DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

LEIC-A, LEIC-T, LETI

Engenharia de Software

2º Semestre – 2015/2016

Enunciado da Primeira Parte do Projeto

Para a primeira parte do projeto pretende-se concretizar as entidades do domínio do problema apresentado no enunciado geral utilizando o mecanismo de persistência de objetos da **fénix-framework**. Para isso, deve ser definido um ficheiro **dml** que descreva a estrutura das entidades do domínio. Esta modelação e concretização ao parcial deverá considerar todas as entidades do domínio descritas, incluindo todos os atributos existentes, as associações entre estas entidades e as possíveis relações de herança que existam. Será também necessário enriquecer a camada de domínio com suporte para as regras de negócio e para as funcionalidades necessárias à visualização do projeto que se apresentam neste enunciado.

1. Casos de uso

Além da definição das entidades e relações do domínio da aplicação descritas, o domínio deve suportar operações para:

O *username* deve ser constituído exclusivamente por letras e dígitos decimais;

Importação e exportação XML : ver descrição na seção 2;

Listagem simples de uma diretoria : dado um caminho devolve uma lista com os nomes dos ficheiros existentes na diretoria;

Leitura do conteúdo de um ficheiro : dado um caminho devolve o conteúdo de um ficheiro de texto;

Apagar um ficheiro (ou diretoria vazia) : dado um caminho remove do sistema um ficheiro de texto ou uma diretoria vazia;

Exceções : auto-explicativas para as situações de erro encontradas.

A primeira parte do projeto não requer a concretização da lógica de negócio responsável por verificar as permissões de acesso aos ficheiros pois todas as operações deverão ser realizadas pelo utilizador *root*.

2. Importação e exportação XML

Para transformar objetos do domínio em XML e vice-versa deve-se utilizar a biblioteca JDOM (`http://www.jdom.org/`). Esta biblioteca permite ler, escrever e modificar documentos XML.

A operação de importação deve criar as entidades do domínio (utilizadores e ficheiros) correspondentes a todos os elementos que integram o documento XML a importar. A operação de exportação corresponderá a converter o conteúdo do sistema de ficheiros e seus utilizadores num documento XML. O documento XML produzido deverá poder ser posteriormente lido, sem erros, para um novo sistema de ficheiros vazio e sem utilizadores regulares. Para não criar dependências, qualquer ficheiro a importar só pode aparecer no doumento XML depois da diretoria a que pertence e depois do utilizador a que pertence.

A estrutura do documento XML, para simplificar, contém todas as entidades diretamente debaixo da raiz (`myDrive`). As operações de importação e exportação devem receber e devolver, respetivamente, um documento XML (`org.jdom2.Document`). O identificador único de cada objeto deve ser codificado como um atributo (`Attribute` em XML), enquanto os restantes elementos são codificados com elementos (`Element` em XML). Para ler e escrever os ficheiros XML sugere-se a utilização das classes `org.jdom2.input.SAXBuilder` e `org.jdom2.output.XMLOutputter`, respetivamente.

Na operação de importação, todos os elementos omissos devem ser substituídos pelos valores por omissão pretinentes. Na importação de um utilizador, além do *username*, apenas devem ser considerados os elementos que representam a palavra chave, nome, diretoria inicial e máscara. No caso dos ficheiros, apenas devem ser considerados os elementos: nome, *username* do dono do ficheiro, permissões, conteúdo (quando aplicável) e caminho para o ficheiro (*path*). Todas as diretorias que não existam no caminho para o ficheiro devem ser criadas como pertencentes ao utilizador *root*, o dono e as permissões aplicam-se apenas ao ficheiro final. É um erro tentar criar utilizadores ou ficheiros que já existam. As permissões do ficheiro são criadas tal como especificadas, sem aplicar a máscara do utilizador. Na operação de exportação devem também ser incluídos o identificador único atribuído ao ficheiro, bem como a data da última modificação.

3. Gestão do projeto

Na realização da primeira parte do projeto, os alunos terão que aplicar uma gestão simplificada do projeto. A gestão do projeto será realizada utilizando o *wiki* do projeto disponível no **github.com** para a definição de *issues* e milestones. Assim, a gestão do projeto a aplicar deve ser a seguinte:

- Criar uma *issue* para cada funcionalidade a desenvolver indicada neste enunciado. A etiqueta da *issue* deverá ser *story* (esta etiqueta deve ser adicionada pela equipa). Na criação da *issue* deve ser indicada uma frase que descreve a funcionalidade a desenvolver.
- A cada história ficará associada uma etiqueta com os pontos da história, a criar com os valores *1 s*, *2 s*, *3 s*, *5 s*, *8 s*, *13 s* e *21 s*. Esta etiqueta representa o esforço relativo previsto de desenvolvimento da história em unidades *s*.
- Deve ser criada uma página no *wiki* do projeto onde se deverá indicar as *issues* a desenvolver para esta parte do projeto. Esta página deve ter o nome *First Sprint*. A cada issue indicada nesta página deve ser associado o link para a página gerida pelo `github.com` com a descrição da atividade relacionada com a *issue* em causa.

- Deve ser criado o *milestone First Sprint* ao qual deverão ser adicionados todos os *issues* a concretizar na primeira parte do projeto.
- Cada issue deve ser atribuído a um membro da equipa utilizando a funcionalidade *Assign* disponível na página Web da *issue* em causa. A funcionalidade pode ser concretizada por um ou mais membros da equipa, sob a responsabilidade do *assignee*.
- Sempre que uma funcionalidade é completamente concretizada, deve ser feito um *commit* com as alterações efetuadas ao repositório e, de seguida, o *commit* deve ser enviado para o repositório central por forma a que os restantes membros da equipa possam aceder às alterações efetuadas. Este *commit* deve incluir na mensagem o texto *closes #id*, onde `id` indica o número da *issue* que foi concretizada. Desta forma, o `github` fecha automaticamente a *issue* correspondente. Deve ainda indicar o número de horas de trabalho envolvido por todos os membros da equipa na concretização da funcionalidade em causa na página *wiki First Sprint*.

Pode ser visto um exemplo da aplicação desta gestão simplificada aplicada à aplicação *PhoneBook* em `https://github.com/tecnico-softeng/phonebook/wiki`.

Note-se que este planeamento apenas tem a indicação das tarefas a realizar. É da responsabilidade da equipa fazer a distribuição temporal das tarefas pelo tempo disponível para realizar o projeto. Todos os elementos da equipa devem participar na realização do projeto, devendo haver uma distribuição o mais equitativa possível do trabalho a realizar. Não deve haver uma especialização do trabalho, ou seja, cada membro da equipa deve participar nos vários tipos de tarefas associadas ao desenvolvimento.

4. Realização do projeto

Todo o código e respetiva documentação, comentários no código ou páginas *wiki* (ver seção 3), devem ser escritos em lingua inglesa.

O código realizado no domínio da aplicação deve estar protegido contra utilização indevida por parte dos programadores que utilizam as classes do domínio.

A aplicação `PhoneBook` disponibilizada, apresenta uma estrutura muito semelhante à da aplicação a desenvolver, incluindo funcionalidades como a importação e exportação para XML.

O ficheiro *fenix-framework-jvstm-ojb.properties*, no diretório *src/main/resources* deve definir os dados a utilizar na ligação ao sistema de bases de dados (ver seção 4.2).

O corpo docente da disciplina de Engenharia de Software prevê que a realização desta parte do projeto exigirá cerca de 12 horas de trabalho a cada um dos membros da equipa. Nesta previsão, o corpo docente assume que os alunos já compreenderam o funcionamento da Fénix Framework antes da realização do trabalho e que existe um planeamento do projeto por forma a perceber o trabalho a realizar e a distribuir o trabalho pelos vários elementos da equipa.

4.1. Visualização

Para a vizualização do projeto, a equipa deve realizar uma classe `pt.tecnico.myDrive.Main` para exercitar o domínio realizado. Esta classe deve ser invocada automaticamente com a opção `exec:java` da ferramenta **maven**. Quando invocada sem argumentos numa base de dados vazia, esta classe deve criar um sistema de ficheiros básico: utilizador *root* e ficheiros a ele associados. Quando invocada com um argumento, o nome de um ficheiro XML acedido

como recurso do projeto, deve reiniciar o sistema de ficheiros e importar o conteúdo do ficheiro indicado.

A aplicação deve realizar as seguintes operações, após a iniciação do sistema e possível importação do ficheiro XML:

1. criar o ficheiro de texto `/home/README` com o conteúdo *lista de utilizadores*.
2. criar a diretoria `/usr/local/bin`.
3. imprimir o conteúdo do ficheiro `/home/README`
4. remover a diretoria `/usr/local/bin`.
5. imprimir a exportação em XML do sistema de ficheiros (ver seção 2)
6. remover o ficheiro `/home/README`
7. imprimir a listagem simples da diretoria `/home`

Com o objetivo de manter sempre atualizado o diagrama de classes UML pedido, sugere-se a utilização da ferramenta `dml2yuml`.

4.2. Avaliação do projeto

Semanalmente haverá uma avaliação da gestão do projeto. Esta avaliação será feita durante a aula de laboratório de cada equipa. Elementos do grupo que não comparecam ao laboratório terão uma avaliação de 0 (zero) nesta componente. Trabalhadores estudantes estão isentos de comparecer nas aulas de avaliação mas devem fornecer, semanalmente, o mesmo material que os restantes colegas. A avaliação terá em conta a qualidade do planeamento do projeto e da gestão do projeto que está a ser aplicada.

Na semana de **29 de Fevereiro a 4 de Março de 2016** , antes da respetiva aula de laboratório, o esqueleto do projeto deve estar submetido ao **github** e cada aluno deverá ter um *clone* atualizado deste na sua área local. O projeto deverá incluir no diretório **info/** o ficheiro inicial de especificação **dml** do projeto de cada aluno, identificado com o seu número mecanográfico (por exemplo `ist99999.dml`). Cada aluno deverá efetuar pelo menos um **commit**, em nome pessoal da sua conta de **github**, do respetivo ficheiro **.dml**.

Na semana de **7 de Março a 11 de Março de 2016** , antes da respetiva aula de laboratório, o projeto deve conter a especificação **.dml** finalizada. O projeto deve apresentar o *wiki* do **1**st **sprint** concluído, incluindo a especificação das histórias e respetivas atribuíções a membros da equipa (de uma forma equilibrada), bem como o respetivo *milestone*. O *wiki* deve incluir a identificação dos membros da equipa e conter pelo menos o nome (primeiro e último), número mecanográfico e nome de utilizador no **github**.

O prazo limite para a entrega da primeira parte do projeto é o dia **14 de Março de 2016** às **20h00** . O projeto a realizar deve apresentar uma estrutura de diretórios semelhante à da aplicação *PhoneBook* apresentada durante as aulas de laboratório. A diretoria **info/** deve conter os ficheiros iniciais de especificação **dml** de cada aluno, além de uma versão atualizada do diagrama UML de classes básico (sem métodos) em formato **.jpg** ou **.pdf** da solução adotada. O ficheiro `pom.xml` do projeto deve conter a descrição da equipa sob a etiqueta `developers` e suportar compilação e execução do projeto com o comando:

```
mvn clean package site exec:java -Dexec.args=drive.xml
```

O código produzido deve ser guardado no repositório Git de cada equipa. Cada equipa, após ter concretizado esta parte do projeto e ter guardado no seu repositório o código respetivo,

deverá criar a *tag* `R_1`. Esta *tag* representará a versão do código produzido para esta parte do projeto que os alunos querem submeter a avaliação.

Para facilitar a execução do código entregue, as equipas **têm** que utilizar os seguintes dados para a definição da ligação à base de dados:

- **username**: *mydrive*
- **password**: *mydriv3*
- **base de dados**: *dirvedb*

4.3. Penalizações

Projetos que guardem ficheiros desnecessários no repositório terão uma penalização na nota. Consideram-se desnecessários os ficheiros `.class` gerados na compilação das classes Java, os ficheiros `_Base.java` automaticamente gerados na compilação da DML, ou ficheiros `.jar`. Para isso, deverão criar o ficheiro *.gitignore* no diretótio base do projeto. Este ficheiro deverá indicar que o diretório *target* não deve ser colocado no sistema de controlo de versões Git.

O `github.com` apresenta estatísticas do desenvolvimento do projeto, por exemplo, número de *commits*, linhas adicionadas e removidas por cada membro da equipa. Pessoas em que o `github.com` mostra uma fraca participação no esforço coletivo de realizar o projeto terão uma penalização na nota.

Apenas a informação pedida deve ser impressa no terminal. Qualquer informação de depuração (*debug* ou *trace*) pode ser impressa no terminal através de uma instância da classe `Logger` do **log4j2** (`http://logging.apache.org/log4j/2.x/`).