1 Introdução

Com o sucesso da versão 1.0 do *software* TornGes decidiu-se avançar para a versão 1.5 adotando o lema *TornGes for Everyone, Everywhere, and at Anytime* (TGE^2A), que tem como objetivo permitir a gestão de torneios de modalidades-Elo a partir de diversos dispositivos com ligações à *internet*.

Nesta versão pretende-se oferecer ao utilizador a possibilidade de aceder às operações da aplicação recorrendo a um navegador *web* e a uma aplicação cliente *desktop*.

Dada a natureza do serviço oferecido e a qualidade pretendida, decidiu-se recorrer a um servidor aplicacional compatível com as especificações Java EE — o *Wildfly*. Tendo em conta a matéria lecionada na disciplina, foi também decidido que a construção da aplicação cliente *web* deve ser feita recorrendo a *Servlets* e *Java Server Pages* e que a construção da aplicação cliente *desktop* deve ser feita recorrendo ao *JavaFX*.

2 O que devemos fazer?

Nesta segunda fase do projeto devem:

- autonomizar a camada de negócio da aplicação de forma a que esta possa ser acedida por vários tipos de clientes, tirando partido do *Java EE EJB Container*;
- adaptar e completar o esqueleto da aplicação *web* que vos é fornecida para interagir com esta camada de negócio, tirando partido do *Java EE Web Container*;
- desenvolver uma aplicação cliente que disponibilize uma interface *GUI* que aceda à camada de negócio utilizando *RMI*, recorrendo ao *Java EE Application Client Container*.

Dada a urgência em lançar esta nova versão para o mercado (que é como quem diz, dado que o semestre está a acabar), decidiu-se que a primeira versão dos clientes *Web* e *desktop* teriam funcionalidades limitadas, cobrindo apenas algumas das operações da aplicação. Mais concretamente:

- o cliente *desktop* deve permitir realizar os casos de uso **Criar Torneio** e **Registar Participante em Torneio**;
- o cliente *web* deve permitir realizar os casos de uso **Registar Resultado** e **Visualizar Calendário de Jogador**

O material de apoio a esta fase do projeto inclui um projeto Java EE que define a estrutura da vossa aplicação em termos de projetos Java EE. Apesar da estrutura ser a da aplicação *TornGes*, o projecto inclui código-fonte da aplicação *SaleSys* de forma a mostrar exemplos funcionais que podem usar como base. O projeto *Enterprise Java Beans* (do tipo *maven*) é composto por cinco sub-projetos:

- *torngest*, que agrega a informação dos restantes (não têm de programar nada neste projeto);
- *torngest-ear*, que contém informação de como assemblar a aplicação empresarial Java a instalar no servidor (não têm de programar nada neste projeto);
- *torngest-business*, que contém a camada de negócio organizada de acordo com o padrão *Domain model* e *Data Mapper* JPA;
- *torngest-web-client*, que é um cliente *Web* desenvolvido de acordo com o padrão *Model-View-Controller* aplicado à `Web`, que usa *JSP* para a visualização de informação (padrão *Server side template* e *Template View*) e *Servlets* para implementar o controlador (padrão *Front controller*). Este cliente acede à camada de negócio utilizando EJB de sessão remotos;
- *torngest-gui-client*, que é um cliente com uma *GUI* programada em JavaFX que acede à camada de negócio via *RMI* recorrendo ao *Java EE Application Client Container*.

Em concreto, devem:

- No projeto correspondente à camada de negócio:
  - Substituir as classes que vos fornecemos pelas classes que desenvolveram na primeira fase deste projeto (ou melhoramentos das mesmas).
  - Identificar quais são os *session beans* a fazer e anotá-los convenientemente.
  - No caso de escolherem usar apenas *stateless session beans* (fortemente recomendado) efetuar, se necessário, as modificações necessárias em classes que têm estado.
  - Identificar quais os *session beans* que precisam de interfaces remotos, identificar o que estes devem incluir, definir estas interfaces e suas implementações.
  - Transformar a interação com a camada JPA de forma a passar a ter *container-managed persistence*, i.e., de forma a tirar partido

dos serviços oferecidos pelo servidor aplicacional. Em particular, devem tirar partido do controle de transações (*Java Transaction API*) e da injeção de contexto e dependências.

  - Identificar se é necessário utilizar as primitivas oferecidas pelo JPA para a gestão de concorrência.

- No projeto correspondente à apresentação utilizando uma aplicação *Web*:

  - Adaptar a aplicação *web* de forma a utilizar a vossa camada de negócio e disponibilizar as funcionalidades indicadas anteriormente. O acesso à camada de negócio deverá ser feito através das interfaces dos *session beans*.
  - O controlador, que segue o padrão *Front controller*, é totalmente reutilizável, mas devem listar a correspondência entre endereços *web* e as vossas ações no ficheiro *app.properties*. Só precisam proceder ao desenvolvimento das ações.
  - A visualização (que recorre a JSP) tem de ser refeita para ir de encontro aos vossos casos de uso e o mesmo acontece com os *helpers* ou *view models*. As modificações que efetuarem devem continuar a respeitar os padrões escolhidos.

- No projeto correspondente à apresentação utilizando um cliente *GUI* feita em JavaFX:

  - Recorrer a uma variante do *MVC* apropriada recorrendo a propriedades e *data binding*.
  - Desenhar as *cenas* utilizando a aplicação *Scene Builder*.
  - Programar os *controladores* seguindo o padrão *Page controller*.
  - O acesso à camada de negócio deve ser feito através da injeção de *session beans* remotos, mas seguindo as regras do Java EE *application client container*.

A execução da aplicação deve recorrer a dois servidores, além da aplicação cliente. O servidor de base de dados está disponível no endereço `http://dbserver.alunos.di.fc.ul.pt` e o acesso é feito como no primeiro projeto; o servidor Java EE (*Wildfly*) estará disponível localmente (nos vossos computadores) ou nas máquinas dos laboratórios.

3 O que recebemos e como usamos?

O material de apoio a esta fase do projeto encontra-se no mesmo repositório que o da Fase 1. Para começar o desenvolvimento deverão:

- Adicionar o repositório original (css000) como remote: `git remote add mainstream git@git.alunos.di.fc.ul.pt:css000/css_tornges.git`
- Importar as actualizações da Fase 2 para o vosso repositório local: `git pull mainstream master` seguido de `git push origin master`
- A pasta `Fase2/torngest/` é um projeto *maven*. Como tal, devem proceder à sua importação para o vosso IDE recorrendo à importação de projetos Maven que o IDE disponibiliza. No caso do *Eclipse*, devem fazer *import from maven*.
- O projeto usa JPA ligado a MySQL. Devem alterar os ficheiros que definem estas ligações para usarem a base de dados do vosso grupo e ter a VPN ligada.
- Se tudo correr como esperado, o projeto importa sem erros. Notem que devem ter acesso à Internet durante o processo de importação para que o *Maven* descarregue as dependências necessárias. Se ocorrer algum problema durante a importação (e tinham rede, vpn, etc.), devem, no Eclipse, escolher a opção *Maven > Update Project...* e depois selecionar quais os projetos a atualizar e selecionar a opção *Force Update of Snapshots/Releases* para forçar que o *plug-in* do *Maven* repita o processo de importação dependências e de configuração do Eclipse. Noutros IDE há que encontrar a operação correspondente.
- Instalar a aplicação no servidor *Wildfly*, fazendo por exemplo, *run on server* do *index.html* do cliente *Web*.
- Aceder à aplicação usando um navegador e o cliente JavaFX fornecido, certificando-se que está tudo a funcionar como é suposto. Note que foi configurado o *drop&create* do esquema da base de dados de cada vez que a aplicação é instalada (por ser o mais apropriado para fazer testes).
- Efetuar as alterações descritas na seção anterior.

4 Como e quando entregamos?

Identifiquem o *commit* como sendo a entrega da Fase 2 (`git tag fase2`) e coloquem essa identificação no servidor gitlab (`git push origin fase2`). Consideramos apenas este tag se o commit tiver uma **data anterior** a **1 de Junho**, data a partir da qual não serão aceites alterações.

O repositório deve conter:

1. o código-fonte do vosso projeto;

2. um documento *PDF* com as decisões importantes que tenham tomado em termos de desenho da aplicação, nomeadamente a forma como resolverem cada um dos aspetos listados na seção sobre o que têm a fazer (por exemplo, que EJB definiram, que interfaces definiram para esses EJBs, etc).

Recordem que o trabalho é em grupo mas a avaliação é individual e que será utilizado o histórico do git para aferir o grau de participação dos diferentes elementos do grupo no trabalho desenvolvido.

5 Notas Finais

Algumas dicas para ajudar a superar alguns problemas recorrentes:

- Execução do cliente JavaFX no *application client container*:
  - Tem de ter a aplicação a correr no servidor para poder usar qualquer uma das aplicações cliente.
  - Em Windows, em vez de executar o *appclient.sh* (do bin do wildfly) têm de executar o *appclient.bat*
  - Para fazer aparecer o ficheiro *torngest-ear/target/torngest-ear-1.0.ear* de que precisamos para executar o cliente (com o script do item anterior) podem fazer o seguinte:
    1. selecionar o projeto *torngest-ear*
    2. escolher a opção *export* no menu de contexto
    3. escolher como destino *"path-to-pai do torngest-ear no git"/torngest-ear/target/torngest-ear-1.0.ear*
    4. correr o comando sh indicado no *client.Main* deste projeto
    5. repetir passos anteriores sempre que fizerem alterações neste projeto
- No cliente JavaFX façam *reorganize imports* em todas as classes deste projeto de forma a não terem referências a classes do business do *SaleSys*
- Quem tem Windows e a placa gráfica Nvidia, vai ter o porto 9990 (por default o porto usado pelo wildfly para a consola de admin) ocupado pela placa gráfica. Devem alterar o ficheiro

  `"path-to-wildfly"/standalone/configuration/standalone.xml`

  e alterar o sítio em que é referida a porta 9990 para outra, por exemplo, 8990 e fazer restart do servidor.
- Se tiverem alguma outra coisa a correr no porto 8080 (onde vai ficar publicado por omissão o cliente web) terminem-na.

- Não se esqueçam que durante da execução do cliente web, o navegador pode usar o que tem na *cache* em vez de ir buscar o código com as alterações que acabaram de fazer. Averiguem como podem forçar o *reload* no navegador que estão a usar. É recomendado usarem o modo de navegação privada durante o desenvolvimento.