DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

LEIC-A, LEIC-T, LETI

Engenharia de Software

2º Semestre – 2015/2016

Enunciado da Segunda Parte do Projeto

A segunda parte do projeto consiste na realização de um conjunto de serviços que irão suportar um determinado conjunto de funcionalidades para serem usadas na camada de apresentação. Será também necessário enriquecer a camada de domínio para que suporte os serviços que vão ser criados. Neste enriquecimento é necessário ter em conta todas as regras de negócio indicadas no primeiro enunciado e as que se apresentam neste. Será também necessário concretizar alguns testes aos serviços criados.

1. Regras de negócio

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

- O acesso à aplicação passa a ser realizada após uma operação de *login* (ver seção 1.1). A entidade que regista a informação associada ao *login* deve estar particularmente defendida contra alterações indevidas. Além disso, o sistema não deve permitir a enumeração ou listagem dos *token*s ativos (ver seção 1.1). Além disso, deve produzir um aviso (`log.warn()`) sempre que se tenta utilizar um *token* não ativo.
- A entidade de *login* tem associada a ela um conjunto de variáveis de ambiente, pares *nome-valor*, onde quer o *nome*, quer o *valor* são cadeias de carateres.
- As operações sobre os ficheiros passam a ser realizadas por utilizadores específicos, pelo que o sistema de permissões descrito anteriormente deve estar completamente operacional.
- O caminho (*path*) para um ficheiro desde a raiz, incluindo o seu nome, passa a ter um limite máximo de 1024 carateres, incluindo os separadores '`/`'.
- O *username* de qualquer utilizador deve ter pelo menos 3 carateres.

1.1. Operação de *login* de um utilizador

A aplicação passa a ter a noção de sessão. A entidade *login* é responsável por gerir o acesso de um utilizador em sessão ao sistema de ficheiros. A aplicação gere os utilizadores que realizaram

o processo de autenticação com sucesso e que por isso podem aceder à aplicação. Esta nova entidade deve ser persistente pelo que também deverá pertencer ao domínio da aplicação. Só os utilizadores que tenham uma sessão válida é que podem aceder à aplicação. Qualquer acesso feito por um utilizador que não esteja em sessão deve ser recusado pela aplicação. A operação de criação de uma sessão recebe o *username* do utilizador e a palavra chave e verifica se a palavra chave é igual à que está definida para o utilizador em causa. Em caso de sucesso, o utilizador é colocado em sessão.

Cada sessão fica ativa durante duas horas. Qualquer acesso feito pelo utilizador enquanto a sessão é válida reinicia o período de validade da sessão do utilizador por mais duas horas. A sessão de um utilizador fica inválida quando o utilizador está mais do que duas horas sem aceder à aplicação. Após ficar com a sua sessão inválida, caso o utilizador queira aceder à aplicação novamente terá que criar uma nova sessão. Por forma a manter a lista de sessões ativas o mais pequena possível, cada vez que um utilizador cria uma sessão são removidas todas as sessões inválidas, de qualquer utilizador, ainda no sistema.

Neste processo de criação de sessão é atribuído um *token* ao utilizador. O *token* é um número inteiro não nulo de $64bits$ (`long`) gerado aleatoriamente e com distribuição aproximadamente uniforme (por exemplo, `new BigInteger(64, new Random()).longValue()`), único de entre todas as sessões ativas no momento da sua geração. Este *token* deve ser depois utilizado nos restantes serviços para identificar o utilizador que quer realizar a operação.

A entidade *login* tem associado o conceito de diretoria corrente (*current working directory*). Todas as operações que especifiquem caminhos (*path*) relativos devem ser consideradas como tendo origem nesta diretoria.

Pode haver mais de uma sessão ativa para um mesmo utilizador, onde cada sessão tem associada o respetivo *token* e diretoria corrente.

2. Camada de serviços

Cada serviço é uma interface com o domínio da aplicação e deve ser executado como uma transação. Durante a execução de um serviço é necessário verificar se o utilizador no contexto do qual o serviço vai ser executado tem uma sessão válida. Caso isso não aconteça, a execução do serviço deve ser abortada. O serviço *Login User* é o único que não precisa de realizar esta verificação.

A concretização destes serviços pode implicar a modificação da camada de negócio por forma a que esta suporte novas funcionalidades necessárias para os serviços a desenvolver. É ainda necessário que o sistema garanta, de uma forma robusta, que o utilizador tem previlégios para realizar a operação, por exemplo, se pode ler ou escrever num ficheiro.

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

- **Login User**
  Cria uma sessão para um utilizador. Este serviço recebe o *username* e a palavra chave. O serviço devolve o *token* da sessão criada.
- **Change Directory**
  Altera a diretoria corrente da sessão. Este serviço recebe o *token* e o caminho, relativo ou absoluto. O serviço devolve o caminho para a nova diretoria corrente. Notar que alterar para a diretoria `'.'` permite saber qual a diretoria corrente.

- **List Directory**
  Efetua a listagem completa da diretoria corrente. Este serviço recebe o *token*. O serviço devolve as carateristicas relevantes de cada ficheiro na diretoria corrente.
- **Create File**
  Cria um ficheiro na diretoria corrente. Este serviço recebe o *token*, o nome a atribuir ao ficheiro, o tipo de ficheiro a criar e, eventualmente, um conteúdo. Notar que a criação de uma diretoria não recebe conteúdo, enquanto um ficheiro de ligação (*link*) recebe obrigatoriamente um caminho. No caso dos restantes tipos de ficheiros, o conteúdo é opcional. O ficheiro é criado com as permissões correspondentes à máscara do utilizador que o criou.
- **Read File**
  Lê o conteúdo de um ficheiro de texto, não de uma diretoria. Este serviço recebe o *token* e o nome do ficheiro. O serviço devolve o conteúdo do ficheiro.
- **Delete File**
  Elimina um ficheiro. Este serviço recebe o *token* e o nome do ficheiro. Caso se trate de uma diretoria, todos ficheiros nela contidos devem ser recursivamente apagados.
- **Write File**
  Altera o conteúdo de um ficheiro de texto, não de uma diretoria. Este serviço recebe o *token*, o nome do ficheiro e um conteúdo. Todo o conteúdo do ficheiro é substituído pelo novo conteúdo.

3. Testes de unidade

É necessário realizar testes de software, ao nível da camada de serviço, para todos os serviços desenvolvidos. O código desenvolvido deve ser realizado seguindo a estratégia *Test First*. Depois da concretização dos casos de teste de um serviço é que deverá ser concretizado o serviço em causa.

A concretização de um serviço implica a realização de código na classe correspondente ao serviço e poderá ainda ser necessário enriquecer a camada de lógica de negócio da aplicação e/ou adicionar novas classes noutros *packages* da aplicação.

Os testes devem ser independentes pelo que cada teste quando termina deve deixar a aplicação no estado em que a encontrou.

3.1. Cobertura

A equipa deve-se assegurar que a bateria de testes tem uma boa cobertura dos serviços realizados. Por forma a avaliar que os testes realizado oferecem uma boa cobertura, a equipa deve recorrer à ferramenta *cobertura*: `mvn cobertura:cobertura`

4. Gestão do projeto

A execução da segunda parte do projeto deve seguir a metodologia SCRUM. A gestão do projeto será feita utilizando as potencialidades do *github.com* já descritas na primeira parte do projeto. Numa página no *wiki* chamada *Second Sprint* constará a lista de todas as histórias (*Sprint Backlog* do SCRUM) segundo projeto, com um *link* para cada *issue* a desenvolver, a que se adiciona:

- Cada história (etiqueta *story*), representada pela respetiva *issue*, deve descrever uma funcionalidade do sistema do ponto de vista do utilizador. O esforço relativo previsto de desenvolvimento da história deve ser representado por uma etiqueta com os pontos da história. Cada história deve ser decomposta numa ou mais tarefas e conter na sua descrição um *link* para cada uma dessas tarefas.

- Para cada tarefa deve ser estimada a sua duração em horas, expressa por uma das etiquetas a criar *0,5 h*, *1 h*, *2 h*, *3 h* e *4 h*. Idealmente, a duração de uma tarefa deverá ser entre uma e duas horas. Cada tarefa, representada pela respetiva *issue*, deverá ser concretizada apenas por um membro da equipa e ter um tipo definido com uma das etiquetas: *feature* (alterações a código já realizado), *new feature* (novas funcionalidades), *test* (casos de teste) ou *bug* (resolução de erros). A tarefa deve conter no título um *link* para a história a que pertence.

- No caso de uma tarefa de teste, a lista dos casos de teste a concretizar deve ser definida na janela de comentário do *issue* correspondente à tarefa. Cada caso de teste deve descrever o que se pretende testar, indicar os valores a aplicar a cada um dos argumentos e o resultado esperado: sucesso (SUC) ou falha (FAIL). Sugere-se a utilização de uma tabela.

- Registo do estado da execução de cada tarefa. Há medida que cada membro termina as suas tarefas, deve ser registado a realização das tarefas no sistema utilizado para gerir o desenvolvimento do projeto. Em qualquer momento, deverá ser possível saber que tarefas é que ainda não estão realizadas e qual o código envolvido na concretização das tarefas já finalizadas.

Todas as *issues*, tarefas e histórias, ficam associadas ao *milestone Second Sprint* e devem ser fechadas com um *commit* contedo o texto *closes #id*. Na janela de comentário desta *issue* deve ser colocado o texto *Implemented by commit* seguido pelo identificador do *commit*.

O planeamento de cada semana, entre aulas de laboratório, consiste numa tabela na página *Second Sprint* no *wiki* com uma linha por membro da equipa e uma coluna com a lista de tarefas a realizar na semana seguinte, seguida de sete colunas com as previsões (em horas) a dispender em cada dia da semana. As duas últimas colunas representam o número de horas efetivamente gasto e as tarefas realmente realizadas na semana anterior. As discrepâncias devem ser justificadas, individualmente, após a tabela.

Pode consultar um exemplo da aplicação desta gestão de projeto para a aplicação *PhoneBook* em `https://github.com/tecnico-softeng/phonebook/wiki/Second-Sprint`.

5. Realização do projeto

Tal como na primeira parte, 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 segunda parte, apresenta uma estrutura muito semelhante à da aplicação a desenvolver, incluindo a realização de serviços e testes.

Todo o código pedido na primeira parte que seja necessário à correta realização da segunda parte deve ser realizado, sob pena de dupla penalização.

O corpo docente de Engenharia de Software prevê que a realização desta parte do projeto exigirá, em média, cerca de 12 horas de trabalho a cada aluno da equipa. Nesta previsão, o

corpo docente assume que os alunos já perceberam o funcionamento da Fénix Framework, da camada de serviços e a utilização da framework de testes JUnit.

5.1. Avaliação do projeto

Semanalmente haverá uma avaliação da gestão do projeto nas mesmas condições indicadas no enunciado da primeira 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.

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

A equipa deve garantir uma boa cobertura da bateria de testes. A visualização consiste somente na execução dos testes: `mvn test`

O prazo de entrega da segunda parte do projeto é o dia **15 de Abril de 2016** às **20h00** . O projeto a realizar deve apresentar uma estrutura de diretórios semelhante à da aplicação *PhoneBook* atualizada para a segunda parte. O ficheiro `pom.xml` do projeto deve suportar compilação e execução do projeto com o comando:

```
mvn clean test cobertura:cobertura site
```

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_2`**. 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*

Aplicam-se as mesmas **penalizações** indicadas no enunciado da primeira parte.