DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

LEIC-A, LEIC-T, LETI, MEIC-T, MEIC-A

Engenharia de Software

2º Semestre – 2014/2015

Enunciado da Segunda Parte do Projecto

1. Segunda Parte do Projecto ES

A segunda parte do projecto 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.

Nesta segunda parte do projecto deve-se ignorar a existência dos serviços externos SD-ID e SD-STORE dado que estes ainda não foram concretizados. A utilização destes serviços será realizada na terceira e quarta parte do projecto.

1.1. Regras de Negócio a Concretizar

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

- Só os utilizadores que podem escrever numa folha de cálculo ou que são o seu criador é que podem alterar os utilizadores que podem ler ou escrever na folha de cálculo.

1.2. Serviços a Concretizar

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

- **Login User**
  Efectua o *login* de um utilizador na aplicação. A aplicação passa a ter a noção de sessão. Esta nova entidade é responsável por gerir 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. O registo em sessão é feito por este serviço, o qual 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. Um utilizador fica com a sessão activa 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 realizar o *login* de novo. Por forma a manter a sessão o mais pequena possível, cada vez que um utilizador efectua um *login* são removidos da sessão todos os utilizadores que lá estejam há mais do que duas horas. Neste proceso de *login* é atribuído um *token* ao utilizador (uma cadeia de caracteres). Este *token* deve ser depois utilizado nos restantes serviços para identificar o utilizaor que quer realizar a operação. Por razões de simplificação, nesta parte do projecto, o *token* é criado na camada de domínio da aplicação e deve resultar da concatenação do *username* do utilizador com um número entre 0 e 9 gerado aleatoriamente.

- **Create User**
  Cria um utilizador. Este serviço recebe o *username*, a palavra chave e o nome do novo utilizador e o *token* do utilizador no contexto do qual este utilizador vai ser criado. Note que apenas o utilizador *root* é que pode criar um utilizador e não é possível criar dois utilizadores com o mesmo nome.
- **Remove User**
  Remove um utilizador. Este serviço recebe o *username* do utilizador a remover e o *token* do utilizador no contexto do qual este utilizador vai ser removido. Note que, tal como no caso anterior, apenas o utilizador *root* é que pode remover um utilizador. Do ponto de vista da lógica de negócio, a remoção de um utilizador implica remover todos as folhas de cálculo criadas pelo utilizador a remover.
- **Create Spread Sheet**
  Cria uma folha de cálculo. Este serviço recebe o número de linhas, colunas e nome da folha de cálculo a criar e o *token* do utilizador no contexto do qual esta folha de cálculo vai ser criada.
- **Assign Literal to Cell**
  Atribui uma constante inteira a uma determinada célula de uma dada folha de cálculo. Este serviço recebe o valor da constante representada por uma cadeia de caracteres, o identificador da célula a alterar, o identificador da folha de cálculo a alterar e o *token* do utilizador que quer realizar a alteração. O serviço deve devolver o novo valor da célula alterada. Note que cada utilizador só pode ler/alterar células de uma folha cálculo desde que tenha os privilégios necessários. O identificador de uma célula é representado por uma *string* com o formato `linha;coluna`.
- **Assign Reference to Cell**
  Atribui uma referência a uma determinada célula de uma dada folha de cálculo. Este serviço recebe o identificador da célula a alterar, o identificador da célula a referenciar, o identificador da folha de cálculo a alterar e o *token* do utilizador que quer realizar a alteração. O serviço deve devolver o novo valor da célula alterada.
- **Export Document**
  Exporta o contéudo de uma folha de cálculo para o formato XML. Este serviço recebe como argumento o identificador da folha de cálculo a exportar e o *token* do utilizador que quer realizar este serviço. Só os utilizadores que podem aceder a uma folha de cálculo (smodo leitura, modo escrita, dono) é que podem exportar uma folha de cálculo.

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 ter abortada. O serviço *Login User* é o único que não precisa de realizar esta verificação.

É necessário ainda verificar, nos serviços em que isso faça sentido, se o utilizador que quer realizar a operação tem privilégios para a realizar, por exemplo, se pode alterar a folha de cálculo no caso do serviço *Assign Reference to Cell*.

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.

1.3. Testes

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*. O corpo docente fornece já um conjunto de casos de testes para os serviços *Create User*, *Delete User* e *Login User*. Desta forma, cada grupo deve desenvolver primeiro estes serviços e depois deverá criar os testes para os restantes serviços. Depois da concretização dos casos de teste é que deverão ser concretizados os serviços 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.

1.4. Estado Persistente Inicial da Aplicação

Para se poder realizar a visualização da segunda entrega, é necessário ter um cenário de teste inserido no estado persistente da aplicação. O cenário de teste é igual ao definido para a primeira entrega do projecto e deve ser concretizado igualmente na classe `pt.tecnico.bubbledocs.BubbleApplication`. Esta classe deverá ser actualizada por forma a que nas situações em seja possível, o acesso às funcionalidades do domínio sejam realizadas através dos serviços criados e não por manipulação directa das entidades do domínio. Note que a inicialização do estado persistente da aplicação poderá afectar a execução dos casos de testes. Pode ser necessário remover a base de dados antes da execução dos casos de testes. ≪ß A utilização de serviços para realizar a inicialização do estado persistente pode exigir a colocação de um ou mais utilizadores em sessão utilizando previamente o serviço *Login User*. As excepções que possam ocorrer durante a execução das acções descritas no método `main` devem ser tratadas por forma a que todas as acções indicadas são realizadas, independentemente de o serem com sucesso ou insucesso.

2. Gestão do Projecto

O corpo docente de Engenharia de Software prevê que a realização desta parte do projecto exigirá, em média, cerca de 10 horas de trabalho a cada aluno do grupo. Nesta previsão, o corpo docente assume que os alunos já perceberam o funcionamento da Fénix Framework, a camada de serviços e a utilização da framework de testes JUnit.

Na realização da segunda parte do projecto, os alunos terão que aplicar uma gestão simplificada do projecto. A gestão do projecto será realizada utilizando o wiki do projecto disponível no `git-hub` e a capacidade de definição de *issues* e milestones no `git-hub`. Assim, a gestão do projecto a aplicar deve ser a seguinte:

- Cada grupo deve primeiro ler o enunciado com atenção;
- Criar uma *issue* para cada funcionalidade a desenvolver indicada neste enunciado. A etiqueta da *issue* deverá ser *story* (esta etiqueta deve ser adicionado pelo grupo). Na criação da *issue* deve ser indicado uma frase que descreve a funcionalidade a desenvolver. Opcionalmente, este descrição breve pode ser indicada na página wiki descrita de seguida.
- Deve ser criada uma página no wiki do projecto de cada grupo onde se deverá indicar as *issues* a desenvolver para a 2ª parte do projecto. Esta página deve ter o nome *First Sprint*. A cada issue indicada nesta página deve ser associado o link para a página gerida pelo `git-hub` com a descrição da actividade relacionada com a *issue* em causa.
- Deve ser criado o *milestone First Sprint* ao qual deverá ser adicionado todos os *issues* a concretizar na segunda parte do projecto.
- Na página *First Sprint* deve ainda ser descrito o conjunto de casos de testes a realizar para cada serviço sem casos de testes disponibilizados pelo corpo docente. Esta enumeração dos casos de testes deve ser feita com a colaboração de todos os elementos do grupo por forma a serem determinados todos os casos de teste a realizar. Esta enumeração deve ser realizada antes da concretização dos casos de teste.Para uma identificação mais fácil dos casos de testes sugere-se o uso de uma tabela.
- Cada issue deve ser concretizado por um ou mais membros da equipa. Essa atribuição deve ser indicada utilizando a funcionalidade *Assign* disponível na página Web da *issue* em causa.
- Sempre que uma funcionalidade é completamente concretizada, deve ser feito um *commit* com as alterações efectuadas 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 efectuadas. Este commit deve ter como mensagem *closes #id*, onde `id` indica o número da *issue* que foi concretizada. Ao realizar este procedimento, a *issue* em causa é fechada automaticamente pelo `git-hub`. 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-distsys-2015/phonebook/wiki`.

Note-se que este planeamento apenas tem a indicação das tarefas a realizar. É da responsabilidade do grupo fazer a distribuição temporal das tarefas pelo tempo disponível para realizar o projecto. Esta distribuição também corresponde a uma tarefa e deve ser a primeira tarefa a realizar pelo grupo e deve ser feita o mais cedo possível. Todos os elementos do grupo devem participar na realização do projecto, devendo haver uma distribuição o mais equitativa possível do trabalho a realizar. Não deve haver uma especialização do trabalho, ou seja, cada membro do grupo deve participar nos vários tipos de tarefas associadas ao desenvolvimento. Por exemplo, realizar testes, concretizar um serviço ou acrescentar uma regra de negócio.

Notas

O `git-hub` apresenta estatísticas do desenvolvimento do projecto, por exemplo, número de *commits*, linhas adicionadas e removidas por cada membro do grupo. Pessoas em que o `git-hub` mostra uma fraca participação no esforço colectivo de realizar o projecto terão uma penalização na nota.

A página wiki da versão v2.2 da aplicação *PhoneBook* disponível no *git-hub* tem um exemplo da aplicação desta gestão de projecto simplificada.

2.1. Entrega da Segunda Parte do Projecto

O prazo de entrega da segunda parte do projecto é até ao início da **sexta** semana de aulas do grupo de **25 a 31 de Março de 2015** . O projecto a realizar deve apresentar uma estrutura semelhante à da aplicação *PhoneBook* com Serviços (ver versão v2.2 da aplicação *PhoneBook*).

O código produzido deve ser guardado no repositório Git de cada grupo. Cada grupo, após ter concretizado esta parte do projecto e ter guardado no seu repositório o código respectivo, deverá criar a *tag* `R_2`. Esta *tag* representará a versão do código produzido para esta parte do projecto que os alunos querem submeter a avaliação.

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

- **username**: *bubble*
- **password**: *bubbl3*
- **base de dados**: *bubbledb*

2.2. Penalizações

Projectos que guardem ficheiros desnecessários no repositório terão uma penalização na nota de 2 a 4 valores. 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 directótio base do projecto. Este ficheiro deverá indicar que o directório *target* não deve ser colocado no sistema de controlo de versões Git.