1 Introdução

Como explicado anteriormente, o que se propõe aos alunos de CSS este ano é o desenvolvimento de um novo módulo de um software de gestão de eventos e atividades desportivas, designado por *AulasGes*, centrado na gestão de aulas de diversas modalidades.

Foi definido que a primeira iteração do *AulaGes* iria focar-se apenas em quatro casos de uso — *Criar Aula, Ativar Aula, Inscrever em Aula, Visualizar Ocupação de Instalação* (*vide* enunciado da entrega anterior). Adicionalmente foi definido que:

- o código do *AulaGes* deve ser organizado em camadas, incluindo uma camada de serviços, uma camada com a lógica de domínio e uma camada de acesso aos dados;
- 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 meta-dados necessários ao JPA e procedendo à persistência dos objetos recorrendo a esta API;
- a camada de serviços tem (por agora) simplesmente o propósito de expor as funcionalidades da aplicação (API) escondendo a organização da camada de negócio.

Para a entrega anterior, focaram o vosso trabalho na concepção e desenvolvimento da camada com a lógica de domínio e da camada de serviços. Neste trabalho pretende-se que se foquem no acesso e persistência dos dados.

2 Que devemos fazer?

Devem equipar a vossa solução do *AulaGes* da entrega anterior de forma a que os dados sejam persistidos. Para efeitos de demonstração da aplicação devem ainda 1) implementar uma classe *SimpleClient* que exercite cada um dos casos de uso em diferentes cenários (sem qualquer leitura do teclado), imprimindo as informações relevantes na consola e 2) ter uma forma expedita de carregar na base de dados os dados necessários para correr este cliente.

Em concreto, devem:

- definir um mapeamento do modelo de objetos da vossa solução num modelo relacional (ORM) e descrever este mapeamento com o JPA;
- recorrendo ainda ao JPA, e aos elementos que permitem definir *application-managed persistence*, desenvolver os catálogos e adaptar os *handlers* dos casos de uso de forma a efetuar o acesso e a persistência das entidades de forma correta;
- gerar a base de dados a partir das anotações JPA;
- disponibilizar um cliente simples que não exija qualquer interação e exercite os quatro casos de uso de acordo com o descrito no anexo;
- disponibilizar uma forma expedita de carregar na base de dados os dados necessários para correr este cliente.

O sistema de gestão de base de dados (e a respectiva base de dados) idealmente é instalada numa máquina diferente da do servidor aplicacional. Assim, 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* e devem continuar a usar o ***SonarLint*** para vos ajudar a controlar a qualidade do vosso código e o ***Git*** para vos ajudar no desenvolvimento cooperativo.

3 Por onde começar?

Um elemento do grupo de trabalho deverá seguir as instruções para fazer *fork* do repositório em https://git.alunos.di.fc.ul.pt/css000/css_meta4. **Importante:** devem definir o vosso *fork* como sendo privado (outros alunos não o poderão ver) e adicionar à lista de membros do projeto o utilizador css000 com o nível de *Reporter*.

O repositório fornecido contém um projeto *maven* com o código de uma versão do sistema *SaleSys* com características semelhantes às que se pretende nesta entrega. Notem que o *.pom* do projeto define que como JPA *provider* é usado o *EclipseLink*. Devem começar por analisar este projeto; existe um README e é exatamente por aí que devem começar!

Uma vez percebida a estrutura do projeto e o que têm a fazer, devem copiar as classes da vossa 3ª entrega para dentro deste projeto e eliminar/adaptar todos os artefactos que dizem respeito ao *SaleSys* (não esquecendo de adaptar o *persistence.xml* com *user*/*password* do vosso grupo).

Na página da disciplina também têm disponível o código e documentação de uma versão do sistema *EspacoGes* realizada por colegas vossos no ano passado (que foi selecionado por cumprir, em grande medida, o que era pretendido).

4 Como e quando entregamos?

Identifiquem o *commit* como sendo a entrega 4 (`git tag entrega4`) e coloquem essa identificação no servidor gitlab (`git push origin entrega4`). O *deadline* para entrega é **3 de Maio**.

O repositório deve conter:

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

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 Critérios de Correção

Para ajudar a guiar o vosso esforço indicam-se alguns critérios de correção que vamos usar:

Geral

- em que medida a leitura do relatório permite saber 1) as operações que, na vossa solução, suportam cada um dos casos de uso; 2) os tipos em torno das quais montaram a vossa solução e as dependências que existem entre eles; 3) as decisões mais importantes que tomaram relativamente ao mapeamento ORM em termos de estrutura
- em que medida a organização do vosso código e do projeto adere aos requisitos que vos foram dados

- em que medida o comportamento do sistema, em cada cenário de teste, é o esperado (isto é feito através da execução do cliente, e envolve não só olhar para o output do mesmo como a inspeção do estado da base de dados no servidor)

Detalhe

- em que medida as decisões que tomaram nos serviços e *handlers* dos casos de uso são consistentes com o que dizem no relatório; três questões especialmente sensíveis: 1) é preciso manter estado durante a execução de um caso de uso? 2) os objetos usados para dar-input/devolver-output são objetos que têm o simples propósito de transferir dados, sem qualquer lógica associada? 3) o input/output da camada dos serviços é exclusivamente realizado através destes objetos (não há qualquer leitura nem escrita)
- em que medida as decisões que tomaram relativamente ao mapeamento ORM são adequadas: uso apenas de chaves sem significado, declaração de restrições aos dados que são importantes, mapeamento adequado para atributos com dados (temporais, *user-defined,* enumerados, etc) e mapeamento adequado das diversas relações, nomeadamente das relações de herança
- em que medida os vossos catálogos e *handlers* fazem uma utilização correta das primitivas oferecidas pelo JPA (uso dos *entity managers*, gestão de transações, etc)
- em que medida definiram as interrogações apropriadas para lidar com as pesquisas necessárias; uma questão importante é que não estão a fazer cálculos sobre o modelo de objetos que podem ser feitos de forma muito mais eficiente no modelo relacional
- em que medida o cliente que escreveram exercita os cenários de teste descritos automaticamente (e não faz qualquer leitura do *stdin*!)

Anexo

Para escrever o *SimpleClient* e para a inicialização da base de dados considerem os dados indicados abaixo. Porque há várias operações que são sensíveis à data corrente e queremos que a execução do *SimpleClient* seja determinística, definam uma classe com um método que é suposto dar a data corrente mas que tem uma implementação *mock* que devolve sempre ***20/04/2020 14:00*** *(que é uma segunda-feira).* Usem esse método para obter a data corrente sempre que ela for precisa (no código do negócio e no *SimpleClient*).

Estado da base de dados

Tipos de Instalações.

Existem 4 tipos de instalações: *Sala de Bicicletas*, *Estúdio*, *Piscina*, *Campo de Ténis*

Instalações

Existem 4 instalações:

	lotação max	*tipo*	*modalidades*
Estúdio 1	*20*	*Estúdio*	*Pilates, Step, GAP*
Estúdio 2	*15*	*Estúdio*	*Pilates*
Biclas	*10*	*Sala de Bicicletas*	*Indoor Cycling*
Piscina 25	*20*	*Piscina*	*Hidroginástica*

Modalidades

Existem 5 modalidades: *Pilates*, *Step, GAP, Indoor Cycling, Hidroginástica.* A duração mínima das aulas destas modalidades é, respetivamente, *50, 45, 50, 55, 45* e o custo por hora é, respetivamente, *7, 10, 10, 10, 15*.

Utentes

Existem 5 utentes.

num	*nome*	*nif*
1	*Ulisses*	*223842389*
2	*David*	*256039682*
3	*Teresa*	*269901841*
4	*Querubim*	*197672337*
5	*Cícero*	*221057552*

Cenário *SimpleClient*

A vermelho estão indicados os casos de uso que devem terminar sem sucesso, e os dados que causam a situação.

1. Criar aula com os seguintes dados:
   modalidade: *Pilates*, nome aula: *PLT001*
   dias da semana: 3ª e 5ª, hora início: *09:15*, duração: *55*
2. Criar aula com os seguintes dados:
   modalidade: *Pilates*, nome aula: *PLT002*
   dias da semana: 3ª e 5ª, hora início: *12:15* duração: *55*
3. Criar aula com os seguintes dados:
   modalidade: *Pilates*, nome aula: *PLT003*
   dias da semana: 2ª, 4ª e 6ª, hora início: *12:15* duração: *55*
4. Criar aula com os seguintes dados:
   modalidade: *GAP*, nome aula: *GAP001*

dias da semana: 2ª, 4ª e 6ª, hora início: *09:00* duração: *45*

5. Criar aula com os seguintes dados:
   modalidade: *GAP*, nome aula: *GAP001*
   dias da semana: 2ª, 4ª e 6ª, hora início: 09:00 duração: 50

6. Criar aula com os seguintes dados:
   modalidade: *STEP*, nome aula: *STP001*
   dias da semana: 2ª, 4ª e 6ª, hora início: *09:15* duração: *45*

7. Ativar aula com os seguintes dados:
   aula: *PLT001* instalação: *Estúdio 1* max: *2*
   início: "*data de hoje*", fim: 31/07/2020

8. Ativar aula com os seguintes dados:
   aula: *PLT002* instalação: *Estúdio 1* max: *2*
   início: "*data de hoje*", fim: 31/07/2020

9. Ativar aula com os seguintes dados:
   aula: *GAP001* instalação: *Estúdio 1* max: *2*
   início: "*data de hoje*", fim: 31/07/2020

10. Ativar aula com os seguintes dados:
    aula: *STP001* instalação: *Estúdio 2* max: *2*
    início: "*data de hoje*", fim: 31/07/2020

11. Ativar aula com os seguintes dados:
    aula: *STP001* instalação: *Estúdio 1* max: *2*
    início: "*data de hoje*", fim: 31/07/2020

12. Fazer inscrição com mensalidade do utente nº1 numa aula de Pilates; escolhendo a primeira aula devolvida na lista (que deve ser a PLT001)

13. Fazer inscrição com mensalidade do utente nº3 numa aula de Pilates; escolhendo a segunda aula devolvida na lista (que deve ser a PLT002)

14. Fazer inscrição com mensalidade do utente nº2 numa aula de Pilates; escolhendo a primeira aula devolvida na lista (que deve ser a PLT001)

15. Fazer inscrição avulso do utente nº4 numa aula de Pilates; escolhendo a primeira aula devolvida na lista (que deve ser a PLT002)

16. Fazer inscrição avulso do utente nº5 numa aula de Pilates; nenhuma escolha possível

17. Fazer inscrição com mensalidade do utente nº5 numa aula de Pilates; escolhendo a primeira aula devolvida na lista (que deve ser a PLT002)

18. Visualizar a ocupação da instalação *Estúdio 1 na* "*data de amanha*"