DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

LEIC-A, LEIC-T, LETI

Engenharia de Software

2º Semestre – 2015/2016

Enunciado da Terceira Parte do Projeto

A terceira parte do projeto consiste na realização da camada de apresentação. Será também necessário enriquecer as camada de domínio e serviços. Neste enriquecimento é necessário ter em conta todas as regras de negócio indicadas nos primeiro e segundo enunciados e as que se apresentam neste. Será também necessário concretizar alguns testes, entre eles teste de integração, testes de sistema e testes de serviços com recurso a *mock-ups*, que requerem operações não realizadas.

1. Regras de negócio

Nesta parte é necessário considerar a concretização das seguintes regras de negócio:

- É criado um novo utilizador predefinido designado por *nobody*, sem palavra passe e com nome *Guest*. Este utilizador não pode alterar ou apagar ficheiros de outros utilizadores, mesmo que as permissões do ficheiro autorizem. O utilizador tem uma máscara inicial `rxwdr-x-`. Este utilizador não tem tempo limite no *login*, logo o seu *token* nunca expira, embora tenha de realizar *login* para obter um *token*. Este utilizador nunca pode ser apagado, nem a sua palavra passe alterada.
- O utilizador *root* passa a ter um tempo limite de 10 minutos, tendo o seu *token* de ser renovado com um novo *login* após esse tempo.
- Passa a existir, para cada utilizador, uma lista de associações entre o nome de uma extensão de ficheiro (uma cadeia de carateres) e um ficheiro. Desta forma, ao executar um ficheiro que não seja uma aplicação é utilizada a sua extensão para selecionar a aplicação que vai executar, recebendo o nome desse ficheiro como argumento. Esta lista é permanente e partilhada por todas as sessões do utilizador.
- As palavras passe passam a ter pelo menos 8 carateres, mesmo quando importadas de ficheiros XML. Utilizadores com palavras passe com menos de 8 carateres ficam impedidos de criar sessões. Os utilizadores predefinidos mantêm as suas palavras chave.

2. Camada de serviços

A concretização destes serviços podem implicar a modificação da camada de negócio por forma a que esta suporte novas funcionalidades necessárias para os serviços a desenvolver.

As funcionalidades a concretizar na camada de serviço são as seguintes:

- **Adiciona variável**
  Adiciona uma variável de ambiente ou redefine uma que já exista com o mesmo nome. Este serviço recebe o *token*, o nome da variável de ambiente e o respetivo valor. O serviço devolve a lista de variáveis de ambiente já definidas.
- **Executa ficheiro**
  Executa um ficheiro de texto. Este serviço recebe o *token*, o caminho para o ficheiro e uma lista de argumentos (cadeias de carateres).

3. Camada de apresentação

A camada de apresentação baseia-se num intrepretador de linha de comandos simples. Os comandos a disponibilizar tem correspondência direta com os serviços realizados na segunda e terceira partes. É da responsabilidade dos comandos processar o pedido do utilizador, efetuando as conversões e adaptações necessárias aos argumentos dos serviços e seus valores de retorno.

As funcionalidades a concretizar na camada de apresentação são as seguintes:

- **Login**: `login username [password]`
  Este comando realiza o serviço de *login*, guardando o *token* devolvido para futura utilização.
- **Change Working Directory**: `cwd [path]`
  Altera a diretoria corrente e imprime o caminho para a nova diretoria corrente.
- **List**: `ls [path]`
  Imprime a informação das entradas existentes na diretoria indicada pelo caminho (*path*) ou, caso este seja omitido, as entradas da diretoria corrente.
- **Execute**: `do path [args]`
  Executa o ficheiro indicado no caminho (*path*) com os argumentos (*args*).
- **Write**: `update path text`
  Altera o conteúdo de um ficheiro indicado no caminho (*path*) para o conteúdo *text*.
- **Environment**: `env [name [value]]`
  Cria ou altera a variável de ambiente com o nome (*name*) indicado e associa-lhe o novo valor (*value*). Se o valor for omitido imprime o valor que já lhe tinha sido associado. Se não forem indicados argumentos, imprime todas as variáveis de ambiente, e os respetivos valores, separados por '='.
- **Key**: `token [username]`
  O interpretador permite manter vários utilizadores em sessão, podendo trocar entre eles com o comando `token`. Quando invocado sem argumentos, imprime o valor do *token* e o *username* do utilizador atual. Quando invocado com um *username*, altera o utilizador atual, atualiza o *token* ativo em conformidade e imprime o seu valor.

Ao iniciar, o interpretador realiza o *login* do utilizador *Guest*. Como o utilizador *Guest* não tem tempo limite, deve ser retirado de sessão pelo interpretador quando qualquer outro utilizador realiza um *login*, bem como quando o interpretador termina.

4. Testes

Tal como na segunda parte do projeto, os novos serviços devem ser realizados seguindo a estratégia *Test First*.

Além destes testes, pretende-se realizar testes para as seguintes funcionalidades, ainda não realizadas:

- **Environment Links**
  Se uma ligação contiver no seu conteúdo o acesso a uma variável de ambiente, por exemplo `/home/$USER/profile`, o valor da variável de ambiente `$USER` deve ser utilizado para determinar o caminho definido pela ligação.
  Não se pretende com este teste realizar a semântica dos *environment links* no projeto. Assim, deverão ser utilizados *mock-ups* que permitam testar as operações relevantes quando um ficheiro de ligação contendo acessos a variáveis de ambiente é utilizado.
- **Execute Association**
  Uma das alterações introduzida na camada do domínio, nesta parte do projeto (ver seção 1), foi a existência de associações entre extensões e ficheiros. Pretende-se testar a sua utilização, sem que ainda tenham sido definidas operações no domínio ou sequer um serviço funcional. Assim, o próprio serviço de execução deve ser *mocked-up* por forma a simular a execução de um ficheiro a partir da sua extensão, sem a indicação da aplicação associada.

Alguns dos testes de unidade deverão ser integrados num teste de integração abrangente e a camada de apresentação deve ser testada com um teste de sistema exaustivo.

O **teste de integração do sistema** exercita a invocação sequencial dos serviços, simulando um cenário de utilização final por parte da camada de apresentação. Note que esta sequência corresponde a um único caso de teste e que os serviços já foram testados isoladamente. Os serviços ao serem invocados em sequência são composição de serviços. Desta forma, deve no projeto de teste ser criado uma *package* de nome *integration*. Os serviços invocados, e que não se encontram completamente realizados, devem utilizar o *JMockit* para substituir as invocações necessárias durante a realização dos testes.

O **teste de sistema** exercita a própria camada de apresentação, numa perspetiva *black-box*, invocando uma sequência de comandos como se de um interpretador se tratasse.

Note que o *teardown* destes testes deve limpar a base de dados uma vez que cada teste é uma sequência de transações, não sendo pois possível executar o teste no contexto de uma única transação que faz *rollback* no fim.

Tal como na segunda parte, a equipa deve assegurar que a bateria de testes tem uma boa cobertura dos serviços realizados.

5. Realização do projeto

Todo o código e respetiva documentação devem ser escritos em lingua inglesa. O código do domínio deve estar protegido contra os programadores. A aplicação `PhoneBook`, atualizada para a terceira parte, apresenta uma estrutura muito semelhante à da aplicação a desenvolver, incluindo a realização da apresentação e dos testes de integração e de sistema.

A gestão do projeto segue a estrutura indicada na segunda parte do projeto, mas com a página *wiki* e a *milestone* designadas por *Third Sprint*. Todo o código pedido nas primeira e segunda partes que seja necessário à correta realização da terceira parte deve ser realizado, sob pena de dupla penalização.

O corpo docente da disciplina de Engenharia de Software prevê que a realização desta parte do projeto exigirá cerca de 15 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, da camada de apresentação e dos testes de integração, de sistema e com *mock-ups* baseados em JMockit.

5.1. Avaliação do projeto

Semanalmente haverá uma avaliação da gestão do projeto nas mesmas condições indicadas no enunciado da segunda parte. Grupos que não comparecam ao laboratório terão uma avaliação de 0 (zero) nesta componente. A divisão semanal e equilibrada das tarefas fica a cargo da equipa, sob a supervisão e avaliação do docente de laboratório. Deverá ser submetida ao *fénix* uma cópia do projeto final entregue, sob pena de penalização. Aplicam-se as mesmas **penalizações** indicadas nos enunciados das partes anteriores.

Cada membro da equipa deve realizar pelo menos **um** teste e **um** comando da camada de apresentação de uma forma verificável (*commit*). Da mesma forma, os testes devem ser realizados antes dos serviços ou comandos que testam, numa perspetiva *test-driven design*.

A equipa deve garantir uma boa cobertura através de uma bateria de testes abrangente. Para a vizualização do projeto, a equipa deve realizar um interpretador de comandos na classe `pt.tecnico.myDrive.presentation.MyDrive`. Esta classe deve ser invocada automaticamente com a opção `exec:java` da ferramenta **maven**. O projeto a realizar deve apresentar uma estrutura de diretórios semelhante à da aplicação *PhoneBook* atualizada para a terceira parte. O prazo de entrega da segunda parte do projeto é o dia **13 de Maio de 2016** às **20h00** . Tal como na primeira parte, pode ser indicado o nome de um ficheiro XML, devendo a aplicação ser iniciada da mesma forma. O ficheiro `pom.xml` do projeto deve suportar compilação, teste e execução do projeto com o comando:

```
mvn clean install 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_3`** e a respetiva *release* no **github.com**. 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**: *drivedb*