de versões;
- integração do aluno num projecto em curso. Pretende-se que o aluno seja confrontado com um trabalho em equipa (como acontece noutros LAPR) mas, neste caso, terá ainda de fazer o seu trabalho sobre um projecto em curso, reutilizando trabalho efectuado anteriormente por outras pessoas.
- na sequência do módulo de gestão de projectos é esperado que o aluno demonstre capacidades de planeamento e gestão de projectos aplicando princípios ágeis;
- os requistos do projecto a desenvolver implicam ainda a aplicação de conceitos, técnicas e práticas das outras unidades curriculares do semestre: SCOMP, RCOMP e LPROG. Em particular pretende-se que o aluno aplique técnicas de comunicação entre processos (e em rede); técnicas de programação paralela e sincronização e especificação e programação de analisadores léxidos, gramáticas e "parsers".

Em função das caracteristicas apresentadas o projecto de LAPR4 de 2012/13 é baseado em:

- utilização do sistema distribuído de controlo de versões Git [5] (o mesmo que é usado em EAPLI);
- utilização do site Bitbucket para manter a cópia central do repositório de cada grupo [2];
- utilização do sistema de controlo de "issues" do Bitbucket para registo de todas as actividades relacionadas com o projecto;
- os requisitos do projecto consistem em novas funcionalidades sobre o projecto Cleansheets (folha de cálculo desenvolvida em linguagem java). O Cleansheets que iremos usar em LAPR4 sofreu algumas alterações relativamente à versão base da autoria de Einar Pehrson. A versão a usar em LAPR4 pode ser obtida num repositório público do Bitbucket [7];
- os alunos devem formar grupos de 3 a 5 elementos (da mesma turma PL). Cada grupo deve ter um chefe de grupo;
- existe um chefe de projecto (atb@isep.ipp.pt) que é o docente recente da UC e que assumirá a representação do "cliente". As dúvidas sobre os requisitos devem ser tiradas junto deste através de um fórum que será dispoibilizado no moodle para esse efeito;
- existe um supervisor do grupo que será um dos docentes de LAPR4. Este docente acompanhará de perto o grupo e os seus elementos no que se refere ao processo de desenvolvimento de software e, particularmente, à utilização dos repositórios e à gestão de "issues";
- existem ainda especialistas (docentes que leccionaram UC do semestre) que estão disponíveis para tirar dúvidas técnicas e também assumem o papel de avaliadores (juntamente com o chefe de projecto e os supervisores).

Na secção seguinte apresentam-se alguns detalhes sobre a organização das equipas.

1.3 Organização das equipas

Os alunos devem formar equipas entre 3 a 5 elementos pertencentes à mesma turma PL. Cada aluno deve criar a sua conta no Bitbucket. Os grupos com apenas 3 elementos devem ter um repositório (fork do Cleansheets) no Bitbucket que vão usar durante o projecto. Os grupos com 4 ou 5 elementos devem ter dois repositórios do Cleansheets no Bitbucket. No Capítulo 2 encontram detalhes sobre como organizar os repositório para ambos os casos.

Os novos desenvolvimentos sobre o Cleansheets vão ser feitos em 3 áreas: persistência (mais focalizado em matéria de EAPLI), linguagens (mais focalizado em matéria de LPROG) e partilha de folha (mais focalizado em matéria de SCOMP e RCOMP). Como os grupos podem ter entre 3 a 5 alunos vão existir 5 trilhos: 2 trilhos na área da persistência; 2 trilhos na área das linguagens e 1 trilho na área da partilha de folha. Os requisitos para cada semana serão apresentados aos alunos num documento dividido nos 5 trilhos. Cada aluno em cada semana assume a responsabilidade de um dos trilhos. Um grupo com 3 alunos apenas trabalha em 3 trilhos; um grupo com 4 alunos apenas trabalha em 4 trilhos e um grupo com 5 alunos trabalha em 5 trilhos. Assim, um aluno ao longo das 3 semanas terá trabalhado num trilho em cada semana. Em termos de avaliação será valorizada a diversificação de áreas. Assim, o ideal é que cada aluno consiga trabalhar em 3 trilhos diferentes, se possível em 3 áreas diferentes.

O chefe do grupo deve submeter inicialmente a constituição do grupo, a conta do bitbucket de cada elemento do grupo e o url dos repositórios do grupo. Devem dar acesso aos repositórios ao regente (atb@isep.ipp.pt) e ao docente supervisor (esta informação estará num endereço a anunciar). A submissão do grupo será efetuada num link do moodle a anunciar.

Capítulo 2

Repositórios

Neste capítulo apresenta-se um guia para ajudar o aluno a entender a sequência de tarefas a executar para estabelecer adequadamente os repositórios do projecto. Os repositórios são em formato Git [5] e para o seu alojamento e gestão de "issues" será utilizado o site Bitbucket [2].

2.1 Grupos e Repositórios

Os grupos devem ter apenas alunos da mesma turma PL. Os grupos devem ser constituídos por 3, 4, ou 5 alunos.

O projecto consiste na evolução de uma aplicação existente pelo que será necessário que o repositório inicial de cada grupo seja construido a partir de um repositório que tem a versão inicial da aplicação.

Para se entender melhor o fluxo de trabalho será apresentada uma pequena simulação de um grupo de trabalho constituído por 5 alunos.

Para além dos alunos cada grupo será seguido por 2 docentes: o professor orientador (que é um dos professores da turma PL, que irá acompanhar o grupo) e o regente da Unidade Curricular.

Consideremos então os seguintes elementos para este tutorial:

nome	**email**	**descrição**
atb	atb@isep.ipp.pt	regente da UC
abraganca	abraganca@gmail.com	professor orientador
alexandre_isep1	alexandre.isep1@gmail.com	aluno 1, **dono do site 1**, "normal"
alexandre_isep2	alexandre.isep2@gmail.com	aluno 2, **dono do site 2**, **"chefe/integrador"**
alexandre_isep3	alexandre.isep3@gmail.com	aluno 3, "normal"
alexandre_isep4	alexandre.isep4@gmail.com	aluno 4, "normal"
alexandre_isep5	alexandre.isep5@gmail.com	aluno 5, "normal"

Relativamente à estrutura dos grupos é necessário considerar duas estruturas:

- **um site**: quando a equipa apenas tem 3 alunos;
- **dois sites**: quando a equipa tem 4 ou 5 alunos.

Na prática, os grupos maiores irão simular dois sites (locais) de desenvolvimento. Cada site terá o seu repositório. Os 2 repositórios deverão ser integrados regularmente. Para grupos de 4 alunos, cada repositório deve ter 2 alunos. Para grupos de 5 alunos, um dos repositórios tem 3 alunos e o outro 2.

2.2 Passo 1: Registo de Utilizadores no Bitbucket

Cada aluno deve registar uma conta pessoal no Bitbucket. Este é o site no qual está hospedado o repositório original do projecto.

Se o aluno já tiver uma conta no Bitbucket deve criar outra!

De preferência o nome do utilizador e o email associado à conta do Bitbucket devem ser os do ISEP. Por exemplo, o aluno Joao com o email 9988999@isep.ipp.pt deve registar-se com os seguinte dados:

- **Username**: joao_9988999;

- **Email**: 9988999@isep.ipp.pt.

Caso o aluno já tenha essa conta no Bitbucket e tenha lá repositórios então deve usar outra conta. Pode, por exemplo, criar uma conta no Google Mail (ou noutro site) e usá-la no registo no Bitbucket. Mais uma vez, o email deve conter o número do aluno no ISEP. Por exemplo:

- **Username**: joao_9988999;
- **Email**: joao_9988999@gmail.com.

Porquê contas do Bitbucket sem repositórios?

As contas de utilizadores que vão ser usadas no projecto são apenas para LAPR4. Vai ser necessário dar acesso a dois docentes: o regente e o orientador. Estes docentes apenas podem ter acesso ao que for desenvolvido no âmbito de LAPR4. Para além disso, os repositórios que os alunos criarem para LAPR4 devem ser privados, apenas com acesso aos elementos do grupo e aos dois docentes referidos anteriormente.

Neste passo todos os utilizadores se registam no Bitbucket. Para os efeitos deste tutorial vamos considerar as contas de alunos apresentadas na tabela anterior.

2.3 Passo 2: Fork do repositório original

Este passo deve ser executado pelo aluno do grupo que for assumir o papel de dono (owner) do primeiro repositório do grupo. Segundo a tabela de utilizadores anterior, será o utilizador alexandre_isep1.

Este utilizador terá de pesquisar pelo repositório do projecto original. Para isso, dentro do Bitbucket, pode usar a área de pesquisa que aparece no canto superior direito e introduzir a seguinte string "atb/cleansheets 2012/2013 Private"[1]. Após encontrar e seleccionar este repositório o Bitbucket irá apresentar uma página semelhante à da figura seguinte. Nesta altura o utilizador deverá seleccionar a opção "Fork", tal como está apresentado na figura.

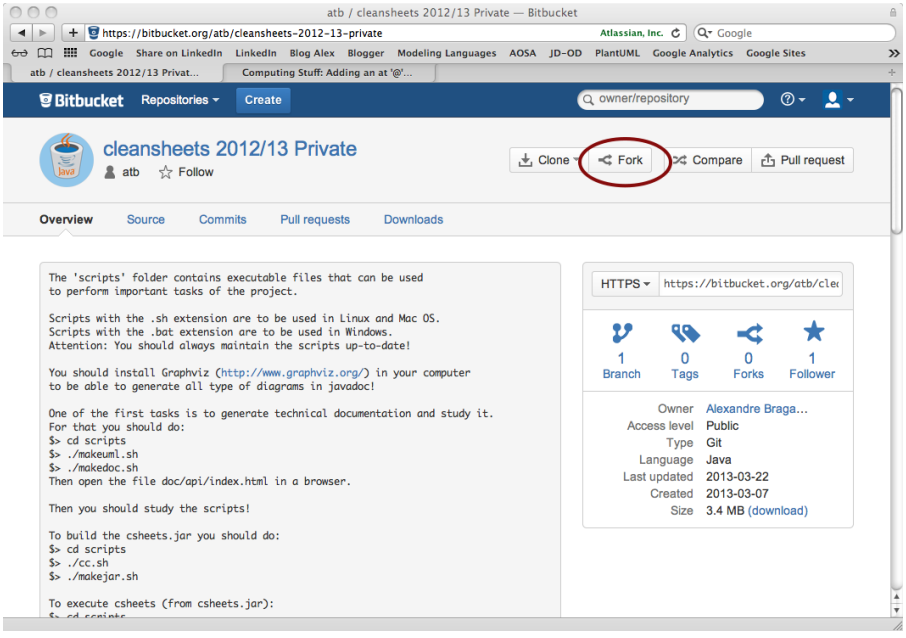

Figura 2.1: Fork do repositório original

Para se criar um fork no Bitbucket é necessário introduzir alguns dados, como o nome e uma pequena descrição do projecto. Como nome sugere-se um que permita identificar claramente o repositório. Para isso deve conter a turma PL, o número do grupo e o número do site (1 ou 2). Na figura seguinte podemos observar um exemplo. Neste exemplo o nome dado ao novo repositório é "lapr4_2de_g1_s1". **As outras opções devem ser preenchidas tal como apresentado na Figura 2.2**. Em particular não se pretende herdar permissões e pretende-se activar o "issue tracking" no novo repositório.

[1] Apesar de no nome aparecer a palavra Private este é um repositório público.

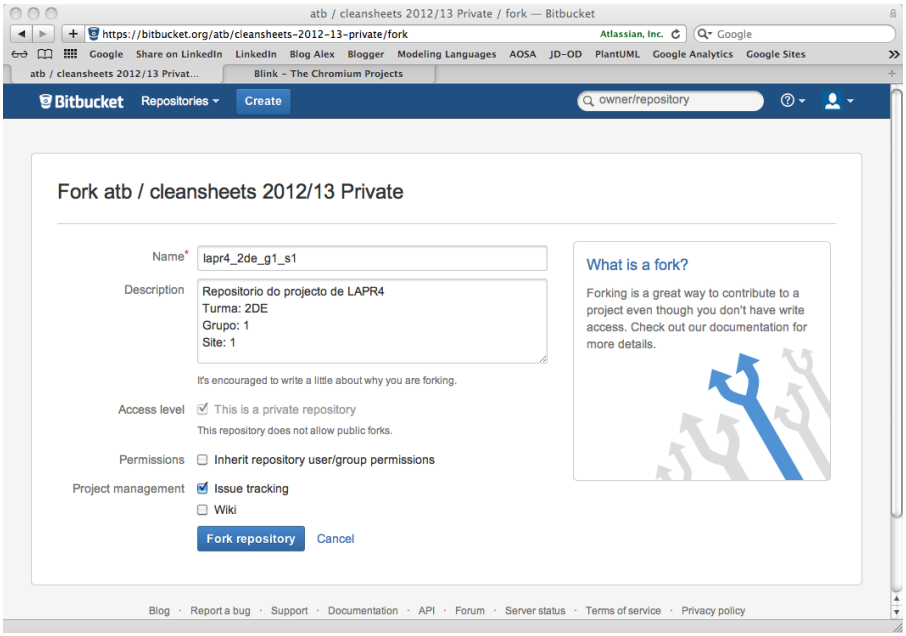

Figura 2.2: Introdução de dados para o primeiro Fork

A Figura 2.3 apresenta a página do Bitbucket após o fork bem sucedido. Podemos verificar que o url alterou-se refletindo o dono (owner) assim como o nome do novo repositório. Pode-se ainda ver que o Bitbucket mantém a relação deste repositório com o original e, ainda, que o novo repositório é privado.

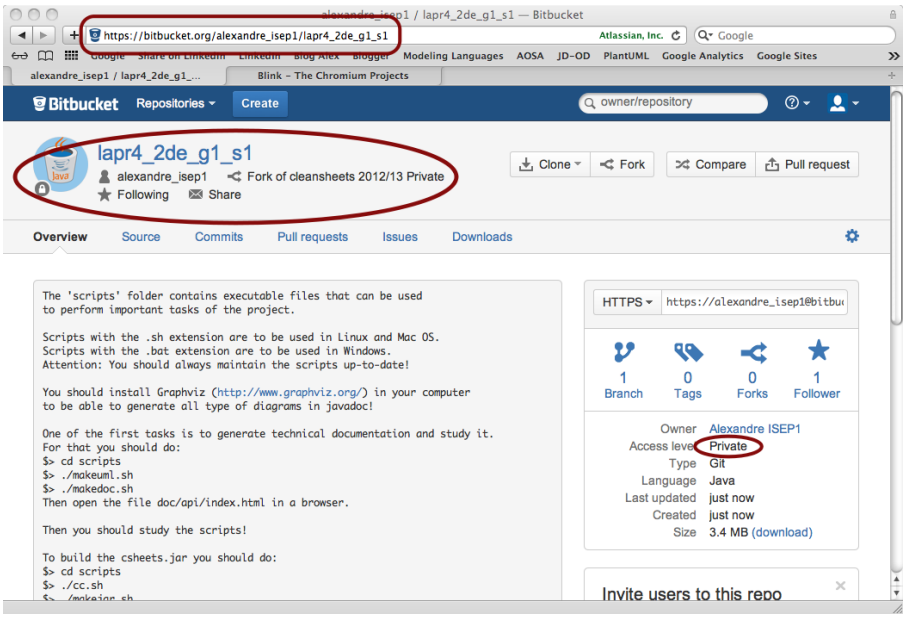

Figura 2.3: O repositório resultante do primeiro Fork

Completamos assim o segundo passo deste tutorial. De acordo com o diagrama apresentado na Figura 2.4 efectuamos o fork que nos permite criar o primeiro repositório do grupo. Falta ainda criar o segundo repositório do grupo. No entanto, este passo que falta é muito importante pois é necessário que seja feito pelo aluno "chefe" do grupo. Este aluno será o único que terá acesso aos dois repositórios do grupo.

É então necessário que o utilizador alexandre_isep1 (dono do repositório actual) atribua direitos de acesso ao seu repositório.

2.4 Passo 3: Atribuição de acessos ao primeiro repositório

Para se atribuir acessos ao repositório é necessário aceder às opções de administração do Bitbucket. Para tal deve-se seleccionar o respectivo ícone, tal como identificado pelo "1º" na Figura 2.5. De seguida selecciona-se a

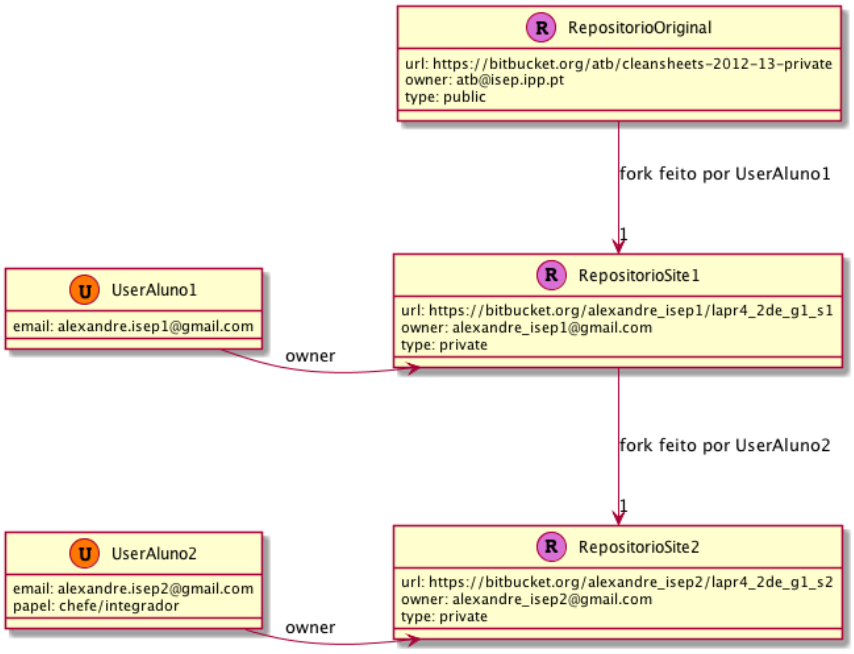

Figura 2.4: Sequência de criação dos repositórios do grupo

opção "Access management". Acede-se então a uma área onde se pode gerir os acessos ao repositório. Inicialmente apenas é apresentado o dono e único utilizador do repositório. A Figura 2.6 apresenta um diagrama com os acessos para o primeiro repositório do grupo.

Para este primeiro repositório do grupo é necessário definir os seguintes acessos:

- Acesso de Leitura: para os dois docentes que necessitam de aceder ao repositório: o regente (atb) e o orientador (neste caso o utilizador abraganca)
- Acesso de Escrita: para os outros 2 alunos que vão trabalhar no primeiro repositório do grupo: o alexandre_isep2 e o alexandre_isep3.

O resultado dessa configuração será o apresentado na Figura 2.7.

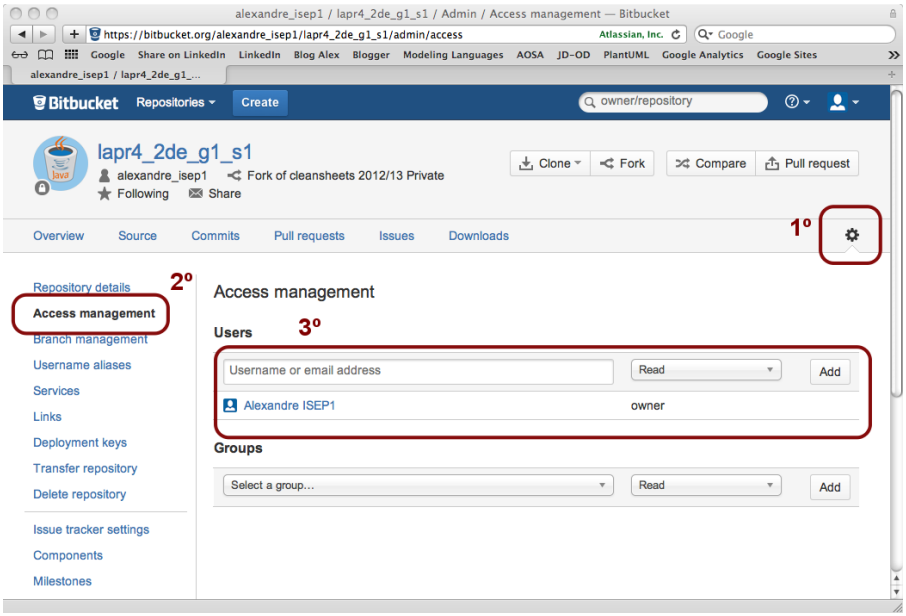

Figura 2.5: Página de atribuição de acessos ao primeiro repositório

Uma vez que o utilizador alexandre_isep2 (que é o chefe do grupo e deve ter acesso aos dois repositórios) já tem acesso ao primeiro repositório este pode agora proceder à criação do segundo repositório do grupo.

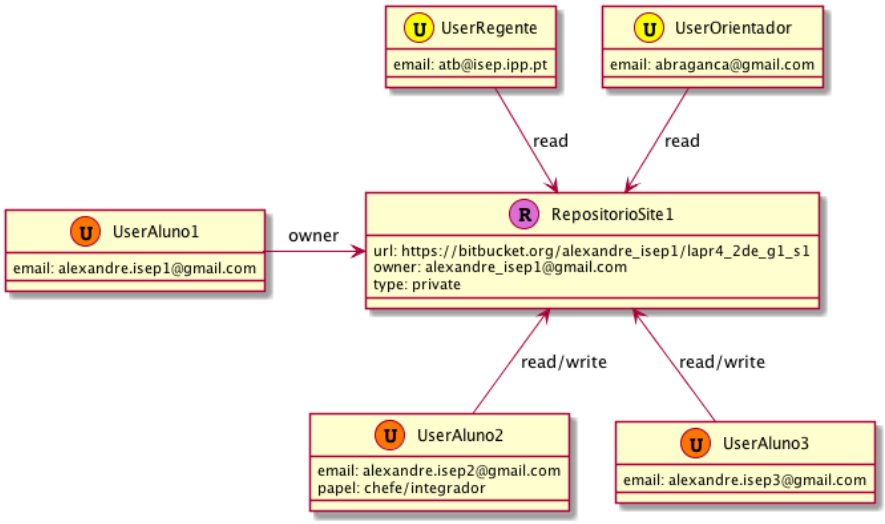

Figura 2.6: Utilizadores para o primeiro site do grupo

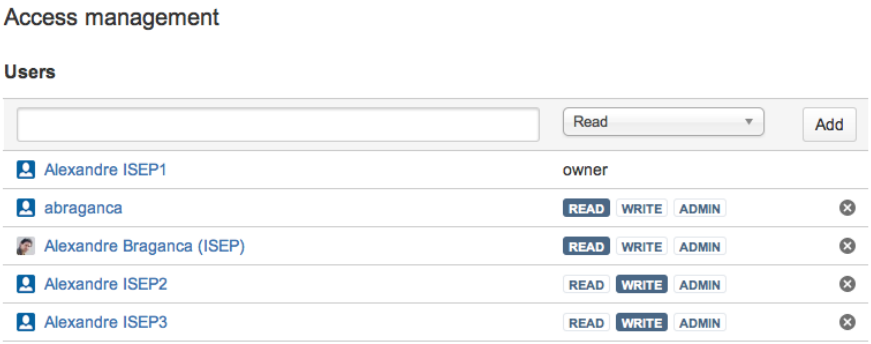

Figura 2.7: Acessos ao primeiro repositório

2.5 Passo 4: Segundo repositório

O chefe do grupo (neste caso, o utilizador alexandre_isep2) que tem neste momento acesso ao primeiro repositório deve fazer o fork deste repositório de forma a criar o segundo repositório do grupo. O chefe do grupo será o único aluno com acesso aos dois repositórios do grupo. Ele terá a responsabilidade de manter os repositórios sincronizados.

O processo é semelhante ao que foi descrito anteriormente. A Figura 2.8 exemplifica como o chefe de grupo deve preencher os dados de criação do segundo repositório do grupo.

Após a criação do segundo repositório o chefe do grupo deve atribuir as permissões. Tal como para o outro repositório, o regente e o orientador devem ter acesso de leitura enquanto os restantes alunos (alexandre_isep4 e alexandre_isep5) acesso de escrita. O resultado é apresentado na Figura 2.9. A Figura 2.10 apresenta o diagrama de acessos para o segundo repositório do grupo.

2.6 Passo 5: Configuração dos "issues"

Um dos aspectos principais ao desenvolvimento do projecto consiste no registo correcto dos eventos associados ao repositório.

O Bitbucket permite de base uma gestão simplificada de "issues". É essa gestão de issues que vai ser usada durante o projecto para o registo de todas as actividades significativas relacionadas com o projecto. Para tal, é necessário que sejam efectuadas algumas configurações iniciais nos repositórios do grupo (se forem 2 é necessário fazer nos 2 repositórios). Deve-se aceder à configuração do Bitbucket, seleccionar "issues tracker settings" e, depois, a opção "Private issues tracket" (tal como ilustrado na Figura 2.11).

O Bitbucket permite configurar 3 conceitos associados aos issues: componentes, milestones e versões.

Para LAPR4 deve ser definida apenas uma versão. Vamos designar a versão que resultará do projecto como "Version 1.5". Devemos definir tal como ilustrado na Figura 2.12

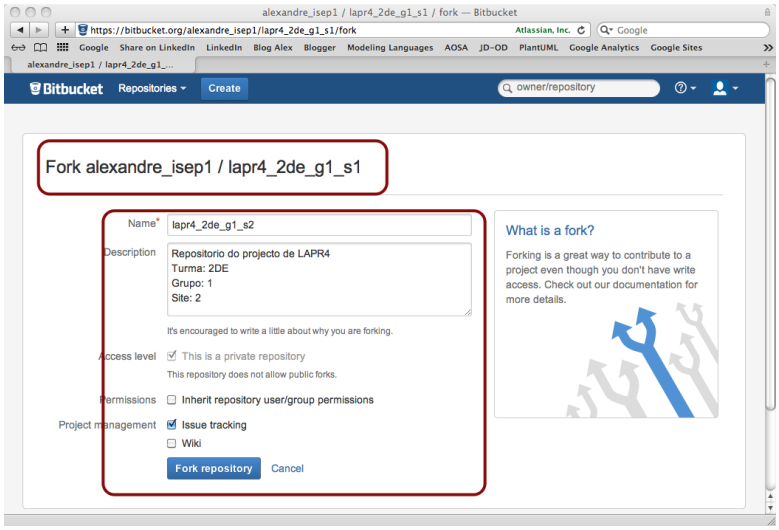

Figura 2.8: Fork para criar o segundo repositório do grupo

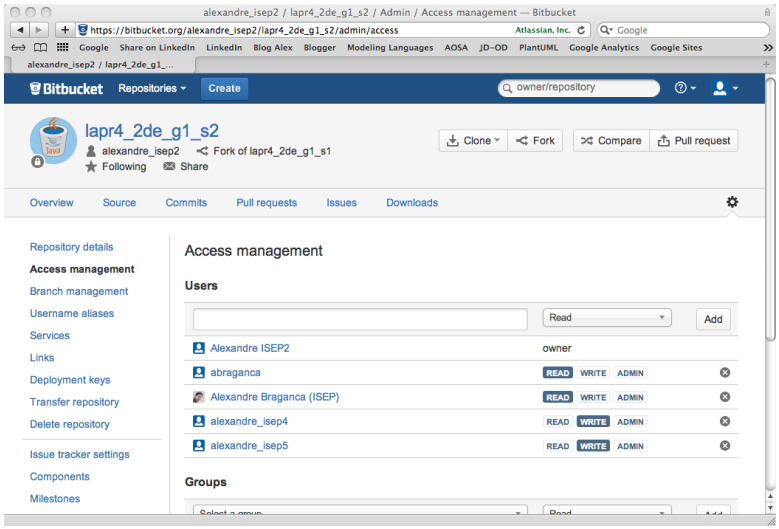

Figura 2.9: Atribuição de permissões para o segundo repositório do grupo

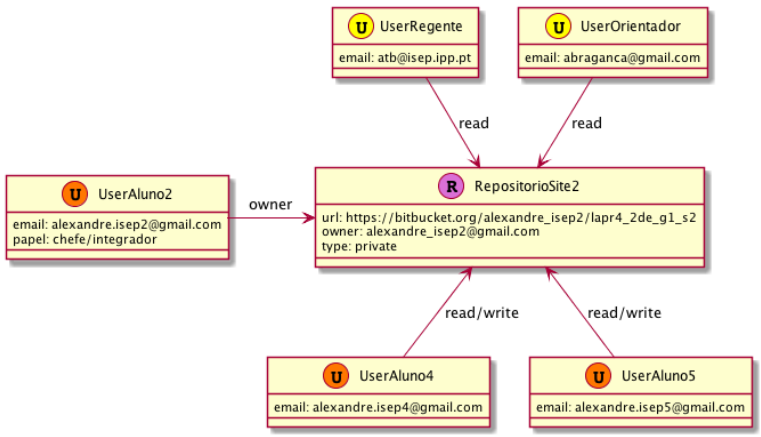

Figura 2.10: Utilizadores para o segundo site do grupo

Relativamente aos milestones devemos configurar 3 milestones, correspondentes às 3 semanas de desenvolvimento do projecto. Esta configuração deve ser realizada tal como ilustrado na Figura 2.13.

Relativamente aos componentes devemos configurar 5 componentes, correspondentes aos 5 trilhos de requisitos do projecto. Esta configuração deve ser realizada tal como ilustrado na Figura 2.14.

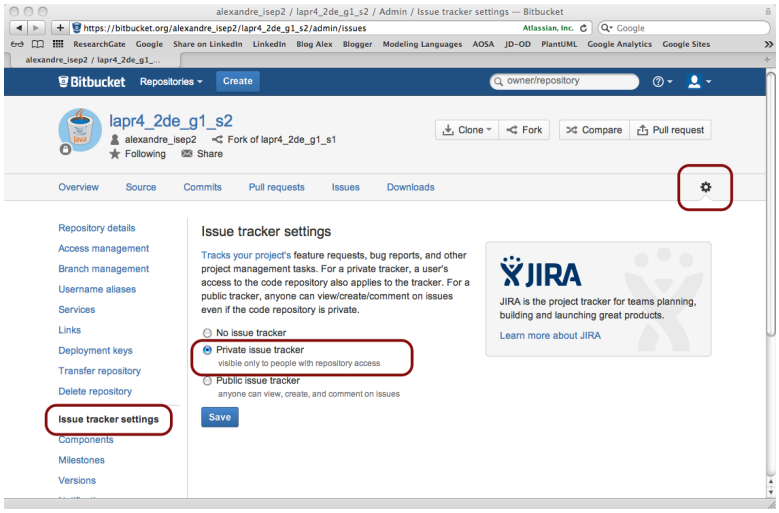

Figura 2.11: Configuração da opção de gestão de issues

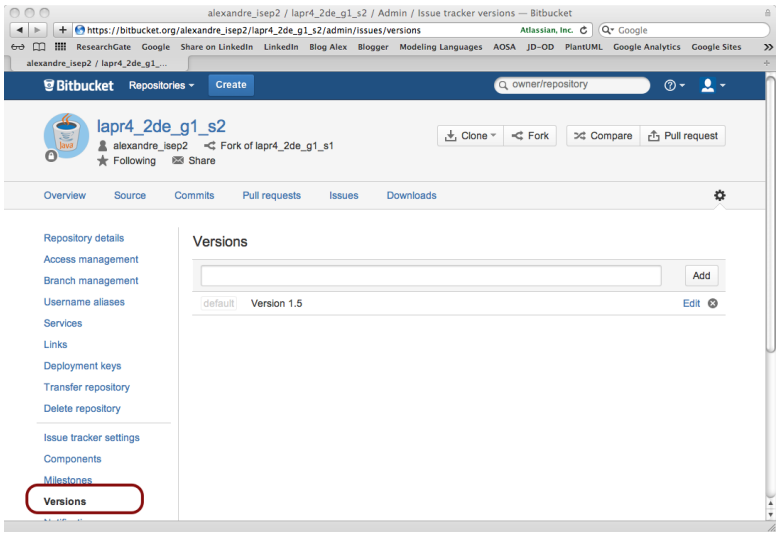

Figura 2.12: Configuração da versão

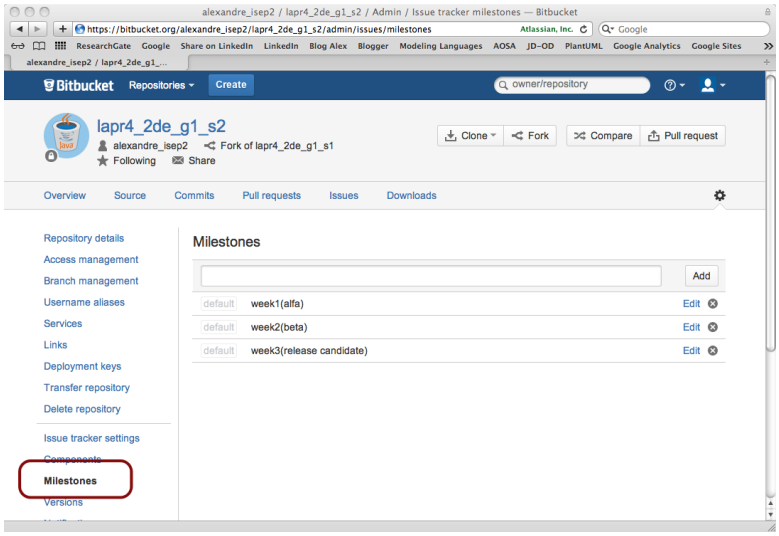

Figura 2.13: Configuração dos milestones

Desta forma, sempre que for criado um issue no Bitbucket, devemos caracterizar estes 3 conceitos: a versão (será sempre a "Version 1.5"); qual o milestone e qual o componente.

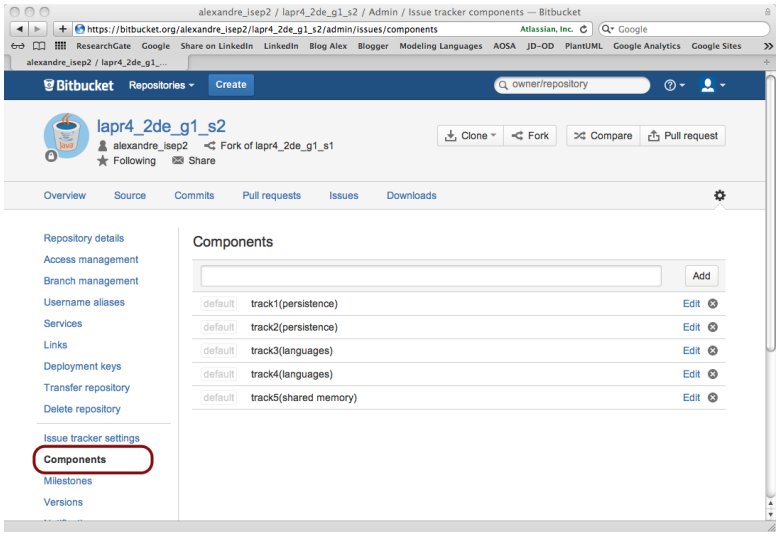

Figura 2.14: Configuração dos componentes

2.7 Passo 6: Criação de um "issue"

Um dos aspectos principais do projecto de LAPR4 será o registo das actividades de cada aluno no âmbito do projecto.

No início de cada semana, e após conhecer o enunciado dessa semana (iteração), no âmbito do planemanto, cada aluno ficará reponsável por um trilho (track). Assim, após a reunião de planeamento do início da semana (segunda-feira), uma das primeira tarefas do aluno será o registo de um issue para o seu trilho dessa semana. Deverá faze-lo tal como é sugerido na Figura 2.15.

O aluno deve escolher um título adequado para o seu issue que reflita o seu trilho. Deve ainda preencher uma descrição que deve conter uma breve enumeração das tarefas planeadas relativas a este issue.

O aluno deve atribuir esse issue a ele próprio.

O aluno deve seleccionar o tipo de issue ("kind") "enhancement" e seleccionar o nível de prioridade.

Deve ainda seleccionar o milestone, componente e versão de acordo com o seu trilho, a semana e a versão (neste caso apenas temos uma).

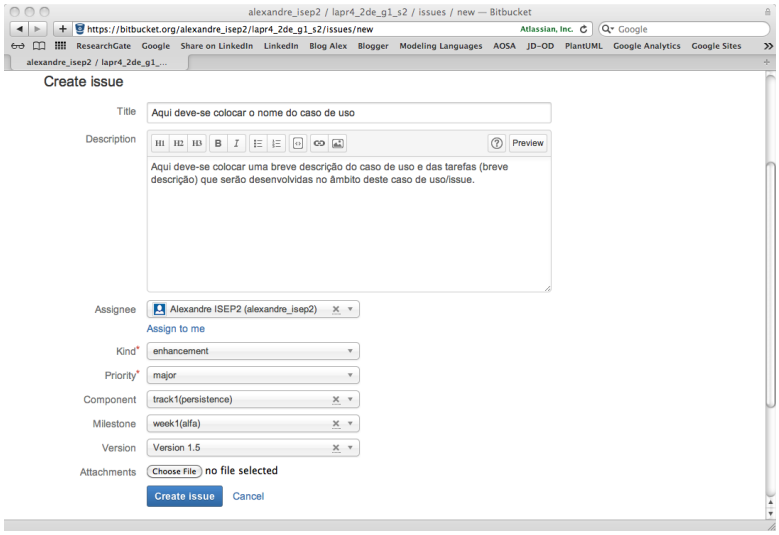

Figura 2.15: Criação de um issue

Podemos aceder aos issues criados através de opção "Issues" do Bitbucket. Com esta opção o Bitbucket mostra-nos uma lista de issues do nosso repositório que podemos filtrar e ordenar para mais facilmente encontrarmos os issues que desejamos. Se clicarmos no link relativo a um issue o Bitbucket irá apresentar o detalhe desse issue, de forma semalhante à que é ilustrada na Figura 2.16.

É particularmente importante o número que identifica o caso de uso. Na ilustração da Figura 2.16 o issue tem o número #1. O número do issue deve ser utilizado sempre que efectamos um commit pois vai permitir

relacionar os commits do repositório com o issue (ou issues) aos quais o commit se refere (ver próxima secção).

Para além dos commits ficarem associados automaticamente a um issue se usarmos a identificação do issue no texto de descrição do commit devemos ainda efectuar comentários no issue (através do Bitbucket) descrevendo eventos ou opções significativas relativas ao issue que são importantes e devem ficar registadas no histórico do issue. Para tal usamos a opção de adicionar um comentário ao issue.

Outra informação importante sobre um issue é o seu estado. O Bitbucket suporta diversos estados relativamente ao ciclo de vida de um issue. Para o caso do projecto de LAPR4 podemos simplicar e resumir apenas a utilização aos seguintes estados:

- new: é o estado de um issue logo após a sua criação.
- open: é o estado em que deve estar um issue quando estamos a trabalhar sobre ele.
- resolved (closed): é o estado de um issue quando o damos por terminado.

De notar que podemos alterar o estados dos issues de 2 formas:

- Bitbucket: utilizando a opção "Wokflow" disponível no Bitbucket ou ainda usando a opção "Edit"
- Commits: sempre que fazemos um commit podemos alterar o estado do issue se usarmos uma sintaxe especial na nossa mensagem de commit (ver secção seguinte)

Outro aspecto importante do issue é a pessoa à qual está atribuido o issue. Quando criamos um issue devemos atribui-lo a um utilizador e mais, tarde, se for necessário, afectá-lo a outro utilizador. Isto pode ser feito através de opção "Edit".

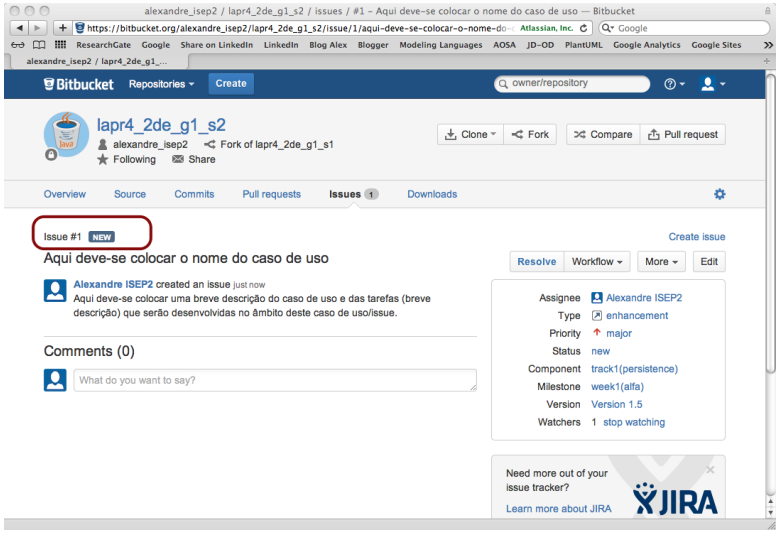

Figura 2.16: Acesso aos issues de um repositório

Deve-se fechar todos os issues antes de submeter o trabalho de uma semana/iteração?

A resposta ideal seria afirmativa. No entanto isso pode não acontecer. Por exemplo, um dos trilhos pode não ter ficado completo mas o projecto de LAPR4 exige que todas as semanas se faça a submissão do projecto. Nessa situação o issue deve ficar aberto.

Issues e Commits quando temos 2 repositórios no grupo

O bitbucket tem um serviço que automaticamente integra os commits com os issues (tal como descrito na secção seguinte). Este serviço é muito interessante mas levanta problemas quando temos 2 repositórios com issues diferentes que têm a mesma referência. O problema surge quando se integram os 2 repositórios no sentido em que podem ser criadas actualizações automáticas na gestão de issues de cada repositório de forma incorrecta. Uma forma de evitar isto é o grupo criar a mesma sequência de issues nos dois repositórios. Por exemplo, em ambos os repositórios criar todos os issues do grupo para essa semana, na mesma sequência. Por exemplo, ficariamos com os issues 1, 2, 3, 4 e 5 nos dois repositórios. No entanto, o issue 1 e 2 apenas é actualizado pelos alunos do repositório 1 e os issues 3, 4 e 5 pelos alunos do repositório 2. Quando o chefe do grupo integrar os dois repositórios o bitbucket vai gerar actualizações de issues "cruzadas" mas, desta forma, elas são sempre disjuntas.

2.8 Passo 7: Efectuar Commits

Se admitirmos o cenário mais vulgar para o projecto teremos que cada aluno irá trabalhar num trilho em cada semana. Para facilitar a gestão do projecto cada trilho em cada semana corresponderá a um issue.

Existem várias tarefas que o aluno poderá necessitar de efectuar no âmbito do trilho. Nem todas as tarefas refletem obrigatoriamente alterações ao repositório. No entanto todas as tarefas devem ficar registadas nos issues do Bitbucket. Por exemplo, se o aluno necessitar de pesquisar, estudar e seleccionar uma biblioteca para usar no projecto isso poderá ficar registado, numa primeira fase, como um comentário no issue (por exemplo, indicando que irá pesquisar e estudar uma solução para um determinado problema) e num momento posterior poderá ficar registado num commit em que se adiciona um "jar" ao projecto relativo à biblioteca seleccionada para resolver o problema.

Em principio os commits devem sempre resultar em incrementos significativos ao projecto. A palavra "significativos" tem por objectivo excluir commits que, por exemplo, apenas acrescentam algumas linhas ao projecto, mas essas linhas não acrescentam nenhuma unidade ao projecto. Por exemplo, não acrescentam uma classe ou um método. Em particular devemos evitar usar o commit como um simples "save" regular. Um sinal que indicia um commit duvidoso é quando temos dificuldade em escrever a frase que descreve o commit.

Tal como se referiu anteriormente é possível associar os commits aos issues do Bitbucket usando para isso uma sintaxe especial no texto da mensagem do commit.

Para associarmos um commit a um issue devemos incluir um par command-issue em qualquer parte da mensagem de commit. O par command-issue tem o seguinte formato: <command> <issue id>

Podemos fazer a associação a um ou vários issues. O <issue id> pode ter os seguintes formatos:

- #4711
- issue #4711
- bug #4711
- ticket #4711

É muito importante incluir sempre o sinal # para garantirmos que o Bitbucket consegue associar o commit ao issue.

Cada acção suporta diversas palavras chave para o <command>. A tabela seguinte ilustra os comandos mais comuns e como podem ser especificados no texto da mensagem de commit:

acção	**command**	**exemplo**
ligar um commit a um issue (sem alterar o estado do issue)	addresses re references ref refs see	re bug #55 see #34 and #456
fechar/resolver um issue	close closed closes closing fix fixed fixes resolve resolves resolved resolving	close #845 fix bug #89 fixes issue #746 resolving #3117
reabrir um issue	reopen reopens reopening	reopen #546 reopening #78

2.9 Passo 8: Tags para marcar as versões finais de cada iteração

No final de cada semana (iteração) o grupo deve preparar e submeter no site do moodle do ISEP a versão correspondente ao "milestone" dessa semana.

Esta submissão é muito importante pois é sobre esta versão que o júri de docentes de LAPR4 fará a avaliação correspondente aos requisitos implementados e à qualidade técnica da solução.

Iremos abordar em mais detalhe as condições e cuidados que devem ser seguidos nas submissões noutro capítulo deste documento. Para já iremo-nos debruçar sobre os aspectos técnicos para a submissão.

Para a submissão da semana o grupo deve ter colocado o projecto num "estado" estável (isto é, o projecto compila e executa -**usando os scripts**- e os requisitos foram testados). Após garantir isso e ter no Bitbucket um commit que corresponde a esse estado o grupo deverá criar uma "tag" para marcar essa versão/milestone. O nome da tag deverá corresponder aos nomes que foram dados aos milestones no Bitbucket: week1(alfa), week2(beta), week3(release candidate).

A geração de tags é muito simples em qualquer ferramenta que estejamos a usar. Basicamente existirá uma opção designada "tag" que pedirá o nome da tag a criar. Poderá ainda pedir para seleccionar o "changeset" para o qual queremos associar a tag. Por omissão deve sugerir o último commit/changeset. Devemos confirmar e de seguida fazer o push para o Bitbucket de forma a ficar registada a nossa tag.

Depois de efectuar o passo anterior, para a submissão no moodle do ISEP, deve-se ir ao Bitbucket, à opção "Downloads", e seleccionar o separador "Tags". O Bitbucket irá apresentar todas as tags existentes e, para cada linha, terá links para descarregar o estado do repositório para essa versão em diversos formatos. Para a submissão no moodle deve-se utilizar o formato "zip" e **não devemos alterar o nome do ficheiro**.

Capítulo 3

Ferramentas

Neste capítulo apresenta-se um guia para ajudar o aluno a montar a cópia do repositório no seu computador (i.e., criar a cópia local do repositório).

Existem várias alternativas para atingir este fim. Uma opção seria a utilização direta do Git através do seus comandos executados directamente numa consola. No entanto não iremos apresentar esta opção uma vez que não é a opção adoptada noutras UC do curso. Em alternativa iremos apresentar uma opção baseada num front-end gráfico (SourceTree [1]) e duas opções baseadas em IDE (NetBeans [3] e Eclipse [4]).

3.1 SourceTree

O SourceTree é uma ferramenta que permite trabalhar com repositórios Git e Mercurial. Inicialmente existia apenas para Mac OSX mas recentemente foi apresentada uma nova versão para Windows (apenas para repositórios Git). Pode, portanto, ser utilizada em ambientes Windows e Mac.

Para explicar a utilização do SourceTree vamos continuar a seguir a equipa descrita no capítulo anterior. Vamos imaginar os possíveis passos do chefe de equipa (utilizador alexandre_isep2) pois este é que terá de realizar as tarefas mais complexas pois tem de trabalhar com 2 repositórios do Bitbucket.

3.1.1 Criação do repositório local

Após lançar o Sourcetree devemos executar a opção New do menu File. O Sourcetree irá apresentar a janela ilustrada na Figura 3.1.

No topo da janela aparecem os ícones das opções de criação. O que desejamos fazer é um clone local do repositório que está hospedado no Bitbucket. Daí devemos usar a opção por omissão "Clone Repository".

Os dados essenciais a introduzir são a fonte e o destino da operação. Na fonte ("Source Path/URL") devemos indicar o url do repositório do Bitbucket. É necessário prestar alguma atenção à estrutura do url. A parte antes do @ indica qual a conta de utilizador que vai ser usada para aceder ao Bitbucket. Neste caso é a do utilizador alexandre_isep2. A parte do url a seguir ao @ é que identifica o repositório no Bitbucket: "bitbucket.org/alexandre_isep2/lapr4_2de_g1_s2.git". Se fosse, por exemplo, o utilizador alexandre_isep5 a fazer o clone na sua máquina a parte do url antes do @ mudaria para alexandre_isep5 mas a parte à direita manter-se-ia.

O campo "Destination Path" deve ser usado para indicar a pasta no disco local aonde desejamos que seja criado o clone do repositório. O campo "Bookmark Name" é apenas o nome que será usado no Sourcetree para identificar o clone criado. Para confirmar clica-se no botão "Clone".

3.1.2 Janela principal

O processo de crição do "clone" pode demorar alguns instantes (depende essencialmente do tamanho do repositório origem). Quando estiver concluido com sucesso o Sourcetree apresentará uma janela que permite trabalhar com o "clone" local assim como efectuar operações de acesso e sincronização com o repositório remoto. A Figura 3.2 apresenta o aspecto dessa janela.

Esta janela tem 3 vistas diferentes: "File Status", "Log/History" e "Search". Podemos seleccionar a vista activa através de ícones no canto superior esquerdo ou por opções de menu [1]. Devemos ter em atenção que os dados que são mostrados na janela podem estar filtrados. Na ilustração da Figura 3.2 o filtro está desactivado ("Show All"), pelo que são visíveis todos os ficheiros do repositório (da cópia de trabalho).

[1] Na versão Windows estes ícones aparecem numa parte inferior da janela.

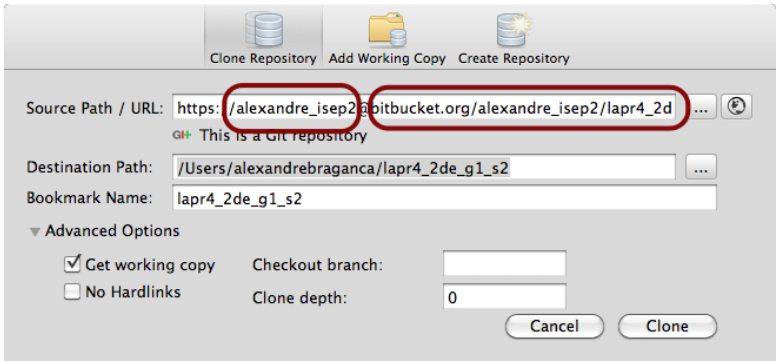

Figura 3.1: Criação da cópia local do repositório do Bitbucket

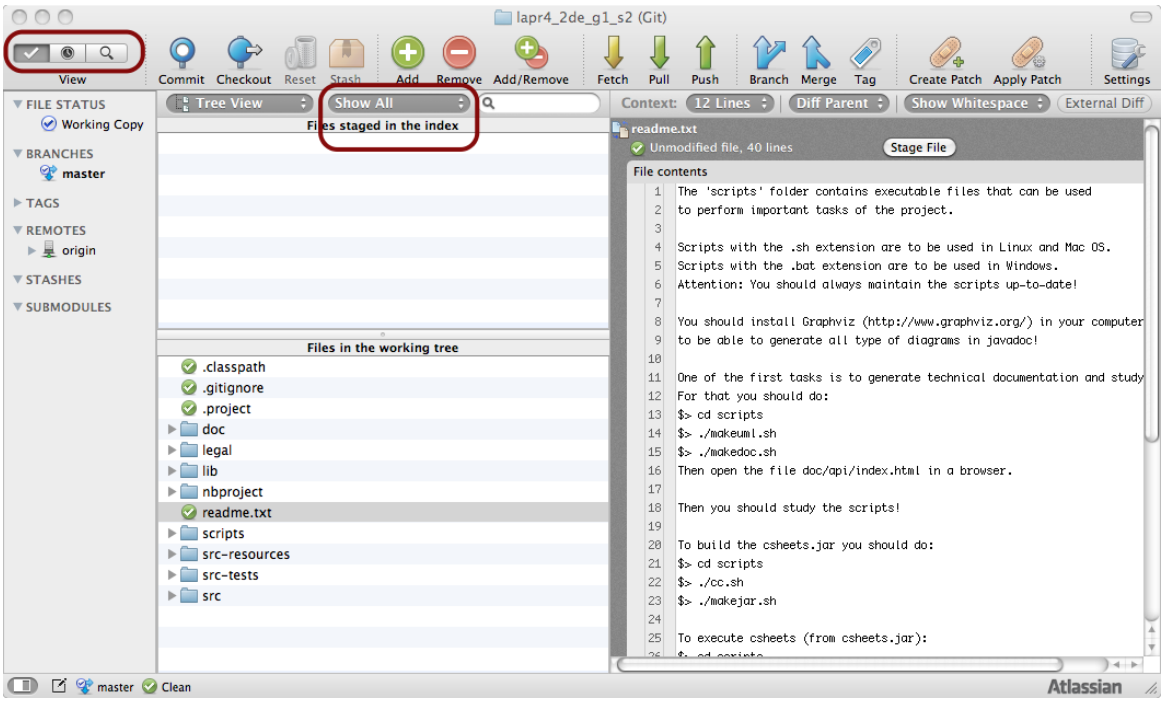

Figura 3.2: Janela de trabalho do Sourcetree

3.1.3 Utilizador para commits

Um dos aspectos mais importantes num ambiente de desenvolvimento colaborativo e distribuído é saber-se claramente quem produziu ou alterou um artefacto. Para não termos surpresas e garantirmos esse objectivo devemos certificarmo-nos que o Sourcetree irá utilizar o registo correcto de autor de commits. Para tal devemos aceder à opção "Repository Settings" do menu "Repository". Teremos então acesso à janela que é apresentada na Figura 3.3.

Nas opções avançadas da janela de configuração de repositórios devemos confirmar que os dados do utilizador são os que pretendemos. No caso apresentado na Figura 3.3 os dados estão correctos. No entanto, muitas vezes os dados que aparecem podem ser da conta por omissão que usamos para aceder a outros repositórios. Nesses casos é necessário corrigir.

3.1.4 Segundo repositório remoto

No caso de um grupo com 4 ou 5 alunos, vão existir 2 repositórios. Nesses casos o chefe do grupo vai necessitar de aceder aos 2 repositórios. No nosso exemplo o utilizador alexandre_isep2 como chefe de grupo deverá aceder aos dois repositórios do grupo.

Uma vez que já foi feito o clone com base num dos repositórios (aquele que tem como dono o alexandre_isep2) esse repositório já está registado como repositório remoto. O que necessitamos é acrescentar um segundo repositório remoto. Podemos fazer isso na janela de configuração de repositórios. A Figura 3.4 apresenta a configuração do segundo repositório remoto (que ficou registado com o nome "site1"). O primeiro repositório (que corresponde ao site 2 do grupo) ficou registado como "origin".

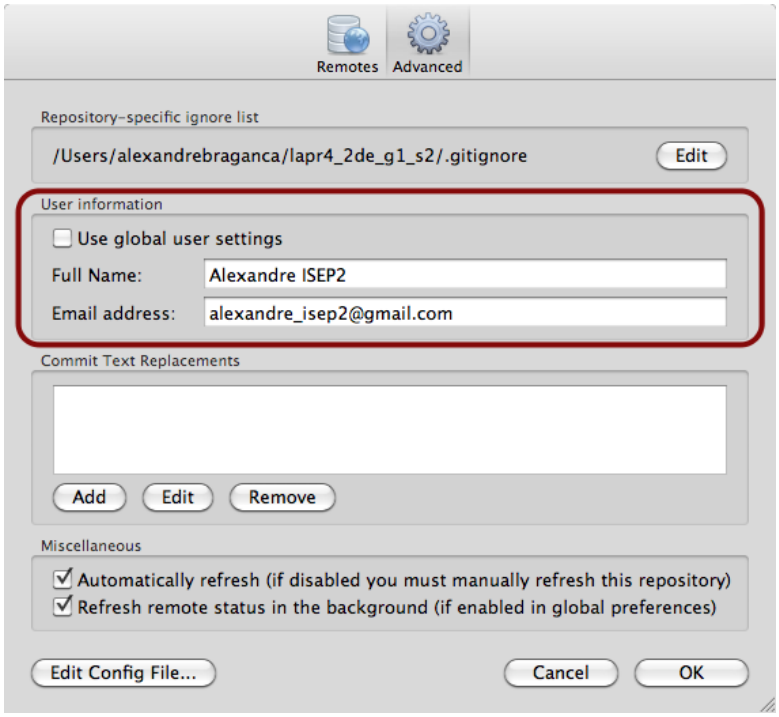

Figura 3.3: Janela de configuração de repositórios

Quando se configura mais do que um repositório remoto é depois possível escolher qual deles se vai usar quando se executam operações com repositórios remotos (por exemplo, "pushes" ou "pulls").

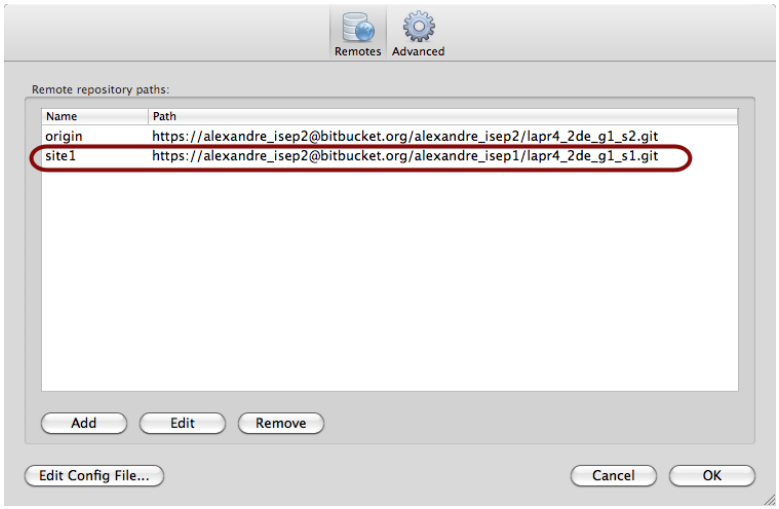

Figura 3.4: Separador "Remotes"da janela de configuração de repositórios

3.2 NetBeans

Para explicar a utilização do Netbeans vamos continuar a seguir a equipa descrita no capítulo anterior. Vamos imaginar os possíveis passos do chefe de equipa (utilizador alexandre_isep2) pois este é que terá de realizar as tarefas mais complexas pois tem de trabalhar com 2 repositórios do Bitbucket.

3.2.1 Criação do repositório local

Após lançar o Netbeans devemos seleccionar a opção "Clone..." no menu "Team->Git". O Netbeans irá apresentar uma janela semelhante à ilustrada na Figura 3.5. Nesta devemos especificar os dados da origem do Clone. No nosso caso são os dados que identificam o url do repositório no Bitbucket para o site 2 do grupo.

Nos campos "User" e "Password" devemos introduzir a identificação do utilizador. Neste caso são as do chefe do grupo.

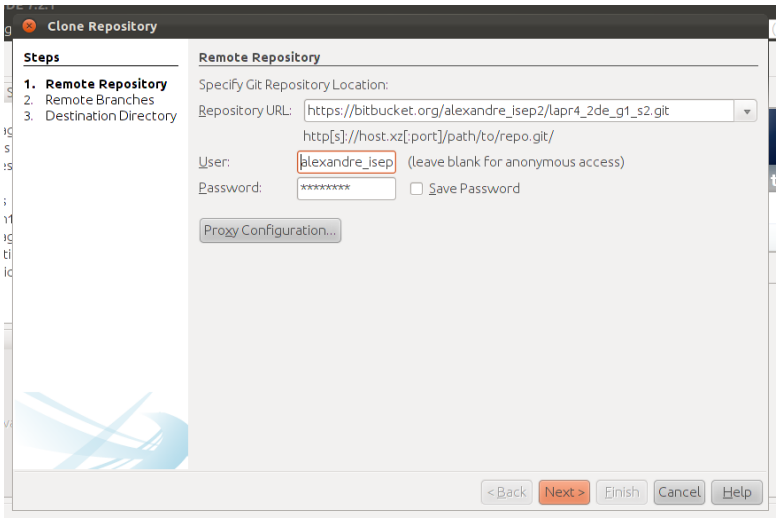

Figura 3.5: Selecção da origem do Clone no Netbeans

Na janela seguinte devemos escolher qual o "branch" (ramo) ao qual desejamos aceder. Neste caso temos apenas um que é o "master". Devemos seleccionar esse.

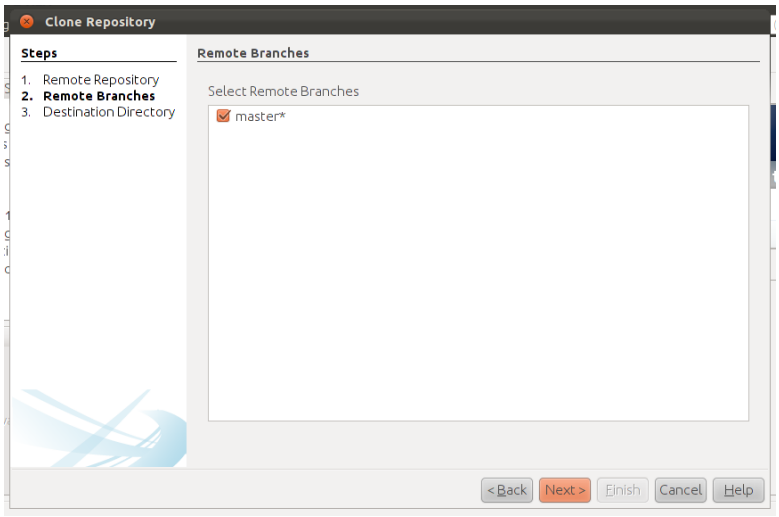

Figura 3.6: Selecção do ramo do Clone no Netbeans

Finalmente devemos introduzir os dados do destino da operação de Clone. Como estamos a trabalhar com o Netbeans ele sugere a pasta onde normalmente temos os projectos do Netbeans. Após confirmarmos com o "Finish" o Netbeans pergunta se deve fazer o "scanning" do clone de forma a encontrar projectos e, nesse caso, se os deve abrir. Devemos reponder afirmativamente pois vamos trabalhar o projecto com o NetBeans.

3.2.2 Commits

Uma das preocupações que devemos ter quando efectuamos commits (para além de uma descrição clara da alteração) é a de identificarmos claramente quem é o autor. Para tal, no Netbeans, a janela de commit permite que especifiquemos não só quem está a fazer o commit como quem é o autor (normalmente são a mesma pessoa). Tal como se pode ver na Figura 3.8 devemos introduzir correctamente a identificação do autor (tal como registamos o utilizador no Bitbucket). O formato é o seguinte: user<email_do_user>.

3.2.3 Segundo repositório remoto

No caso de um grupo com 4 ou 5 alunos, vão existir 2 repositórios. Nesses casos o chefe do grupo vai necessitar de aceder aos 2 repositórios. No nosso exemplo o utilizador alexandre_isep2 como chefe de grupo deverá aceder aos dois repositórios do grupo.

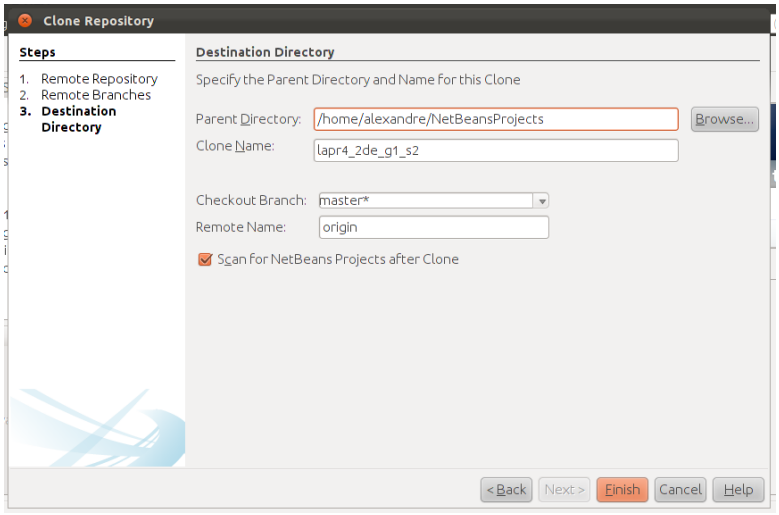

Figura 3.7: Selecção do destino do Clone no Netbeans

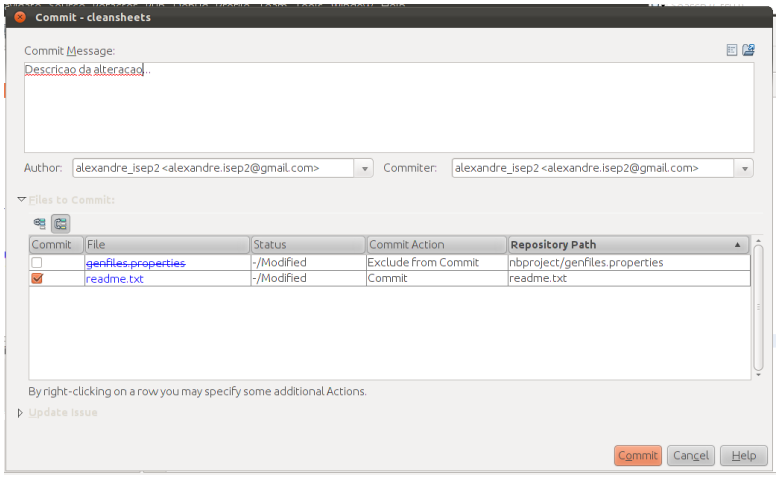

Figura 3.8: Identificação do autor no commit

No Netbeans, sempre que se acede a um repositório remoto, quer para fazer pull quer para fazer push podemos identificar o repositório remoto. Caso não o façamos ele vai usar o repositório a partir do qual fizemos originalmente o clone. A Figura 3.9 ilustra uma situação em que o utilizador alexandr_isep2 efectua um push das alterações do repositório local para o primeiro repositório (i.e., site) do grupo.

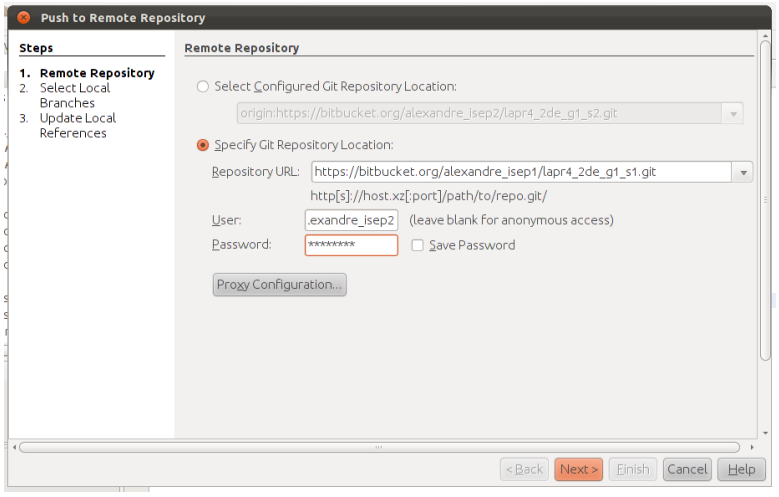

Figura 3.9: Selecção do destino do Clone no Netbeans

3.3 Eclipse

Para explicar a utilização do Eclipse vamos continuar a seguir a equipa descrita no capítulo anterior. Vamos imaginar os possíveis passos do chefe de equipa (utilizador alexandre_isep2) pois este é que terá de realizar as tarefas mais complexas pois tem de trabalhar com 2 repositórios do Bitbucket.

3.3.1 Criação do repositório local

Após arrancar o Eclipse devemos usar a opção Import do menu File para acedermos à opção "Projects from GIT". Esta opção permite fazer o clone de um repositório remoto GIT. Tal como ilustradao na Figura 3.10 devemos seleccionar a opção uri para especificarmos o endereço remoto do repositório do Bitbucket do qual queremos fazer o clone.

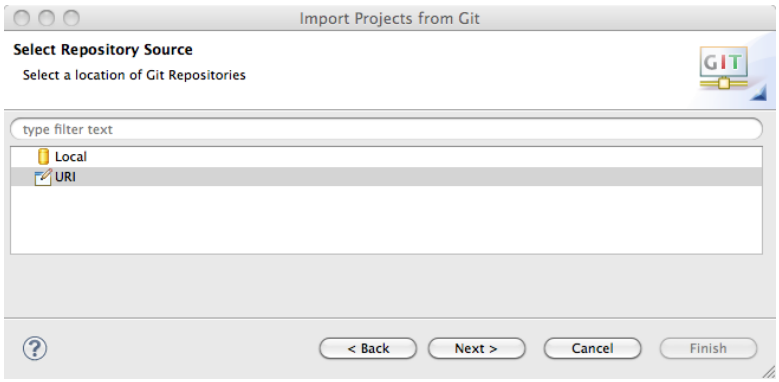

Figura 3.10: Seleccionar a fonte da importação GIT

De seguida devemos introduzir os dados de identificação do repositório remoto assim como de acesso ao Bitbucket, tal como ilustrado na Figura 3.11

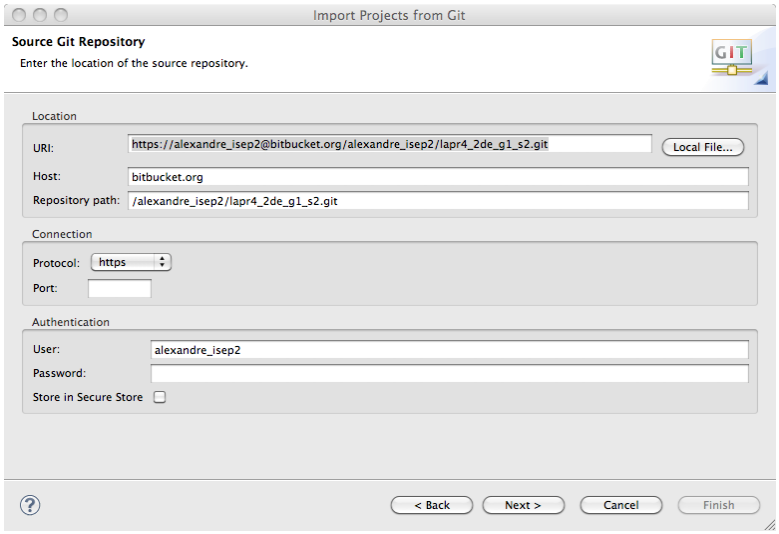

Figura 3.11: Dados do repositório origem

No passo seguinte, o Eclipse apresenta os ramos (branch) que detectou no repositório destino. No nosso caso apenas existirá um ramo. Devemos seleccionar esse ramo, tal como ilustrado na Figura 3.12.

O passo seguinte consiste em especificar o destino. Para tal devemos especificar uma pasta no disco local, tal como ilustrado na Figura 3.13. É muito importante que a pasta na qual será colocado o repositório não seja a pasta do workspace do eclipse nem uma sua subpasta. Deve usar outra pasta que não a do workspace do eclipse.

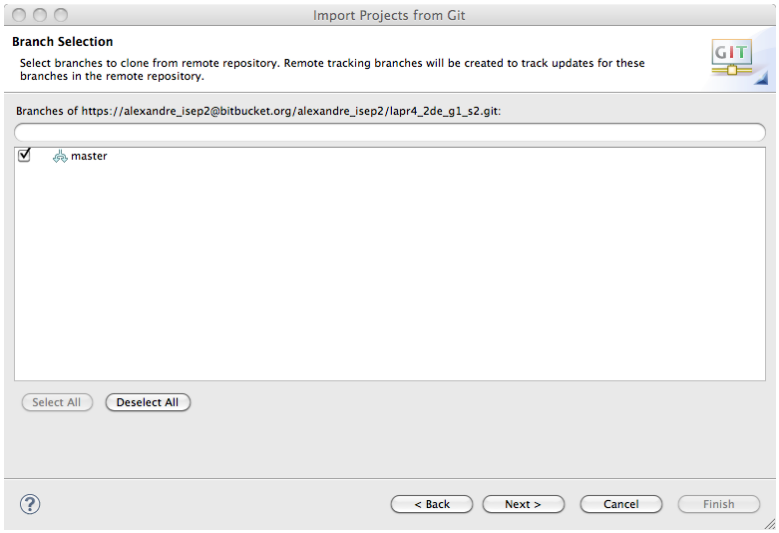

Figura 3.12: Seleccionar o branch

Porquê colocar o repositório fora do workspace (segundo o EGit User Guide)?

It is a good idea to keep your Repository outside of your Eclipse Workspace. If you don't do this, the new Repository will consider the complete folder structure of the Eclipse workspace as (potential) content. This can result in performance issues, for example when calculating the changes before committing (which will scan the complete .metadata folder, for example); more often than not, the workspace will contain dead folders (e.g. deleted projects) which semantically are not relevant for EGit but can not be excluded easily.

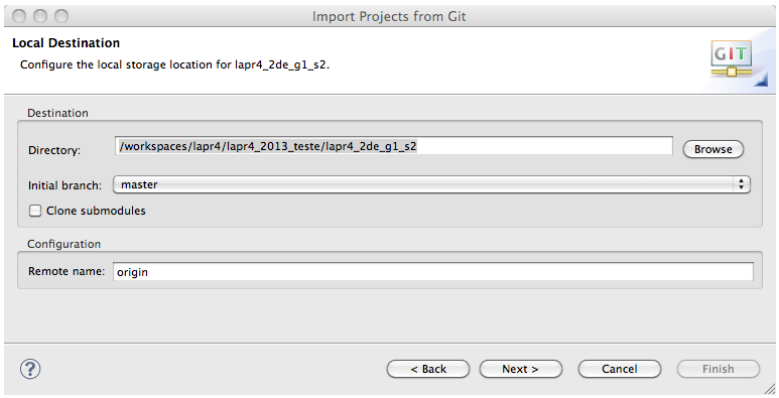

Figura 3.13: Dados do destino

Finalmente o eclipse apresenta os projectos que detectou no repositório local e pede para se confirmar a sua abertura. Tal como ilustrado na Figura 3.13 devemos confirmar a abertura do projecto.

3.3.2 Commits

Uma das preocupações que devemos ter quando efectuamos commits (para além de uma descrição clara da alteração) é a de identificarmos claramente quem é o autor. Para tal, no Eclipse, a janela de commit permite que especifiquemos não só quem está a fazer o commit como quem é o autor (normalmente são a mesma pessoa). Tal como se pode ver na Figura 3.15 devemos introduzir correctamente a identificação do autor (tal como registamos o utilizador no Bitbucket). O formato é o seguinte: user<email_do_user>.

Podemos aceder às opções do Git no eclipse através do menu de contexto que se obtém com um clique do botão direito sobre o projecto. A opção "Team" que aparece permite aceder às operações relativas ao Git.

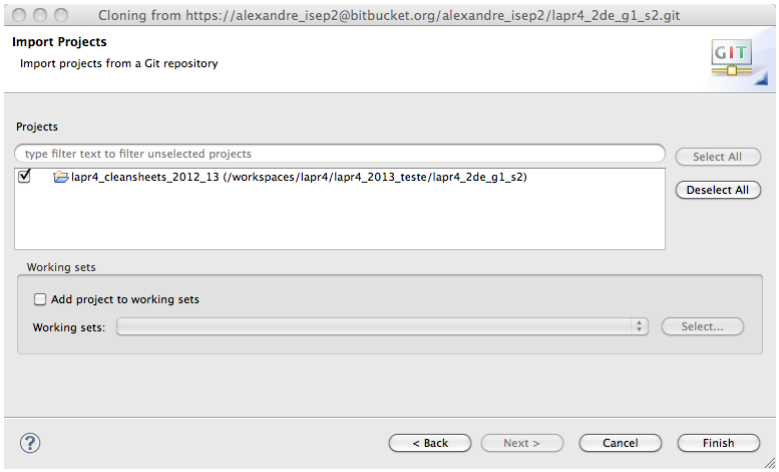

Figura 3.14: Importar projectos

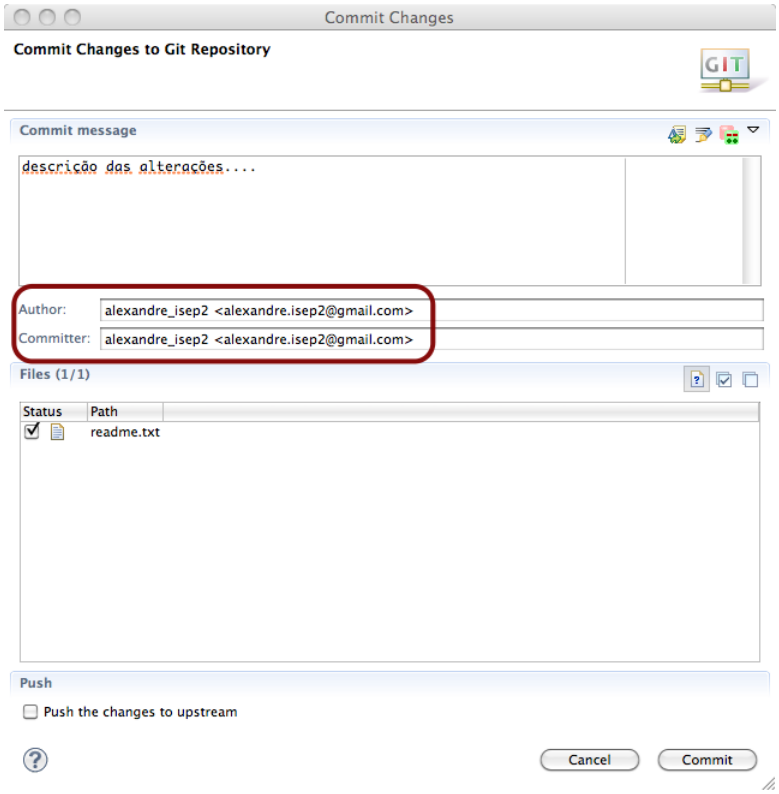

Figura 3.15: Janela de commit

3.3.3 Segundo repositório remoto

No caso de um grupo com 4 ou 5 alunos, vão existir 2 repositórios. Nesses casos o chefe do grupo vai necessitar de aceder aos 2 repositórios. No nosso exemplo o utilizador alexandre_isep2 como chefe de grupo deverá aceder aos dois repositórios do grupo.

No Eclipse, sempre que se acede a um repositório remoto, quer para fazer pull quer para fazer push podemos identificar o repositório remoto. Caso não o façamos ele vai usar o repositório a partir do qual fizemos originalmente o clone. A Figura 3.16 ilustra uma situação em que o utilizador alexandr_isep2 efectua um push das alterações do repositório local para o primeiro repositório (i.e., site) do grupo.

Como o nosso repositório pode ter vários ramos é necessário seleccionar numa janela seguinte qual o ramo ou ramos a incluir no push. No nosso caso podemos usar o botão "Add All Branches Spec"....

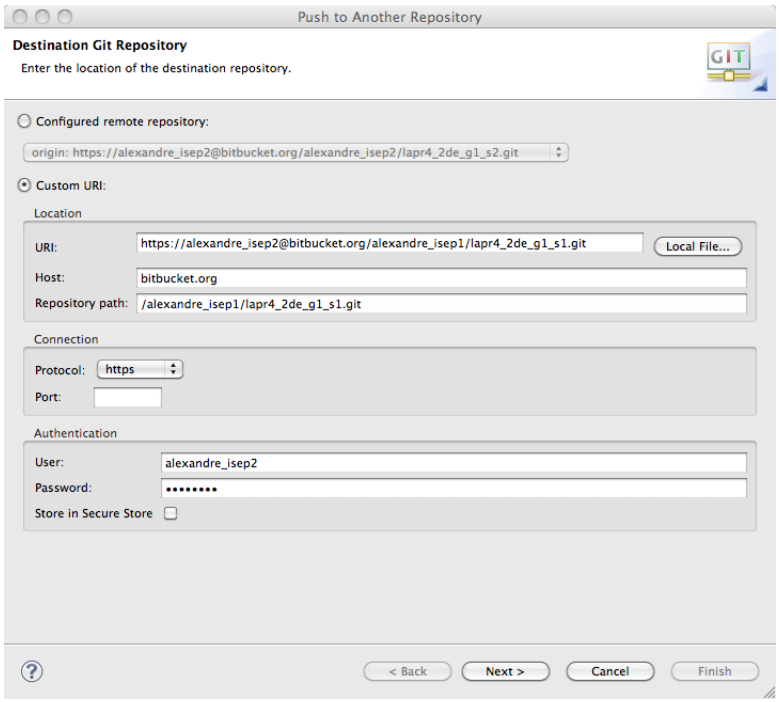

Figura 3.16: Janela de push

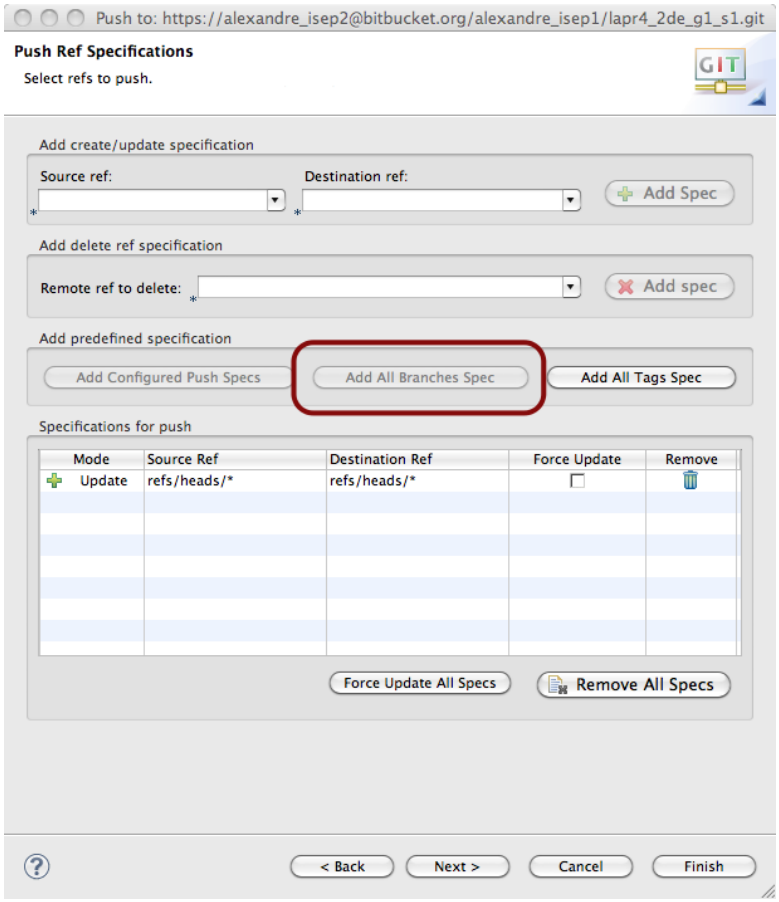

Figura 3.17: Janela de push: selecção de ramos

Capítulo 4

Tarefas

Capítulo sobre gestão de tarefas e issues

4.1 Enunciado

O enunciado do projecto será apresentado de uma forma iterativa ao longo das 3 semanas dedicadas à realização do projecto.

Isto significa que os alunos vão descobrir detalhes do enunciado ao longo das semanas. Pretende-se desta forma simular situações reais nas quais os requisitos são incompletos e podem mudar ao longo do tempo.

Com base no enunciado publicado a cada semana os alunos devem identificar os requisitos envolvidos (quer funcionais quer não funcionais).

De forma a facilitar a gestão do projecto e também a avaliação o enunciado estará dividido em 5 trilhos. Cada trilho está incluído numa de 3 áreas funcionais. As áreas funcionais são as seguintes: Persistência e I/O (2 trilhos); Linguagens (2 trilhos) e Partilha de Folha (1 trilho).

Em cada semana cada aluno do grupo deverá assumir a responsabilidade dos requisitos de 1 trilho. Como existem 3 áreas funcionais e 3 semanas de trabalho, espera-se que cada aluno assuma a reponsabilidade de trilhos de 3 áreas funcionais diferentes, uma por cada semana.

O número de trilhos que cada grupo deve atacar corresponde ao número de elementos do grupo. Um grupo de 3 alunos apenas necessita de atacar 3 trilhos, enquanto um grupo de 5 alunos necessita de atacar 5 trilhos.

Faz parte do planeamento de cada grupo definir quais os trilhos que deseja atacar, sendo que nunca se deve esquecer que a maximização de nota acontece quanto o aluno aborda as 3 áreas funcionais.

4.2 Primeiro contacto com o Projecto

Após obter uma cópia ("clone") local do repositório (i.e., uma cópia no seu computador) o aluno deve seguir dois passos que são essenciais: ler o conteúdo do ficheiro readme.txt e consultar a documentação inicial do projecto.

O ficheiro readme.txt deve estar no diretório no qual foi criada a cópia local do repositório. Este ficheiro explica os passos essenciais para se começar a trabalhar com o projecto.

Uma dos princípios importantes é que o projecto contém no subdiretório "scripts" um conjunto de scripts executáveis que permitem executar as tarefas automáticas de construção do projecto, como por exemplo, a geração da documentação ou a construção do ficheiro jar.

A documentação do projecto é baseada em javadoc. Assim, esta é construida a partir de comentários especiais que se encontram no próprio código. Num projecto desta natureza (de uma Unidade Curricular muito focalizada no processo de desenvolvimento de software) é muito importante incluir na documentação técnica diagramas técnicos (por exemplo, diagramas UML). Sendo assim, é utilizada uma ferramenta designada PlantUML para esse fim. O ficheiro readme.txt alerta para o facto da ferramenta PlantUML depender de um software designado Graphviz [6] que é necessário instalar no computador.

Depois de seguidas as instruções (no ficheiro readme.txt) para gerar a documentação o aluno deve abrir o ficheiro principal da documentação (doc/api/index.html). Nele encontrará documentação técnica sobre o projecto que completa a documentação contida neste documento.

Porque é que a documentação está em língua Inglesa?

Como o projecto original do Cleansheets foi desenvolvido com documentação em língua Inglesa optou-se por manter a utilização da língua Inglesa nos acrescentos que foram feitos para LAPR4. O aluno pode continuar a manter esta opção ou usar a língua Portuguesa.

4.3 Organização

Cada grupo é constituído por 3 a 5 alunos. Para além dos alunos o grupo conta com mais 2 elementos que são "observadores": o chefe do projecto (que é o professor regente da UC: atb@isep.ipp.pt) e o supervisor do grupo (que é um docente da UC que será anunciado a cada grupo). Estes 2 docentes são apenas observadores e apenas necessitam de ter acesso de leitura aos repositórios do grupo no Bitbucket (tal como descrito no Capítulo 2)

Cada grupo deve escolher um dos alunos para assumir o papel de chefe do grupo. O chefe do grupo assume uma responsabilidade prática significativa nos grupos de 4 ou 5 alunos pois será ele que deve efectuar a integração dos dois repositórios da sua equipa (de acordo com o processo descrito no Capítulo 2).

4.4 Ciclo Semanal

O ciclo de trabalho de cada equipa (i.e., grupo) desenvolve-se de acordo com as iterações que são semanais.

Assim, no ínicio de cada semana o enunciado dessa semana é publicado. O primeiro dia da semana deve ser essencialmente um dia dedicado ao planeamento. Todas as segundas-feiras os alunos terão aula TP de planeamento com os docentes de Gestão. Devem fazer o planeamento da semana com base no enunciado e aplicando as melhores práticas leccionadas no módulo de gestão. A folha de cálculo usada para o planemanto deve fazer parte do repositório (pode ser colocada, por exemplo, na pasta "doc").

Um dos constrangimentos que devem ter em atenção quando estiverem a fazer o planeamento é que se espera que cada aluno trabalhe numa área funcional diferente em cada semana. Não é algo que seja obrigatório, mas é algo que será tomado em conta na avaliação.

Para além disso, cada aluno do grupo deve ficar responsável nessa semana por um trilho do enunciado. Não quer isto dizer que apenas ele irá desenvolver tarefas relacionadas com esse trilho. Apenas que é o responsável e como tal se espera que as tarefas nucleares de concepção, implementação, documentação técnica e testes unitários sejam realizadas maioritariamente por ele. No entanto diversas outras tarefas podem ser partilhadas (ou realizadas noutros trilhos) como, por exemplo: tarefas de análise; tarefas de pesquisa e estudo; tarefas de documentação; tarefas de experimentação; tarefas de execução de casos de teste, etc.

Logo no inicio da semana, após a reunião de planeamento semanal, cada aluno deve registar no Bitbucket um issue relativo ao requisito (ou requisitos) que vai atacar num determinado trilho. Espera-se que o caso mais comum seja apenas um issue semanal para cada aluno (relativo ao trilho que será da sua responsabilidade).

4.5 Ciclo Diário

É importante que os alunos integrem com frequência o trabalho que realizam na sua cópia local do repositório. Idealmente deveriam conseguir uma integração diária. Para além disso cada grupo deve fazer uma pequena reunião no início de cada dia de trabalho para fazer uma avaliação do progresso do trabalho e das dificuldades.

Como sugestão de organização do trabalho sugere-se que cada grupo se reuna no iníco de cada dia e, presencialmente, façam a integração dos respectivos trabalhos no Bitbucket. Assim, se houverem problemas na integração estarão todos presentes e poderão ajudar na resolução dos problemas. Após esta integração todos os elementos devem actualizar as suas cópias locais do repositório com a última versão do repositório integrado do Bitbucket. Quando terminarem esta tarefa sugere-se que façam logo a reunião diária de ponto de situação.

Os alunos devem ter em atenção que o último dia da semana terá um ciclo ligeiramente diferente, uma vez que é necessário prever tempo para a tarefa de execução de casos de teste. Estes casos de teste consistem em testar os requisitos da semana após estes terem sido dados como "fechados" pelo responsável pelo "issue". Os casos de teste devem ser executados por um aluno diferente daquele que ficou reponsável pelo "issue". Note-se que caso seja detectado um problema este deve ficar registado na gestão de issues do Bitbucket e o issue deve ser "reaberto".

Cada aluno deve primar por desenvolver o seu trilho isento de erros. Cada aluno que execute casos de teste deve primar por identificar o máximo de erros possível.

4.6 Trilhos e Casos de Uso

Um trilho normalmente dará origem a um caso de uso (embora possa dar origem a mais). Em termos de gestão de issues, um caso de uso corresponde a um issue.

Portanto, a situação mais comum será a de haver uma correspondência direta entre o requisito de um trilho numa semana e um issue. Se assim for, quando o aluno der o issue por concluído então é porque o trilho que é da sua responsabilidade está concluído. Isto também permite facilmente assinalar ao colega do grupo que vai testar o caso de uso que este está completo e que se podem fazer testes. Se dos testes resultar a identificação de erros o issue deve ser reaberto para que os erros possam ser corrigidos.

4.7 Testes

No contexto de LAPR4 os alunos devem incorporar 2 tipos de testes no seu desenvolvimento: testes unitários (com o junit) e casos de teste.

Os testes unitários correspondem a testes automáticos que podem ser realizados sobre as classes desenvolvidas. O seu desenvolvimento é da responsabilidade do responsável pelo "issue".

Os casos de teste consistem basicamente em testes interactivos à funcionalidade do "issue" sendo baseados na descrição do respectivo caso de uso. Devem ser realizados após o "issue" ser fechado pelo seu responsável. Um outro elemento do grupo testa a funcionalidade seguindo a descrição do caso de uso e usando dados de teste. Deve tentar identifcar problemas com o "issue" e, se for caso de disso', reportar os problemas encontrados ao responsável do "issue" através da reabertura do "issue".

4.8 Integração de repositórios

Os alunos devem ter o cuidado de integrar o trabalho dos diversos elementos do grupo com uma frequência adequada. Podem fazê-lo uma vez por dia ou então, por exemplo, sempre que tenham um "commit significativo". Desta forma minimizam-se os problemas que podem surgir quando se integra apenas, por exemplo, no final da semana de forma a se submeter a versão semanal.

Nesta secção vamos abordar este tema e apresentar possíveis receitas para se conseguir a integração.

4.8.1 Integração com 1 Repositório Bitbucket

Esta é a situação mais simples que acontece com grupos de apenas 3 alunos. Neste caso o procedimento é semelhante ao que os alunos usaram em EAPLI.

Basicamente cada aluno, após ter efectuado o commit no seu repositório local deve fazer um "pull" sobre o repositório central do bitbucket para verificar se está a trabalhar sobre a "última" versão central. Caso isso aconteça o "pull" não resulta em nenhuma alteração local e o aluno pode efectuar o "push" das suas alterações locais para o repositório central do Bitbucket. Caso isso não aconteça, do "pull" vai resultar a necessidade de efectuar "merge" das alterações locais com as remotas. Algumas são feitas automaticamente mas outras requerem a intervenção do utilizador. Após finalizado o "merge" pode-se fazer o "commit" do resultado e o "push" para o repositório central do Bitbucket.

Todos os alunos do grupo devem fazer estes passos de forma sequencial.

Após todos terem feito estas operações o repositório central estará com uma versão integrada do trabalho dos alunos. Todos os alunos devem agora fazer o "pull" para obterem todos no seu repositório local a versão que acabaram de integrar.

4.8.2 Integração com 2 Repositórios Bitbucket

Num grupo com 4 ou 5 alunos vão existir 2 repositórios no Bitbucket. Para exemplificar como pode ser efectuada a integração nestes casos vamos usar a estrutura de grupo ilustrada na Figura 4.1.

Neste caso o chefe do grupo vai ter acesso aos dois repositórios centrais que o grupo terá no Bitbucket. Este terá a responsabilidade de manter os dois repositórios sincronizados (com a ajuda dos outros elementos do grupo).

Seguindo então a estrutura ilustrada na Figura 4.1 um processo possível para a integração poderá ser o descrito de seguida.

O grupo começa por fazer a integração num dos repositórios, por exemplo, o ReposiorioSite2. Os utilizadores deste repositório (alexandre_isep2, alexandre_isep4 e alexandre_isep5) procedem à integração tal como descrito na secção anterior para o caso de só existir um repositório.

De seguida os utilizadores do RepositorioSite1 (com excepção do utilizador alexandre_isep2) fazem a integração no seu repositório central.

É agora necessário que o utilizador alexandre_isep2 (o chefe de projecto e reponsável pela integração), que tem o seu repositório local sincronizado com o RepositorioSite2, faça a integração com o que está no RepositórioSite1. Para isso pode fazer um "pull" ao RepositorioSite1 e efectuar o "merge" com este. Com esta operação o chefe do grupo tem no seu repositório local a integração dos dois repositórios (admitindo que entretanto nenhum dos elementos do grupo fez alterações nos repositórios). Deve então fazer um "push" dessa integração para o RepositorioSite1 e de seguida para o RepositórioSite2.

Após esta operação todos os alunos devem fazer um "pull" ao respectivo repositório para obterem a última versão (que é igual nos dois repositórios do grupo).

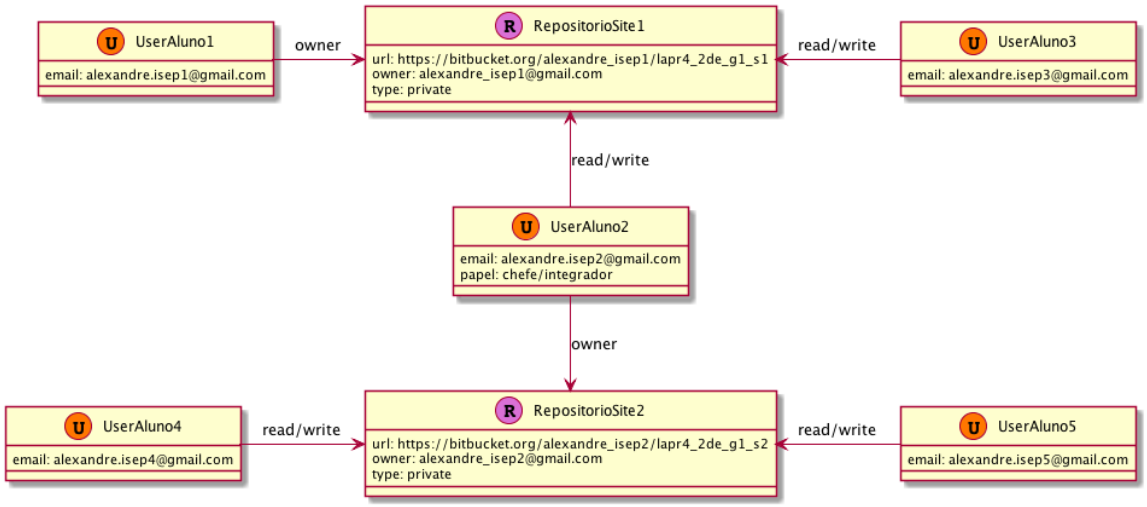

Figura 4.1: Grupos com 2 repositórios no Bitbucket

4.9 Submissões

As submissões oficiais do trabalho devem ser efectuadas no final de cada semana num link do moodle a anunciar.

O objectivo é que o grupo tenha conseguido integrar todo o trabalho (mesmo que hajam requisitos que não estejam completos). Devem ter em atenção que essa entrega deve ter sido marcada no repositório através de uma "tag". O conteúdo a submeter será o obtido de acordo com as indicações apresentadas na secção 2.9 do Capítulo 2.

IMPORTANTE: Confirmar o funcionamento dos scripts antes da submissão!

Uma tarefa de extrema importância é a verificação que todos os scripts estão a funcionar antes de submeter o projecto. Os scripts serão usados nas avaliações para efectuar todas as tarefas: construção do executável; construção da documentação; execução de testes unitários; execução da aplicação, etc. No caso de existirem instruções especificas para a execução dos scripts estas devem ser descritas no ficheiro readme.txt.

Capítulo 5

Avaliação

A avaliação de LAPR4 é efectuada de acordo com os princípios gerais enunciados na Ficha da Unidade Curricular.

Na secção seguinte apresentam-se notas sobre os critérios de avaliação especificos relativos ao projecto e à forma da sua concretização. Note-se que esta é apenas uma componente da avaliação: existe ainda a componente de gestão de projecto.

5.1 Avaliação do Projecto na FUC

Avaliação do projeto irá considerar as seguintes componentes:

- **1**. a adoção e domínio do processo de desenvolvimento de software (incluindo a gestão de actividades), reportado através da entrega periódica de artefactos (incluindo código e testes);
- **2**. a qualidade da documentação técnica do projeto;
- **3**. o nível de satisfação dos requisitos;
- **4**. a qualidade (técnica) da solução;
- **5**. o conhecimento sobre o projeto, sua apresentação e defesa;

As componentes **1** e **2** são avaliados principalmente pelo professor supervisor do projeto (e também pelo regente). Este tem acesso aos repositórios dos alunos e fará essa avaliação com base nos registos de commits assim como no registo de issues de cada aluno e no respectivo planeamento.

As componentes **3** e **4** são avaliadas por professores especialistas em função do trilho que está a ser avaliado.

A componente **5** é avaliada numa sessão de defesa do projeto perante um júri constituido por docentes de LAPR4. Esta sessão não consiste numa sessão típica de apresentação de projeto mas numa defesa individual. O aluno deverá demonstrar conhecimento sobre os diversos aspectos do projeto respondendo de forma individual a pergunta(s) sobre o seu trabalho e sobre o projecto.

Os alunos com uma nota possível superior a 17 devem defender seu projeto e trabalho frente a um segundo júri, mais extenso, que inclui o regente.

De notar que em todos os critérios os alunos são avaliados individualmente pois a avaliação dos critérios **1** a **4** é feita ao nível de detalhe da semana e do trilho e a defesa é individual.

Bibliografia

[1] Atlassian. Sourcetree. http://www.sourcetreeapp.com/.

[2] Bitbucket. Site bitbucket. https://bitbucket.org/.

[3] Oracle Corporation. Netbeans. https://netbeans.org/.

[4] The Eclipse Foundation. Eclipse. http://www.eclipse.org/.

[5] Git. Git. http://git-scm.com/.

[6] Graphviz. Graphviz. http://www.graphviz.org/.

[7] Einar Pehrson. Cleansheets. https://bitbucket.org/atb/cleansheets-2012-13-private.