Engenharia de Aplicações

Modo de funcionamento do trabalho prático

1 Introdução

O projeto será desenvolvido por cada turma PL. Em cada aula, o docente em conjunto com os alunos planeiam o trabalho dessa sessão. Cada par de alunos ficará responsável por uma tarefa, tendo que a analisar, desenhar, desenvolver, testar e no fim colocar no repositório comum devidamente integrado. A turma pode ser organizada em pequenas equipas por área funcional.

No repositório deverão ser guardados os artefactos de design criados, o código e os testes. Na parte final de cada aula os alunos integrarão o seu contributo no repositório do projeto e atualizarão a execução das tarefas (ex., fechar *issues* concluídos) para que a mesma possa ser utilizada na próxima sessão.

2 Setup (1ª sessão):

1. O professor de cada turma PL cria um repositório no Bitbucket, ex., EAPLI_PL_2015_2DD.
2. Cada aluno deve indicar ao docente a sua conta Bitbucket
3. O docente adiciona esses alunos como colaboradores desse repositório
4. Cada aluno (ou par de alunos) faz o *clone* do repositório

3 Sessão de análise:

1. Os alunos em equipa discutem os requisitos apresentados pelo cliente e constroem o modelo de domínio **identificando os pontos de integração** entre as áreas funcionais.
2. O modelo de domínio é adicionado à documentação do repositório e o *issue* correspondente é comentado e marcado como concluído.
3. Em seguida, os alunos em equipa discutem os vários casos de uso que constituem as **prioridades atuais** do cliente com vista a identificar:
    a. **Pontos em aberto** que seja necessário mais esclarecimento do cliente
    b. Casos de uso que possam ser **divididos em passos mais pequenos** para melhor divisão do trabalho e progresso nos ciclos de duas semanas
    c. **Dependências entre casos de uso** para melhor gestão do trabalho
    d. Se a equipa assim o entender, dividem-se em equipas mais pequenas por cada área funcional do projeto
4. A equipa deve decidir a **organização dos menus da aplicação** para evitar sobreposições e incoerências. Devem registar na wiki e num *issue* do Bitbucket o resultado dessa discussão.
5. Em equipa decidem os **aspetos base do projeto**
    a. Camadas a criar.
    b. Regras de nomenclatura de classes em cada camada.

c. Padrões gerais a utilizar em cada camada e nas interações entre camadas, ex., Factory e interfaces para esconder o acesso ao mecanismo de persistência.
d. Alguém na equipa é escolhido para fazer o *commit* da base do projeto: projeto(s) java, packages iniciais, classes comuns entre as várias áreas funcionais (estas classes pode estar vazias neste momento).
e. As decisões são documentadas na wiki do projeto.

6. Para cada caso de uso (original ou resultante da divisão do passo anterior) **criam o *issue* correspondente** garantindo que o *issue* tem sempre como prefixo a identificação do caso de uso e área funcional (ex., "UC-D-002:"). Em cada issue devem registar em comentários as diferentes tarefas:
    a. Análise
    b. Design
    c. Planeamento de testes
    d. Desenvolvimento e testes
    e. Integração

4 Em todas as sessões de desenvolvimento

1. Cada par de alunos faz o *pull* das últimas alterações para o seu repositório local por forma a garantir que tem a base de código mais atual.
2. O professor discute com a equipa (a turma PL) as tarefas existentes (resultantes do planeamento anterior). Para tal utilizam a ferramenta de *issues* do repositório escolhido, ex., Bitbucket.
3. Cada par de alunos escolhe uma tarefa/*issue* e faz o *assign* dessa tarefa a um dos elementos
4. Cada par analisa, desenha, desenvolve e testa.
    a. Ter em conta as dependências com outros membros da equipa, e conversar/combinar os pontos comuns.
    b. Todos os artefactos devem ser colocados no repositório na pasta "documentation" ou wiki e os *issues* devem ser atualizados com os comentários e anexos adequados.
5. Quando concluída a tarefa, fazem o *commit* no repositório local com uma mensagem expressiva do trabalho efetuado, indicando o número dos alunos envolvidos nessa tarefa e fechando o *issue* ("fixes #999").
6. Em seguida tentam fazer o *push* para o repositório partilhado.
7. Em caso de conflito,
    a. efectuam o *pull*,
    b. realizam o *merge*,
    c. compilam e testam novamente o seu código,
    d. fazem *commit*
    e. fazem *push*

Os alunos que trabalhem em pares devem alternar a conta do repositório com que fazem *commit*. Não esquecer de colocar o número dos alunos envolvidos na mensagem de *commit*.

Ao trabalhar em pares é importante seguir as recomendações sobre “pair programming” em que cada elemento vai alternando no papel de “condutor” e de “navegador”.

Periodicamente a equipa faz sessões de “code review” onde analisam extratos de código e identificam pontos a melhorar criando os *issues* correspondentes. Ex., ao fazer revisão do código de um caso de uso notam que existem métodos duplicados ou que uma dada classe está num package desadequado. Devem em seguida criar um *issue* correspondente a esse *refactoring*.