1 Introdução

A organização de eventos com uma componente competitiva, que vão desde provas desportivas a torneios de jogos de tabuleiro ou cartas, está a afirmar-se como uma atividade importante e com potencial lucrativo considerável. No entanto, a organização destes eventos tem diversos intervenientes com interesses diferentes e apresenta desafios complexos, não só na definição e gestão dos eventos, como na comunicação com as diversas partes interessadas.

Embora existam algumas ofertas no mercado de aplicações para apoiar a organização de eventos, estas centram-se na assistência logística, por exemplo, na venda de bilhetes, e não na parte de gestão das competições dos eventos. Assim, aproveitando esta necessidade de mercado, CSS lançou-se na construção de uma aplicação que apoie a organização de eventos competitivos nas suas várias vertentes, a saber, gestão das competições, logística, divulgação e financeira.

No ano letivo 16/17, os alunos de CSS produziram uma primeira iteração de um módulo[1] de um software de gestão de eventos. Este módulo, designado por DesporGes, é centrado apenas no apoio à gestão de provas desportivas de tipo *taça* e *campeonato*. O que se propõe aos alunos de CSS este ano é a construção de um novo módulo centrado na gestão de torneios que assentam na classificação de jogadores com o sistema de Elo[2].

2 TornGes

Na primeira iteração, o TornGes vai centrar-se no apoio à gestão de torneios de modalidades cujos jogadores são classificados pelo sistema de Elo e em equipas de jogadores destas modalidades (classificadas da mesma forma). Vão ser considerados dois tipos de torneio: individual e de equipa. Em ambos os casos, um torneio envolve vários encontros entre dois participantes. Nos torneios individuais os encontros são entre os jogadores registados enquanto que nos torneios de equipa os encontros são entre os jogadores das equipas registadas.

Especificamente, o TornGes vai disponibilizar quatro casos de uso cuja descrição breve é fornecida abaixo.

[1] Pode ler sobre esta forma de modularização centrada no domínio em `https://martinfowler.com/bliki/PresentationDomainDataLayering.html`

[2] `https://en.wikipedia.org/wiki/Elo_rating_system`

Criar torneio — O gestor do torneio indica que quer criar um novo torneio. O sistema indica as designações das modalidades existentes e pede a modalidade, a designação do torneio, o seu tipo (torneio de equipa ou torneio individual), o número de participantes, o número de encontros entre cada par de oponentes, a data de início do torneio e a sua duração em dias. O gestor fornece esta informação. O sistema verifica que:

1. a modalidade indicada é válida,
2. não existe outro torneio com a mesma designação,
3. o número de participantes e o número de encontros entre cada par de oponentes são maiores que zero e pares,
4. no caso de torneio ser de equipa, o torneio dura pelo menos 2 dias e apanha um fim-de-semana

e o sistema cria um novo torneio com os dados fornecidos que fica aberto para o registo de participantes.

Nesta primeira iteração não se vão desenvolver os casos de uso para criar modalidades, jogadores e equipas. Tal ficará para uma iteração futura. Parta do princípio que existe um leque fixo de modalidades e que todos os jogadores e equipas de cada modalidade já foram de alguma forma introduzidos. Equipas e jogadores de cada modalidade têm um número de inscrição no sistema, um nome e o número de pontos (calculado com o sistema de Elo). Os jogadores podem pertencer a uma equipa e definem o diferencial máximo de pontos relativamente aos jogadores com que aceitam jogar, que apenas se aplica nos torneios individuais.

Registar Participante em Torneio — O gestor indica que quer registar um participante (jogador ou equipa) num torneio, fornecendo a designação do torneio pretendido e o número de inscrição do participante no sistema. O sistema verifica que:

1. a designação do torneio fornecida é de um torneio que ainda está aberto,
2. o número de inscrição fornecido é de um participante da modalidade do torneio e tem o tipo certo (é uma equipa se e só se o torneio é de equipa),
3. o participante não está registado em nenhum outro torneio que decorra nos mesmos dias,
4. no caso de torneios de equipa, a equipa participante tem um número de elementos que está acima de um dado limite (o qual varia com a modalidade),

e regista o participante no torneio. No caso de ter sido atingido o número de participantes no torneio, o sistema fecha o registo no torneio e cria todos os

seus encontros. Seja N o número de encontros entre cada par de oponentes do torneio e M o limite mínimo do tamanho da equipa na respetiva modalidade. No caso de torneios de equipa, cada equipa tem um confronto com cada uma das outras equipas participantes. Cada confronto consiste em N encontros entre os M melhores jogadores com o mesmo *rank* nas duas equipas, ou seja, N encontros entre os dois jogadores com mais pontos em cada equipa, N encontros entre os dois segundos de cada equipa, etc.. Cada participante é o primeiro a jogar em metade dos encontros. No caso de torneios individuais, os confrontos são apenas entre jogadores cujo diferencial na pontuação esteja dentro dos limites impostos por ambos os jogadores. Cada confronto consiste em N encontros entre os dois jogadores e cada jogador é o primeiro a jogar em metade dos encontros.

Registar Resultado de Encontro — O gestor indica que quer registar o resultado de um encontro num torneio, fornecendo a designação do torneio pretendido. Se a designação for válida e a data corrente não for anterior à data de início do torneio, o sistema mostra os encontros do torneio sem resultados, mais especificamente, o número do encontro, o nome do primeiro e segundo oponente. Os resultados devem estar ordenados por número de encontro. O gestor escolhe o número de um encontro e indica o resultado de entre três possibilidades: vitória, empate, derrota (do primeiro oponente). O sistema verifica que o número do encontro é válido e efetivamente ainda não tem um resultado registado, regista o resultado e atualiza as pontuações dos jogadores intervenientes e, das suas equipas, no caso de se tratar de um torneio de equipa. O processo de registo de resultados para o torneio repete-se até o gestor indicar que quer terminar o caso de uso.

Visualizar Calendário de Jogador — O diretor indica que quer visualizar o calendário de confrontos de um jogador fornecendo o seu número de inscrição e uma data. O sistema valida que o número de inscrição não é de uma equipa e mostra uma lista ordenada por data com os confrontos do jogador em torneios posteriores a essa data. Devem ser considerados tanto os torneios individuais como os de equipa a que pertence. As entradas da lista devem conter a designação do torneio e a sua data de início, o jogador adversário e o diferencial de pontos, e o número de encontros.

3 Que devemos fazer?

Nesta primeira fase do projeto deve desenvolver os 4 casos de uso da aplicação TornGes, sem se preocupar ainda como será feita a apresentação. A aplicação deve estar organizada em camadas, com uma camada de acesso aos dados, uma camada com a lógica de domínio e uma camada de serviços. A camada de acesso aos dados deve, recorrendo ao JPA, seguir o padrão *data mapper*; a camada com a lógica de domínio deve seguir o padrão *domain model*, fornecendo os metadados necessários ao JPA e procedendo à persistência

dos objetos recorrendo a esta API; e a camada de serviços terá simplesmente o propósito de expor as funcionalidades da aplicação (API) escondendo a organização da camada de negócio.

Para efeitos de cálculo das pontuações com o sistema Elo podem seguir `http://goo.gl/Z79YD5` e usar $K = 32$.

Para efeitos de demonstração da aplicação deve implementar ainda uma classe *SimpleClient* que exercite cada um dos casos de uso (sem qualquer leitura do teclado), imprimindo as informações relevantes na consola e ter uma forma expedita de carregar na base de dados os dados necessários para correr este cliente. Em anexo é descrito em detalhe o que deve fazer este programa.

Em concreto, devem:

- elaborar os SSD e o modelo de domínio de forma a identificar as operações do sistema, os conceitos relevantes e associações entre estes;
- pensar no desenho do sistema, elaborar um modelo de classes, e proceder à sua implementação seguindo o padrão *Domain Model*;
- baseado no modelo de classes, definir um ORM e anotar as entidades de forma a poderem utilizar a camada de acesso aos dados baseada em JPA (*application-managed persistence*);
- adaptar os vossos *handlers* dos casos de uso de forma a efetuar a persistência das entidades de forma correta;
- gerar a base de dados a partir das anotações JPA;
- disponibilizar um cliente simples que exercite os quatro casos de uso.

Idealmente, o sistema de gestão de base de dados (e a respectiva base de dados) devem ser instaladas numa máquina diferente da do servidor aplicacional. Para tal, devem utilizar a máquina disponível no endereço `http://dbserver.alunos.di.fc.ul.pt` acessível via VPN da FCUL. Nesta máquina têm instalado a *MariaDB* e o *phpmyadmin*.

A aplicação deve ser descrita por um projeto Maven Java. Devem recorrer ao plugin *SonarLint* para o Eclipse[3] para controlarem a qualidade do código da vossa aplicação.

O código deve ser desenvolvido em equipa e deve ser utilizado o sistema de controlo de versões *git*, disponível no servidor `git.alunos.di.fc.ul.pt`. Um elemento do grupo de trabalho deverá seguir as instruções para fazer *fork* do repositório em `http://git.alunos.di.fc.ul.pt/css000/css_tornges/`. O repositório pode pertencer a qualquer um dos elementos do grupo, deve ser privado, e só os elementos do grupo e o utilizador *css000* devem ter acesso à informação do repositório. O utilizador *css000* só necessita de permissões de leitura dos dados (ou seja, pode ser programador).

[3] `https://www.sonarlint.org/eclipse/` e instalável através do *Eclipse Marketplace*

4 Como e quando entregamos?

Identifiquem o *commit* como sendo a entrega da Fase 1 (`git tag fase1`) e coloquem essa identificação no servidor gitlab (`git push origin fase1`). Consideramos apenas este tag se o commit tiver uma **data anterior** a **30 de Abril**, data a partir da qual não serão aceites alterações.

O repositório deve conter na pasta Fase1:

1. o código fonte do vosso projeto (o que inclui o .pom);
2. um **único** documento **PDF** com os SSDs, o modelo de classes, o modelo com o ORM e outras decisões importantes que tenham tomado em termos de desenho da aplicação.

5 Notas Finais

Na página da disciplina encontram disponível uma implementação de uma primeira versão do DesporGes, que cobre os requisitos da 1ªparte do projeto de CSS do ano passado e que possivelmente vos poderá ajudar a tomar algumas decisões de desenho do TornGes.

Usem o forum de discussão para esclarecer questões sobre projeto, nomeadamente sobre o enunciado. Dessa forma os outros grupos, eventualmente com as mesmas questões, poderão beneficiar dos nossos esclarecimentos. Podem evidentemente também usar o horário de atendimento semanal dos docentes da disciplina.

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.

Anexo

Considere os seguintes dados para escrever o *SimpleClient* e para a inicialização da base de dados.

Estado da base de dados

Modalidades Existe uma única modalidade, o Xadrez, e nesta modalidade o limite mínimo para o tamanho da equipa é 2.

Jogadores Existem 10 jogadores de Xadrez:

nInscrição	Nome	ELO	Team	Δ ELO
10	Garry Kasparov	2000	URSS	1000
11	Anatoly Karpov	1900	URSS	1000
12	Bobby Fisher	1800	USA	200
13	Paul Morphy	1700	USA	200
14	Magnus Carlsen	1500	Norway	200
15	Jon Ludvig Hammer	1600	Norway	200
16	Ding Liren	1500	China	1000
17	Yu Yangyi	1300	China	100
18	Reuben Fine	100	USA	100
19	Alexander Onischuk	100		100

A terceira coluna refere-se à pontuação inicial segundo o sistema ELO, a segunda coluna refere-se à equipa a que pertence. O número de inscrição de cada equipa é a ordem por que aparece nesta tabela: URSS - 0, USA - 1, Norway - 2, China - 3. Todas as equipas começam com 1000 pontos.

Cenário *SimpleClient*

- Crie um torneio com os seguintes dados: nome: Torneio A, modalidade: Xadrez, tipo: Individual, participantes: 8, número de encontros: 4, data: 2018/03/28, duração: 1 dia
- Registe no torneio *Torneio A* todos os jogadores com números de inscrição de 10 a 17, de modo a fechar o torneio.
- Visualize o calendário do jogador 10 (Garry Kasparov) desde a data 2018/02/01. Deverá imprimir no *standard output* a lista de confrontos obtida, um confronto por linha. Cada linha, como ilustrado abaixo, deve ter o nome do torneio, a data, o jogador adversário, o número de encontros e a diferença de pontuação entre eles.

  ```
  Torneio A | 18/03/28 | Anatoly Karpov | 4 | +100
  ```

  ...

- Registe, pela ordem indicada, os seguintes resultados de encontros do *Torneio A*:
  - Primeiro encontro Garry Kasparov x Anatoly Karpov: Vitória (para Garry Kasparov).
  - Primeiro encontro Garry Kasparov x Ding Liren: Derrota.
  - Segundo encontro Garry Kasparov x Ding Liren: Derrota.
- Volte a visualizar o calendário do jogador 10 (Garry Kasparov) desde a data 2018/02/01.
- Crie um torneio com os seguintes dados: nome: Torneio E, modalidade: Xadrez, tipo: Equipa, participantes: 4, número de encontros: 6, data: 2018/05/18, duração: 4 dias
- Registe todas as equipas que foram aqui apresentadas no *Torneio E* (número de inscrição de 0 a 3).
- Visualize o calendário do jogador 10 (Garry Kasparov) a partir da data 2018/02/01.
- Visualize o calendário do jogador 18 (Reuben Fine) a partir da data 2018/02/01.